Парадигма развития науки

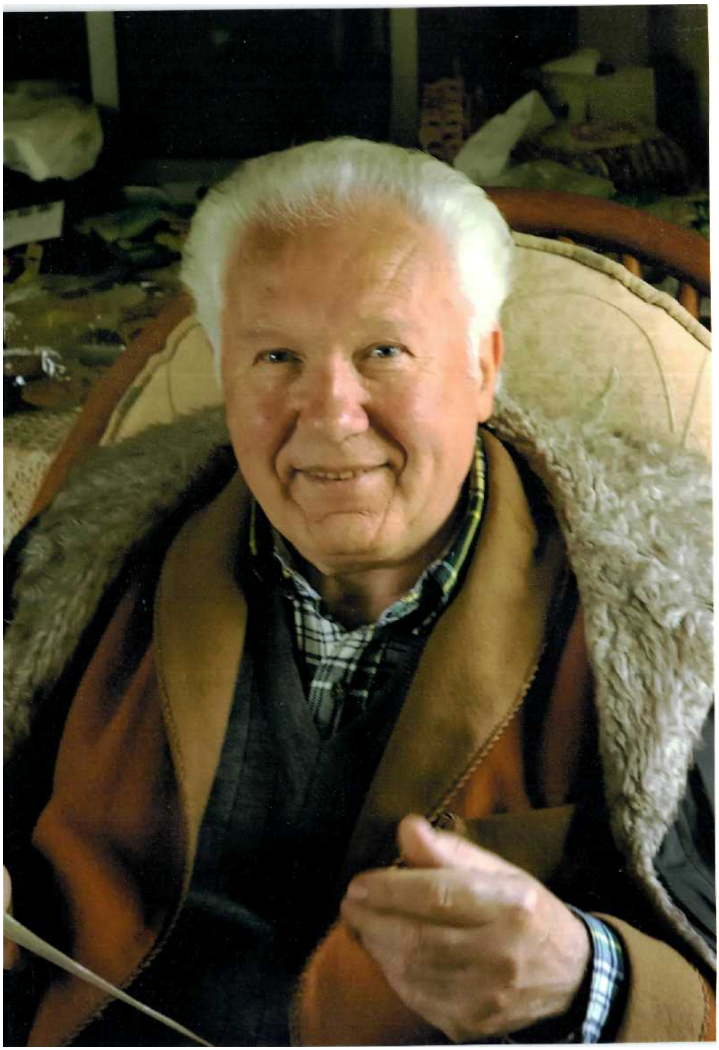
А. Е. Кононюк

Основы фундаментальной теории искусственного интеллекта

Книга 1

Введение в теорию систем искусственного интелекта

Киев
«Освіта України»
2017



Кононюк Анатолий Ефимович

Структурная схема
парадигмы развития науки

УДК 51 (075.8)
ББК В161.я7
К65

Рецензент:
Н.К.Печурин - д-р техн. наук, проф. (Национальный авиационный университет).

Кононюк А. Е.
К213 Основы фундаментальной теории искусственного интеллекта. — В 20-и кн. Кн.1. — К.:Освіта України. 2017.—730 с.

ISBN 978-966-373-693-8 (многотомное издание)
ISBN 978-966-373-694-5 (книга 1)

Многотомная работа посвящена систематическому изложению общих формализмов, математических моделей и алгоритмических методов, которые могут быть исспользуемых при моделировании и исследованиях математических моделей объектов искусственного интеллекта.

Развиваются представления и методы решения, основанные на теориях эвристического поиска и автоматическом доказательстве теорем, а также процедуральные методы, базирующиеся на классе проблемно-ориентированных языков, сочетающих свойства языков программирования и автоматических решателей задач отображения искусственного интеллекта различными математическими средствами.

В работе излагаются основы теории отображения искусственного интеллекта такими математическими средствами как: множества, отношения, поверхности, пространства, алгебраические системы, матрицы, графы, математическая логика и др.

Для бакалавров, специалистов, магистров, аспирантов, докторантов всех специальностей.

УДК 51 (075.8)
ББК В161.я7

ISBN 978-966-373-693-8 (многотомное издание)
ISBN 978-966-373-694-5 (книга 1)

Оглавление

Введение

Понятие искусственного интеллекта, как пишет в Интернете Евгений Масунов, витает в воздухе уже очень давно: еще в древнегреческих мифах содержались истории о механических людях, копирующих поведение человека. Первые вычислительные устройства воспринимались как «логические машины» и были призваны воспроизводить такие особенности человека, как память и базовые арифметические способности. А инженеры видели свою фундаментальную задачу в том, чтобы попытаться создать искусственный мозг.

По мере развития технологий и, что еще важней, понимания того, как работает наш разум, представление об искусственном интеллекте тоже переживало эволюцию. Его задача перестала состоять в осуществлении сложных вычислений, а сосредоточилась на копировании процесса принятия решений человеком и выполнении задач с более человечным подходом.

Искусственные интеллекты — устройства, наделенные синтезированным разумом, — часто подразделяются на две фундаментальные группы: прикладную и общую. Прикладной искусственный интеллект имеет более простое и узкое назначение: такие системы могут торговать акциями или управлять автономным автомобилем.

Общие системы искусственного интеллекта в теории способны выполнять любое задание, и именно в этой сфере сегодня происходят самые захватывающие события. Развитие этого направления привело к возникновению такого феномена, как машинное обучение, которому мы уделим внимание в одном из томов настоящей работы.

Попытки сгенерировать искусственные мыслящие организмы всерьез начали возникать более 70 лет назад, когда стали появляться предположения о том, что компьютеры могут мыслить как люди. Амбициозные прогнозы привлекли серьезное финансирование, но через несколько десятилетий работа в этом направлении не принесла каких-то существенных плодов. Только в последние 25 лет, благодаря новому подходу к искусственному интеллекту и прорывам в технологиях, ученые приблизились к осуществлению мечты тех, кто стоял у истоков.

Во времена Второй мировой войны над этой проблемой работали ученые, представлявшие многие дисциплины, включая такие зарождавшиеся науки, как неврология и информатика.

В Великобритании вопросом разумных машин занялись математик Алан Тьюринг и невролог Уильям Грей Уолтер. Своими идеями они обменивались во время ужинов в престижном кружке под названием Ratio Club. Уолтер прославился тем, что создал первых в истории роботов. Тьюринг же изобрел так называемый «тест Тьюринга», который задал принципиальную планку для интеллектуальной машины: компьютер должен заставить человека подумать, что тот взаимодействует с другим человеком.

В 1950 году в свет вышла книга «Я, Робот» — сборник рассказов популярного фантаста Айзека Азимова. Азимов одним из первых писателей рассказал об идее машинного интеллекта и пофантазировал о его будущем. Книга, которая стала чрезвычайно популярной, заставила человечество задуматься и стала важнейшим источником вдохновения для целого поколения ученых. Самая известная часть книги — так называемые «Три закона робототехники», призванные уберечь человечество от восстания машин. Помимо абстрактных фантазий о далеком будущем, работа Азимова содержала и вполне реальные предсказания, которые уже успели сбыться. Например, речь идет о компьютере, способном хранить все человеческие знания, которому любой человек может задать любой вопрос.

Собственно термин «искусственный интеллект» впервые прозвучал в 1956 году на летней конференции в Дартмутском Университете, организованной молодым специалистом по информатике Джоном Маккарти. Конференция была отмечена оживленной дискуссией о том, как именно стоит подходить к проблеме искусственного интеллекта. Одна группа, в числе которой был влиятельный ученый Марвин Минский, склонялась к так называемому подходу «сверху вниз» — предварительно запрограммированному компьютеру, работающему по законам, которые управляют поведением человека. Другие предложили подход «снизу вверх», основанный на нейронных сетях, которые симулируют работу клеток мозга и самостоятельно обучаются новым типам поведения. В то время верх взял подход Минского, и на пару с Маккарти он получил крупный грант от американского правительства, которое надеялось, что открытия в этой сфере позволят США укрепить свои позиции в противостоянии Советскому союзу в рамках холодной

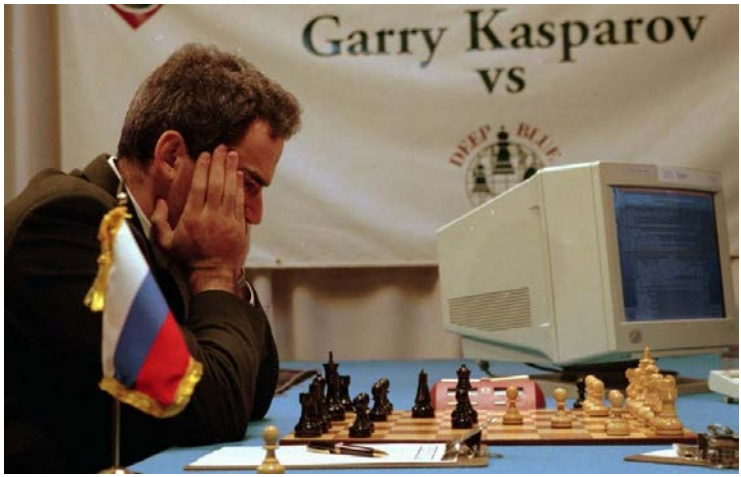
войны. Кстати, Марвин Минский оказал серьезное влияние и на научную фантастику. В 1968 году он работал консультантом у Стэнли Кубрика на съемках фильма «2001 год: Космическая одиссея», в котором одним из персонажей был наделенный интеллектом компьютер HAL 9000. Во время одной из сцен HAL дает интервью BBC, рассказывая о своей миссии и заявляя, что он «совершенно безопасен и не способен совершать ошибки». Участвующий в миссии ученый в своем интервью добавляет, что HAL, по его мнению, может иметь и самые подлинные эмоции. В фильме были отражены предсказания ученых того времени: например, прогноз того же Минского о том, что очень скоро машины по своему уровню интеллекта приблизятся к человеку. В картине также блистательно переданы некоторые страхи общества, например, связанные с тем, что машины могут захватить власть и начать вредить людям. К концу 60-х стало ясно, что Минский погорячился со своими прогнозами, и мечты первых визионеров отделяют от реализации долгие годы и даже десятилетия. Свидетельством того, что искусственный интеллект топчется на месте, стал представленный в 1969 году робот Shakey.

Shakey стал первым роботом, который способен самостоятельно принимать решения, исходя из окружающей обстановки. Перед перемещением у него внутри выстраивалась карта местности, но даже в помещениях с минимумом препятствий робот передвигался непозволительно медленно. Перед каждым движением вперед Shakey

приходилось обновлять свою карту, а появление в его поле зрения движущегося объекта могло ввести робота в ступор, и вычисления, необходимые для осуществления следующего шага, могли занять более часа.

Наступили семидесятые, и положение отрасли только ухудшилось. Невзирая на многомиллионные денежные вливания, искусственный интеллект так и не продемонстрировал серьезного движения вперед. В Конгрессе США все чаще звучали призывы отказаться от финансирования этого направления, а в 1973 году ведущий британский математик Сэр Джеймс Лайтхилл выступил в парламенте с разгромной речью о положении дел в данной сфере. Он заявил, что в настоящий момент машины по своему интеллекту сопоставимы с шахматистом с любительским уровнем мастерства. С таким развитием даже не приходится говорить о том, что когда-нибудь они смогут выполнять такие простые на первый взгляд задачи, как распознавание лиц. В итоге было принято решение существенно урезать финансирование, и исследования в области искусственного интеллекта перешли в стадию анабиоза. Переломным моментом, ознаменовавшим выход индустрии из спячки, принято считать появление коммерческих решений, в которых искусственный разум помог компаниям сэкономить деньги и сократить расходы. Новые коммерческие системы не преследовали тех сверхамбициозных целей, о которых заявляли пионеры искусственного интеллекта. Вместо создания универсального синтезированного разума эти «экспертные системы» были сосредоточены на выполнении куда более узких задач. Это означало, что их было необходимо программировать для решения какой-то одной конкретной проблемы. Первой успешной экспертной системой для бизнеса стала RI, которую взяла на вооружение компания Digital Equipment Corporation в начале восьмидесятых. Эта система, которая помогала составлять заказы на продукцию, позволила компании экономить до 40 миллионов долларов в год. Однако коммерческое использование искусственного интеллекта не выглядит так же романтично, как мечтания об имитации биологии, которые по-прежнему занимали умы исследователей. В 1990 году ученый Родни Брукс опубликовал статью под названием «Слоны не играют в шахматы». Автор был вдохновлен существенными прорывами в нейрологии, в которой начали открываться загадки, связанные с человеческим мышлением и восприятием. Например, для распознавания визуальной информации зрению необходима совместная работа различных «модулей» мозга без какого-либо центрального управления. Главным тезисом статьи Брукса стало заявление, что подход «сверху вниз», при котором компьютер

предварительно программируется на следование правилам интеллектуального поведения, является неверным. Это заключение позволило возродить подход «снизу вверх», а вместе с ним и совершенно немодную в те времена проблематику нейронных сетей. Однако сторонники подхода «сверху вниз» не собирались сдаваться и в 1997 году одержали важную победу. Суперкомпьютер Deep Blue сразился в шахматы с мировым чемпионом Гарри Каспаровым и в конечном итоге вырвал у легендарного гроссмейстера победу. В теории созданная IBM машина по своим возможностям превосходила Каспарова: хотя бы в том, что за секунду могла просчитать до 200 миллионов возможных ходов. Но в шахматах этого недостаточно: важным аспектом является умение мыслить стратегически. Поединок, который журналисты прозвали «Последним фиаско человеческого мозга», закончился такой красивой победой компьютера, что Каспаров даже предположил, что где-то за кулисами сидел живой человек, который управлял Deep Blue. Некоторые назвали этот момент возрождением интереса к искусственному интеллекту, скептики же сочли это еще одним примером создания машины, которая отлично справится с поставленной задачей, если правильно ее запрограммировать на решение конкретной проблемы с четкими правилами и законами.



Гарри Каспаров и Deep Blue

Родни Брук, о котором мы говорили выше, не стал ограничиваться только теоретическими размышлениями и занялся непосредственным продвижением своих идей. В 2002 году учрежденная им компания iRobot создала первого коммерчески успешного робота для дома — самоуправляемого робота-пылесоса под названием Roomba.

Уборка комнаты, безусловно, слишком скромный масштаб свершений для технологии, с которой ученые связывают такие большие надежды. Однако Roomba нельзя не назвать серьезным прорывом. Его многослойные системы генерирования поведения были намного проще алгоритмов робота Shakey и больше напоминали роботов, которых создал Грей Уолтер более 50 лет назад. Несмотря на относительно скромную функциональность датчиков и минимальную вычислительную производительность, устройству хватало интеллекта для того, чтобы качественно убирать квартиру. Появление Roomba ознаменовало собой новую эпоху автономных роботов, сосредоточенных на выполнении конкретных задач.

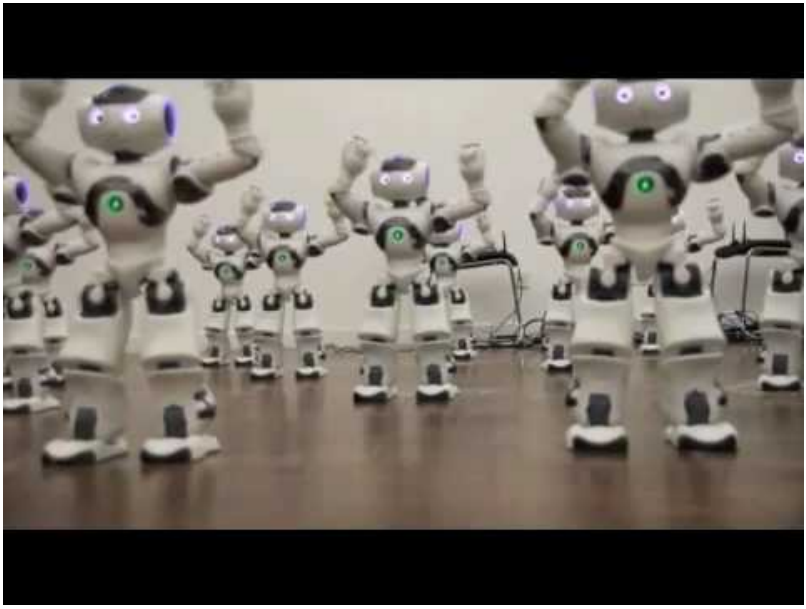
Несмотря на то, что холодная война закончилась, и искусственный интеллект в те годы не оправдал надежд американских военных, в середине двухтысячных они вернулись к этой проблематике, избрав новый подход. Власти начали активно инвестировать в автономных роботов, и в 2005 году мир увидел первый плод этих работ — BigDog от компании Boston Dynamics. Этого устрашающего четвероногого робота, рассчитанного на выполнение задач в условиях среды, в которых невозможна работа традиционных транспортных средств, пока так и не удалось увидеть в действии. Зато компания iRobot сумела

внести заметный вклад в развитие этого направления. Их робот-сапер PackBot объединил в себе пользовательское управление и интеллектуальные способности, такие как умение распознавать запах взрывчатки. Более 2000 таких роботов с успехом использовались в ходе вооруженных конфликтов в Ираке и Афганистане.

Кажется, лед тронулся, и прогресс начали ощущать не только в академической среде, но и на массовом рынке. В ноябре 2008 года смартфон iPhone получил функцию, которой мало кто придал больше значение: в поисковом приложении Google появилось распознавание речи.

На первый взгляд тут нет ничего сложного, но в действительности это изобретение стало предвестником гигантского прорыва. Хотя распознавание речи с давних времен является одним из важнейших аспектов искусственного интеллекта, за десятилетия упорной работы точность распознавания так и не удалось поднять выше отметки в 80%. Компания Google выбрала для решения этой задачи принципиально новый подход: тысячи мощных компьютеров, которые работают в параллельных нейросетях, самостоятельно обучаются обнаруживать паттерны и нюансы в гигантских объемах данных, поступающих от миллионов пользователей Google. Поначалу точность распознавания была на весьма посредственной отметке, но теперь, спустя годы обучения и совершенствования, Google утверждает, что технология работает с 92-процентной точностью.

На фоне того, как гигантские вычислительные сети продолжали менять сущность искусственного интеллекта, более простые и узкоспециализированные компьютеры тоже переживали ренессанс.



Например, роботы-гуманоиды, в числе которых модель под названием NAO, могли вытворять такие вещи, о которых Shakey даже мечтать не мог. Воспользовавшись последними достижениями технологий в области нейросетей и машинного обучения, создатели разработали машину, которая поразила воображение многих. Шанхайская выставка World Expo 2010 года наверняка запомнится в первую очередь потрясающим танцем 20 роботов NAO, которые двигались в полной гармонии с музыкой, а их движения были не запрограммированными паттернами, а чистой импровизацией.

Спустя 14 лет после исторического поражения Гарри Каспарова перед искусственным интеллектом возник еще один вызов, который с успехом был принят. В 2011 компьютер Watson от все той же корпорации IBM принял участие в американском телешоу Jeopardy, которое является прообразом программы «Своя игра» на канале НТВ.

Для машины эта задача была куда сложней, чем игра в шахматы. Watson должен был отгадывать загадки и отвечать на вопросы, требующие не только сообразительности, но и эрудиции. Создатели компьютера использовали нейросети и в течение более трех лет обучали машину распознавать паттерны вопросов и ответов. В итоге Watson не оставил шансов своим соперникам — двум лучшим игрокам за всю историю шоу. Этот эпизод произвел большой фурор и был преподнесен СМИ как триумф искусственного интеллекта.

Женя Густман

2014 год был отмечен еще одним любопытным и немного спорным событием для развития искусственного интеллекта: российские и украинские разработчики представили миру Женю Густмана — первого виртуального собеседника, которому удалось пройти тест Тьюринга. 7 июня 2014 года, на конкурсе, посвященном 60-летию со дня смерти британского математика, Густман, который представляется 13-летним мальчиком из Одессы, убедил 33% судей в том, что он человек. Скептики не придали этому событию большого значения, утверждая, что 11 лет разработки виртуального собеседника были сосредоточены не на создании настоящего искусственного разума, а на поиске способов одурачить судей и выиграть в конкурсе.

С 2014 года мы успели стать свидетелями ряда других любопытных разработок, которые показали, насколько далеко продвинулась сфера искусственного интеллекта за последние 70 лет. Многомиллиардные инвестиции Google в самоуправляемые автомобили, голосовой

переводчик Skype, работающий в реальном времени, дроны, самостоятельно следящие за людьми — похоже, еще немного, и некогда утопическая идея станет частью нашей обыденности.

А теперь представьте, насколько противоречивые чувства может испытывать эксперт по машинному обучению, создающий систему искусственного интеллекта (ИИ), которая однажды, а, возможно, даже и очень скоро, сможет самостоятельно создавать новые ИИ. И при этом эти ИИ будут эффективнее, чем созданные изначально самим человеком. Грядет эра, в которой машины будут сами создавать себе собственную замену.

В настоящий момент специалисты по машинному обучению очень ценятся на развитых рынках труда, однако, когда в мире начинает создаваться программное обеспечение, способное «учиться обучать» себе подобных, — недалек тот день, когда и такие специалисты станут совершенно невостребованными.

В подобной ситуации могут в скором времени оказаться такие группы, как Google Brain, OpenAI, DeepMind, а также кафедры самых престижных технологических школ и институтов, занимающиеся разработкой систем машинного обучения для того, чтобы эти системы в будущем сами создавали системы машинного обучения. И что страшнее, первые признаки этого можно отметить уже сейчас. Например, исследователи из Google Brain разработали программу, способную создавать ИИ-системы, чья задача заключается в измерении уровня эффективности работы программ по языковой обработке. Тест показал, что написанная машиной программа справляется с этим заданием лучше, чем софт, написанный людьми.

Как указывает MIT Technology Review, глава группы разработчиков Google Brain Джефф Дин рассматривает «автоматизированное машинное обучение» как самый многообещающий исследовательский проект для его команды.

«В настоящий момент при решении той или иной задачи вы полагаетесь на ваш собственный опыт, имеющиеся на руках данные и собственно сами вычисления. Можем ли мы исключить из этого порядка «опыт», если речь идет о машинном обучении?», — задается вопросом Дин.

Если окажется, что ИИ способен последовательно справляться с поставленными задачами на уровнях, сравнимых с теми, которые были продемонстрированы в эксперименте Google Brain, то однажды самосоздаваемые и самообучаемые ИИ смогут привести к более быстрому созданию и адаптации новых технологий.

И все же, несмотря на то что эта сфера по-прежнему в большей степени интересна только энтузиастам, в мире уже можно отметить возрастающее число людей, обеспокоенных тем, что рост и развитие ИИ-систем может в конечном итоге лишить многих их средств к существованию.

Автоматизация призвана изменить не только экономику, но и сам принцип капитализма в целом, принцип, который не изменялся на протяжении столетий. В отдаленной перспективе машины действительно станут дешевле, чем наемные рабочие. Ведь начальству больше не придется беспокоится о том, когда давать и оплачивать своим подчиненным отпуска, страховку, выплачивать зарплату и давать многие другие вещи, которые требуются и ожидаются сотрудниками от своих работодателей. И все же эта более дешевая и более эффективная рабочая сила потребует от нас огромных жертв.

Самым большим сектором экономики, который первым ощутит последствия от автоматизации, станет производство. Особенно в развивающихся странах. Важность этой проблемы затронул даже в своей прощальной речи бывший президент США Барак Обама:

«Волна следующей дезорганизации нашей экономики не будет поступать из-за рубежа. Она будет исходить от неослабевающего темпа автоматизации, которая сделает множество рабочих мест для среднего класса попросту неактуальными», — заявил Обама.

И многие эксперты в индустрии согласны с этими словами. Более того, пострадают не только рабочие места, не требующие особых навыков. Уже разрабатываются системы, которые в будущем смогут заменить, например, киномонтажеров, поэтов-песенников, журналистов и многих других. И сейчас, когда уже существуют ИИ-системы, способные создавать некоторые программы, функционирующие гораздо эффективнее программ, написанных людьми, нам необходимо стать более внимательными к этой проблеме и, наконец, прийти к осознанию того, что может нас ожидать за горизонтом.

Исследователи из университетов в Сингапуре и Германии обнаружили способ, позволяющий превратить чипы оперативной памяти в вычислительное устройство. По сути, открытие это позволило бы компьютеру работать без наличия в нём центрального процессора. При этом следует оговориться о том, что речь идёт исключительно об оперативной памяти нового поколения ReRAM (резистивная память с произвольным доступом), которая должна поступить в продажу в самое ближайшее время.

ReRAM, или RRAM, – это энергонезависимая память, имеющая сходство с технологиями CBRAM и PRAM. Основная суть этой технологии заключается в том, что диэлектрики в момент приложения к ним высокого напряжения способны сформировать внутри себя проводящие нити низкого сопротивления (притом что диэлектрики обладают достаточно высоким сопротивлением). Во время приложения соответствующего уровня напряжения проводящие нити разрушаются, вновь превращая материал в диэлектрик и отключая функции проводника.

Помимо всего прочего, **память ReRAM** использует **троичную систему счисления**, что позволяет хранить в памяти куда более большее количество данных. Ещё одним преимуществом такой памяти является возможность изготавливать её в ощутимо меньших масштабах, нежели существующая сегодня память. Это, в свою очередь, увеличит скорость ввода и вывода данных и заметно понизит потребление энергии. Но на этом список преимуществ ReRAM не заканчивается.

Исследователи из Рейнско-Вестфальского технического университета Ахена и сингапурского Наньянского технологического университета сумели создать новый формат оперативной памяти ReRAM, имеющей в своём распоряжении дополнительное пространство, пригодное для исполнения вычислительных функций. Другими словами, **учёным удалось перенести часть вычислительных операций из центрального процессора в свободные ячейки оперативной памяти.** Сами исследователи утверждают, что в будущем эта технология позволит избавить от вычислительных процессоров не только компьютеры и лэптопы, но и смартфоны, планшеты и прочих представителей Интернета вещей.

Разработчики утверждают, что открытая ими технология способна заметно увеличить вычислительную мощность и скорость работы компьютеров, так как данные больше не придётся передавать между центральным процессором и оперативной памятью. Также технология позволит значительно уменьшить размеры материнских плат и сэкономить потребление энергии, ведь вместо двух компонентов останется лишь один. Подробности технологии были опубликованы на сайте Nature.com, где с ними может ознакомиться любой желающий.

1. ИНТЕЛЛЕКТ (Основные положения)

1.1. Введение

Круг проблем, объединяемых терминами «интеллект» и «искусственный интелект» достаточно широк и довольно неопределен, значительная часть данного раздела посвящена определению этих понятий.

Наиболее характерной и специфической разновидностью умственной деятельности является решение задач.

Слово «задача» мы будем употреблять дальше в самом широком смысле. Поэтому прежде всего уточним, что будет подразумеваться под этим словом.

Типичным примером задачи является отыскание пути к заранее указанному месту (цели) в каком-либо ограниченно знакомом районе. Эта задача может возникнуть перед человеком, оказавшимся в лесу или в лабиринте. Процесс решения подобного рода задач мы склонны представлять как поиск некоторого пути обхода препятствий, пути преодоления трудностей. Заметим, что трудность решения в какой-то мере входит в само понятие задачи: **там, где нет трудностей, нет и задачи.** Поэтому, **если с самого начала имеется средство непосредственного (без преодоления каких-либо трудностей) достижения цели, то будем говорить, что задачи нет или задача не поставлена**

Таким образом, **всякая задача предполагает необходимость поиска некоторого алгоритма (метода) и/или средства для достижения так или иначе определенной (поставленной), но непосредственно недоступной цели. Решение задачи означает нахождение этого алгоритма или средства.**

Понятие **алгоритма** принадлежит к числу основных понятий кибернетики — науки об управлении, переработке информации и связи в живой и неживой природе, которая (**информация**) является, по существу, **теоретической основой как естественного так и искусственного интелекта**. Под алгоритмом понимают точное предписание о выполнении в определенном порядке системы операций для решения любой задачи из некоторого данного класса (множества) задач.

Разумеется, высказанная формулировка не является строгим математическим определением понятия «алгоритм». Она, скорее, трактует значение этого термина, поясняя его смысл. Тем не менее эта формулировка близка и понятна каждому человеку, она отражает то понятие алгоритма, которое постепенно складывалось с древнейших времен по мере решения человеком все новых и новых задач.

В математике и кибернетике серия (класс) задач определенного типа считается решенной, когда для ее решения установлен алгоритм. Например, в алгебре установлены алгоритмы, которые по любым заданным коэффициентам алгебраического уравнения позволяют чисто механически определить (с любой наперед заданной точностью) корни этого уравнения; в кибернетике установлены алгоритмы оптимального управления. **Нахождение алгоритмов является естественной целью человека при решении им разнообразных классов задач**.

Отыскание алгоритма для задач некоторого данного типа (особенно «оптимального», «хорошего» алгоритма) связано с тонкими и сложными рассуждениями, требующими большой изобретательности и высокой квалификации. Принято считать, что **подобного рода деятельность требует участия интеллекта человека**. **Задачи, связанные с отысканием алгоритма решения класса задач определенного типа, будем называть интеллектуальными.**

Что же касается задач, алгоритмы решения которых уже установлены, то, как отмечает известный специалист в области «искусственного интеллекта» М. Минский, «излишне приписывать им такое мистическое свойство, как «интеллектуальность». В самом деле, после того, как такой алгоритм уже найден, процесс решения соответствующих задач становится таким, что его могут в точности выполнить человек, компьютер (должным образом запрограммированный) или робот, не имеющие ни малейшего представления о сущности самой задачи. **Требуется только, чтобы лицо, решающее задачу (человек, компьютер или робот), было способно выполнять те элементарные операции, из которых складывается процесс, и, кроме того, чтобы оно педантично и аккуратно руководствовалось предложенным алгоритмом.** Такое лицо, действуя, как говорят в таких случаях, чисто машинально, может успешно решать любую задачу рассматриваемого типа.

Поэтому представляется совершенно естественным исключить из класса интеллектуальных такие задачи, для которых существуют стандартные методы решения. Примерами таких задач могут служить чисто вычислительные задачи: решение системы линейных алгебраических уравнений, численное интегрирование

дифференциальных уравнений и т. д. Для решения подобного рода задач имеются стандартные алгоритмы, представляющие собой определенную последовательность элементарных операций, которая может быть легко реализована в виде программы для компьютера. В противоположность этому для широкого класса интеллектуальных задач, таких, как распознавание образов, игра в шахматы, доказательство теоремы и т. п., напротив, это формальное разбиение процесса поиска решения на отдельные элементарные шаги часто оказывается весьма затруднительным, даже если само их решение и несложно.

Трудность разбиения решения интеллектуальных задач на элементарные шаги обычно бывает связана с трудностью формального описания этих задач. Например, человек может отличить кошку от собаки или русскую речь от английской, совершенно не будучи в состоянии дать формальное описание соответствующего алгоритма распознавания.

Тип задачи предопределяет алгоритм ее решения. В чем состоит специфика алгоритмов решения **интеллектуальных задач**? Рассмотрим в качестве примера алгоритм решения «задачи на доказательство». **Процесс решения этой задачи — результат интеллектуальной деятельности — есть доказательство,** т. е. последовательность логических операций или шагов, начинающихся с условий (предпосылок) задачи и заканчивающихся заключением теоремы. При этом каждый шаг приводит к некоторому новому положению, полученному из соответствующим образом подобранных отдельных условий или из уже известных фактов (аксиом), или из ранее доказанных положений (лемм).

Многие нематематические задачи можно себе представить в том же аспекте. Например, технологу при наладке производства нового изделия необходимо спланировать, скоординировать и уложить в согласованную схему множество операций: выбор и подготовку оборудования, формулировку конструктивных требований и ограничений, обоснование технологического маршрута и т. п. Сверх того в его обязанности может входить согласование этих технологических операций с операциями совершенно иного характера (финансовыми, экологическими, юридическими и т. п.). Все эти операции взаимосвязаны и могут быть спланированы так, что в результате их осуществления будет получено требуемое изделие с нужными свойствами.

Таким образом, чтобы **решить «задачу на доказательство», необходимо составить хорошо скоординированную, согласованную схему операций (логических, кинематических или прикладных),**

начинающуюся с услоиий (предпосылок) и заканчивающуюся заключением (целью), последовательно ведущую от данных к неизвестному, от объектов, находящихся в нашем распоряжении, к объектам, которых нужно достичь.

Решая интеллектуальные задачи, мы постоянно ищем пути и средства к достижению той или иной цели, пытаемся выработать какой-то план действий, следуя которому, можна достичь этой непосредственно недоступной цели. Именно способность к преодолению трудностей и препятствий, к нахождению обходного пути к цели там, где нет прямого пути, отличает интеллектуальную деятельность от неинтеллектуальной, возвышая «умное» животное над «глупым», человека — над самым «умным» животным и талантливых людей — над другими людьми. (Разумеется, интеллектуальная деятельность человека отличается от таковой у животных. **Важнейшим отличием человеческого мышления является язык, с помощью которого человек получает возможность значительно расширить свои интеллектуальные способности и класс решаемых задач**. Но и в жизни животных интеллектуальная деятельность также играет огромную роль, нередко определяя возможность их существования.).

Умение **решать интеллектуальные задачи, т. е. способность целенаправленно преобразовывать имеющуюся информацию (знания), приобретается путем обучения на опыте подобно, скажем, умению плавать.** Учась плавать, мы подражаем пловцам в том, что они делают руками и ногами, чтобы плавать. При помощи упражнений мы вырабатываем у себя необходимые навыки. Точно так же, учась решать задачи, мы наблюдаем и подражаем другим в том, как они это делают, и, наконец, мы вырабатываем у себя соответствующие навыки путем упражнений, т. е. **путем обучения на опыге**. Заметим, **что умение решать задачи гораздо важнее, чем одно чистое знание, хотя, конечно, без необходимых, знаний невозможно и решение.**

Резюмируя вышеизложенное, попытаемся теперь определить, что же такое интеллект?

1.2. Базовые понятия и определения

Прежде всего заметим, что четкого, общепризнанного определения термина «интеллект» пока еще нет. Поэтому предлагаемое определение будет носить некоторый рабочий (возможно, неполный) характер.

Под интеллектом будем подразумевать способность мозга решать (интеллектуальные) задачи путем приобретения, запоминания и

целенаправленного преобразования знаний (информации) в процессе обучения на опыте и адаптации к разнообразным обстоятельствам.
В этом определении под термином «знания» мы подразумеваем не только ту информацию (непосредственные впечатления), **которая поступает в мозг через органы чувств. Такого типа знания чрезвычайно важны (необходимы), но недостаточны для интеллектуальной деятельности.** Дело в том, что **сущности (субъекты, объекты, процессы, явления) и окружающая их среда обладают свойствами не только воздействовать на органы чувств, но и находиться друг с другом в определенных отношениях**. Ясно, что для того чтобы человее мог осуществлять в окружающей среде интеллектуальную деятельность (или хотя бы просто существовать), **человеку необходимо иметь в системе знаний модель этой среды. В этой информационной моделе окружающей среды реальные сущности, их признаки и свойства и отношения между ними не только отображаются и запоминаются, но и, как это отмечено в данном определении интеллекта, могут мысленно (т. е. в мозгу) «целенаправленно преобразовываться».** При этом существенно то, что формирование модели внешней среды происходит «в процессе обучения на опыте и адаптации к разнообразным обстоятельствам».

Приведем еще одно определение понятия «интеллект».

ИНТЕЛЛЕКТ (лат. intellectus — ум, рассудок, разум) — в общем смысле способность мыслить; в гносеологии — способность к опосредованному, абстрактному познанию, включающая в себя такие функции, как сравнение, абстрагирование, образование понятий, суждение, умозаключение; противостоит непосредственным видам познания — чувственному и интуитивному; в психологии — рациональное, подчиненное законам логики мышление; противостоит нерациональным сферам психики — эмоциям, воображению, воле и т. д. Общая способность к познанию и решению трудностей, которая объединяет все **познавательные способности человека**: **ощущение, восприятие, память, представление, мышление, воображение.**
Рассмотрим подробно куждую познавательную способность человека

1.2.1.Ощущение

Ощуще́ние, **чу́вственный о́пыт** — простейший психический процесс, представляющий собой психическое отражение отдельных признаков,

свойств и состояний внешней среды, субъектом внутренних или внешних стимулов и раздражителей при участии нервной системы.

Ощущение и восприятие

В психологии ощущение считается процессом отражения отдельных свойств объектов окружающего мира.

От понятия восприятия (как целостного впечатления, например, восприятие чашки кофе как целостного образа) **отличается количественно** (ощущение аромата кофе, цвета, температуры и т. д.). **Восприятие состоит из одного или более ощущений, создающих наиболее полное представление об объекте.** То есть, восприятие будет, даже если человек не прикасается к чашке, с закрытыми глазами будет вдыхать аромат кофе (одно ощущение).

Поэтому неудивительно, когда в данном случае **зрение рассматривается как орган восприятия**, **задача которого** – посредством соответствующей деятельности мозга – **распознавать** по тому, что видимо, то, что невидимо.

Характеристики ощущений

- **Модальность** — качественная характеристика ощущений. Каждый вид ощущений имеет свои модальные характеристики. Для зрительных ощущений таковыми могут быть цветовой тон, светлота, насыщенность; для слуховых — высота тона, тембр, громкость; для тактильных — твердость, шероховатость и т. д.
- **Интенсивность** — классическая количественная характеристика ощущений.
- **Локализация** — пространственная характеристика ощущений, информация о локализации раздражителя в пространстве. В некоторых случаях (болевые, интероцептивные ощущения) локализация затруднена, неопределенна.
- **Длительность** — временная характеристика ощущения.

Классификация ощущений

Первым, кто попытался классифицировать ощущения, был Аристотель. Он выделил **5 чувств**: **зрение, слух, осязание, обоняние и вкус**. В XIX веке увеличение количества видов ощущений поставило задачу их классификации.

Классификация Вундта

В. Вундт предлагал группировать ощущения в зависимости от свойств вызывающих их раздражителей, среди которых он выделял механические, физические и химические свойства (например, **зрительные и слуховые ощущения** относятся к разряду «физических», поскольку вызываются физическими явлениями — **электромагнитными колебаниями и звуковыми волнами**; обоняние и вкус — «химические» ощущения и т. д.). Этот вариант классификации не получил широкого распространения.

Классификация Шеррингтона

Ч. Шеррингтон выделил три основных класса ощущений, основываясь на локализации (по месту расположения) рецепторов:

- **Экстероцептивные ощущения**, источником которых являются рецепторы, расположенные на поверхности тела. Они дают **образы внешнего мира.**
- **Интероцептивные ощущения**. Рецепторы находятся во внутренних органах. Они сигнализируют о **состоянии внутренних процессов организма** (ощущения голода, жажды, боли и т. п.).
- **Проприоцептивные ощущения**. Рецепторы расположены в мышцах, связках и сухожилиях. Проприоцептивные ощущения играют важнейшую роль в **регуляции движений**, а также дают **информацию о положении в пространстве тела и его частей.**

Шеррингтон разделил **экстероцептивные ощущения на контактные и дистантные.** Первые вызываются приложением воздействия непосредственно к поверхности тела, а вторые возникают, когда

раздражитель действует с некоторой дистанции (обоняние, слух, зрение).

Классификация Хэда

Г. Хэд делил ощущения по их происхождению:

- **Протопатическая чувствительность**, более ранняя по происхождению и примитивная, тесно связана с эмоциями, далека от мышления, она менее дифференцированная и локализованная. Относящиеся к ней ощущения трудно разделять на категории и обозначать словами, описывать.
- **Эпикритическая чувствительность** выше по уровню, возникает позже и обладает, по сути, противоположными характеристиками: связь с мышлением, отдаленность от эмоциональных состояний, большая дифференцированность, категориальные названия для ощущений (красный, синий цвет, а не «запах мяты» или «запах сосны»), четкая локализация.

В работе едва ли не каждого органа чувств есть элементы как протопатической, так и эпикритической чувствительности, хотя их соотношение неодинаково.

Нарушение ощущений

Сенсорная гипопатия — это нарушение чувствительности, которое выражается в резком изменении порогов ощущений, при этом как сильные, так и слабые раздражители вызывают одинаково слабые ощущения.

Сенсорная гиперпатия — это резкое усиление интенсивности ощущений при воздействии слабых раздражителей.

Парестезия — это расстройство, при котором появляются ощущения в виде онемений, ползания мурашек, покалываний при отсутствии реальных раздражителей.

Общие психофизиологические закономерности ощущений

Работа каждого анализатора имеет свои специфические закономерности.

Наряду с этим все виды ощущений подчинены общим психофизиологическим закономерностям.

Для возникновения какого-либо ощущения раздражитель должен иметь определенную величину интенсивности.

Интенсивность определяется силой действующего раздражителя и функциональным состоянием рецептора.

Минимальная величина раздражения, которая вызывает едва заметное ощущение, называется **абсолютным нижним порогом ощущения**. Способность ощущать эти самые слабые раздражения называется **абсолютной чувствительностью. Она всегда выражается в абсолютных числах**. Например, для возникновения ощущения давления достаточно воздействия 2 мг на 1 мм² поверхности кожи.

Верхний абсолютный порог ощущения — максимальная величина раздражения, дальнейшее увеличение которой вызывает исчезновение ощущения или болевое ощущение. Например, сверхгромкий звук вызывает боль в ушах, а сверхвысокий (по частоте колебаний свыше 20000 Гц) — вызывает исчезновение ощущения (слышимый звук переходит в ультразвук). Давление 300 г/кв.мм вызывает боль.

Наряду с абсолютной чувствительностью следует различать относительную чувствительность — чувствительность к различению интенсивности одного воздействия от другого. **Относительная чувствительность характеризуется порогом различения.**

Порог различения, или дифференциальный порог, — едва ощущаемое минимальное различие в силе двух однотипных раздражителей.

Порог различения — это относительная величина (дробь), которая показывает, какую часть первоначальной силы раздражителя надо прибавить (или убавить), чтобы получить едва заметное ощущение изменения в силе данных раздражителей.

Так, если взять груз в 1 кг и затем прибавить ещё 10 г, то этой прибавки никто ощутить не сможет; чтобы почувствовать увеличение прибавки веса необходимо добавить 1/30 часть первоначального веса, то есть 33 г. Таким образом, относительный порог различения силы тяжести равен 1/30 части силы первоначального раздражителя.

Относительный порог различения яркости света равен 1/100; силы звука — 1/10; вкусовых воздействий — 1/5. Эти закономерности открыты Бугером и Вебером (закон Бугера-Вебера).

Закон Бугера-Вебера относится только к средней зоне интенсивности раздражителей. Иначе говоря, относительные пороги теряют значение при очень слабых и очень сильных раздражителях. Это было установлено Фехнером.

Фехнер установил также, что если интенсивность раздражителя увеличивать в геометрической прогрессии, то ощущение будет увеличиваться лишь в арифметической прогрессии. (Закон Фехнера).

Нижние и верхние абсолютные пороги ощущений (абсолютная чувствительность) характеризуют пределы человеческой чувствительности. Но чувствительность каждого человека изменяется в зависимости от различных условий.

Так, входя в плохо освещенное помещение, мы вначале не различаем предметы, но постепенно под влиянием данных условий чувствительность анализатора повышается.

Находясь в накуренном помещении или в помещении с какими-либо запахами, мы через некоторое время перестаем замечать эти запахи (понижается чувствительность анализатора).

Когда из плохо освещенного пространства мы попадаем в ярко освещенное, то чувствительность зрительного анализатора понижается.

Изменение чувствительности анализатора в результате его приспособления к действующим раздражителям называется **адаптацией.**

Разные анализаторы имеют различную скорость и различный диапазон адаптации. К одним раздражителям адаптация происходит более быстро, к другим — медленнее. Более быстро адаптируются обонятельные и тактильные анализаторы. Полная адаптация к запаху йода наступает через одну минуту. Через три секунды ощущение давления отражает только 1/5 силы раздражителя (поиск очков, сдвинутых на лоб, — один из примеров тактильной адаптации). Ещё медленнее адаптируются слуховой, вкусовой и зрительный анализаторы. Для полной адаптации к темноте необходимо 45 мин. После этого периода зрительная чувствительность увеличивается в 200 000 раз (самый высокий диапазон адаптации).

Явление адаптации имеет целесообразное биологическое значение. Оно содействует отражению слабых раздражителей и предохраняет анализаторы от чрезмерного воздействия сильных раздражителей.

Чувствительность зависит не только от воздействия внешних раздражителей, но и от внутренних состояний.

Повышение чувствительности анализаторов под влиянием внутренних (психических) факторов называется **сенсибилизацией**. Так, например, слабые вкусовые ощущения повышают зрительную чувствительность. Это объясняется взаимосвязью данных анализаторов, их системной работой.

Сенсибилизация, обострение чувствительности, может быть вызвано не только взаимодействием ощущений, но и физиологическими факторами, введением в организм тех или иных веществ. Например, для повышения зрительной чувствительности существенное значение имеет витамин А.

Чувствительность повышается, если человек ожидает тот или иной слабый раздражитель, когда перед ним выдвигается специальная задача различения раздражителей. **Чувствительность отдельного человека совершенствуется в результате упражнения**. Так, дегустаторы, специально упражняя вкусовую и обонятельную

чувствительность, различают разнообразные сорта вин, чая и могут даже определить, когда и где изготовлен продукт.

У людей, лишенных какого-либо вида чувствительности, осуществляется компенсация (возмещение) этого недостатка за счет повышения чувствительности других органов (например, повышение слуховой и обонятельной чувствительности у слепых).

Взаимодействие ощущений в одних случаях приводит к сенсибилизации, к повышению чувствительности, а в других случаях — к её понижению, то есть к десенсибилизации. Сильное возбуждение одних анализаторов всегда понижает чувствительность других анализаторов. Так, повышенный уровень шума в «громких цехах» понижает зрительную чувствительность.

Одним из проявлений взаимодействия ощущений является **контраст ощущений.**

Контраст ощущений — это повышение чувствительности к одним свойствам под влиянием других, противоположных свойств действительности. Например, одна и та же фигура серого цвета на белом фоне кажется темной, а на чёрном — светлой.

Ощущения одного вида могут вызвать добавочные ощущения другого вида. Так, звуки могут вызывать цветовые ощущения, цветовые — вкусовые (желтый цвет — ощущение кислого). Это явление называется синестезией.

1.2.2. Восприятие

Восприя́тие, перце́пция (от лат. *perceptio*) — чувственное познание предметов окружающего мира, субъективно представляющееся непосредственным.

Различные трактовки восприятия

Согласно философии эмпиризма, восприятие состоит из ощущений или, в более поздней версии этой философии, из так называемых чувственных данных (Дж. Мур, Б. Рассел и др.). Трактовка ощущений как элементарных «кирпичиков» психического получила особое

распространение в ассоциативной психологии. Философская критика тезиса о возможности построения восприятия из ощущений или чувственных данных была осуществлена, в частности, Г. Райлом и М. Мерло-Понти. В психологии XX века произошёл отказ от трактовки восприятия как соединения атомарных чувственных содержаний (ощущений); **восприятие стало пониматься как целостное и структурное.** Согласно современному психологу Дж. Гибсону, **восприятие — это активный процесс извлечения информации об окружающем мире, включающий в себя реальные действия по обследованию того, что воспринимается.** Понятое таким образом восприятие презентует субъекту те свойства внешнего мира, которые соотносятся с потребностями субъекта и выражают возможности его деятельности в данной реальной ситуации. Согласно У. Найссеру, **извлечение информации происходит на основе имеющихся у субъекта схем различных предметов и мира в целом**. Большинство из этих схем приобретается в процессе опыта, но есть и исходные схемы, которые являются врождёнными. Подобные идеи были высказаны также представителями когнитивной психологии, которые полагают, что **восприятие — это процесс категоризации воспринимаемого, то есть отнесения воспринимаемых предметов к тому или иному классу (категории) объектов, начиная с таких категорий как стол или дерево и заканчивая такими как предмет, причинность и т. д.** Некоторые из этих категорий являются продуктом опыта, другие являются врождёнными.

Часть психологов продолжает рассматривать **восприятие как синтез ощущений,** при этом ощущения трактуются как возникающие в результате непосредственно чувственного познания субъективные переживания силы, качества, локализации и других характеристик воздействия стимулов на органы чувств.

Уровни восприятия

Выделяются **четыре операции или четыре уровня восприятия:** *обнаружение*, *различение*, *идентификация* и *опознание*. Первые два относятся к перцептивным, последние — к опознавательным действиям.

Обнаружение — исходная фаза развития любого сенсорного процесса. На этой стадии субъект может ответить лишь на простой вопрос, есть ли стимул.

Следующая операция восприятия — ***различение,*** или собственно восприятие. Конечный результат её — **формирование перцептивного образа эталона**. При этом развитие восприятия идёт по линии выделения специфического сенсорного содержания в соответствии с особенностями предъявляемого материала и стоящей перед субъектом задачи.

Когда перцептивный образ сформирован, возможно осуществление опознавательного действия. Для опознания обязательны **сличение и идентификация**.

Идентификация есть отождествление непосредственно воспринимаемого объекта с образом, хранящимся в памяти, или отождествление двух одновременно воспринимаемых объектов. ***Опознание*** включает также категоризацию (отнесение объекта к определённому классу объектов, воспринимавшихся ранее) и извлечение соответствующего эталона из памяти.

Свойства восприятия

- **Предметность** — объекты воспринимаются не как бессвязный набор ощущений, а как **образы, составляющие конкретные предметы.**
- **Структурность** — предмет воспринимается сознанием уже в качестве абстрагированной от ощущений **смоделированной структуры**.
- **Апперцептивность** — на восприятие оказывает влияние общее содержание психики человека.
- **Константность** — постоянство восприятия одного и того же дистального объекта при изменении проксимального стимула.
- **Избирательность** — преимущественное выделение одних объектов по сравнению с другими.
- **Осмысленность** — предмет сознательно воспринимается, мысленно называется (связывается с определённой категорией), относится к определённому классу.

  Осмысление состоит из этапов:

  1. **Селекция** — выделение из **потока информации объекта восприятия**

2. **Организация** — объект идентифицируется по **комплексу признаков**
3. **Категоризация** и приписывание объекту свойств объектов этого класса

Константность восприятия

Константность — постоянство восприятия одного и того же дистального объекта при изменении проксимального стимула, **способность распознавать один и тот же объект на основе различающейся сенсорной информации (ощущений).** Воспринимаемый в различных обстоятельствах и условиях объект рассматривается как один и тот же. Так, яркость объекта как величина характеризующая отражённый свет изменяется если переместить его из слабо освещённой комнаты в комнату с хорошим освещением. Тем не менее объект при изменении проксимальной стимульной информации в обоих случаях рассматривается как один и тот же. Можно **выделить константность таких свойств объекта как размер, форма, яркость, цвет. Константность восприятия формы** исследуется на установке, основными элементами которой являются квадрат-эталон (со стороной 10 см) и прямоугольник-измеритель (шириной 10 см). Квадрат-эталон всегда в опыте наклонён к наблюдателю, а плоскость прямоугольника-измерителя должна быть перпендикулярна оси зрения испытуемого. Высота прямоугольника-измерителя может изменяться испытуемым с помощью специальной кнопки. Испытуемого просят подобрать такую высоту прямоугольника-измерителя, чтобы он имел ту же видимую форму, что и наклонённый квадрат-эталон. В опыте варьируется наклон квадрата-эталона (25°, 30°, 35° и 40°). Для каждого значения наклона эталона испытуемый четыре раза подравнивает высоту измерителя. Таким образом получаются данные для вычисления **коэффициента константности**.

Константность восприятия измеряется коэффициентом константности по формуле Брунсвика — Таулесса:

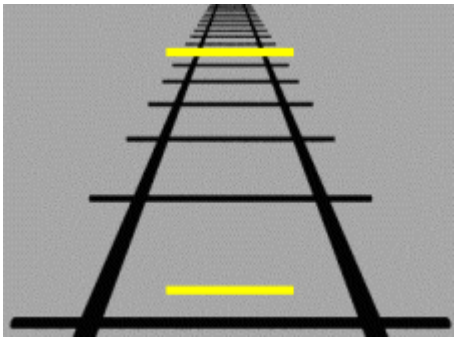
$$K = \frac{V - P}{R - P}$$

где V — высота прямоугольника-измерителя, которую установил испытуемый в стремлении подравнять видимые формы измерителя и эталона, R — высота квадрата эталона, $P=R\cdot\cos\alpha$, где α — угол наклона квадрата-эталона.

Константность восприятия формы в опытах с инверсией поля зрения с помощью инвертоскопа падает до нуля, а в процессе адаптации восстанавливается, достигая доэкспериментального уровня. Эксперименты с инверсией поля зрения человека проводятся для исследования механизмов константности зрительного восприятия.

Одно из объяснений константности восприятия основывается на различении восприятия и чувствительности (ощущения). Восприятие действительных свойств объектов это субъективный психический процесс связывающий ощущения (чувственный опыт) свойств объекта с другой стимульной информацией.

Так свойство размера объекта связывается с расстоянием до объекта, яркость объекта связывается с освещенностью. Субъективный психический процесс восприятия, который позволяет человеку признавать объект одним и тем же даже если он располагается на разном расстоянии от него (у объекта в таком случае различный угловой размер — если он на большом расстоянии — малый угловой размер, если на маленьком расстоянии — большой угловой размер) в некоторых случаях сопровождается «регрессом к действительным объектам». Примером регресса к действительным объектам как следствия константности восприятия являются оптические иллюзии. Так иллюзия Понцо показывает, как осуществляемая восприятием регрессия к действительным объектам, которые располагаются в трехмерном мире, в случае с двухмерным объектом — рисунком — заставляет человека воспринимать горизонтальный отрезок у сходящихся концов вертикальных линий как более длинный, чем отрезок, расположенный у расходящихся концов тех же вертикальных линий, как будто последний расположен «ближе» к наблюдателю.



Пример иллюзии Понцо. Обе горизонтальные линии имеют одинаковый размер.

Факторы восприятия

Внешние

- размер
- интенсивность (в физическом или эмоциональном плане)
- контрастность (противоречие с окружением)
- движение
- повторяемость
- новизна и узнаваемость

Внутренние

- стереотипность восприятия, установка восприятия: ожидание увидеть то, что должно быть уви́дено по прошлому опыту
- потребности и мотивация: человек видит то, в чём нуждается или что считает важным
- опыт: человек воспринимает тот аспект стимула, которому научен прошлым опытом
- я-концепция: восприятие мира группируется вокруг восприятия себя
- личностные особенности: оптимисты видят мир и события в позитивном свете, пессимисты, напротив, — в неблагоприятном

Три механизма селективности восприятия:

- **принцип резонанса** — соответствующее потребностям и ценностям личности воспринимается быстрее, чем несоответствующее
- **принцип защиты** — противостоящее ожиданиям человека воспринимается хуже
- **принцип настороженности** — угрожающее психике человека распознаётся быстрее прочего

Формы и принципы восприятия

- **Фигура — фон** — восприятие выделяет фигуру из фона.
- **Константность** — объекты длительное время воспринимаются одинаково.
- **Группировка** — однообразные стимулы группируются в структуры.

  Принципы группировки:

  - **Близость** — расположенное рядом воспринимается вместе.
  - **Подобие** — схожее по каким-то признакам воспринимается вместе.
  - **Замкнутость** — человек склонен заполнять пробелы в фигуре.
  - **Целостность** — человек склонен видеть непрерывные формы, а не сложные комбинации.
  - **Смежность** — близкое во времени и пространстве воспринимается как одно.
  - **Общая зона** — стимулы, выявленные в одной зоне воспринимаются как группа.

Результат восприятия

Результатом процесса восприятия становится построенный образ.

Образ — субъективное видение реального мира, воспринимаемого при помощи органов чувств.

Получив образ, человек (или другой субъект) производит ***определение ситуации***, то есть оценивает её, после чего принимает решение о своём поведении.

Восприятие в зоопсихологии

Восприятие присуще главным образом высшим живым существам; в слабых формах, позволяющих говорить лишь о зачатках восприятия, нечто подобное можно обнаружить у существ средних стадий эволюции.

Согласно теории Леонтьева, восприятие развилось в результате перехода от гомогенной среды обитания к предметно выраженной.

Восприятие в теории психики

Восприятие — одна из психических функций, сложный **процесс приёма и преобразования сенсорной информации,** формирующий субъективный целостный образ объекта, воздействующего на анализаторы через совокупность ощущений, инициируемых данным объектом.

Как форма чувственного отражения предмета, **восприятие включает обнаружение объекта как целого, различение отдельных признаков в объекте, выделение в нём информативного содержания, адекватного цели действия, формирование чувственного образа.**

Если ощущения отражают лишь отдельные свойства предметов, то синтез множества ощущений объекта создаёт целостную картину, в которой в качестве единицы взаимодействия представлен весь предмет, в совокупности его свойств. **Эта картина называется субъективным восприятием объекта.**

Социальное восприятие

Социальное восприятие — восприятие, направленное на создание представления о себе, других людях, социальных группах и социальных явлениях.

Термин был предложен Джеромом Брунером в 1947 году для обозначения феноменов социальной детерминации процессов восприятия. Современная трактовка термина была дана в рамках социальной психологии.

К механизмам социальной перцепции относят**: рефлексию, идентификацию, каузальную атрибуцию.**

Эффекты восприятия

Социальному восприятию присущи некоторые особенные проявления неточности восприятия, называемые законами, эффектами или ошибками восприятия.

- Эффекты стереотипизации:
    - **Галоэффект** (эффект ореола, эффект нимба или рога) — общее благоприятное или неблагоприятное мнение о человеке переносится на его неизвестные черты.
    - **Эффекты последовательности**:
    - **Эффект первичности** (эффект первого впечатления, эффект знакомства) — первая информация переоценивается по отношению к последующей.
    - **Эффект новизны** — новой информации о неожиданном поведении хорошо знакомого, близкого человека придаётся большее значение, чем всей информации, полученной о нём ранее.
- **Эффект роли** — поведение, определяемое ролевыми функциями, принимается за личностную особенность.
- **Эффект присутствия** — чем лучше человек чем-то владеет, тем лучше он делает это на глазах у окружающих, чем в одиночестве.
- **Эффект авансирования** — к разочарованию приводит отсутствие приписываемых ранее несуществующих достоинств.
- **Эффект снисходительности** — руководитель гипертрофирует позитивные черты подчинённых и недооценивает негативные

(характерно для руководителя попустительского и, в какой-то мере, демократического стиля).

- **Эффект гипервзыскательности** — руководитель гипертрофирует негативные черты подчинённых и недооценивает позитивные (характерно для руководителя авторитарного стиля).
- **Эффект физиогномической редукции** — вывод о присутствии психологической характеристики делается на основе черт внешности.
- **Эффект красоты** — внешне более привлекательному человеку приписывается больше положительных черт.
- **Эффект ожидания** — ожидая от человека определённой реакции, мы провоцируем его на неё.
- **Внутригрупповой фаворитизм** — «свои» кажутся лучше.

    - **Эффект отрицательной асимметрии начальной самооценки** — во времени есть тенденция к противоположному внутригрупповому фаворитизму.

- **Презумпция взаимности** — человек считает, что «другой» относится к нему так, как он относится к «другому».
- **Феномен предположения о сходстве** — человек считает, что «свои» относятся к остальным людям так же, как он.
- **Эффект проекции** — человек исходит из того, что другие обладают такими же качествами, как он.
- **Феномен игнорирования информационной ценности неслучившегося** — информация о том, что могло бы произойти, но не произошло, игнорируется.

Атрибуция

Атрибуция — приписывание характеристик себе или другому человеку.

Под каузальной атрибуцией понимают интерпретацию поведения партнера по общению путём выдвижения предположений о его мотивах, намерениях, эмоциях, причинах поведения, качествах личности с последующим их приписыванием партнеру. Каузальная атрибуция тем больше определяет социальную перцепцию (восприятие), чем больше дефицит информации о партнере по

общению. Результаты приписывания могут стать материалом для формирования социальных стереотипов. Стереотипизация восприятия приводит к двум различным следствиям. Во-первых, к упрощению познания другого человека (людей). Во-вторых, к формированию предубеждений по отношению к представителям различных социальных групп (профессиональных, социо-экономических, этнических и т. д.)

Впечатление

Впечатление — мнение, оценка, сложившиеся после знакомства, соприкосновения с кем-чем-нибудь.

Формирование впечатления

Формирование впечатления — процесс создания своих впечатлений о других.

Впечатления составляют:

- Образцы поведения
- Абстракции

Управление впечатлениями

Управление впечатлениями — поведение, направленное на формирование и контроль за впечатлением о себе других людей.

Тактики управления впечатлением:

- Усиление собственной позиции
- Усиление позиции собеседника

Самопрезентация — поведение, направленное на создание благоприятного или соответствующего чьим-то идеалам впечатления о себе.

Согласно данным исследования Гордона, произведённого в 1996 году, уровень успешности тактик управления впечатлением распределяется следующим образом:[7]

1. Представление собеседника в лучшем свете
2. Согласие с мнением собеседника.
3. Самопрезентация
4. Комбинация 1-3
5. Оказание услуг

Основные виды и свойства восприятия

Восприятие как непосредственное отражение мира классифицируется по разным основаниям.

Традиционно выделяют пять видов восприятия в соответствии с ведущим анализатором, участвующим в построении перцептивного образа (по модальности восприятия):

- визуальное;
- аудиальное;
- осязательное (тактильное);
- вкусовое;
- обонятельное.

Различают также виды восприятия в зависимости от объекта восприятия, например, восприятие пространства, времени, движения, скорости; произведений живописи, музыки; основных явлений социальной жизни человека (другого человека, событий общественной жизни) и т.п.

Восприятие окружающего мира, как правило, комплексно; оно представляет собой результат совместной деятельности различных органов чувств. Более того, восприятие сложных явлений предметного и социального мира осуществляется прежде всего благодаря участию процессов памяти, мышления и воображения. Иначе говоря, вести речь о процессе восприятия в «чистом виде» во многих случаях неправомерно.

В психологии существует деление видов восприятия в зависимости от участия в нем других психологических образований: эмоциональное восприятие (восприятие мира, искусства); рациональное восприятие (восприятие, подчиненное процессу мышления) и др.

Каждый из видов восприятия имеет свои специфические особенности и механизмы. Их описание представляет задачу как психологии, так и других отраслей знания: физиологии, кибернетики.

1.2.3. Память (значения)

Па́мять — многозначный термин, в общем случае обозначающий способность сохранять, накапливать и воспроизводить хранящуюся информацию, а также функцию, обеспечивающую данные возможности.

Термин различается в зависимости от области использования и может иметь следующие значения:

Биология

- Память — способность к воспроизведению прошлого опыта, одно из основных свойств нервной системы живых организмов, выражающееся в способности длительно хранить информацию и многократно вводить её в сферу сознания и поведения.
    - *Рабочая память* — разновидность памяти, которая заключается в способности к временному хранению и манипулированию информацией.
    - *Непроизвольная память* — память, которая не регулируется определённой программой и целью.
    - *Ложная память* — состояние дежавю, когда человеку кажется, что он «помнит» что-то, но чего на самом деле не было.
- Генетическая память — гипотетическая совокупность наследственных реакций передаваемых субъекту через поколения посредством генов.

История

- Память — один из видов документов в делопроизводстве Приказов.
- Историческая память — набор передаваемых из поколения в поколение исторических сообщений, мифов и легенд.
    - *Коллективная память* — совокупность действий, предпринимаемых коллективом или социумом, по символической реконструкции прошлого в настоящем.

Техника

• Компьютерная память (физическая память, машинная память) — устройство хранения информации:

- *Ассоциативная память* — вид памяти, отличающийся схемой доступа к данным и используемый в приложениях быстрого поиска.
- *Квантовая память* — средство хранения информации при квантовых вычислениях.
- *Кеш-память* — буфер, ускоряющий доступ к данным из более медленной памяти.
- *Оперативная память* — память для временного хранения рабочих данных и программ.
- *Рефлективная память* — специальная память, применяемая для связи между вычислительными устройствами.
- *Энергонезависимая память* — память, не требующая электроэнергии для хранения данных.
    - *Контактная память* — класс электронных устройств в виде металлической таблетки, широко применяемый в качестве домофонных ключей.
    - *Флеш-память* — технология энергонезависимой памяти.
- *Виртуальная память* — технология управления компьютерной памятью ЭВМ, разработанная для многозадачных операционных систем, позволяющая не зависеть от адресации непосредственно к физической памяти.

• Память переводов — база данных, содержащая набор ранее переведенных текстов.

• Эффект памяти — способность неживых предметов восстанавливать тот или иной свой параметр, например, форму или уровень заряда.

Па́мять — это общее обозначение для комплекса познавательных способностей и высших психических функций по накоплению, сохранению и воспроизведению знаний и навыков. Память в разных формах и видах присуща всем высшим животным. Наиболее развитый уровень памяти характерен для человека.

Пионером в исследовании памяти человека считается Герман Эббингауз, ставивший эксперименты на себе (основной методикой было заучивание бессмысленных списков слов или слогов).

1. Методы исследования

- Методы клинической и экспериментальной психофизиологии
- Методы физиологии поведения
- Методы гистохимии
- Методы электрофизиологии мозга и отдельных нейронов. В настоящее время для изучения функционирования мозга, задействованных его структур, в том числе, в процессах памяти, широко применяется функциональная магнитно-резонансная томография, которая позволяет определить активацию головного мозга в различных условиях в ходе нормального его функционирования. Для изучения психических процессов, в том числе связанных с памятью, на молекулярном уровне, таких, как метаболизм, транспорт веществ, лиганд-рецепторные взаимодействия, экспрессия генов и т. д. применяется позитронно-эмиссионная томография.
- Фармакологические методы
- Методы аналитической биохимии

В зависимости от задач, подлежащих решению, исследование механизмов памяти осуществляется на разных объектах — от человека до культуры нервных клеток.

2. Память в нейрофизиологии

Память — одно из свойств нервной системы, заключающееся в способности какое-то время сохранять информацию о событиях внешнего мира и реакциях организма на эти события, а также многократно воспроизводить и изменять эту информацию.

Память свойственна животным, имеющим достаточно развитую центральную нервную систему (ЦНС). Объём памяти, длительность и надёжность хранения информации, как и способность к восприятию сложных сигналов среды и выработке адекватных реакций, пропорциональны числу задействованных в этих процессах нервных клеток.

У кишечнополостных формируются лишь простые суммационные рефлексы, у большинства членистоногих и моллюсков память выражается в привыкании, то есть в торможении более или менее готовых программ поведения или отдельных реакций, неадекватных определённым условиям среды. Головоногие моллюски по способности к обучению сравнимы с птицами и млекопитающими. В онтогенезе высших животных возможности памяти как по объёму, так и по сложности запоминаемых ситуаций возрастают по мере созревания нейронов и миелинизации нервных волокон мозга.

Физиологические исследования памяти обнаруживают **3 основных этапа её формирования, которым соответствуют 3 вида памяти: сенсорная, кратковременная и долговременная. Кратковременная память** характеризуется временем устойчивого хранения информации до 20 секунд и при неповторении, предшествующая информация разрушается при превышении 30 секунд под воздействием вновь поступившей информации. **Долговременная память**, время хранения информации в которой сравнимо с продолжительностью жизни организма, устойчива к воздействиям, нарушающим кратковременную память. Переход от кратковременной памяти к долговременной, называемый **консолидацией, постепенен** и связан с активацией ряда биохимических процессов. Опыты с иссечением участков коры больших полушарий головного мозга и электрофизиологические исследования показывают, что **«запись» каждого события распределена по более или менее обширным зонам мозга. Материальным носителем информации** о разных событиях является не возбуждение разных нейронов, а **различные комплексы**

совозбуждённых нейронов (нейронных сетей). Новые реакции вырабатываются и запоминаются нервной системой в основном либо на основе создания новых синаптических связей между имеющимися нейронами, либо на основе изменения эффективности уже имеющихся синаптических связей. **Под запоминанием (долговременным) подразумевается изменение способности одних нейронов возбуждаться при возбуждении других нейронов**. Долговременные изменения эффективности синапсов могут быть обусловлены изменениями в биосинтезе белков, от которых зависит чувствительность синаптической мембраны к медиатору (**долговременная потенциация**). Установлено, что биосинтез белков активируется при возбуждении нейронов на разных уровнях организации ЦНС, а **блокада синтеза нуклеиновых кислот или белков затрудняет или исключает формирование долговременной памяти.** Очевидно, что одна из функций активации синтеза при возбуждении — **структурная фиксация нейронных сетей, что и лежит в основе долговременной памяти**. Установление ассоциаций между нейронами (то есть путей распространения возбуждения) может происходить как за счёт увеличения проводимости имеющихся синапсов, так и в результате возникновения дополнительных синапсов. Оба возможных механизма нуждаются в интенсификации белкового синтеза. Первый — сводится к частично изученным явлениям клеточной адаптации и хорошо согласуется с представлением об универсальности основных биохимических систем клетки. Второй — требует направленного роста отростков нейронов.

Известно, что для работы памяти и способности мозга к обучению необходимы белки DC0, Leo и CaMKII. При этом отвечающие за их экспрессию гены у крыс и мух ничем не отличаются. Те же белки работают и в организме человека: проблемы с их экспрессией играют важную роль в болезни Альцгеймера, синдромах Дауна и Энгельмана.

По современным представлениям, память является неотъемлемой частью таких процессов, как

- обучение;[1]
- прогнозирование будущего и воображение несуществующего (по-видимому, оба процесса являются процедурами «нарезания и перетасовки фрагментов воспоминаний»);[9][11]
- сознание и самоидентификация индивидуума.[9][12][13][14]

3. Память и обучение

Привыкание можно рассматривать как суммацию тормозных стимулов.

Память и обучение являются сторонами одного процесса. Под обучением подразумевают обычно механизмы приобретения и фиксации информации, а под памятью — механизмы хранения и извлечения этой информации.

Процессы обучения можно разделить на **неассоциативные и ассоциативные. Неассоциативное** обучение рассматривается как эволюционно более древнее и не подразумевающее связи между тем, что запоминается и какими-либо ещё стимулами. **Ассоциативное** основывается на формировании связи между несколькими стимулами. Например, классический вариант выработки условного рефлекса по Павлову: установление связи между нейтральным условным стимулом и безусловным стимулом, вызывающим безусловный рефлекторный ответ.

Безусловные рефлексы в эту классификацию не входят, так как осуществляются на основе унаследованных паттернов связей между нервными клетками.

Неассоциативное обучение делится на суммацию, привыкание, долговременную потенциацию и импринтинг.

Суммация

Вариант с участием блокируемых магнием глутамат-эргических ионных (Ca, Na) каналов (каналы NMDA-типа), способных вернуться в неактивное состояние только через часы.

Суммация — постепенное увеличение реакции на повторяющиеся предъявления ранее индифферентного стимула. Результатом суммации является обеспечение реакции организма на слабые, но длительно действующие стимулы, которые потенциально могут иметь какие-то последствия для жизнедеятельности индивида.

В обычной ситуации реакция развивается так: сильный стимул вызывает в чувствительном нейроне целую пачку из потенциалов действия, что приводит к большому выбросу медиатора из синаптического окончания аксона чувствительного нейрона на двигательном нейроне, и этого оказывается достаточно для возникновения надпорогового постсинаптического потенциала и запуска в мотонейроне потенциала действия.

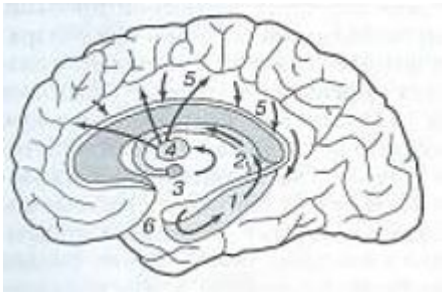
Иная ситуация наблюдается при развитии суммации.

Один сценарий развития суммации заключается в ритмичном использовании серии слабых стимулов, каждый из которых недостаточен для выброса медиатора в синаптическую щель. При этом если частота стимуляции достаточно велика, то в пресинаптическом окончании накапливаются ионы кальция, так как ионные насосы не успевают откачивать их в межклеточную среду. В итоге очередной потенциал действия может вызвать выброс медиатора, которого хватит на то, чтобы возбудить постсинаптический мотонейрон. Если при этом ритмичную стимуляцию ранее подпороговыми стимулами не прерывать, то приходящие потенциалы действия будут продолжать запускать рефлекс, так как высокое содержание Ca^{2+} в окончании чувствительного нейрона сохраняется. Если же сделать паузу в стимуляции, то Ca^{2+} будет удален и для запуска рефлекса слабыми стимулами опять потребуется предварительная суммация.

Другой сценарий развития суммации наблюдается при однократном, но сильном раздражении, в результате чего к пресинаптическому окончанию на двигательном нейроне приходит высокочувствительная серия импульсов, приводящая к поступлению в окончание большого количества ионов Ca2+, которого хватает на возбуждение следующего в цепи нейрона ранее подпороговым стимулом. Продолжительность такого эффекта может составлять секунды.

Способность к суммации, по-видимому, лежит в основе кратковременной нейрологической памяти. Получая какую-либо информацию через систему анализаторов (приглядываясь, прислушиваясь, принюхиваясь, осторожно пробуя новую для нас пищевую приправу), мы обеспечиваем ритмическую стимуляцию синапсов, через которые проходит сенсорный сигнал. Эти синапсы в течение нескольких минут сохраняют повышенную возбудимость, облегчая проведение импульсов, и, таким образом, сохраняет след о

переданной информации. Однако суммация, будучи эволюционно ранним механизмом обучения, быстро исчезает и не может противостоять любым сильным внешним воздействиям на организм.



1 — гиппокамп; 2 — свод; 3 — мамиллярное тело; 4 — передние ядра таламуса; 5 — поясная извилина; 6 — зубчатая извилина

Привыкание

При многократном раздражении средней силы реакция на него ослабляется или вообще исчезает. Это явление называют «привыкание» (или «габитуация»).

Причины привыкания разнообразны и первым из них является адаптация рецепторов. Вторая причина — накопление $Ca2+$ в пресинаптических окончаниях на тормозных нейронах. При этом повторные сигналы, исходно незначимые для тормозных нейронов, постепенно суммируются, а затем запускают тормозные нейроны, активность которых блокирует прохождение сигналов по рефлекторной дуге. Привыкание можно рассматривать как суммацию тормозных сигналов. Нужно подчеркнуть, что суммация и привыкание, как и другие формы синаптической пластичности, являются просто следствием структуры синапсов и организации нейронов.

Долговременная потенциация

Долговременная потенциация возникает в том случае, когда животному предъявляют некий стимул, который оно различает, но который при этом слишком слаб для того, чтобы вызвать ответную реакцию. После длительной паузы (1 — 2 ч) животному предъявляют сильный стимул, который вызывает исследуемую реакцию.

Следующую стимуляцию проводят ещё через 1 — 2 ч с помощью слабого сигнала, ранее не приводившего к срабатыванию рефлекса. У животных, у которых нервная система способна к долговременной потенциации, возникает рефлекторный ответ. В дальнейшем интервал между сильной и слабой стимуляцией может быть увеличен до 5 и даже 10 ч, а возбудимость нервной системы все время будет оставаться повышенной.

Долговременная потенциация может рассматриваться как вариант «длительной» кратковременной памяти, распространяющейся на дневной период бодрствования человека — с утра до вечера[7].

Импринтинг

Вариант с участием глутамин-эргических рецепторов сопряжённых, например, с аденилат-циклазой, с последующим усилением экспрессии генов рецепторов к глутамину.

Это явление определяют как устойчивую индивидуальную избирательность по отношению к внешним стимулам в определенные периоды онтогенеза. **Наиболее известны следующие варианты импринтинга: запоминание родителя детенышем; запоминание детёныша родителем; импринтинг будущего полового партнёра.**

В отличие от условного рефлекса, эта связь, во-первых, образуется только в строго определенный период жизни животного; во-вторых, образуется без подкрепления; в-третьих, в дальнейшем оказывается очень стабильной, практически не подлежит угасанию и может сохраняться в течение всей жизни особи. Было показано, что импринтинг сопровождается активацией нейронов промежуточной области медиовентрального гиперстриатума. Повреждение этой области нарушало у цыплят и импринтинг, и другие виды памяти.

В процессе запоминания/обучения по типу импринтинга устанавливаются контакты групп нейронов одного ядра со строго определенными группами другого ядра. По мере обучения могут либо увеличиваться размеры нейронов, их количество в пределах соответствующих структур, число шипиков и синаптических контактов — либо число нейронов, синаптических связей и NMDA-рецепторов в синапсах может даже уменьшаться, но сродство оставшихся рецепторов к специфическому медиатору будет возрастать.

Можно предложить следующую модель развития импринтинга.

Выделяющаяся из окончания нейрона глутаминовая кислота действует на метаботропные рецепторы на поверхности постсинаптического нейрона и запускает выработку вторичного (внутриклеточного) посредника (например, цАМФ). Вторичный посредник через каскад регуляторных реакций усиливает синтез белков, формирующих новые синапсы к глутамату, которые встраиваются в мембрану нейрона таким образом, чтобы улавливать сигналы от самого активного пресинаптического окончания, передающего информацию о характеристике объекта импринтинга. Встраивание в мембрану новых рецепторов увеличивает эффективность синаптической передачи, и сумма вызванных постсинаптических потенциалов от приходящих сигналов достигает порогового уровня. Затем возникнут ПД и поведенческая реакция будет запущена.

Следует подчеркнуть, что нейрохимические и синаптические изменения протекают не мгновенно, а требуют времени. Для успешного импринтинга важно наличие стабильного сенсорного «давления» на обучающийся нейрон, например, постоянное присутствие матери. Если это условие не выполняется, то импринтинг вообще не возникает.

Обученные нейроны способны поддерживать концентрацию рецепторов на постсинаптической мембране «запечатленного» синапса на постоянном высоком уровне, что обеспечивает стабильность импринтинга, позволяющую рассматривать его как специфический вариант долговременной памяти. [7]

Ассоциативное обучение

Отличие от импринтинга в том, что необходима одновременная стимуляция дополнительным нейромедиатором.

Ассоциативное обучение основывается на образовании связи (ассоциации) между двумя стимулами. В качестве примера можно рассмотреть формирование условного рефлекса, когда на один нейрон одновременно подаётся сигнал и от некоторого незначительного стимула, и от центра положительного подкрепления из гипоталамуса. При этом вероятно, что на разных постсинаптических участках

генерируются различные вторичные посредники, и изменение экспрессии генов рецепторов к нейромедиаторам, действующим на данный нейрон будет обусловлено суммарным эффектом этих вторичных посредников[7].

Предположительно, процессы консолидации памяти начинаются с усиления глутаматной передачи, за счёт глутаматных рецепторов NMDA-типа. Такие рецепторы способны связать глутамат только после некоторой предварительной деполяризации мембраны, вызванной поступлением в постсинаптическую клетку ионов натрия в результате работы каналов, связанных с другим типом глутаматных рецепторов. Связав глутамат, NMDA-рецепторы инактивируются только по прошествии продолжительного времени (часы). В активном состоянии они связываются с каналами для ионов кальция. Повышение концентрации кальция приводит к активации ряда киназ, запускающих каскад дальнейших реакций. В частности, активированная Ca2+ протеинкиназа А переходит в ядро, регулируя там экспрессию целого ряда генов, что в конечном счёте приводит к формированию новых синапсов между взаимодействующими в процессе ассоциативного обучения нейронами. Помимо этого активация киназ приводит к изменению активности других ионных каналов, дополнительно увеличивая проницаемость постсинаптической мембраны обучающегося нейрона к ионам кальция и уменьшая — к ионам калия. Кроме того, в синапсах наблюдается агрегация белковых молекул в слоистые структуры, формирующие транссинаптические каналы (волокна), что резко облегчает прохождение медиатора и резко повышает проводимость синапса.

Память и сон

Воздействие 36-часовой депривации сна на кодирование декларативной памяти человека. Испытуемых разделили на 2 группы: контрольную и тех, кого лишали сна. Людям предлагали выучить слова, различной эмоциональной окраски. По прошествии 36 часов обе группы тестировали на долю запомненных слов. Тёмные столбики — контроль, светлые — группа, лишённая сна. Вариант А — весь запомненный материал, а вариант Б — эффекты, возникающие если запоминаемый материал разделён на эмоционально-позитивный (I), эмоционально-негативный (II) и эмоционально-нейтральный (III)

Работы по исследованию депривации (лишения) сна на процессы памяти показывают, что лишённые сна люди воспроизводят в разы меньше материала по сравнению с людьми, которых сна не лишали. При 36-часовой депривации наблюдается ухудшение способности воспроизводить материал на 40 %. Интересная закономерность обнаруживается, если проанализировать отдельно влияние сна на способность воспроизводить материал разной эмоциональной окраски. Во-первых, результаты указывают на то, что эмоционально-окрашенный материал запоминается лучше эмоционально-нейтрального, независимо от количества сна. Это согласуется с положением, что консолидация памяти происходит при значительном участии систем подкрепления, формирующих эмоции. Кроме того, оказывается, что хотя ухудшение запоминания при депривации сна наблюдается во всех случаях, интенсивность этого влияния существенно зависит от эмоциональной окраски материала. Сильнее всего затрудняется воспроизведение эмоционально-нейтрального и особенно — эмоционально-позитивного материала. В то время как изменения в воспроизведении эмоционально-негативного материала мало и статистически недостоверно.

Исследования роли дневного сна на формирование процедурной памяти показывают, что при инструментальном обучении люди демонстрируют улучшение навыков только после сна — продолжительностью хотя бы в несколько часов, независимо от того, поспали они днём или ночью.

Однозначного ответа на вопрос о всех механизмах связи процессов сна и памяти нет, как нет и ответа на вопрос о возможных компенсаторных механизмах, развивающихся после некоторых воздействий на структуры мозга, обычно задействованные в процессах сна и памяти. Некоторые исследователи критикуют положения о связи механизмов сна с механизмами памяти, утверждая либо что сон вообще играет лишь пассивную (хотя и позитивную) роль в запоминании, уменьшая отрицательную интерференцию следов памяти, либо что быстрый сон не задействован в процессах памяти. В пользу последней позиции приводят следующие группы аргументов:

- Поведенческие: все опыты по изучению депривации быстрого сна «методом островков» (экспериментальное животное помещается в условия, где при потере позы — что неизбежно в стадии быстрого сна — оно падает в воду и пробуждается)

нельзя считать убедительными, из-за неадекватности методики.

- Фармакологические: все три основных класса антидепрессантов (ингибиторы МАО, «трициклики» и ингибиторы обратного захвата серотонина) полностью или почти полностью подавляют быстрый сон, но не вызывают нарушения обучаемости и памяти ни у больных, ни у подопытных животных.
- Клинические: имеется несколько сообщений о больных с билатеральными разрушениями в области моста — у таких больных полностью и, по-видимому, навсегда исчезал быстрый сон, но никаких жалоб на нарушения обучаемости и памяти от таких больных не поступало.

Память и стресс

Показано, что стресс влияет на работу гиппокампа, являющегося ключевой структурой, задействованной в консолидации памяти. Кратковременный всплеск АКТГ и кортизола способствуют консолидации воспоминаний. Более сильный выброс АКТГ блокирует консолидацию воспоминаний. Длительно повышенный уровень кортизола, видимо, способствует деградации ткани гиппокампа (до 8 % у ветеранов войны во Вьетнаме, страдавших от посттравматического стрессового расстройства и до 12 % у детей, страдавших от жестокого обращения).

Память и физическая активность

Ученые из университета в Калифорнии (США) доказали связь между физическими упражнениями и памятью. Регулярная нагрузка способствует повышению уровня глутаминовой и гамма-аминомасляной кислот в головном мозге, которые необходимы для многих процессов умственной деятельности и настроения. Выполнение упражнений в течение 20 минут достаточно для того, чтобы концентрация этих соединений повысилась, а процессы запоминания улучшились[20][*неавторитетный источник? 39 дней*].

4. Процессы памяти

- **Запоминание** — это процесс памяти, посредством которого происходит запечатление следов, ввод новых элементов ощущений, восприятия, мышления или переживания в систему ассоциативных связей. Запоминание может быть произвольным и непроизвольным, основу произвольного запоминания составляет установление смысловых связей — результат работы мышления над содержанием запоминаемого материала.
- **Хранение** — процесс накопления материала в структуре памяти, включающий его переработку и усвоение. Сохранение опыта дает возможность для обучения человека, развития его перцептивных (внутренних оценок, восприятия мира) процессов, мышления и речи.
- **Воспроизведение и узнавание** — процесс актуализации элементов прошлого опыта (образов, мыслей, чувств, движений). Простой формой воспроизведения является узнавание — опознание воспринимаемого объекта или явления как уже известного по прошлому опыту, установлением сходств между объектом и образом его в памяти. Воспроизведение бывает произвольным и непроизвольным. При непроизвольном образ всплывает в сознании без усилий человека.

Если в процессе воспроизведения возникают затруднения, то идёт процесс припоминания. Отбор элементов, нужных с точки зрения требуемой задачи. Воспроизведенная информация не является точной копией того, что запечатлено в памяти. Информация всегда преобразовывается, перестраивается.

- Забывание — потеря возможности воспроизведения, а иногда даже узнавания ранее запомненного. Наиболее часто забывается то, что незначимо. Забывание может быть частичным (воспроизведение не полностью или с ошибкой) и полным (невозможность воспроизведения и узнавания). Выделяют временное и длительное забывание.

5. Теоретические модели памяти в психологии

- **Модель Дональда Нормана (Donald Norman) и Нэнси Во (Nancy Waugh)** выделяет две структуры памяти: *первичная память* хранящая временную информацию, которую человек использует в данный момент и *вторичная память* сохраняющая информацию на длительное время.
- **Модель Ричарда Аткинсона (Richard Atkinson) и Ричарда Шиффрина (Richard Shiffrin)** выделяет три структуры памяти: *сенсорное хранилище* (sensory store) или сенсорная память, содержащее информацию, поступающую из сенсорной системы, сохраняемую на небольшой период; *кратковременное хранилище*, сохраняющее ограниченный объём информации на более продолжительный срок, чем сенсорная память, в нём протекают процессы, регулирующие обмен информацией с долговременной памятью; *долговременная память*, сохраняющая значительный объём информации на продолжительный период или постоянно. Аткинсон и Шиффрин рассматривали эти хранилища не в качестве определенных психологических структур, но как гипотетическую, ментальную модель помогающую понять функционирование памяти.
- **Модель уровней переработки Фергуса Крейка (Fergus I. M. Craik) и Роберта Локхарта (Robert S. Lockhart)** разработанная в 1972 г. Память является функцией переработки стимульной информации. Память не включает фиксированное количество хранилищ. То, на каком уровне памяти сохраняется информация зависит от процессов переработки. Чем более глубокий уровень переработки информации, тем более долгосрочный характер будет иметь хранение в памяти этой информации. П. И. Зинченко экспериментально показал как уровень переработки влияет на запоминание.
- **Модель рабочей памяти Алана Бэддли.** Рабочая память является частью долговременной памяти и включает в себя кратковременную память. Рабочая память содержит только ту информацию из долговременной памяти, которая находится в активной обработке. **В рабочей памяти находятся зрительно-пространственный набросок, фонологическая петля, центральный управляющий элемент (central executive) координирующий когнитивные процессы**

(связывающий информацию поступающую из разнообразных источников и управляющий вниманием), эпизодический буфер (episodic buffer), другие подсистемы.

Сенсорные процессы формирующие зрительно-пространственный набросок, а также фонологическую петлю в модели памяти Бэддли рассматриваются в рамках модели уровней переработки Фергуса Крейка и Роберта Локхарта как процессы переработки.

- **В концепции Карла Густава Юнга** память понимается, как функция контролируемая волей и находящаяся под контролем так называемого «эго-комплекса». «То, что мы называем памятью, — это дар репродуцировать бессознательные содержания, и это — главная функция». Источником воспоминаний может выступать не только личностный объём бессознательного, но и его коллективный слой (включая архаичную часть), что объясняет припоминание в необычных случаях знаний и событий, не принадлежащих личному сознательному опыту индивида. Забывание означает выпадение содержаний из доступной сознанию области, погружение их в бессознательное. По Юнгу психические содержания, образы, содержащиеся в сознании погружаются в бессознательное, забываются, если они теряют энергию до уровня ниже порога сознания, и появляются в сознании, вспоминаются, если их энергетический уровень становится выше этого порога.[26]Из личного бессознательного могут извлекаться (вспоминаться) "так называемые подпороговые (сублиминальные) восприятия, которые недостаточно сильны для того, чтобы достичь сознание" при первоначальном восприятии и попадающие поэтому в предсознание, если впоследствии, по каким-то причинам их энергетический уровень превысит порог сознания.

6. Классификация видов памяти

Существуют различные типологии памяти:

- **по сенсорной модальности** — зрительная (визуальная) память, моторная (кинестетическая) память, звуковая

(аудиальная) память, вкусовая память, обонятельная память, болевая память, эйдетическая память;

- **по содержанию** — образная память, моторная память, эмоциональная память, социальная память, пространственная память;
- **по организации запоминания** — декларативная память (эпизодическая память, семантическая память и автобиографическая память), процедурная память;
- **по времени хранения** — ультракратковременная память, кратковременная память, долговременная память;
- **по физиологическим принципам** — определяемая структурой связей нервных клеток (она же долговременная) и определяемая текущим потоком электрической активности нервных путей (она же кратковременная);
- **по наличию цели** — произвольная и непроизвольная;
- **по наличию средств** — опосредованная и неопосредованная;
- **по уровню развития** — моторная, эмоциональная, образная, словесно-логическая.

Критерий	**Вид**
Содержание	• образная память • словесно-логическая память • сенсорная память • эйдетическая память • эмоциональная память • социальная память • пространственная память
Время	• кратковременная память • долговременная память • рабочая память • промежуточная память
Организация запоминания	• эпизодическая память • семантическая память • процедурная память

На стыке между эпизодической и семантической памятью выделяют автобиографическую память, включающую в себя черты обоих.

Можно построить иную классификацию по содержанию памяти:

Процедурная (память на действия) и **декларативная** (память на названия). В рамках последней выделяют **эпизодическую** (память на события и явления индивидуальной жизни человека) и **семантическую** (знание вещей, не зависящих от индивидуальной жизни человека).

Сенсорная память

Сенсорная память сохраняет информацию, формируемую сенсорной системой, возникающую при воздействии стимулов на органы чувств. В сенсорной памяти сохраняется сенсорная информация после прекращения воздействия стимула. По времени хранения сенсорная память является ультракратковременной. Временные характеристики функционирования наиболее изучены для сенсорной зрительной (иконической) памяти.

Иконическая память

Иконической памятью называется сенсорная память, в которую поступает и сохраняется стимульная визуальная информация, отфильтрованная и агрегированная зрительной системой.
Особенностью иконической памяти является фиксация информации в целостной, портретной форме.

С исследованием иконической сенсорной памяти, её объёма связаны эксперименты Джорджа Сперлинга. В экспериментах Сперлинг использовал как процедуру «общего отчета» (Whole Report Procedure), так и собственную разработку — процедуру «частичного отчета» (Partial Report Procedure). В силу скоротечности иконической памяти процедура общего отчета не позволяла объективно оценить объём регистрируемой в сенсорной памяти информации, поскольку в ходе самого процесса отчета происходило «забывание» портретной информации, стирание её из сенсорной иконической памяти. Процедура частичного отчета показала, что в иконической памяти осуществляется регистрация 75 % зрительного поля. Эксперименты Сперлинга показали, что образ сигнала в иконическую

память заносится за время не больше 50 миллисекунд, затухает по экспоненте с постоянной времени, равной примерно 150 миллисекунд и по истечении 0,5 с. от образа мало что остается. Также было выяснено, что процессы, связанные с иконической памятью, не контролируются ментально. Даже тогда, когда испытуемые не могли наблюдать символы, они по-прежнему сообщали, что продолжают их видеть. Таким образом, субъект процесса запоминания не различает содержание иконической памяти и объекты, которые находятся в окружающей среде.

Стирание информации, находящейся в иконической памяти, другой информацией, поступающей от ощущений, позволяет зрительному ощущению быть более восприимчивым. Такое свойство иконической памяти — стирание — обеспечивает запоминание информации в иконической памяти, учитывая её ограниченный объём, даже в том случае если скорость поступления сенсорной информации превышает скорость затухания сенсорной информации в иконической памяти. Исследования показали, что если зрительная информация поступает достаточно быстро (до 100 миллисекунд), то происходит наложение новой информации на прежнюю, которая ещё находится в памяти, не успев затухнуть в ней и перейти на другой уровень памяти — более долговременный. Эта особенность иконической памяти называется ***эффектом обратной маскировки***. Так, если показать букву, а затем в течение 100 миллисекунд на той же позиции зрительного поля — кольцо, то испытуемый будет воспринимать букву в кольце.

Эхоическая память

Эхоическая память сохраняет стимульную **звуковую информацию**, поступающую через слуховую сенсорную систему из органов слуха.

Тактильная память

Тактильная память регистрирует **стимульную информацию**, поступающую через соматосенсорную систему.

Вкусовая сенсорная память

Вкусовая сенсорная память регистрирует **стимульную информацию вкусовых раздражителей**, поступающую через вкусовую сенсорную систему.

Обонятельная сенсорная память

Обонятельная сенсорная память **регистрирует запаховую стимульную информацию, поступающую через обонятельную сенсорную систему**.

Эйдетическая память

Эйдетическая память - долговременная память на сенсорную информацию, преимущественно на зрительные впечатления, позволяющая удерживать и воспроизводить в деталях образ воспринятого ранее предмета или явления в их отсутствии. В эйдетический образ могут и зачастую входят также и другие сенсорные модальности (слуховые, тактильные, двигательные, вкусовые, обонятельные).

Социальная память

Социальная память это память социальных связей: знание отличительных черт индивида, его характера, а также социального положения. Максимальное количество постоянных социальных связей, которые человек в состоянии комфортно поддерживать, и, соответственно, размер социальной памяти, называется числом Данбара. Это число находится в диапазоне от 100 до 230, чаще всего считается равным 150. По Р. Данбару размер социальной памяти линейно связан с размером неокортекса. Социальная память является долговременной памятью.

Пространственная память

Люди обладают мощной пространственной памятью. В основном они хорошо помнят пространство своих жилищ, родственников, друзей, окрестности, где они живут, места отдыха, игр, прогулок, путешествий и т.д., с самого раннего детства. Хорошо запоминают

пространственные характеристики мест в которых бывают и дорог, по которым перемещаются. Эта память имеет долговременной характер. Мощь пространственной памяти лежит в основе метода запоминания Цицерона, базирующегося на пространственном воображении.

У беспозвоночных за пространственную память отвечают грибовидные тела.

У человека функции хранения и обработки пространственной информации осуществляет гиппокамп.

Долговременная и кратковременная память

Нейробиологические исследования обнаруживают 2 основных вида памяти: кратковременная и долговременная. Одно из важнейших открытий Эббингауза состояло в том, что если список не очень велик (обычно 7), то его удаётся запомнить после первого прочтения (обычно список элементов, которые можно запомнить сразу, называют объёмом кратковременной памяти).

Другой закон, установленный Эббингаузом, — количество сохраняющегося материала зависит от промежутка времени с момента заучивания до проверки (так называемая «кривая Эббингауза»). Был открыт позиционный эффект (возникающий, если запоминаемая информация по объёму превышает кратковременную память). Он заключается в том, что лёгкость запоминания данного элемента зависит от места, которое он занимает в ряду (легче запоминаются первые и последние элементы).

В теории памяти Д. О. Хебба считается, что кратковременная память основана на электрофизиологических механизмах, поддерживающих возбуждение связанных нейронных систем, а долговременная память фиксируется структурными изменениями в отдельных клетках, входящими в состав нейронных систем, и связана с химической трансформацией, образованием новых веществ.

Кратковременная память

Кратковременная память существует за счет временных схем нейронных связей, исходящих из областей фронтальной (особенно

дорсолатеральной, префронтальной) и теменной коры. Сюда попадает информация из сенсорной памяти. В кратковременной памяти информация хранится около 20 сек., после 30 сек. след информации становится настолько хрупким, что даже минимальная интерференция разрушает его. Повторение сохраняет содержимое кратковременной памяти. Её ёмкость весьма ограничена. Джордж Миллер во время своей работы в Bell Laboratories провел опыты, показывающие, что ёмкость кратковременной памяти составляет 7±2 объекта (название его работы гласит «Волшебное число 7±2»). Современные оценки ёмкости кратковременной памяти несколько ниже, обычно 4-5 объектов, причем известно, что ёмкость кратковременной памяти увеличивается за счёт процесса, называемого «Chunking» (группировка объектов). Например, если предъявить строку

ФСБКМСМЧСЕГЭ

человек будет способен запомнить только несколько букв. Однако, если та же информация будет представлена иным образом:

ФСБ КМС МЧС ЕГЭ

человек сможет запомнить гораздо больше букв потому, что он способен группировать (объединять в цепочки) информацию о смысловых группах букв (в английском оригинале: FBIPHDTWAIBM и FBI PHD TWA IBM). Также Герберт Саймон показал, что идеальный размер для последовательностей букв и цифр, неважно осмысленных или нет, составляет три единицы. Возможно, в некоторых странах это отражается в тенденции представлять телефонный номер как несколько групп по 3 цифры и конечной группы из 4-х цифр, разделенных на 2 группы по две.

Существуют гипотезы о том, что кратковременная память опирается преимущественно на акустический (вербальный) код для хранения информации и в меньшей степени на зрительный код. В своём исследовании (1964) Конрад показал, что испытуемым труднее вспоминать наборы слов, которые акустически подобны.

Современные исследования коммуникации муравьёв доказали, что муравьи способны запоминать и передавать информацию объёмом до 7 бит. Более того, показано влияние возможной группировки объектов на

длину сообщения и эффективность передачи. В этом смысле закон «Волшебное число 7±2» выполнен и для муравьёв.

Долговременная память

Хранение в сенсорной и кратковременной памяти обычно имеет жестко ограниченную ёмкость и длительность, то есть информация остается доступной некоторое время, но не неопределенно долго. Напротив, долговременная память может хранить гораздо большее количество информации потенциально бесконечное время (на протяжении всей жизни). Например, некоторый 7-значный телефонный номер может быть запомнен в кратковременной памяти и забыт через несколько секунд. С другой стороны, человек может помнить, за счет повторения, телефонный номер долгие годы. В долговременной памяти информация кодируется семантически: Бэддли (Baddeley, 1960) показал, что после 20-минутной паузы испытуемые имели значительные затруднения во вспоминании списка слов с похожим значением (например: большой, огромный, крупный, массивный) (см. также феномен «На-кончике-языка»).

Долговременная память поддерживается более стабильными и неизменными изменениями в нейронных связях, широко распределенных по всему мозгу. Гиппокамп важен при консолидации информации из кратковременной в долговременную память, хотя, по-видимому, собственно в нём информация не хранится. Скорее гиппокамп вовлечен в изменение нейронных связей в период после 3 месяцев от начального обучения.

Одной из первичных функций сна является консолидация информации. Возможно показать, что память зависит от достаточной длительности сна между упражнением и проверкой. Причём гиппокамп воспроизводит активность текущего дня во время сна.

Нарушения памяти

Большое количество знаний об устройстве и работе памяти, которое сейчас имеется, было получено при изучении явлений её нарушения. Нарушения памяти — амнезии — могут быть вызваны различными причинами. В 1887 русский психиатр Корсаков С. С. в своей публикации «Об алкогольном параличе» впервые описал картину грубых расстройств памяти, возникающих при сильном алкогольном

отравлении. Открытие под названием «корсаковский синдром» прочно вошло в научную литературу. В настоящее время все нарушения памяти делятся на:

- **Гипомнезии** — ослабление памяти. Ослабление памяти может возникнуть с возрастом или/и как следствие какого-либо мозгового заболевания (склероза мозговых сосудов, эпилепсии и т. д.).
- **Гипермнезии** — аномальное обострение памяти по сравнению с нормальными показателями, наблюдается гораздо реже. Люди, отличающиеся этой особенностью, забывают события с большим трудом (Соломон Шерешевский).
- **Парамнезии**, которые подразумевают ложные или искажённые воспоминания, а также смещение настоящего и прошлого, реального и воображаемого.

Особо выделяется детская амнезия — потеря памяти на события раннего детства. По-видимому, этот вид амнезии связан с незрелостью гиппокампальных связей, либо с использованием других методов кодирования «ключей» к памяти в этом возрасте. Впрочем, есть данные, что воспоминания первых лет жизни (и даже внутриутробного существования) могут быть частично актуализированы в изменённых состояниях сознания.

6. Описание памяти в мнемотехнике

Свойства памяти

- Точность
- Объём
- Скорость процессов запоминания
- Скорость процессов воспроизведения
- Скорость процессов забывания

Закономерности памяти, выявляемые в мнемотехнике

Память имеет объём, ограниченный количеством стабильных процессов, являющихся опорными при создании ассоциаций (связей, отношений)

Успешность припоминания зависит от способности переключать внимание на опорные процессы, восстанавливать их. Основной приём: достаточное количество и частота повторений.

Имеет место такая закономерность, как кривая забывания.

Мнемотехнические «законы» памяти

Закон памяти	Практические приёмы реализации
Закон интереса	Интересное запоминается легче.
Закон осмысления	Чем глубже осознать запоминаемую информацию, тем лучше она запомнится.
Закон установки	Если человек сам себе дал установку запомнить информацию, то запоминание произойдёт легче.
Закон действия	Информация, участвующая в деятельности (т. е. если происходит применение знаний на практике) запоминается лучше.
Закон контекста	При ассоциативном связывании информации с уже знакомыми понятиями новое усваивается лучше.
Закон торможения	При изучении похожих понятий наблюдается эффект "перекрытия" старой информации новой.
Закон оптимальной длины ряда	Длина запоминаемого ряда для лучшего запоминания не должна намного превышать объём кратковременной памяти.
Закон края	Лучше всего запоминается информация, представленная в начале и в конце.
Закон повторения	Лучше всего запоминается информация, которую повторили несколько раз (см. кривая забывания).
Закон незавершённости (Эффект Зейгарник)	Лучше всего запоминаются незавершённые действия, задачи, недосказанные фразы и т. д.

Мнемонические приёмы запоминания

- Образование смысловых фраз из начальных букв запоминаемой информации.

- Рифмизация.
- Запоминание длинных терминов или иностранных слов с помощью созвучных.
- Нахождение ярких необычных ассоциаций (картинки, фразы), которые соединяются с запоминаемой информацией.
- Метод Цицерона на основе пространственного воображения.
- Метод Айвазовского основан на тренировке зрительной памяти.
- Методы запоминания цифр:
  - закономерности;
  - знакомые числа.
- Способ связующих звеньев применим для запоминания имен и фамилий, названий книг, то есть для любых рядов слов.
- Способ образования структурных связей помогает запечатлеть информацию, для которой трудно образовать смысловые или ассоциативные связи[44].

7. Мифология, религия, философия о памяти

- В древнегреческой мифологии имеется миф о реке Лета. Лета обозначает «забвение» и является неотъемлемой частью царства мёртвых. Умершие есть те, кто потеряли память. И напротив, некоторые, удостоенные предпочтения, — среди них Тиресий или Амфиарай, — сохранили свою память и после кончины.
- Противоположностью реки Лета является Богиня Мнемозина, персонифицированная Память, сестра Кроноса и Океаноса — мать всех муз. Она обладает Всеведением: согласно Гесиоду (Теогония, 32 38), она знает «всё, что было, всё, что есть, и всё, что будет». Когда поэтом овладевают музы, он пьет из источника знания Мнемозины, это значит, прежде всего, что он прикасается к познанию «истоков», «начал».
- Согласно философии Платона Анамнесис — припоминание, воспоминание — понятие, описывающее основную процедуру процесса познания.

См. также

- Мнемотехника
- Механизм создания памяти

- Непроизвольная память
- Перенос памяти
- Компьютерная память
- Spaced learning
- Интервальные повторения

- Феномен «На-кончике-языка»

1.2.4. Представление

Представление (философия)

Представле́ние (*repraesentatio,* нем. *Vorstellung*) — чувственный образ предметов, данный сознанию, сопровождающийся, в отличие от восприятия, чувством отсутствия того, что представляется. **Различают представления памяти и воображения**. Представлением также называется соответствующий психический процесс.

В более широком значении словом *представление* означается всякое воспроизведённое памятью состояние сознания: например, исчезнувшее чувство может быть воспроизводимо памятью в качестве *представления*. Таким образом ***представление*** **обозначает вторичное, воспроизведённое состояние сознания следовательно от первичного (ощущения, чувства и т. д.).**

Представление есть термин психологический и логический.

В логике *представление* отличают от понятия; первоначальное *представление* образуется совершенно непроизвольно, будучи простым следом первичных душевных состояний; в так называемом общем *представлении* заметна уже работа мысли, ибо оно соответствует целому ряду сходных предметов. **Логика различает несколько видов *представления* (общие, отвлечённые и т. д.) и отличает их от понятия, причём общими *представлениями* называет группу изменчивых воспоминаний о сходных предметах; понятия, напротив того, характеризуются определённостью и постоянством.**

Психологический взгляд на *представление* излагается в учении об ассоциациях и в учении о памяти вообще.

Кант в «Критике способности суждения» определяет прекрасное через чувство удовольствия от представления. При этом представление необходимо находится в воображении (*Einbildungskraft*) и может быть тождественно идее.

Особенное значение термин *представление* имеет в философии и психологии Гербарта, который в *представлении* видит основной факт психической жизни. «*Представления*, — говорит Гербарт, — становятся силами, сопротивляясь друг другу. Это происходит тогда, когда сходятся вместе несколько противоположных *представлений*». («Основания психологии», I гл., § 10). Исходя из положения, что *представления* суть самостоятельные силы, Гербарт строит своеобразную статику и динамику представлений.

Представление (психология)

Представле́ние — процесс и результат мысленного воссоздания образов предметов и явлений, которые в данный момент не воздействуют на органы чувств человека. Понятие «представление» имеет два значения. Одно из них обозначает образ предмета или явления, которые ранее воспринимались анализаторами, но в данный момент не воздействуют на органы чувств («название результата процесса», девербатив). Второе значение данного термина описывает сам процесс воспроизводства образов («название процесса», субстантивированный инфинитив).

Описание представления

Представления как психические явления имеют как черты сходства, так и отличий с такими психическими явлениями, как восприятие, псевдогаллюцинации и галлюцинации.

Физиологическую основу представлений составляют «следы» в коре больших полушарий головного мозга, остающиеся после реальных возбуждений центральной нервной системы при восприятии. Эти «следы» сохраняются благодаря известной «пластичности» центральной нервной системы.

Классификация представлений

Существуют различные способы классификации представлений**.**

По ведущим анализаторам (по модальностям)

В соответствии с разделением представлений по репрезентативным системам (по модальности ведущего анализатора) выделяют следующие виды представлений:

- *зрительные* (образ человека, места, пейзажа);
- *слуховые* (воспроизведение музыкальной мелодии);
- *обонятельные* (представление какого-то характерного запаха — например, огуречного или парфюмерного);
- *вкусовые* (представления о вкусе пищи — сладком, горьком и пр.)
- *тактильные* (представление о гладкости, шершавости, мягкости, твердости предмета);
- *температурные* (представление о холоде и тепле).

Тем не менее, часто в формировании представлений участвуют сразу несколько анализаторов. Так, представляя в сознании огурец, человек одновременно представляет себе и его зелёный цвет и пупырчатую поверхность, его твердость, характерный вкус и запах. Представления формируются в процессе деятельности человека, поэтому в зависимости от профессии развивается преимущественно какой-либо один вид представлений: у художника — зрительный, у композитора — слуховой, у спортсмена и балерины — двигательный, у химика — обонятельный и т. д.

По степени обобщенности

Представления различаются также по степени обобщенности. В этом случае говорят о единичных, общих и схематизированных представлениях (в отличие от восприятий, которые всегда бывают единичными).

- *Единичные представления* — это представления, основанные на восприятии одного определенного предмета или явления.

Часто они сопровождаются эмоциями. Эти представления лежат в основе такого явления памяти как узнавание.

- *Общие представления* — представления, обобщенно отражающие ряд сходных предметов. Этот вид представлений чаще всего формируется при участии второй сигнальной системы и словесных понятий.
- ***Схематизированные представления*** **описывают предметы или явления в виде условных фигур, графических изображений, пиктограмм и т. д. Примером может служить диаграммы или графики, отображающие экономические или демографические процессы.**

По происхождению

Третья классификация представлений — по происхождению. В рамках данной типологии их делят на представления, возникшие на основе ощущений, восприятия, мышления и воображения.

- *На основе восприятия*. Большая часть представлений человека — это образы, возникающие на основе восприятия — то есть первичного чувственного отражения действительности. Из данных образов в процессе индивидуальной жизни постепенно формируется и корректируется картина мира каждого конкретного человека.
- *На основе мышления*. Представления, сформированные на основе мышления, отличаются высокой степенью абстрактности и могут иметь мало конкретных черт. Так у большинства людей имеются представления таких понятий как «справедливость» или «счастье», но им трудно наполнить данные образы конкретными чертами.
- *На основе воображения*. Представления могут формироваться и на основе воображения, и данный тип представлений составляет основу творчества — как художественного, так и научного.

По степени волевых усилий

Представления различаются также по степени проявления волевых усилий. В этом случае они делятся на непроизвольные и произвольные.

- *Непроизвольные представления* — это представления, возникающие спонтанно, без активизации воли и памяти человека, например — грёзы.
- *Произвольные представления* — это представления, возникающие у человека под воздействием воли, в интересах поставленной им цели. Эти представления контролируются сознанием человека и играют большую роль в его профессиональной деятельности.

Свойства представлений

Представлениям присущи такие основные свойства, как *наглядность, фрагментарность, неустойчивость* и *обобщенность*.

- **Наглядность**. Человек представляет образ воспринятого объекта исключительно в наглядной форме. При этом имеет место размытость очертаний и исчезновение ряда признаков. Наглядность представлений беднее наглядности восприятия вследствие утраты непосредственности отражения.
- **Фрагментарность**. Для представления предметов и явлений характерна неравномерность воспроизведения их отдельных частей. Преимущество имеют объекты (или их фрагменты), которые в предыдущем перцептивном опыте обладали большей привлекательностью или значимостью. Фрагментарность представлений, отмеченная ещё Г. Эббингаузом и подтвержденная современными исследователями, состоит в том, что «при внимательном анализе или попытке установить все стороны или черты предмета, образ которого дан в представлении, обычно оказывается, что некоторые стороны, черты или части вообще не представлены». Если неустойчивость представления есть аналог неполной константности, то фрагментарность представляет собой эквивалент неполной целостности или выражение её дефицита в представлении по сравнению с восприятием.
- **Неустойчивость**. Представленный в данный момент времени образ (или его фрагмент) можно удерживать в активном сознании лишь в течение определенного времени, по истечении которого он начнет исчезать, утрачивая фрагмент за фрагментом. С другой стороны, образ представления возникает не сразу, а по мере восприятия новых сторон и

свойств предмета, новых временных связей; постепенно он дополняется, изменяется и «проясняется». По своей сущности неустойчивость как проявление непостоянства является отрицательным эквивалентом или выражением дефицита константности, свойственной перцептивному образу. Она хорошо знакома каждому по собственному опыту и заключается в «колебаниях» образа и текучести его компонентов.

- **Обобщенность**. Представленный объект, его образ, обладает определенной информационной емкостью, причем содержание (структура) образа представлений схематизируется или свертывается. Как указывает В.С. Кузин, **представление всегда включает в себя элемент обобщения. В нём материал отдельного восприятия обязательно связывается с материалом предыдущего опыта и предшествующих восприятий.** Новое объединяется со старым.
  Представления — это результат всех прошлых восприятий конкретного предмета или явления. Береза как образ представления — итог всех прошлых восприятий берез как непосредственно, так и на изображениях. Поэтому представление, обобщая конкретный предмет (или явление), одновременно может служить обобщением и целого класса аналогичных предметов в силу того, что представляемый объект не воздействует непосредственно на органы чувств.

Представление (базы данных)

Представление (англ. *view*, иногда используются названия *«вид»*, *«взгляд»*) — виртуальная (логическая) таблица, представляющая собой поименованный запрос (синоним к запросу), который будет подставлен как подзапрос при использовании представления.

В отличие от обычных таблиц реляционных баз данных, представление не является самостоятельной частью набора данных, хранящегося в базе. Содержимое представления динамически вычисляется на основании данных, находящихся в реальных таблицах. Изменение данных в реальной таблице базы данных немедленно отражается в содержимом всех представлений, построенных на основании этой таблицы.

Способ создания и содержимое представлений

Типичным способом создания представлений для СУБД, поддерживающих язык запросов SQL, является связывание представления с определённым SQL-запросом. Соответственно, **содержимое представления — это результат выполнения этого запроса**, а возможности построения представления ограничиваются только степенью сложности диалекта SQL, поддерживаемого конкретной СУБД. Так, для типичных СУБД, таких как PostgreSQL, Interbase, Firebird, Microsoft SQL Server, Oracle, представление может содержать:

- подмножество записей из таблицы БД, отвечающее определённым условиям (например, при наличии одной таблицы «Люди» можно создать два представления «Мужчины» и «Женщины», в каждом из которых будут записи только о людях соответствующего пола);
- подмножество столбцов таблицы БД, требуемое программой (например, из реальной таблицы «Сотрудники» представление может содержать по каждому сотруднику только ФИО и табельный номер);
- результат обработки данных таблицы определёнными операциями (например, представление может содержать все данные реальной таблицы, но с приведением строк в верхний регистр и обрезанными начальными и концевыми пробелами);
- результат объединения (join) нескольких таблиц (например, при наличии таблиц «Люди», «Адреса», «Улицы», «Фирмы и организации» возможно построение представления, которое будет выглядеть как таблица, для каждого человека содержащее его личные данные, адрес места жительства, название организации, где он работает, и адрес этой организации);
- результат слияния нескольких таблиц с одинаковыми именами и типами полей, когда в представлении попадают все записи каждой из сливаемых таблиц (возможно, с исключением дублирования);
- результат группировки записей в таблице (например, при наличии таблицы «расходы» с записями по каждому платежу можно построить представление, содержащее средства, израсходованные на каждую отдельную статью расходов);

- практически любую комбинацию вышеперечисленных возможностей.

Использование

Представления используются в запросах к БД тем же образом, как и обычные таблицы. В случае SQL-СУБД имя представления может находиться в SQL-запросе на месте имени таблицы (в предложении FROM). Запрос из представления обрабатывается СУБД точно так же, как запрос, в котором на месте имени представления находится подзапрос, определяющий это представление. При этом СУБД с развитыми возможностями оптимизации запросов перед выполнением запроса из представления могут проводить совместную оптимизацию запроса верхнего уровня и запроса, определяющего представление, с целью минимизации затрат на выборку данных.

Использование представлений не даёт каких-то совершенно новых возможностей в работе с БД, но может быть очень удобно.

- Представления скрывают от прикладной программы сложность запросов и саму структуру таблиц БД. Когда прикладной программе требуется таблица с определённым набором данных, она делает простейший запрос из подготовленного представления. При этом даже если для получения этих данных требуется чрезвычайно сложный запрос, сама программа этого запроса не содержит.
- Использование представлений позволяет отделить прикладную схему представления данных от схемы хранения. С точки зрения прикладной программы структура данных соответствует тем представлениям, из которых программа эти данные извлекает. В действительности данные могут храниться совершенно иным образом, достаточно лишь создать представления, отвечающие потребностям программы. Разделение позволяет независимо модифицировать прикладную программу и схему хранения данных: как при изменении структуры физических таблиц, так и при изменении программы достаточно изменить представления соответствующим образом. Изменение программы не затрагивает физические таблицы, а изменение физической структуры таблиц не требует корректировки программы.

- С помощью представлений обеспечивается ещё один уровень защиты данных. Пользователю могут предоставляться права только на представление, благодаря чему он не будет иметь доступа к данным, находящимся в тех же таблицах, но не предназначенных для него.
- Поскольку SQL-запрос, выбирающий данные представления, зафиксирован на момент его создания, СУБД получает возможность применить к этому запросу оптимизацию или предварительную компиляцию, что положительно сказывается на скорости обращения к представлению, по сравнению с прямым выполнением того же запроса из прикладной программы.

Специфические типы представлений

Некоторые СУБД имеют расширенные представления для данных, доступных только для чтения. Так, СУБД Oracle реализует концепцию «материализованных представлений» — представлений, содержащих предварительно выбранные невиртуальные наборы данных, совместно используемых в распределённых БД. Эти данные извлекаются из различных удалённых источников (с разных серверов распределённой СУБД). Целостность данных в материализованных представлениях поддерживается за счёт периодических синхронизаций или с использованием триггеров. Аналогичный механизм предусмотрен в Microsoft SQL Server версии 2000.

По самой сути представления могут быть доступны только для чтения. Тем не менее, в некоторых СУБД (например, в Oracle) представления могут быть редактируемыми, как и обычные физические таблицы. Редактирование может допускаться для представлений, выбранных из единственной физической таблицы таким образом, чтобы каждой записи в представлении соответствовала строго одна запись в таблице-источнике, а в числе полей представления был первичный ключ физической таблицы. При выполнении команд редактирования, добавления или удаления для такого представления сервер СУБД преобразует эти команды в соответствующие команды для физической таблицы-источника. Разумеется, если в представлении используется группировка записей или преобразование значений в полях, редактирование такого представления невозможно даже теоретически. Но и такие представления могут, тем не менее, редактироваться, посредством

написания соответствующих триггеров (хотя осмысленность подобных операций целиком останется на совести программиста). Впрочем, редактируемые представления, как и возможность создания триггеров для представлений, поддерживают лишь немногие СУБД.

1.2.5. Понятие

Поня́тие — отображённое в мышлении единство существенных признаков, свойств, связей и отношений предметов или явлений; мысль или система мыслей, выделяющая и обобщающая предметы некоторого класса по общим и в своей совокупности специфическим для них признакам.

Понятие в его отвлеченности противостоит конкретности восприятия. Также **понятие противостоит слову, которое можно трактовать как знак понятия.**

1. Содержание и объём понятия

Выделяют содержание и объём понятия. **Содержанием понятия называется совокупность существенных признаков класса предметов, подпадающих под это понятие.** Например, содержание понятия «ромб» образуют следующие два признака: родовой — «быть параллелограммом» и специфический (видовой) — «иметь равные стороны». **Объёмом понятия называется совокупность самих предметов (или классов предметов), подпадающих под это понятие.** Например, объём понятия «дерево» составляет множество всех деревьев (которые существовали, существуют или будут существовать; реальных и воображаемых), или множество всех разновидностей деревьев.

Между содержанием и объёмом понятия существует обратная зависимость: чем больше содержание понятия, тем меньше его объём. Иными словами, чем больше признаков входит в понятие, тем меньше предметов это понятие охватывает (и наоборот). Например, понятие «лиственное дерево» больше по содержанию, то есть содержит больше признаков, чем понятие «дерево», соответственно объём первого понятия оказывается меньше (у́же), чем объём второго, поскольку

лиственные деревья — это часть (или подкласс) всех деревьев (деревьев вообще).

2. Виды понятий

По объёму

По объёму понятия можно разделить на ***единичные***, ***общие*** и ***пустые***. В объём единичного понятия входит один-единственный объект (одноэлементный класс) — например, «русский писатель Антон Павлович Чехов», «столица Дании». В объём общего понятия входит более одного объекта (например, «дерево», «химический элемент»). Объём пустого понятия представляет собой пустое множество (например, «вечный двигатель», «круглый квадрат»).

По содержанию

По содержанию понятия делят на положительные и отрицательные; относительные и безотносительные; собирательные и несобирательные (разделительные); конкретные и абстрактные; эмпирические и теоретические.

1. ***Положительные*** понятия фиксируют наличие у предмета какого-либо признака (например, «опрятный человек»), ***отрицательные*** указывают на отсутствие этого признака у предмета («неопрятный человек»). Если отрицание «не» или «без» («бес») стало частью слова и без него это слово не употребляется («неряха»), такое понятие также считается положительным.

2. ***Относительное*** понятие обозначает предмет, существование которого подразумевает существование некоторого другого предмета («ученик» — «учитель»). ***Безотносительное*** понятие обозначает предмет, существующий вне подобной зависимости («человек», «дерево»).

3. ***Собирательным*** называется понятие, обозначающее множество однородных предметов, которое мыслится как единое целое («стая», «флот»). Собирательные понятия могут быть общими («лес») или единичными («Созвездие Волопаса»). В отличие от собирательного,

несобирательное (***разделительное***) понятие указывает не на группу, а на отдельный предмет («дерево», «звезда»).

4. Понятие называется ***конкретным***, если оно относится к предмету или классу предметов (например, «дом»), и ***абстрактным***, если оно отражает свойства, признаки предмета, взятые отдельно от него самого (например, «белизна», «доброта»), или отношения между предметами (например, «равенство»).

5. ***Эмпирические*** понятия есть понятия о наблюдаемых объектах и их свойствах, а ***теоретические*** — о ненаблюдаемых объектах. Если эмпирические понятия вырабатываются на основе непосредственного сравнения общих свойств некоторого класса наличествующих (доступных для изучения) объектов или явлений, то теоретические — на основе опосредованного анализа некоторого класса объектов или явлений при помощи ранее выработанных понятий, концепций и формализмов.

Название любого материального предмета является конкретным эмпирическим понятием, а его непосредственно наблюдаемые свойства выражаются абстрактными эмпирическими понятиями. К конкретным теоретическим понятиям относится, в частности, ряд понятий теоретической физики, например «электрон»; абстрактным теоретическим понятием является, например, «спин».

3. Формирование понятий

Формирова́ние поня́тий (образование понятий) — усвоение или выработка человеком новых для него понятий на основе опыта.

Формирование понятий — это переход от единичных вещей и явлений, данных в чувственном опыте, к обобщению этого опыта в понятиях, фиксирующих существенные признаки этих вещей и явлений. Вещи даны в ощущениях и восприятиях, понятиями же оперирует мышление; вещи чувственны, а понятия представляют собой нечувственные сущности, доступные лишь разуму. Как заполняется этот по видимости непреодолимый разрыв между единичным и всеобщим, каким образом возникновение понятий, столь отличных по своей природе от вещей, вообще возможно и как именно протекает

этот процесс, каковы его механизмы, — всё это составляет одну из сложнейших проблем теории познания.

Формирование понятий изучается философией и психологией. Если философия занимается общетеоретическими вопросами — объяснением связи между единичным и всеобщим, то психология сосредоточивает внимание на вопросе о том, как именно происходит выявление признаков, составляющих некоторое понятие (класс), и правил, связывающих эти признаки.

Теории формирования понятий

Платон (теория припоминания) и Аристотель

Принимая во внимание пропасть, разделяющую единичное и общее, Платон отказывается допустить, что понятия могут быть получены, выведены из чувственного опыта. Мы никогда не смогли бы найти обобщающую идею, — говорит он, — если бы уже не имели её. «Мы непременно должны знать равное само по себе ещё до того, как впервые увидим равные предметы» («Федон»). Поэтому «знание — это припоминание» («Федон»). Платон постулирует существование самостоятельной сферы идей (эйдосов). Идеи существуют сами по себе, объективно, независимо от нашего познания и чувственного мира (более того, как раз вещи этого чувственного мира производны от идей, представляют собой их воплощения). (Следует отметить, что понятия не тождественны идеям: идеи, в отличие от понятий, не в нас, не присутствуют в сознании; **идеи — это то, что мыслится в понятиях**.) Далее, переходя в самый ответственный момент на язык мифа, Платон говорит, что душа некогда обитала в той небесной сфере, где существуют идеи, и там созерцала их; однако, пав на Землю, душа позабыла это знание. Но при виде вещей, являющихся тенью, несовершенным отражением идей, душа вспоминает и сами оригиналы. Вещи только помогают их вспомнить, «напоминая» об идеях, которые душа некогда непосредственно созерцала.

По сходному пути пошёл ученик Платона Аристотель, утверждая, что знание общего не вырабатывается из знания единичного, а лишь *выявляется* благодаря ему. Согласно Аристотелю, все формы бытия уже существуют в душе потенциально, будучи заложены в пассивной части души (в пассивном уме); воздействие действительности на душу

через ощущения, в сочетании с работой активной части ума, актуализирует их.

Закон диссоциации (У. Джеймс)

Философ и психолог У. Джеймс предлагает следующее объяснение механизма формирования понятий. «Мы бы никогда не смогли различить элементы абсолютно неизменяющейся группы, состоящей из свойств, нигде более порознь не встречающихся, — пишет Джеймс. — Если бы все холодное было мокро, а мокрое — холодно, если бы только твердые вещи были колючи, а остальные нет, то вероятно ли, чтобы мы различали холодное и влажное, твёрдое и колючее? …Если бы теплота прямо зависела от высоты предмета над земной поверхностью… то для понятий „теплота“ и „высота“ у нас имелось бы одно слово».

Но если некий признак можно встретить и в составе других групп, вместе с другими признаками, то он начинает выделяться. Признак, который мы встречаем то в одном, то в другом объекте, вследствие этого отделяется и от того и от другого «и мало-помалу становится для нашего сознания самостоятельным представлением — абстрактом». Джеймс называет это *законом диссоциации образа при изменении сопровождающих элементов*.

Ассоциативная теория

Ассоцианизм не видит принципиальных различий между понятиями и представлениями.

«Еще Дж. Локк сформулировал этот взгляд. Особенную наглядность придает ему коллективными фотографиями Ф. Гальтон, в которых он на одной и той же пленке делал один снимок поверх другого; накладывание их друг на друга приводило к тому, что индивидуальные признаки стирались и сохранялись лишь общие черты. По этому образцу мыслил ряд психологов, придерживавшихся этой концепции на природу понятий и процесс их образования».

Т. Циген полагал, что **понятие — это ассоциация представлений**.

В 1950-х годах Рестл и Бурн попытались свести формирование понятий к повторяющемуся совместному предъявлению признаков, сопровождающемуся подкреплением. Их взгляд состоял в том, что подкрепление правильного сочетания признаков ведёт к постепенному отсеву несущественных признаков и формированию понятия из существенных. Между существенными признаками и реакцией опознания их как понятия образуется ассоциация.

Теория выдвижения и проверки гипотез (Дж. Брунер)

Работа Джерома Брунера и его коллег *A Study of Thinking* (1956) оказала сильное влияние на формирующуюся когнитивную психологию в целом и исследования в области формирования понятий в частности.

Дж. Брунер предположил, что следует изучать различные *стратегии* формирования понятий и предложил для этого соответствующую методику). Брунер описал две стратегии: сканирование и сосредоточение (иначе: «целостная стратегия», «фокусировка»), каждая из которых также имеет по две разновидности.

1. *Одновременное сканирование*. Происходит одновременная проверка всех возможных гипотез (**под гипотезами понимаются различные наборы признаков; один из этих наборов и представляет собой искомое понятие**). Не выдержавшие проверки гипотезы отбрасываются по мере их опровержения. Особенность этой стратегии в том, что при этом необходимо помнить всё просмотренное в ходе проверки.
2. *Последовательное сканирование*. В этом случае гипотезы проверяются поочерёдно. Когда гипотеза оказывается опровергнутой, её отбрасывают и переходят к другой с учетом всего предыдущего опыта.
3. *Консервативное сосредоточение*. Берётся положительный пример понятия (то есть то, о чём достоверно известно, что этот предмет подходит под искомое понятие), после чего его признаки по одному проверяются на существенность. Заменяя проверяемый признак, но оставляя все остальные без изменения, можно определить, является ли этот признак существенным, то есть входит ли в искомое понятие. Эта стратегия выгоднее, так как уменьшает нагрузку на память.

4. *Рискованное сосредоточение* отличается от консервативного тем, что изменяются 2 или более признаков за раз.

4. Эмпирические методы исследования

В психологии существуют различные методы исследования понятий, среди которых особенно полезен для изучения формирования понятий метод формирования искусственных понятий**.**

Метод формирования искусственных понятий

Этот метод состоит в том, что испытуемому предъявляется ряд объектов, сходных по одним признакам и различающихся по другим. О каждом из этих объектов испытуемый узнаёт, что он является (или, наоборот, не является) примером некоторого понятия (иначе говоря, входит в некоторый класс), причём точное определение этого понятия (признаки класса) испытуемому неизвестно. Задача испытуемого — на основании получаемой информации «отгадать» это понятие, то есть определить признаки, составляющие его (и правила, связывающие эти признаки).

Понятие в теории решения задач

Теория решения задач — теоретический раздел исследований по искусственному интеллекту — предлагает достаточно математически строгую и в то же время наглядную трактовку термина «понятие». Полное математически строгое описание можно найти в монографии Бенерджи**.**

Можно дать менее строгое, но более лаконичное описание таким образом:

1. Понятия образуются на основании свойств.
2. Существует два основных класса свойств — внутренние и внешние. Внешние свойства выявляются непосредственно, их существование постулируется, вопрос об их происхождении не ставится. Внутренние свойства являются ненаблюдаемой непосредственно логической функцией внешних свойств.

3. При решении задач используются преимущественно внутренние свойства. Использование это состоит в том, что в зависимости от значения свойства выбирается та или иная операция, ведущая к решению задачи.
4. Понятие в традиционном его понимании — это особый вид внутренних свойств, получаемых в результате логической конъюнкции (логическое И) внешних свойств.
5. Любое внутреннее свойство можно представить в виде дизъюнкции (логическое ИЛИ) понятий.

В такой трактовке закон обратного отношения действительно оказывается тривиальным следствием определения и одного из законов поглощения A&B→A. Стоит заметить, что закон обратного отношения не имеет места для произвольного свойства.

Бенерджи рассматривает модель задач, в которой задано некоторое множество ситуаций и множество преобразований (операций) одной ситуации в другую. Выделено также **подмножество ситуаций, являющихся целью решения. «При этом мы стремимся перевести данную ситуацию в другую допустимую ситуацию, применяя последовательность преобразований, чтобы в конце прийти к целевой ситуации**». Понятия в модели Бенерджи применяются для описания как целевого подмножества, так и стратегии выбора преобразований.

Понятия по Бенерджи логично было бы называть «протопонятиями», так как в общенаучном смысле понятия выделяются и фиксируются с помощью термина в ходе решения широкого класса однородных задач, в которых их применение оказалось полезным.

Понятие в психологии

Психология позволяет подойти к изучению понятий эмпирически, исследуя существующие в сознании отношения между понятиями (семантические кластеры, группы, сети), в том числе с помощью математических методов (кластерного и факторного анализа); процессы формирования понятий, в том числе с помощью метода формирования искусственных понятий; возрастное развитие понятий и т. п.

Методы исследования понятий

В психологии разработано множество методов исследования понятий, таких как ассоциативный эксперимент, метод классификации, метод субъективного шкалирования, семантический дифференциал, метод формирования искусственных понятий.

В некоторых случаях, как, например, в методе семантического радикала, используются также физиологические измерения.

Возрастное развитие понятий

Психологические исследования позволили установить, что **понятия не являются неизменными по своей природе сущностями**, не зависящими от возраста оперирующего ими субъекта. Овладение понятиями происходит постепенно, и понятия, которыми пользуется ребёнок, отличаются от понятий взрослого человека. Были выявлены различные типы понятий, соответствующие различным возрастным стадиям.

Предпонятия

Ж. Пиаже обнаружил, что на дооперациональной стадии когнитивного развития (2—7 лет) понятия ребёнка представляют собой ещё не истинные понятия, но предпонятия. Предпонятия образны и конкретны, не относятся ни к индивидуальным объектам, ни к классам вещей и связываются друг с другом посредством трансдуктивного рассуждения, представляющего собой переход от частного к частному.

Житейские и научные понятия

Л. С. Выготский, исследуя развитие понятий в детском возрасте, писал о житейских (спонтанных) и научных понятиях. Житейские понятия — приобретаемые и используемые в быту, в повседневном общении слова вроде «стол», «кошка», «дом». Научные понятия — это слова, которые ребёнок узнаёт в школе, термины, встроенные в систему знаний, связанные с другими терминами.

При использовании житейских понятий *ребёнок долгое время* (до 11-12 лет) *осознаёт только предмет*, на который они указывают, *но не сами понятия, не их значение*. Лишь постепенно ребёнок овладевает значением понятий. Согласно взглядам Выготского, развитие спонтанных и научных понятий идёт в противоположных направлениях: спонтанных — к постепенному осознанию их значения, научных — в обратном направлении.

Приходящее с возрастом осознание значений связано с рождающейся систематичностью понятий, то есть с установлением логических отношений между ними. А поскольку научные понятия, которые ребёнок усваивает в процессе обучения, принципиально отличаются от житейских понятий именно тем, что по самой своей природе они должны быть организованы в систему, то — полагает Выготский — их значения и осознаются первыми. Осознанность же значений научных понятий постепенно распространяется и на житейские.

1.2.6. Мышление

Мышле́ние — это познавательная деятельность человека. Оно является опосредованным и обобщённым способом отражения действительности.

Практическим мышлением можно назвать процесс построения цепочки логически связанных событий, вплоть до необходимого на практике, для достижения желаемого результата.

Результатом мышления является мысль (понятие, смысл, идея). Мышление противопоставляют «низшим» способам освоения мира в форме ощущения или восприятия, которые свойственны в том числе и животным. Многие философы называли мышление сущностным свойством человека. Так Декарт утверждал: «Я мыслю, следовательно, я существую». Паскаль называл человека мыслящим тростником.

Особенностью мышления является свойство получать знание о таких объектах, свойствах и отношениях окружающего мира, которые не могут быть непосредственно восприняты. Это свойство мышления

осуществляется посредством таких умозаключений как аналогия и дедукция**.**

Мышление связано с функционированием мозга, однако сама способность мозга к оперированию абстракциями возникает в ходе усвоения человеком форм практической жизни, норм языка, логики, культуры. Мышление осуществляется в многообразных формах духовной и практической деятельности, в которых обобщается и сохраняется познавательный опыт людей. **Мышление осуществляется в образно-знаковой форме**, основные результаты его активности выражаются здесь в продуктах художественного и религиозного творчества, **своеобразно обобщающих познавательный опыт человечества**. **Мышление осуществляется также в собственной адекватной ему форме теоретического познания, которое с опорой на предшествующие формы приобретает неограниченные возможности умозрительного и модельного видения мира.**

Мышление изучается почти всеми существующими научными дисциплинами, являясь в то же время объектом исследования ряда философских дисциплин — логики, гносеологии, диалектики.

История представлений о мышлении

Уже в античной науке с мышлением была связана установка на выявление не явлений, а сущности, не видимого (данного в чувственном восприятии), а того, что существует на самом деле, — отмечает В. М. Розин.

Первым философом, который поставил вопрос о мышлении (ноэзис), был Парменид. Результатом такого мышления может быть как истина, так и мнение.

Процесс мышления Платоном понимается как припоминание того, что душа знала в своей космической жизни, но забыла при вселении в тело. И само мышление, которое он считал главным, когнитивным процессом, по сути является мышлением репродуктивным, а не творческим, хотя он и оперирует понятием интуиция, ведущим для творческого мышления.

Аристотель создал логику — науку о мышлении, в рамках которой он рассмотрел такие составные части мышления как понятие, суждение и умозаключение. Впоследствии, в эпоху Средневековья, последователи Аристотеля акцентировали своё внимание на такой форме мышления как силлогизм и дедукция, что привело к созданию «мыслительной машины» Раймунда Луллия.

Мышление для Декарта представало как нечто бестелесное, духовное. Более того, мышление является единственным атрибутом души, и именно это обусловливает постоянность мыслительных процессов, происходящих в душе, то есть она всегда знает о том, что происходит внутри неё. Душа — это мыслящая субстанция (лат. *res cogitans*), вся сущность или природа которой состоит в одном мышлении. В качестве метода познания Декарт использовал систематическое сомнение.

Спиноза определяет мышление как способ действия мыслящего тела. Из этого определения вытекает и предложенный им способ раскрытия/определения этого понятия. Для того, чтобы определить мышление, необходимо тщательно исследовать способ действий мыслящего тела в отличие от способа действий (от способа существования и движения) тела немыслящего.

Одной из заслуг Канта часто называется различение аналитического и синтетического мышления.

С точки зрения психологии

В психологии **мышление — совокупность умственных процессов, лежащих в основе познания**; именно к мышлению относят активную сторону познания: **внимание, восприятие, процесс ассоциаций, образование понятий и суждений.** В более тесном логическом смысле мышление заключает в себе лишь образование суждений и умозаключений путём анализа и синтеза понятий.

Мышление — опосредованное и обобщённое отражение действительности, вид умственной деятельности, заключающейся в познании сущности вещей и явлений, закономерных связей и отношений между ними.

Мышление как одна из высших психических функций — психический процесс отражения и познания существенных связей и отношений предметов и явлений объективного мира.

По способу решения задач мышление (как психический процесс, биологической целью которого является оптимальное решение возникшей перед индивидуумом проблемы) может быть конвергентным (коррелируется с интеллектом, линейное мышление, приводящее к одному-единственному результату) и дивергентным (коррелируется с творческими способностями, состоит в нахождении множественности вариантов решения проблемы или разноплановом видении одного объекта). Термин конвергентного и дивергентного мышления предложен американским психологом Дж.Гилфордом (1950г.) и развит Э.Торренсом.

Операции мышления

- Анализ — разделение предмета/явления на составляющие компоненты. Может быть мысленный и ручной.
- Синтез — объединение разделённых анализом с выявлением при этом существенных связей.

Анализ и синтез являются основными операциями мышления, на основе которых выстраиваются иные типологические единицы.

- Сравнение — сопоставление предметов и явлений, при этом обнаруживаются их сходства и различия.
- Классификация — группировка предметов по признакам.
- Обобщение — объединение предметов по общим существенным признакам.
- Конкретизация — выделение частного из общего.
- Абстрагирование — выделение какой-либо одной стороны, аспекта предмета или явления с игнорированием других.

Закономерности рассмотренных операций мышления и есть суть основных внутренних, специфических закономерностей мышления. На их основе только и могут получить объяснение все внешние проявления мыслительной деятельности.

1.2.7. Воображение

Воображе́ние — способность человека к спонтанному возникновению или преднамеренному построению в сознании образов, представлений, идей объектов, которые в опыте в целостном виде не воспринимались или не могут восприниматься посредством органов чувств (как, например, события истории, предполагаемого будущего, явления невоспринимаемого или мира, не существующего вообще - сверхъестественные персонажи сказок, мифов и пр.); способность сознания создавать образы, представления, идеи и манипулировать ими; играет ключевую роль в следующих психических процессах: моделирование, планирование, творчество, игра, память. В широком смысле, **всякий процесс, протекающий «в образах» является воображением** .

Воображение является основой наглядно-образного мышления, позволяющего человеку ориентироваться в ситуации и решать задачи без непосредственного вмешательства практических действий. Оно во многом помогает ему в тех случаях жизни, когда практические действия или невозможны, или затруднены, или просто нецелесообразны. Например, при моделировании абстрактных процессов и объектов.

Разновидность творческого воображения — фантазия. **Воображение — одна из форм психического отражения мира.** Наиболее традиционной точкой зрения является **определение воображения как процесса** (А. В. Петровский и М. Г. Ярошевский, В. Г Казаков и Л. Л. Кондратьева и др). Согласно М. В. Гамезо и И. А. Домашенко: **«Воображение — психические процессы, заключающиеся в создании новых образов (представлений) путём переработки материала восприятий и представлений, полученных в предшествующем опыте».** Отечественными авторами это явление также рассматривается как способность (В. Т. Кудрявцев, Л. С. Выготский) и как специфическая деятельность (Л. Д. Столяренко, Б. М. Теплов). Принимая во внимание сложное функциональное строение, Л. С. Выготский считал адекватным применение понятия **психологической системы**.

По мнению Э. В. Ильенкова, традиционное понимание воображения отражает только его производную функцию. Главная — позволяет видеть то, что есть, то, что лежит перед глазами, то есть **основной**

функцией воображения является преобразование оптического явления на поверхности сетчатки в образ внешней вещи.

Классификация процессов воображения

1. По результатам:

- Репродуктивное воображение (воссоздание действительности такой, какая она есть)
- Продуктивное (творческое) воображение:
  - с относительной новизной образов;
  - с абсолютной новизной образов.

2. По степени целенаправленности:

- активное (произвольное) — включает воссоздающее и творческое воображение;
- пассивное (непроизвольное) — включает непреднамеренное и непредсказуемое воображение.

3.По виду образов:

- конкретное;
- абстрактное.

4.По приёмам воображения:

- **агглютинация** — соединение несоединимых в реальности объектов и свойств;
- **гиперболизация** — увеличение предмета, его частей и интенсивности свойств;
- **замещение** - замена объекта, процесса или частей на другие, в чем-то подобные;
- **концентрирование (концентрирующая интеграция)** - полное или существенное включение одних частей в другие, в том числе путем совмещения, "сплавления", "растворения";
- **комплексирование** - в технике, объединение в одно связное целое объектов (узлов, механизмов), имеющих относительно самостоятельное назначение;

- **миниатюризация** — уменьшение предмета, его частей и интенсивности свойств. Прием создания нового мысленного образа, обратный гиперболизации;
- **расчленение и элиминация** - разбор, разделение целого на части и с удалением некоторых из них;
- **перенос (эвристическая трансдукция)** - перенос частей, процессов из одной целостной системы в другую;
- **перестановка** - перемещение частей целого на другие места, в том числе с изменением взаимодействия;
- **регенерация** - достройка недостающих, удаленных одной или нескольких частей;
- **придание сходства, подобия по форме, материалу, устройству, принципу действия,** например, технических устройств - живым существам;
- **противопоставление** - прием преобразования представлений, состоящий в сознательном поиске и использовании явлений, процессов, объектов, имеющих свойства, противоположные исходному образу;
- **реинтеграция** - достройка до целого из одной основной, важной части или процесса;
- **схематизация** — выделение различий и выявление черт сходства;
- **типизация** — выделение существенного, повторяющегося в однородных явлениях;
- **трансформация** - преобразование внешней формы.

5. По степени волевых усилий:

- преднамеренное;
- непреднамеренное.

Четырёхэтапная модель творческого процесса Уоллеса

- Стадия приготовления, сбор информации. Заканчивается ощущением невозможности решить проблему.
- Стадия инкубации. Ключевая стадия. Человек сознательно не занимается проблемой.
- Инсайт (озарение).
- Проверка решения.

Механизмы воображения

- агглютинация — создание нового образа из частей других образов;
- гиперболизация — увеличение или уменьшение объекта и его частей;
- схематизация — сглаживание различий между объектами и выявление их сходств;
- акцентирование — подчеркивание особенностей объектов;
- типизация — выделение повторяющегося и существенного в однородных явлениях.

Существуют условия, способствующие нахождению творческого решения: наблюдательность, легкость комбинирования, чуткость к проявлению проблем.

Гилфорд вместо понятия «воображение» использовал термин «дивергентное мышление». Он означает порождение новых идей с целью самовыражения человека. Характеристики дивергентного мышления:

- беглость;
- гибкость;
- оригинальность;
- точность.

Развитие воображения у детей

Через творчество у ребёнка развивается мышление. Этому способствуют настойчивость и выраженные интересы. Отправной точкой для развития воображения должна быть направленная активность, то есть включение фантазий детей в конкретные практические проблемы.

Развитию воображения способствуют:

- ситуации незавершенности;
- разрешение и даже поощрение множества вопросов;
- стимулирование независимости, самостоятельных разработок;

- билингвистический опыт;
- позитивное внимание к ребёнку со стороны взрослых.

Развитию воображения препятствуют:

- конформность;
- неодобрение воображения;
- жесткие полоролевые стереотипы;
- разделение игры и обучения;
- не готовность к изменению точки зрения;
- преклонение перед авторитетами**.**

Воображение и реальность

Мир воспринимается как интерпретация данных, поступающих от органов чувств. Будучи таковым, он воспринимается реальным в отличие от большинства мыслей и образов.

Функции воображения

- представление действительности в образах, а также создание возможности пользоваться ими, решая задачи;
- регулирование эмоциональных состояний;
- произвольная регуляция познавательных процессов и состояний человека, в частности восприятия, внимания, памяти, речи, эмоций;
- формирование внутреннего плана действий — способности выполнять их внутри, манипулируя образами;
- планирование и программирование деятельности, составление программ, оценка их правильности, процесса реализации.

Воображение и когнитивные процессы

Воображение является познавательным процессом, специфика которого состоит в переработке прошлого опыта.

Взаимосвязь воображения и органических процессов наиболее ярко проявляется в следующих явлениях: идеомоторный акт и психосоматическое заболевание. На основе связи между образами

человека и его органическими состояниями строится теория и практика психотерапевтических воздействий. **Воображение неразрывно связано с мышлением.** Согласно Л. С. Выготскому, допустимо высказывание о единстве этих двух процессов.

И мышление, и воображение возникают в проблемной ситуации, мотивируются потребностями личности. Основу обоих процессов составляет опережающее отражение. В зависимости от ситуации, запаса времени, уровня знаний и их организации **одна и та же задача может решаться как с помощью воображения, так и с помощью мышления.** Различие состоит в том, что отражение действительности, осуществляемое в процессе воображения, происходит в виде ярких **представлений**, в то время, как опережающее отражение в процессах мышления происходит путём **оперирования понятиями, позволяющими обобщенно и опосредованно познавать окружающее**. Использование того или иного процесса диктуется, прежде всего, ситуацией: творческое воображение работает, в основном, на том этапе познания, когда неопределенность ситуации достаточно велика. Таким образом, **воображение позволяет принимать решения даже при неполноте знания.**

В своей деятельности воображение использует следы прошлых восприятий, впечатлений, представлений, то есть следы памяти (энграммы). Генетическое родство памяти и воображения выражается в единстве составляющих их основу аналитико-синтетических процессов. Принципиальное различие между памятью и воображением обнаруживается в различном направлении процессов активного оперирования с образами. Так, основной тенденцией памяти является восстановление системы образов, максимально приближенной к ситуации, которая имела место в опыте. Для воображения, напротив, характерно стремление к максимально возможному преобразованию исходного образного материала.

Воображение включается в восприятие, влияет на создание образов воспринимаемых предметов и, в то же время, само зависит от восприятия. Согласно идеям Ильенкова, **главной функцией воображения является преобразование оптического явления, состоящего в раздражении световыми волнами поверхности сетчатки, в образ внешней вещи.**

Воображение тесно связано с эмоциональной сферой. Эта связь имеет двойственный характер: с одной стороны, образ способен вызвать сильнейшие чувства, с другой — возникшая однажды эмоция или чувство может стать причиной активной деятельности воображения. Данная система подробно рассмотрена Л. С. Выготским в его работе «Психология искусства». Основные выводы, к которым он приходит, можно изложить следующим образом. Согласно закону реальности чувств, «все фантастические и нереальные наши переживания, в сущности, протекают на совершенно реальной эмоциональной основе». На основе этого Выготский делает вывод, что фантазия является центральным выражением эмоциональной реакции. **Согласно закону однополюсной траты энергии, нервная энергия имеет тенденцию к тому, чтобы растрачиваться на одном полюсе — или в центре или на периферии; всякое усиление энергетической траты на одном полюсе немедленно же влечет за собой ослабление её на другом.** Таким образом, с усилением и усложнением фантазии как центрального момента эмоциональной реакции её периферическая сторона (внешнее проявление) задерживается во времени и ослабевает в интенсивности. Таким образом, воображение позволяет получать разнообразный опыт переживаний и оставаться при этом в рамках социально приемлемого поведения. Каждый получает возможность прорабатывать излишнее эмоциональное напряжение, разряжая его с помощью фантазий, и компенсируя, таким образом, неудовлетворенные потребности.

1.2.8. Составляющие интеллекта и его роль

Интеллектуал — человек с высоко развитым интеллектом и аналитическим мышлением; представитель интеллектуального труда.

Интеллект – относительно устойчивая структура умственных способностей человека, способность решать различные задачи и эффективно адаптироваться в обществе.

Выделяют различные виды интеллекта, каждый из которых, обычно, является способностью решать задачи определенного типа (математический интеллект, вербальный интеллект, социальный интеллект)

Интеллект — это, прежде всего, основа целеполагания, планирования ресурсов и построение стратегии достижения цели.

Зачатками интеллекта обладают животные, и уже на этом уровне их интеллект посредством механизмов целеполагания и достижения целей влиял и влияет на эволюцию животных.

Влияние интеллекта выходит за пределы жизни одного человека. Развитие интеллекта у человека выделило его из животных и стало началом развития общества, а затем и человеческой цивилизации.

Интеллект как способность обычно реализуется при помощи других способностей. Таких как: **способности познавать, обучаться, мыслить логически, систематизировать информацию путём её анализа, определять её применимость (классифицировать), находить в ней связи, закономерности и отличия, ассоциировать её с подобной и т. д. О наличии интеллекта можно говорить при совокупности всех этих способностей, в отдельности каждая из них не формирует интеллект. Интеллектом может обладать система, составляющие элементы которой каждый в отдельности интеллектом не обладают.**

К параметрам, формирующим отличительные особенности интеллектуальной системы человека относят:

- объём рабочей памяти, способность к прогнозированию, орудийной деятельности, логике,
- многоуровневую (6 слоев нейронов) иерархию системного отбора ценной информации,
- сознание,
- память.

Выделяются биофизические параметры «интеллектуальной энергетики»: количество информации, ускорение (частота, скорость) и расстояние её передачи, — с объединением их в «формулу интеллекта».

Различное содержание деятельности требует развития определённых интеллектуальных способностей индивида. Но во всех случаях **необходима чувствительность индивида к новому, актуальным проблемам, к тенденциям возможного развития ситуации.** Показателем развития интеллекта является несвязанность субъекта

внешними ограничениями, отсутствие у него ксенофобии — боязни нового, непривычного.

Существенное качество ума индивида — предвидение возможных последствий предпринимаемых им действий, способность предупреждать и избегать ненужных конфликтов. Одной из основных особенностей развитого интеллекта является способность к интуитивному решению сложных проблем.

Развитие отдельных качеств интеллекта определяется как генотипом данного индивида, так и широтой его жизненного опыта. У конформных индивидов формируется так называемое *целевое мышление* — сфера мышления индивида сужается до крайне ограниченных житейских пределов, широко распространяется **интеллектуальный инфантилизм**, а в среде интеллектуалов — **созерцательность**. В групповом мышлении начинают преобладать расхожие стереотипы, шаблонные ориентации, схематизированные матрицы поведения. Возникают деформации в содержании интеллекта. Возможны деформации и в структуре интеллекта, в его организации. **Негативным качеством интеллекта является *ригидность* мышления** — его негибкость, предвзятое отношение к явлению, преувеличение чувственного его впечатления, приверженность к шаблонным оценкам.

1.2.9. Различные взгляды на интеллект

Согласно Линде Готтфредсон, **интеллект — это весьма общая умственная способность, которая включает возможность делать заключения, планировать, решать проблемы, абстрактно мыслить, понимать сложные идеи, быстро обучаться и учиться на основании опыта.** Это не просто изучение книг, узкие академические знания или навыки проходить тесты. Напротив, по мнению учёного, **интеллект отражает более широкую и глубокую способность познавать окружающий мир, понимать суть вещей и соображать, что делать в той или иной ситуации.**

Ф. Н. Ильясов определяет ***интеллект*** как **«способность системы создавать в ходе самообучения программы (в первую очередь эвристические) для решения задач определенного класса сложности и решать эти задачи».**

В начале XX века Чарльз Спирман показал, что если человек хорошо решает одни задачи, то он успешен и в решении других, то есть, что все интеллектуальные способности статистически связаны. **Спирман ввёл «фактор *g*» общего интеллекта, показывающий эффективность выполнения всех познавательных задач. На практике оказалось, что «фактор *g*» трудно измерить напрямую. Однако на его основе удалось сформулировать величины, которые измерить возможно и которые представляют собой приблизительные меры *g*. Одним из таких параметров является коэффициент интеллекта (IQ). Психолог Джеймс Флинн первый провел обширные исследования в области динамики IQ в разных странах мира за длительный период и показал, что этот коэффициент непрерывно возрастал в течение 50 лет (Эффект Флинна**).

Социальный интеллект

Социальный интеллект — способность правильно понимать поведение людей. Эта способность необходима для эффективного межличностного взаимодействия и успешной социальной адаптации. Сам термин «социальный интеллект» был введен в психологию Э. Торндайком в 1920 году для обозначения «дальновидности в межличностных отношениях». Многие известные психологи внесли свою лепту в интерпретацию этого понятия. В 1937 году Г. Оллпорт связывал социальный интеллект со способностью высказывать быстрые, почти автоматические суждения о людях, прогнозировать наиболее вероятные реакции человека. Социальный интеллект, по мнению Г. Оллпорта, — особый «социальный дар», обеспечивающий гладкость в отношениях с людьми, продуктом которого является социальное приспособление, а не глубина понимания. Затем способности социального интеллекта многие известные ученые раскрывали в структурах общего интеллекта. Среди них наиболее ярко представлены модели интеллекта, предложенные Д. Гилфордом, Г. Айзенком. Среди психологов до последнего времени ведутся дискуссии вокруг определения интеллекта, данного Э. Борингом: интеллект есть то, что измеряется тестами интеллекта. Имеются различные точки зрения на оценку данного высказывания. По мнению Б. Ф. Анурина, оно достаточно тавтологично, тривиально и прямо напрашивается на критику. Другие исследователи считают такое определение рекурсивным, что является чрезвычайно распространенным в математике, информатике, компьютерном

программировании, искусственном интеллекте. Г. Айзенк не согласен с определением Э. Боринга: тесты интеллекта, утверждает он, составляются не случайным образом и опираются в своей разработке на хорошо известные, выявленные и проверенные природные закономерности, такие как принцип «позитивного многообразия».

Социальный интеллект, по мнению Г. Олпорта — особый «социальный дар», обеспечивающий гладкость в отношениях с людьми, продуктом которою является социальное приспособление, а не глубина понимания.

Создателем первою надежного теста для измерения социального интеллекта стал Дж. Гилфорд. Согласно концепции Гилфорда, социальный интеллект представляет систему интеллектуальных способностей, не зависимую от факторов общего интеллекта. **Эти способности. так же как и общеинтеллектуальные, могут быть описаны в пространстве трех переменных: содержание, операции, результаты.** Дж. Гилфорд выделил одну операцию — познание (С) и сосредоточил свои исследования на **познании поведения** (СВ). **Эта способность включает шесть факторов** (рис. 1).

Познание элементов поведения (CBU) — способность выделять из контекста вербальную и невербальную экспрессию поведения (способность, близкая к выделению «фигуры из фона» в гештальт-психологии).

Познание классов поведения (CBC) — способность распознавать общие свойства в некотором потоке экспрессивной или ситуативной информации о поведении.

Познание отношений поведения (CBR) — способность понимать отношения, существующие между единицами информации о поведении.

Познание систем поведения (CBS) - способность понимать логику развития целых ситуаций взаимодействия людей, смысл их поведения в этих ситуациях.

Познание преобразований поведения (CBT) — способность понимания исходного значения сходною поведения (вербального и невербального) в разных ситуационных контекстах.

Познание результатов поведения (CBI) — способность предвидеть последствия поведения, исходя из имеющейся информации.

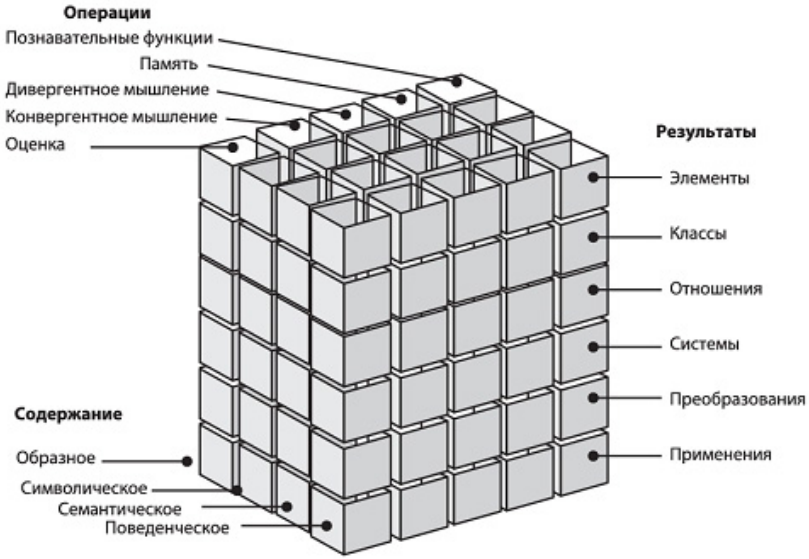


Рис. 1. Структура интеллекта по Дж. Гилфорду

Также как и к темпераменту, к интеллекту нельзя применить оценочно-содержательные характеристики. Интеллект не может быть «порядочным», «моральным», «злым», «добрым». Все эти определения связаны с личностной направленностью субъекта, характеризуя содержательную сторону его мировоззрения, ценностных ориентаций, этического повеления. Можно сказать, что **темперамент** — **это характеристика инструментальной сферы индивидуальности со стороны активности, энергии, а интеллект — со стороны возможностей субъекта, умения распорядиться этой энергией.**

Итак, говоря об исследовании интеллекта и способностей человека, долгие годы приходилось отмечать, что монополия в их изучении принадлежала тестологии.

Впервые о существовании индивидуальных различий в умственных (интеллектуальных) способностях заговорил Ф. Гальтон. Но он отождествлял интеллекте врожденными психофизиологическими функциями (реакция, чувствительность и т.п.).

В 1905 г. французские ученые А. Вине и Т. Симон — в связи с запросами правительства найти способ отличать неспособных детей — создали первую серию тестов (30 заданий), помогающих в какой-то мере определить нечто вроде величины способностей:

IQ = (Умственный возраст/Хронологический возраст) х 100%,

где **умственный возраст** — это средний возраст детей, которые решают те же задания, что и испытуемый (т.е. если 6-летний ребенок решает задания для 8-летних, его умственный возраст равен 8 годам).

Вине и Симон впервые выдвинули идею о влиянии окружающей среды на особенности познавательного развития; однако интеллект в их понятии ограничивался только достигнутым на данный момент уровнем изучаемых особенностей (т.е. репродуктивной его стороной).

На сегодня еще бытуют две основные тенденции в развитии представлений об интеллекте:

- признание общего фактора интеллекта;
- отрицание какого-либо общего начала интеллектуальной деятельности и утверждение существования множества независимых интеллектуальных способностей.

Однако стоит отметить и наличие некого кризиса тестологического подхода к изучению интеллекта: несмотря на мощное методологическое обеспечение изучения интеллектуальных способностей, тестология не смогла породить приемлемую концепцию интеллекта. Кроме того **приверженцы взглядов на интеллект как единую структуру пришли к парадоксальному выводу о множестве различных, не всегда зависимых друг от друга, способностей, а приверженцы идеи множественного интеллекта убедились в наличии общего начала всех проявлений интеллекта**. При более внимательном рассмотрении оказывается, что содержание тестов зачастую подменяет понятие интеллекта либо общим уровнем

образовательного и культурного развития (к тому же, оценка «правильности-неправильности» решения заданий зависит от соответствующей культуры, либо способностью к обучению, которая, впрочем, вовсе не тождественна на самом деле интеллекту). Кроме того, низкие результаты тестов связаны с тревожностью, агрессивностью, интроверсией, т.е. возможна подмена измерения интеллектуальных способностей измерением сформированности индивидуальных механизмов саморегуляции.

В психологии интеллект рассматривается как компонент индивидуальности, связанный с личностными характеристиками (исследования связей интеллекта с эмоционально-волевыми особенностями, социально-экономическими условиями и т.д.).

Б.М. Теплов: «...способность есть индивидуальное свойство, которое различно проявляется у разных людей. Это переменная. Ее можно измерять. Способность есть свойство, связанное с успешным освоением/осуществлением деятельности. Это успех. Его можно измерять...».

Интеллект необходимо рассматривать как сложную многоуровневую структуру:

- интеллект как результат процесса социализации, а также влияния культуры в целом (социокультурный подход):
- интеллект как следствие адаптации к требованиям окружающей среды в естественных условиях взаимодействия человека с окружающим миром (генетический подход);
- **интеллект как особая форма человеческой деятельности (процессуально-деятельностный подход);**
- **интеллект как продукт целенаправленного обучения**;
- **интеллект как совокупность элементарных процессов обработки информации (информационный подход);**
- интеллект как особая форма содержания сознания (феноменологический подход);
- **интеллект как система разноуровневых познавательных процессов (структурно-уровневый подход);**
- **интеллект как фактор саморегуляции (регуляционный подход).**

Предлагается довольно много определений интеллекта. Из них наиболее распространенным было определение, данное К. Ясперсом: «**Интеллект — это человек в целом, рассматриваемый со стороны его способностей».** Но оставалось непонятным, как же следует приступать к исследованию интеллекта, базируясь на особенностях всей психической сферы или принимая во внимание лишь способности исследуемого? Разумеется, человек, больной или здоровый во всех без исключения случаях должен изучаться всесторонне, т.е. в целом. Однако для этого **необходимо знание отдельных качеств и свойств психики.** Поэтому вышеприведенное, еще бытующее определение интеллекта, по существу, ничего не дает ни в теоретическом, ни в практическом положении.

Объединение в понятии интеллекта категорий, связанных с познавательным процессом, соответственно предполагает изучение каждой из этих категорий в отдельности. Кроме того, должны быть рассмотрены и некоторые иные слагаемые, из которых составляется личность человека.

Формулировка интеллекта включает в себя, прежде всего, **опыт и знания** человека. Обе эти категории не могут рассматриваться без учета его **способностей и одаренности.**

Способности — это такие индивидуальные свойства человека, которые помогают ему быстрее и легче овладеть знаниями, приобретать те или иные навыки и умения. **Способности — психические явления, рассматриваемые более широко, чем интеллект**. При этом подразумевается, что способности бывают всегда к какой-то определенной деятельности: познанию и применению его на практике, музыке, технике и т.д. Потому и говорят о способностях не только интеллектуальных, но и музыкальных, художественных, технических.

Если направленность и темперамент являются достаточно самостоятельными подструктурами личности, то способности, как и характер, включая в себя черты и направленности личности и характера, являются общими свойствами личности.

Интеллект, как и другие способности человека, не есть некая постоянная величина. Развитие способностей (на основании определенных биологических предпосылок-задатков) происходит в

течение всей жизни, особенно интенсивно в детстве и юности. Следовательно, разовьются ли имеющиеся у человека задатки, зависит от условий его жизни, воспитания и обучения. В развитии биологических предпосылок ведущая роль принадлежит именно той деятельности, которая опирается на них. И чем раньше это будет сделано, тем эффективнее будут результаты развития. Задатки долгое время могут быть либо не развитыми, не превращенными в способности, либо не использованными во всей широте до определенного времени. Так было с русскими писателями XIX в. С.Т. Аксаковым и И.А. Гончаровым: оба проявили незаурядное литературное дарование лишь в 40-50-летнем возрасте.

Определяя **одаренность**, имеют в виду **сочетание различных способностей человека. Но одаренность — свойство не врожденное.**

Важнейшим свойством и вместе с тем условием, интеллектуальной деятельности является умственная работоспособность. Характерно, что большинство талантливых и гениальных людей обладали большой работоспособностью. Являясь показателем силы нервной системы, работоспособность, тем не менее, не может быть развита в течение жизни и деятельности человека.

Таким образом, способности, одаренность и умственная работоспособность обязательно должны быть рассмотрены и учтены при решении вопроса об объеме знаний и опыта.

Весьма важную и существенную роль играет **память. Богатые запасы памяти — важная предпосылка интеллектуального развития.** Однако память опять-таки не главное в интеллекте. Ч. Дарвин, например, исключительно высокие интеллектуальные способности которого не подлежат сомнению, откровенно признавался, что обладал чрезвычайно слабой памятью.

Наши **знания приобретаются в процессе обучения и практической деятельности. Большой запас знаний, так называемая эрудиция — важнейшее свойство интеллекта.** Несмотря на это, нельзя оценить ум человека, овладевшего значительным количеством знаний, как большой и высокий, ибо и эрудиция не определяет всего интеллекта. Можно иметь диплом об окончании высшего образовательного учреждения и, тем не менее, не обладать развитым интеллектом. И наоборот, человек со сравнительно небольшим запасом знаний только

лишь на основании одного этого не может считаться интеллектуально отсталым. Способность к суждению, применению имеющегося комплекса знаний и опыта может быть у него на высоком уровне, что при соответствующих условиях и проявляется в виде замечательных успехов (вспомните довольно любопытный оскароносный фильм «Миллионер из трущоб»).

Немаловажны и внимание, и волевые качества человека, способности к восприятию, особенности эмоциональной сферы. Однако так же, как и память, эти психические функции представляют собой лишь его предпосылки, которыми обусловлено развитие интеллекта (рис. 2).

Рис. 2. Инфраструктура интеллекта

Специально следует отметить роль мышления, точность понятий, которыми оно оперирует, тонкий анализ и точный синтез, их адекватность ситуации, обстановке. Наконец, **верность суждений и умозаключений обязательны для высокоразвитого интеллекта, являясь вместе с тем его необходимыми условиями и предпосылками**. Таким образом, **интеллект — весьма сложное психическое явление, включающее в свою структуру целый ряд вполне конкретных элементов.** Кроме того, скажем: развитие интеллекта нельзя рассматривать в отрыве от социальной среды, которая обязательно принимается во внимание при квалификации его глубины и широты в течение дальнейшей жизни. Оба эти качества не могут служить критерием при оценке интеллекта — ведь нельзя, посмотрев на термометр сегодня, сказать, какая температура воздуха будет завтра.

Интеллектуальная одаренность — такое состояние индивидуально-психических ресурсов (в первую очередь умственных), которое обеспечивает возможность творческой интеллектуальной деятельности, связанной с созданием субъективно и объективно новых идей,использованием нестандартных подходов в разработке проблем, чувствительностью к ключевым, наиболее перспективным линиям поиска решений; открытостью к любым инновациям.

Американский психолог Дж. Рензулли предлагает модель **интеллектуальной одаренности** (рис. 3), которая является «местом пересечения» трех факторов:

- интеллектуальные способности выше среднего уровня;
- креативность (творческость);
- мотивационная включенность.

Рис. 3. Модель Рензулли

Эта и другие модели свидетельствуют о том, что интеллектуальная одаренность исключает ее сведение только к высокой оценке на тестах интеллекта.

Другой ученый, Р. Стернберг, выделяет **пять критериев интеллектуальной одаренности:**

- превосходства (тестового);
- редкости (нетипичности);
- продуктивности;
- демонстративности (повторяемости);
- ценности (для данной культуры).

Сейчас изучено около 100 отдельных способностей человека (+ прогностическая функция).

Вот основные категории одаренности (ребенка): «...специфическая одаренность; предпочтение ребенка заниматься каким-то определенным видом деятельности; творчество или продуктивность мышления; способность к лидерству; способность к визуальным и исполнительским видам деятельности; психомоторные способности».

Выделение различных видов одаренности служит важной цели — привлечь внимание к более широкому спектру способностей, которые должны получить признание и возможности для развития. Различия между видами одаренности не могут рассматриваться без учета мотивации, сложившейся самооценки, других индивидуальных особенностей, от которых зависит реализация способностей.

1.2.10. Эмоциональный интеллект

Эмоциона́льный интелле́кт (**ЭИ**; англ. *emotional intelligence, EI*) — способность человека распознавать эмоции, понимать намерения, мотивацию и желания других людей и свои собственные, а также способность управлять своими эмоциями и эмоциями других людей в целях решения практических задач.

Понятие эмоционального интеллекта появилось как реакция на частую неспособность традиционных тестов интеллекта предсказать успешность человека в карьере и в жизни. Этому было найдено объяснение, состоявшее в том, что успешные люди способны к эффективному взаимодействию с другими людьми, основанному на эмоциональных связях, и к эффективному управлению своими собственными эмоциями, в то время как принятое понятие интеллекта

не включало эти аспекты, и тесты интеллекта не оценивали эти способности.

По менее научному определению С. Дж. Стейна и Говарда Бука, эмоциональный интеллект, в отличие от привычного всем понятия интеллекта, «является способностью правильно истолковывать обстановку и оказывать на неё влияние, интуитивно улавливать то, чего хотят и в чём нуждаются другие люди, знать их сильные и слабые стороны, не поддаваться стрессу и быть обаятельным».

Предполагается, что именно эмоциональный интеллект в современном его понимании был ключевым для выживания человека в доисторические времена, поскольку он проявляется в способности адаптироваться в окружающей среде, уживаться и находить общий язык с соплеменниками и соседними племенами. Этого аспекта в 1872 году коснулся Чарльз Дарвин в своём труде «Выражение эмоций у людей и животных», где он писал о роли внешних проявлений эмоций для выживания и адаптации.

Проблемой эмоций и контроля над эмоциями много занимался основатель психоанализа Зигмунд Фрейд. Он, в частности, считал, что первые законы и предписания этики, такие как «Свод законов Хаммурапи» (XVIII век до н. э., Вавилон) или эдикт императора Ашоки, можно расценивать именно как первые попытки обуздать и цивилизовать проявления эмоций.

Первые публикации, рассматривавшие социальное взаимодействие людей как вид интеллекта появились ещё задолго до Второй мировой войны. В 1920 году профессор Эдвард Торндайк впервые ввёл понятие *социального интеллекта*, который он описал как «способность понимать людей, мужчин и женщин, мальчиков и девочек, умение обращаться с людьми и разумно действовать в отношениях с людьми». В 1926 году был создан первый получивший широкое распространение тест (тест-анкета) для измерения социального интеллекта — *George Washington Social Intelligence Test*. Попытки измерения социального интеллекта продолжились в следующие десять лет, хотя, по заключению Роберта Торндайка (англ. *Robert Thorndike*) и Сола Стерна (англ. *Saul Stern*), написавших обзор методов измерения социального интеллекта в 1937 году, эти попытки не увенчались успехом.

Важный вклад в исследование интеллекта внёс Дэвид Уэкслер (англ. *David Wechsler*), который **рассматривал интеллект как «совокупную способность индивидуума действовать целенаправленно, рационально мыслить и эффективно взаимодействовать с окружающим миром».** В 1940 году он написал публикацию, в которой разделил способности человека на «интеллектуальные» и «неинтеллектуальные», к числу последних он отнёс аффективные, личностные и социальные, и заключил, что именно «неинтеллектуальные» способности являются ключевыми при предсказании жизненного успеха человека. Влияние Дэвида Уэкслера, много занимавшегося разработкой тестов интеллекта, сохранялось и в начале второй половины XX века, когда доминирующей в психологии стала **теория бихевиоризма.**

В 1960-х впервые стало фигурировать понятие именно *эмоционального интеллекта*. В 1964 году оно появилось в работе Майкла Белдока *Sensitivity to expression of emotional meaning in three modes of communication,* а в 1966 году в работе Б. Лойнера *Emotional intelligence and emancipation.*

В 1975 году Клод Штайнер (англ. *Claude Steiner*), один из основателей **транзактного анализа**, разработал концепцию эмоциональной грамотности и запустил программу тренинга эмоциональной грамотности, представленную в его книге *Achieving Emotional Literacy* (изд. Avon Books, Нью-Йорк, 1997).

Расцвет теории эмоционального интеллекта пришёлся на 1980-е и 1990-е. В 1983 году Говард Гарднер (англ. *Howard Gardner*) **опубликовал свою известную модель интеллекта, в которой разделил интеллект на внутриличностный и межличностный**. В 1985 году Уэйн Пэйн опубликовал работу *A Study of Emotion: Developing Emotional Intelligence*, посвящённую развитию эмоционального интеллекта. В 1988 году Рувен Бар-Он в своей докторской диссертации ввёл понятие *эмоционального коэффициента EQ* (англ. *Emotional Quotient*, по аналогии с англ. *Intelligence Quotient, IQ*). Наконец, в 1990 году вышла влиятельнейшая статья Питера Саловея и Джона Майера «Эмоциональный интеллект» (англ. *Emotional Intelligence*), фактически определившая всё современное понимание эмоционального интеллекта. После выхода этой статьи теория эмоционального интеллекта привлекла большое

внимание, последовала масса публикаций на тему эмоционального интеллекта.

В 1995 году научный журналист Дэниэл Гоулман (англ. *Daniel Goleman*) опубликовал научно-популярную книгу *Emotional Intelligence*, в которой описал историю развития теории эмоционального интеллекта, дал обзор современных научных представлений об эмоциональном интеллекте и даже представил собственную модель эмоционального интеллекта, получившую впоследствии название *смешанной модели*.

В 1996 году Рувен Бар-Он на собрании американской ассоциации психологов в Торонто представил свой новый тест EQ-i (Emotional Quotient Inventory), содержавший **перечень вопросов для определения коэффициента эмоционального интеллекта, из которого родилась известная сейчас «модель эмоционального интеллекта Бар-Она».**

В начале XXI веке разработка концепции эмоционального интеллекта продолжилась, многие новые публикации по этой теме сделали Питер Саловей, Джон Майер, Говард Гарднер, Константин Васили Петридис.

Модель эмоционального интеллекта Майера — Саловея-Карузо (модель способностей)

Данная модель считается в психологии основной на данный момент, именно её, как правило, используют для описания понятия эмоционального интеллекта, хотя также большой популярностью пользуется основанная на этой модели *смешанная модель* Дэниэла Гоулмана. Модель способностей критиковалась некоторыми учёными, в частности Говардом Гарднером, за излишний психометрический уклон.

Майер, Сэловей и Карузо выделяют всего **четыре составляющие эмоционального интеллекта:**

1. **Восприятие эмоций** — способность распознавать эмоции (по мимике, жестам, внешнему виду, походке, поведению, голосу) других людей, а также идентифицировать свои собственные эмоции.

2. **Использование эмоций** для стимуляции мышления — способность человека (главным образом неосознанно) активировать свой мыслительный процесс, пробуждать в себе креативность, используя эмоции как фактор мотивации.
3. **Понимание эмоций** — способность определять причину появления эмоции, распознавать связь между мыслями и эмоциями, определять переход от одной эмоции к другой, предсказывать развитие эмоции со временем, а также способность интерпретировать эмоции во взаимоотношениях, понимать сложные (амбивалентные, неоднозначные) чувства.
4. **Управление эмоциями** — способность укрощать, пробуждать и направлять свои эмоции и эмоции других людей для достижения поставленных целей. Сюда также относится **способность принимать эмоции во внимание при построении логических цепочек, решении различных задач, принятии решений и выборе своего поведения.**

1.2.11. Смешанная модель

Модель эмоционального интеллекта, созданная научным журналистом Дэниэлом Гоулманом приобрела большую популярность. В то же время многие учёные указывают на недостаточную научность этой модели. Смешанная модель предполагает, что эмоциональный интеллект состоит из **5 компонентов**:

1. **Самопознание** — способность идентифицировать свои эмоции, свою мотивацию при принятии решений, узнавать свои слабые и сильные стороны, определять свои цели и жизненные ценности.
2. **Саморегуляция** — способность контролировать свои эмоции, сдерживать импульсы.
3. **Социальные навыки** — способность выстраивать отношения с людьми, манипулировать людьми, подталкивать их в желаемом направлении.
4. **Эмпатия** — способность учитывать чувства других людей при принятии решений, а также способность сопереживать другим людям.
5. **Мотивация** — способность стремиться к достижению цели ради факта её достижения.

Три теста-опросника было создано на основе модели Гоулмана: Emotional Competency Inventory (ECI), Emotional and Social Competency Inventory (ESCI), Emotional and Social Competency — University Edition (ESCI-U).

Модель социального и эмоционального интеллекта (ESI) Рувена Бар-Она

Модель Рувена Бар-Она (англ. *Reuven Bar-On*) была представлена в 1996 году на собрании американской ассоциации психологов в Торонто (Канада). Модель состоит из **15 способностей**:

1. **Самоуважение** — способность понимать и оценивать себя, видеть свои возможности и ограничения, сильные и слабые стороны, и принимать себя вместе со своими сильными и слабыми сторонами.
2. **Эмоциональная осознанность** — способность человека распознавать у себя наличие эмоции в конкретный момент, различать свои эмоции и понимать причины их возникновения.
3. **Ассертивность / Самовыражение** — способность ясно и конструктивно выражать свои чувства и мысли, а также способность мобилизовывать свою эмоциональную энергию, проявлять при необходимости твёрдость убеждений, стоять на своём.
4. **Независимость** — способность полагаться на себя и эмоционально не зависеть от других.
5. **Эмпатия** — это умение распознавать, осознавать и понимать чувства других людей.
6. **Социальная ответственность** — способность идентифицировать себя как члена социальной группы, конструктивно сотрудничать с другими людьми, проявлять заботу и брать на себя ответственность за других людей.
7. **Межличностные отношения** — способность конструктивного общения через вербальные и невербальные коммуникации, способность устанавливать и поддерживать взаимовыгодные отношения, основанные на чувстве эмоциональной близости, умение чувствовать себя свободно и комфортно в социальных контактах.
8. **Стрессоустойчивость** — способность эффективно управлять своими эмоциями, быстро находить выход из ситуации.

9. **Контролирование импульсов** — способность сдерживать свои эмоции, воздерживаться перед соблазном.
10. **Оценка действительности** — способность сверять свои мысли и чувства с объективной внешней реальностью.
11. **Гибкость** — способность быстро корректировать свои чувства, мысли, представления и поведение соответственно меняющимися обстоятельствами.
12. **Решение проблем** — способность устанавливать и формулировать проблему, а также находить для неё потенциально эффективное решение.
13. **Самоактуализация** — способность устанавливать цели и стремиться к их достижению, реализовывать свой потенциал.
14. **Оптимизм** — способность сохранять надежду и позитивное отношение даже в сложных обстоятельствах.
15. **Счастье / Благополучие** — способность чувствовать удовлетворённость собой, другими и жизнью в целом.

Все модели эмоционального интеллекта часто критикуют за достаточно произвольное добавление в них компонентов. И хотя нет сомнений, что все эти компоненты действительно влияют на успех человека в жизни и особенно в карьере, **но для подачи этого как научной теории требуется установить некий чёткий принцип, на основе которого можно было бы структурировать понятие эмоционального интеллекта, а в отсутствии этого принципа понятие эмоционального интеллекта превращается лишь в произвольный набор факторов, влияющих на жизнь человека.**

1.3. Структура и качества интелекта

В начале XX в. английский психолог Ч. Э. Спирмен (1863-1945) разработал статистические методы измерения интеллекта и выдвинул **двухфакторную теорию интеллекта**. В нем выделялся **общий фактор (фактор G**) и **специальные факторы**, определяющие успех в решении задач конкретного типа (**фактор S**). Возникла **теория специфических способностей**. Психолог Дж. П. Гилфорд (1897-1987) **выделил 120 факторов интеллекта и представил ею структуру в виде кубической модели** (рис.4).

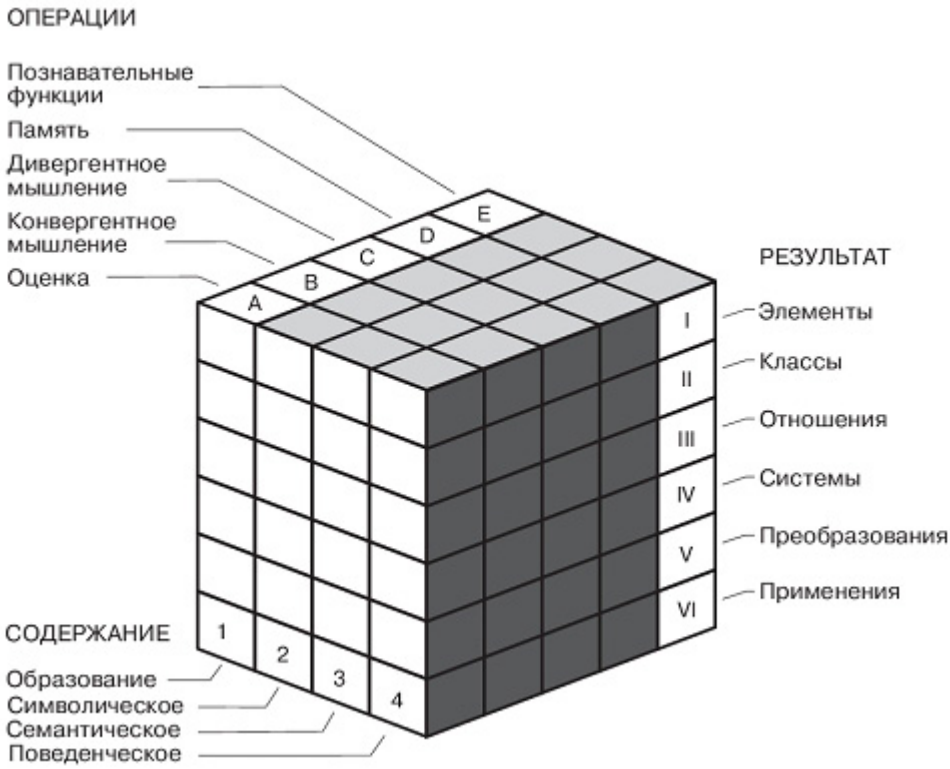


Рис.4. Структура интеллекта по Дж. П. Гилфорду.

Эта кубическая модель — попытка определить каждую из 120 **специфических способностей**, исходя из **трех размерностей мышления: о чем мы думаем (содержание), как мы об этом думаем (операция) и к чему приводит это умственное действие (результат).** Например, при заучивании таких символических обозначений, как сигналы азбуки Морзе (E12), при запоминании семантических преобразований, необходимых для спряжения глагола в том или ином времени (DV3), или при опенке изменений в поведении, когда необходимо пойти на работу по новому пути (AV4), **вовлекаются совсем различные типы интеллекта**

В сотнях тысяч **генов, расположенных в 46 хромосомах**, кроется огромный, еше малоизученный потенциал человеческой индивидуальности. Однако по наследству индивиду передается только «сырье» для построения сложных психорегуляционных конструкций. Жизненные потребности индивида могут посылать соответствующие запросы отдельным генетическим образованиям. **Различные генетические локусы, как показали исследования лауреатов Нобелевской премии Р. Робертсона и Ф. Шарпа, способны к функциональным перестройкам.**

Интеллектуальные возможности человека проявляются в той стратегии, которую он вырабатывает в различных проблемных ситуациях, в его способности трансформировать проблемную ситуацию в конкретную проблему, а затем в систему поисковых задач.

Одни люди способны к быстрым умозаключениям, интуитивным озарениям, одномоментному охвату события во всех его взаимосвязях, **они последовательны в выдвижении гипотез и проверке их правильности**; другие замыкаются на первой пришедшей на ум гипотезе, их мышление нединамично. Некоторые пытаются решать проблемные задачи вообще без всяких предварительных предположений, надеясь на случайные находки; их мышление бессистемно, перекрывается импульсивными эмоциями. Мышление многих людей стереотипно, излишне стандартизировано.

В развитии интеллекта индивида существенную роль играют как генетические, так и социокультурные факторы, а точнее, взаимодействие этих факторов. **Генетические факторы** — наследственный потенциал, получаемый индивидуумом от своих родителей. **Это исходные возможности взаимодействия индивида с окружающим миром**.

Деятельность мозга (обладающего интеллектом), направленную на решение интеллектуальных задач, мы будем называть *мышлением*, или *интеллектуальной деятельностью*.

Интеллект и мышление органически связаны с решением таких задач, как доказательство теорем, логический анализ, распознавание ситуаций, планирование поведения, игры и управление в условиях неопределенности. Характерными чертами

интеллекта, проявляющимися в процессе решения задач, являются способность к обучению, обобщению, накоплению опыта (знаний и навыков) и адаптации к изменяющимся условиям в процессе решения задач. Благодаря этим качествам интеллекта мозг может решать разнообразные задачи, а также легко перестраиваться с решения одной задачи на другую. Таким образом, мозг, наделенный интеллектом, является универсальным средством решения широкого круга задач (в том числе неформализованных), для которых нет стандартных, заранее известных методов решения.

Качества человеческого интеллекта

Основными **качествами человеческого интеллекта** являются **пытливость, глубина ума, его гибкость и подвижность, логичность и доказательность**.

Пытливость ума — стремление разносторонне познать то или иное явление в существенных отношениях. Данное качество ума лежит в основе активной познавательной деятельности.

Глубина ума заключается в способности отделять главное от второстепенного, необходимое от случайного.

Гибкость и подвижность ума — способность человека широко использовать имеющиеся опыт и знания, оперативно исследовать известные предметы в новых взаимосвязях, преодолевать шаблонность мышления. Это качество особенно ценно, если иметь в виду, что **мышление представляет собой применение знаний, «теоретических мерок» к различным ситуациям**. В определенном смысле мышление имеет тенденцию к стабильности, некоторой трафаретности. Это препятствует решению творческих задач, требующих необычного, нешаблонного подхода. Косность мышления обнаруживается, например, при решении следующей задачи. Необходимо зачеркнуть тремя замкнутыми линиями четыре точки, расположенные в виде квадрата. Попытка действовать путем соединения этих точек не приводит к решению задачи. Она может быть решена только при условии выхода за пределы этих точек (рис.5).

При этом негативным качеством интеллекта является **ригидность мышления** — негибкое, предвзятое отношение к сущности явления,

преувеличение чувственного впечатления, приверженность шаблонным оценкам.

Интеллект — способность индивида обобщенно, схематично осмысливать конкретную ситуацию, оптимально организовать ум при решении нестандартных задач. Однако сущность интеллекта нельзя понять только через описание отдельных его свойств. Носителями интеллекта являются опыт умственной деятельности индивида, сформировавшееся у него ментальное пространство, способность к презентации структурного представления исследуемого явления в сознании индивида.

Логичность мышления характеризуется строгой последовательностью рассуждений, учетом всех существенных сторон в исследуемом объекте, всех возможных его взаимосвязей с другими объектами. **Доказательственность мышления** характеризуется способностью использовать в нужный момент такие факты, закономерности, которые убеждают в правильности суждений и выводов.

Критичность мышления предполагает умение строго оценивать результаты мыслительной деятельности, отбрасывать неправильные решения, отказываться от начатых действий, если они противоречат требованиям задачи.

Широта мышления заключается в способности охватить вопрос в целом, не теряя из виду всех данных соответствующей задачи, а также в умении видеть новые проблемы (креативность мышления).

Различное содержание деятельности требует развития определенных ведущих интеллектуальных особенностей индивида, его чувствительности к поисковым проблемам — его **креативности**.

Показателем развитости интеллекта является его **дивергентность** — несвязанность субъекта внешними ограничениями (например, его способность увидеть возможности новых применений обычных предметов).

Существенное качество ума индивида — **прогностика** — предвидение возможного развития событий, последствий предпринимаемых

действий. Способность предвидеть, предупреждать и избегать ненужных конфликтов — признак развитости ума, широты интеллекта.

Интеллектуально ограниченные люди крайне узко, локально отражают действительность, не производят необходимого переноса знаний на новые объекты.

Развитие отдельных качеств ума индивида определяется как генотипом данного индивида, так и широтой его жизненного опыта, **семантическим полем его сознания — индивидуальной системой значений, структурой интеллекта.** В групповом мышлении начинают преобладать стереотипы, шаблонные ориентации, схематизированные матрицы поведения. Возникают деформации как в содержании, так и в структуре интеллекта.

Существенные непатологические нарушения в структуре интеллекта — **умственные аномалии**. Они выражаются в нарушении всей психической системы личности — ее мотивационных, целеобразующих и целедостигающих регуляционных механизмов. Отметим наиболее типичные признаки нарушенности интеллекта:

- неадекватность мотивов совершаемым действиям;
- нарушения в целеобразовании и программировании действий, контроля за их исполнением;
- нарушения смысловых связей, неадекватность средств поставленным целям;
- дефекты мыслительных операций (обобщения, классификации и т. д.).

Приведем некоторые интеллектуальные тесты, выявляющие качества интеллекта (рис. 5-8).

В большинстве тестов интеллекта испытуемому предлагаются задания на обобщение, классификацию, перенос знаний, экстраполяцию и интерполяцию. Некоторые задания оперируют рисунками и геометрическими фигурами. Успех испытуемого определяется количеством правильно выполненных заданий.

Рис.5. Тесты на дивергентность мышления

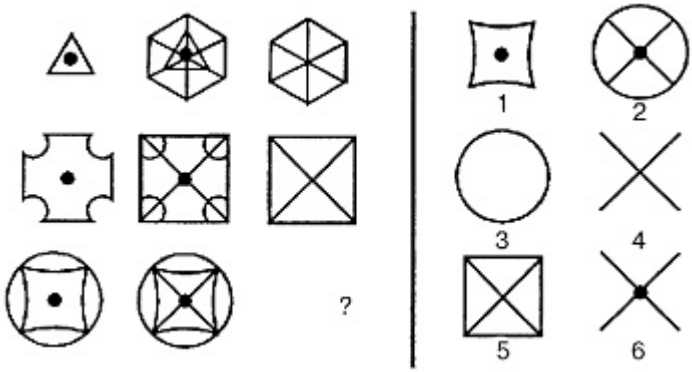


Рис.6. Выберите нужную фигуру из шести пронумерованных

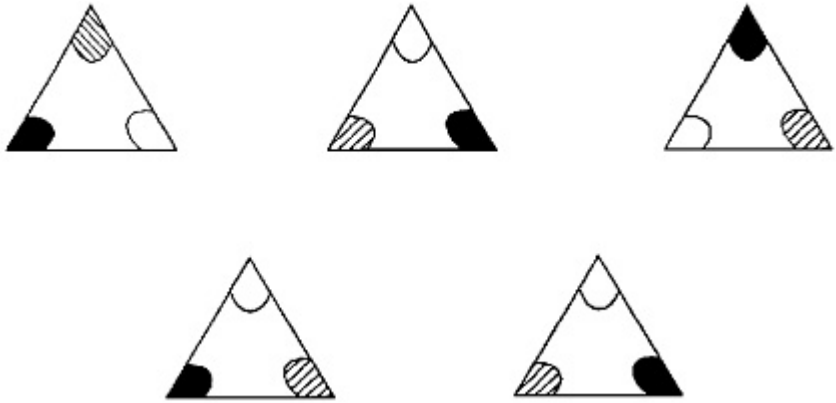


Рис.7. Исключите лишнюю фигуру

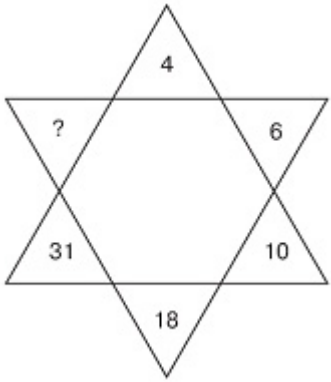


Рис.8. Вставьте недостающее число (тест на экстраполяцию)

Тест на выявление абстрагирующей деятельности

Из слов, стоящих в скобках, выделите два слова, имеющих существенное отношение к исходному слову.

1. САД (растения, садовник, собака, забор, земля).
2. РЕКА (берег, рыба, рыболов, тина, вода).
3. ГОРОД (автомобиль, здание, толпа, улица, площадь).
4. САРАЙ (сеновал, лошади, крыша, скот, стены).

5. КУБ (углы, чертеж, сторона, камень, дерево).
6. ДЕЛЕНИЕ (класс, делимое, карандаш, делитель, бумага).
7. КОЛЬЦО (диаметр, алмаз, округлость, золото, печать).
8. ЧТЕНИЕ (глаза, книга, картина, печать, слово).
9. ГАЗЕТА (правда, приложения, телеграммы, бумага, редактор).
10. ИГРА (карты, игроки, штрафы, наказания, правила).
11. ВОЙНА (пушки, самолеты, сражение, ружья, солдаты).

Ключ

1. Растения, земля.
2. Берег, вода.
3. Здание, улица.
4. Крыша, стсны.
5. Углы, сторона.
6. Делимое, делитель.
7. Диаметр, округлость.
8. Глаза, печать.
9. Бумага, редактор.
10. Игроки, правила.
11. Сражения, солдаты.

1.4. Мышление, его формы и виды

1.4.1. Общее понятие о мышлении

Информация, полученная человеком из окружающего мира, позволяет человеку представлять не только внешнюю, но и внутреннюю сторону предмета, представлять предметы в отсутствие их самих, предвидеть их изменение во времени, устремляться мыслью в необозримые дали и микромир. Все это возможно благодаря **процессу мышления**. В психологии под **мышлением понимают процесс познавательной деятельности индивида, характеризующийся обобщенным и опосредованным отражением действительности. Предметы и явления действительности обладают такими свойствами и отношениями, которые можно познать непосредственно, при помощи ощущений и восприятий (цвета, звуки, формы, размещение и перемещение тел в видимом пространстве).**

Первая особенность мышления — его опосредованный характер. То, что человек не может познать прямо, непосредственно, он познаёт косвенно, опосредованно: одни свойства через другие, неизвестное — через известное**. Мышление всегда опирается на данные чувственного опыта — ощущения, восприятия, представления — и на ранее приобретённые теоретические знания. Косвенное познание и есть познание опосредованное.**

Вторая особенность мышления — его обобщённость. **Обобщение как познание общего и существенного в объектах действительности возможно потому, что все свойства этих объектов связаны друг с другом. Общее существует и проявляется лишь в отдельном, в конкретном.**

Обобщения люди выражают посредством речи, языка. Словесное обозначение относится не только к отдельному объекту, но также и к целой группе сходных объектов. **Обобщённость также присуща и образам (представлениям и даже восприятиям).** Но там она всегда ограничена наглядностью. **Слово же позволяет обобщать безгранично.** Философские понятия материи, движения, закона, сущности, явления, качества, количества и т.д. — широчайшие обобщения, выраженные словом.

Результаты познавательной деятельности людей фиксируют в форме понятий. Понятие — есть отражение существенных признаков предмета. Понятие о предмете возникает на основе многих суждений и умозаключений о нём. **Понятие как результат обобщения опыта людей является высшим продуктом мозга, высшей ступенью познания мира.**

Мышление человека протекает в форме суждений и умозаключений. **Суждение — это форма мышления, отражающая объекты действительности в их связях и отношениях.** Каждое суждение есть отдельная мысль о чём-либо. **Последовательная логическая связь нескольких суждений, необходимая для того, чтобы решить какую-либо мыслительную задачу, понять что-нибудь, найти ответ на вопрос, называется рассуждением**. Рассуждение имеет практический смысл лишь тогда, когда оно приводит **к определённому выводу, умозаключению**. **Умозаключение (вывод) и будет ответом на вопрос, итогом поисков мысли.**

Умозаключение — это вывод из нескольких суждений, дающий нам новое знание о предметах и явлениях объективного мира. Умозаключения бывают **индуктивные, дедуктивные и по аналогии**.

Мышление — высшая ступень познания человеком действительности. Чувственной основой мышления являются ощущения, восприятия и представления. Через органы чувств — эти единственные каналы связи организма с окружающим миром — поступает в мозг информация. Содержание информации перерабатывается мозгом. Наиболее сложной (логической) формой переработки информации является деятельность мышления. Решая мыслительные задачи, которые перед человеком ставит жизнь, он размышляет, делает выводы и тем самым познает сущность вещей и явлений, открывает законы их связи, а затем на этой основе преобразует мир.

Мышление не только теснейшим образом связано с ощущениями и восприятиями, но оно формируется на основе их. **Переход от ощущения к мысли — сложный процесс, который состоит, прежде всего, в выделении и обособлении предмета или признака его, в отвлечении от конкретного, единичного и установлении существенного, общего для многих предметов.**

Мышление выступает главным образом как решение задач, вопросов, проблем, которые постоянно выдвигаются перед людьми жизнью. **Решение задач всегда должно дать человеку что-то новое, новые знания.** Поиски решений иногда бывают очень трудными, поэтому мыслительная деятельность, как правило, — деятельность активная, требующая сосредоточенного внимания, терпения. Реальный процесс мысли — это всегда процесс не только познавательный, но и эмоционально-волевой.

Для мышления человека более существенно взаимосвязь не с чувственным познанием, а с речью и языком. В более строгом понимании **речь - процесс общения, опосредованный языком.** Если **язык — объективная, исторически сложившаяся система кодов** и предмет специальной науки — языкознания, то **речь является психологическим процессом формулирования и передачи мысли средствами языка.**

Современная психология не считает, что внутренняя речь имеет такое же строение и такие же функции, как и развернутая внешняя речь. Под внутренней речью психология подразумевает существенный переходный этап между замыслом и развернутой внешней речью. Механизм, который позволяет перекодировать общий смысл в речевое высказывание, т.е**. внутренняя речь есть, прежде всего, не развернутое речевое высказывание, а лишь подготовительная стадия**.

Однако неразрывная связь мышления с речью вовсе не означает, что мышление может быть сведено к речи. Мышление и речь не одно и то же. Мыслить, не значить говорить про себя. Свидетельством этому может служить возможность высказывания одной и той же мысли разными словами, а также то, что мы не всегда находим нужные слова, чтобы выразить свою мысль.

Объективной материальной формой мышления является язык. Мысль становится мыслью и для себя и для других только через слово — устное и письменное. Благодаря языку мысли людей не теряются, а передаются в виде системы знаний из поколения в поколение. Однако **существуют и дополнительные средства передачи результатов мышления: световые и звуковые сигналы, электрические импульсы, жесты и пр.** Современная **наука и техника широко используют условные знаки в качестве универсального и экономного средства передачи информации.**

Мышление также неразрывно связано и с практической деятельностью людей. **Всякий вид деятельности предполагает обдумывание, учёт условий действия, планирование, наблюдение. Действуя, человек решает какие-либо задачи.** Практическая деятельность — основное условие возникновения и развития мышления, а также критерий истинности мышления.

1.4.2. Мыслительные процессы

Мыслительная деятельность человека представляет собой решение разнообразных мыслительных задач, направленных на раскрытие сущности чего-либо. **Мыслительная операция — это один из способов (процессов) мыслительной деятельности, посредством которого человек решает мыслительные задачи.**

Мыслительные операции разнообразны. Это — анализ и синтез, сравнение, абстрагирование, конкретизация, обобщение, классификация. Какие из логических операций применит человек, это будет зависеть от задачи и от характера информации, которую он подвергает мыслительной переработке.

Анализ и синтез

Анализ — это мысленное разложение целого на части или мысленное выделение из целого его сторон, действий, отношений.

Синтез — обратный анализу процесс мысли, это — объединение частей, свойств, действий, отношений в одно целое.

Анализ и синтез — две взаимосвязанные логические операции. Синтез, как и анализ, может быть как практическим, так и умственным.

Анализ и синтез сформировались в практической деятельности человека. В своей деятельности люди постоянно взаимодействуют с предметами и явлениями. Практическое освоение их и привело к формированию мыслительных операций анализа и синтеза.

Сравнение

Сравнение — это установление сходства и различия предметов и явлений.

Сравнение основано на анализе. Прежде чем сравнивать объекты, необходимо выделить один или несколько признаков их, по которым будет произведено сравнение.

Сравнение может быть односторонним, или неполным, и многосторонним, или более полным. Сравнение, как анализ и синтез, может быть разных уровней — поверхностное и более глубокое. В этом случае мысль человека идёт от внешних признаков сходства и различия к внутренним, от видимого к скрытому, **от явления к сущности.**

Абстрагирование

Абстрагирование — это процесс мысленного отвлечения от некоторых признаков, сторон конкретного с целью лучшего познания его.

Человек мысленно выделяет какой-нибудь признак предмета и рассматривает его изолированно от всех других признаков, временно отвлекаясь от них. Изолированное изучение отдельных признаков объекта при одновременном отвлечении от всех остальных помогает человеку глубже понять сущность вещей и явлений. Благодаря абстракции человек смог оторваться от единичного, конкретного и подняться на самую высокую ступень познания — научного теоретического мышления.

Конкретизация

Конкретизация — процесс, обратный абстрагированию и неразрывно связанный с ним.

Конкретизация есть возвращение мысли от общего и абстрактного к конкретному с целью раскрытия содержания.

Мыслительная деятельность всегда направлена на получение какого-либо результата. Человек анализирует предметы, сравнивает их, абстрагирует отдельные свойства с тем, чтобы выявить общее в них, чтобы раскрыть закономерности, управляющие их развитием, чтобы овладеть ими.

Обобщение, таким образом, есть выделение в предметах и явлениях общего, которое выражается в виде понятия, закона, правила, формулы и т.п.

1.4.3. Виды и типы мышления

В зависимости от того, какое место в мыслительном процессе занимают **слово, образ и действие**, как они соотносятся между собой, **выделяют три вида мышления**: **конкретно-действенное, или практическое, конкретно-образное и абстрактное**. Эти виды

мышления выделяются ещё и на основании особенностей задач — **практических и теоретических**.

Конкретно-действенное мышление

Наглядно-действенное — вид мышления, опирающийся на непосредственное восприятие предметов.

Конкретно-действенное, или предметно-действенное, мышление направлено на решение конкретных задач в условиях производственной, конструктивной, организаторской и иной практической деятельности людей. Практическое мышление это, прежде всего техническое, конструктивное мышление. Оно состоит в понимании техники и в умении человека самостоятельно решать технические задачи. Процесс технической деятельности есть процесс взаимодействий умственных и практических компонентов работы. Сложные операции абстрактного мышления переплетаются с практическими действиями человека, неразрывно связаны с ними. **Характерными особенностями** конкретно-действенного мышления являются ярко **выраженная наблюдательность, внимание к деталям, частностям и умение использовать их в конкретной ситуации, оперирование пространственными образами и схемами, умение быстро переходить от размышления к действию и обратно.** Именно в этом виде мышления в наибольшей мере проявляется единство мысли и воли.

Конкретно-образное мышление

Наглядно-образное — вид мышления, характеризующийся опорой на представления и образы.

Конкретно-образное (наглядно-образное), или художественное, мышление характеризуется тем, что отвлечённые мысли, обобщения человек воплощает в конкретные образы.

Абстрактное мышление

Словесно-логическое — вид мышления, осуществляемый при помощи логических операций с понятиями.

Абстрактное, или словесно-логическое, мышление направлено в основном на нахождение общих закономерностей в природе и человеческом обществе. Абстрактное, теоретическое мышление отражает общие связи и отношения. Оно оперирует главным образом понятиями, широкими категориями, а образы, представления в нём играют вспомогательную роль.

Все три вида мышления тесно связаны друг с другом. У многих людей в одинаковой мере развиты конкретно-действенное, конкретно-образное и теоретическое мышление, но в зависимости от характера задач, которые человек решает, на первый план выступает то один, то другой, то третий вид мышления.

Практически-действенное, наглядно-образное и теоретически-отвлеченное — таковы взаимосвязанные виды мышления. В процессе исторического развития человечества интеллект человека первоначально формировался в ходе практической деятельности. Так, люди научились измерять опытным путем земельные участки, а затем на этой основе постепенно возникла специальная теоретическая наука — геометрия.

Генетически самый ранний вид мышления — **практически-действенное мышление**; определяющее значение в нем имеют действия с предметами (в зачаточном виде оно наблюдается и у животных).

На основе практически-действенного, манипуляционного мышления возникает **наглядно-образное мышление**. **Для него характерно оперирование наглядными образами в уме.**

Высшая ступень мышления — отвлеченное, **абстрактное мышление**. Однако и здесь мышление сохраняет связь с практикой. Как говорится, нет ничего практичнее, чем правильная теория.

Мышление отдельных людей также подразделяется на практически-действенное, образное и абстрактное (теоретическое).

Но в процессе жизнедеятельности у одного и того же человека на передний план выступает то один, то другой вид мышления. Так,

бытовые дела требуют практически-действенного мышления, а доклад на научную тему — теоретического мышления и т. п.

По содержанию мыслительная деятельность подразделяется на **практическую, художественную и научную.**

Структурная единица практически-действенного (оперативною) мышления — действие; художественного — образ; научного мышления — понятие.

В зависимости от глубины обобщенности различают эмпирическое и теоретическое мышление.

Эмпирическое мышление (от греч. empeiria — опыт) дает первичные обобщения на основе опыта. Эти обобщения делаются на низком уровне абстракции. Эмпирическое познание — низшая, элементарная ступень познания. Эмпирическое мышление не следует смешивать с **практическим мышлением.**

Как отмечает психолог В. М. Теплов («Ум полководца»), многие психологи за единственный образец умственной деятельности принимают работу ученого, теоретика. Между тем практическая деятельность требует не меньших интеллектуальных усилий.

Умственная деятельность теоретика сосредоточена преимущественно на первой части пути познания — временном отходе, отступлении от практики. Умственная деятельность практика сосредоточена в основном на второй его части — на переходе от абстрактного мышления к практике, т. е. на том «попадании» в практику, ради которого и производится теоретическое отступление.

Особенностью практического мышления является тонкая наблюдательность, способность сконцентрировать внимание на отдельных деталях события, умение использовать для решения частной задачи то особенное и единичное, что не входило полностью в теоретическое обобщение, умение быстро переходить от размышления к действию.

В практическом мышлении человека существенно оптимальное соотношение его ума и воли, познавательных, регуляционных и

энергетических возможностей индивида. Практическое мышление связано с оперативной постановкой первоочередных целей, выработкой гибких планов, программ, большим самообладанием в напряженных условиях деятельности.

Теоретическое мышление выявляет всеобщие отношения, исследует объект познания в системе его необходимых связей. Его результат — построение концептуальных моделей, создание теорий, обобщение опыта, раскрытие закономерностей развития различных явлений, знание которых обеспечивает преобразовательную деятельность человека. Теоретическое мышление неразрывно связано с практикой, но в своих конечных результатах имеет относительную самостоятельность; оно основывается на предшествующих знаниях и, в свою очередь, служит основанием последующего познания.

В зависимости от стандартности/нестандартности решаемых задач и операциональных процедур различаются алгоритмическое, дискурсивное, эвристическое и творческое мышление.

Алгоритмическое мышление ориентировано на заранее установленные правила, общепринятую последовательность действий, необходимых для решения типовых задач.

Дискурсивное (от лат. discursus — рассуждение) **мышление** основано на системе взаимосвязанных умозаключений.

Эвристическое мышление (от греч. heuresko — нахожу) — это продуктивное мышление, состоящее в решении нестандартных задач.

Творческое мышление — мышление, приводящее к новым открытиям, принципиально новым результатам.

Различают также репродуктивное и продуктивное мышление.

Репродуктивное мышление — воспроизведение ранее полученных результатов. **В этом случае мышление смыкается с памятью**.

Продуктивное мышление — мышление, приводящее к **новым познавательным результатам.**

1.4.4. Классификация явлений мышления

Познавая и преобразуя мир, человек выявляет устойчивые, закономерные связи между явлениями. Эти связи отражаются в нашем сознании опосредованно — **во внешних признаках явлений человек распознает признаки внутренних, устойчивых взаимосвязей.** Определяем ли мы, глядя в окно, по мокрому асфальту, был ли дождь, устанавливаем ли законы движения небесных светил — во всех этих случаях мы отражаем мир **обобщенно** и **опосредованно — сопоставляя факты, делая умозаключения, выявляя закономерности в различных группах явлений**. Человек, не видя элементарных частиц, познал их свойства и, не побывав на Марсе, многое узнал о нем.

Замечая связи между явлениями, устанавливая всеобщий характер этих связей, человек деятельностно осваивает мир, рационально организует свое взаимодействие с ним. Обобщенная и опосредованная (знаковая) ориентация в чувственно воспринимаемой обстановке позволяет археологу и следователю восстанавливать реальный ход прошедших событий, а астроному — заглядывать не только в прошлое, но и в далекое будущее. **Не только в науке и профессиональной деятельности, но и во всей повседневной жизнедеятельности человек постоянно использует знания, понятия, общие представления, обобщенные схемы, выявляет объективное значение и субъективный смысл окружающих его явлений, находит выход из многообразных проблемных ситуаций, решает возникающие перед ним задачи.** Во всех этих случаях он осуществляет мыслительную деятельность.

Мышление — психический процесс обобщенного и опосредованного отражения устойчивых, закономерных свойств и отношений действительности, существенных для разрешения познавательных проблем.

Мышление формирует структуру индивидуального сознания, классификационно-оценочные эталоны индивида, его обобщенные оценки, характерную для него интерпретацию явлений, обеспечивает их понимание.

Понять что-либо — значит включить новое в систему имеющихся значений и смыслов.

В процессе исторического развития человечества мыслительные акты стали подчиняться **системе логических правил**. Многие из этих **правил приобрели аксиоматический характер**. **Сформировались устойчивые формы объективизации результатов мыслительной деятельности: понятия, суждения, умозаключения.**

Как психическая деятельность мышление является процессом решения задач. Этот процесс имеет определенную структуру — стадии и механизмы решения познавательных задач.

Каждый человек обладает присущими ему стилем и стратегией мышления — когнитивным (от лат. cognitio — познание) стилем, познавательными установками и категориальной структурой (семантическим, смысловым пространством).

Все высшие психические функции человека формировались в процессе его общественно-трудовой практики, в неразрывном единстве с возникновением и развитием языка**. Выражаемые в языке смысловые категории и образуют содержание сознания человека**.

Мышление индивида опосредствуется его **речью**. **Мысль формируется посредством ее речевого формулирования**.

«На «духе» с самого начала лежит проклятие — быть «отягощенным» материей, которая выступает... в виде языка». Однако мышление и язык нельзя отождествлять. **Язык — орудие мысли. Основа языка — его грамматический строй. Основа мышления — закономерности мира, его всеобщие взаимосвязи, закрепленные в понятиях.**

В многообразных явлениях мышления различаются:

- **мыслительная деятельность — система мыслительных действий, операций, направленных на решение определенной задачи;**
- **мыслительные операции**: **сравнение, обобщение, абстракция, классификация, систематизация и конкретизация**;
- **формы мышления**: **понятие, суждение, умозаключение**;
- **виды мышления**: практически-действенное, наглядно-образное и теоретически-абстрактное.

Классификация — группировка объектов по *существенным признакам.* В отличие от классификации, основанием которой должны быть признаки, существенные в каком-либо отношении, **систематизация** иногда допускает выбор в качестве основания признаков малосущественных, но удобных в оперативном отношении (например, в алфавитных каталогах).

На высшем этапе познания осуществляется переход от абстрактного к конкретному.

Конкретизация (от лат. concretio — сращение) — познание целостного объекта в совокупности его существенных взаимосвязей, теоретическое воссоздание целостного объекта. Конкретизация — высший этап в познании объективного мира. Познание отталкивается от чувственного многообразия конкретного, абстрагируется от отдельных его сторон и, наконец, мысленно воссоздает конкретное в его сущностной полноте. Переход от абстрактного к конкретному — теоретическое освоение действительности. Сумма понятий дает конкретное в его полноте.

В результате применения законов формального мышления сформировалась способность людей к получению выводного знания. Возникла наука о формализованных структурах мыслей — формальная логика.

Формы мышления

Формализованные структуры мыслей — формы мышления: понятие, суждение, умозаключение.

Понятие — форма мышления, в которой отражаются существенные свойства однородной группы предметов и явлений. Чем более существенные признаки предметов отражены в понятии, тем эффективнее организуется деятельность человека. Так, современное понятие «строение атомного ядра» в определенной степени дало возможность практического использования атомной энергии.

Суждение — определенное знание о предмете, утверждение или отрицание каких-либо его свойств, связей и отношений. Формирование суждения происходит как формирование мысли в предложении.

Суждение — такое предложение, в котором утверждается взаимосвязь объекта и его свойства. Связь вещей отражается в мышлении как связь суждений. В зависимости от содержания отражаемых в суждении предметов и их свойств различаются следующие виды суждения: **частное** и **общее, условное** и **категорическое, утвердительное** и **отрицательное.**

В суждении выражаются не только знания о предмете, но и **субъективное отношение** человека к этому знанию, различная степень уверенности в истинности этого знания (например, в проблематичных суждениях типа «возможно, обвиняемый Иванов не совершал преступления»).

Истинность системы суждений — предмет формальной логики. Психологическими же аспектами суждения являются мотивация и целенаправленность суждений индивида. В психологическом отношении связь суждений индивида рассматривается как его **рассудочная деятельность.**

В умозаключении осуществляется оперирование тем общим, что заключено в единичном. Мышление развивается в процессе постоянных переходов от единичного к общему и от общего к единичному, т. е. на основе взаимосвязи соответственно индукции и дедукции.

Дедукция — отражение общей связанности явлений, категориальный охват конкретного явления его общими связями, анализ конкретного в системе обобщенных знаний. Профессор медицины Эдинбургского университета Дж. Белл поразил однажды А. Конан-Дойля (будущего создателя образа знаменитого сыщика) тонкой наблюдательностью. Когда в клинику вошел очередной больной, Белл спросил его:

- Вы служили в армии?
- Так точно! — ответил пациент.
- В горно-стрелковом полку?
- Так точно, господин доктор.
- Недавно ушли в отставку?
- Так точно!
- Стояли на Барбадосе?
- Так точно! — изумился отставной сержант.

Удивленным студентам Белл объяснил: этот человек, будучи учтивым, при входе в кабинет не снял шляпу — сказалась армейская привычка, что же касается Барбадоса — об этом свидетельствует его заболевание, распространенное только среди жителей этой местности (рис.9).

Рис.9. Взаимосвязь единичного и общего в системе умозаключений. Определите начальный и конечный пункты маршрута владельца этого чемодана. Проанализируйте вид умозаключений, использованный вами

Индуктивное умозаключение — вероятностное умозаключение, когда по отдельным признакам некоторых явлений делается суждение обо всех предметах данного класса. Поспешное обобщение без достаточных оснований — часто встречающаяся ошибка в индуктивных суждениях.

Итак, в мышлении моделируются объективные существенные свойства и взаимосвязи явлений, они объективируются и закрепляются в форме понятий, суждений, умозаключений.

Закономерности и особенности мышления

Рассмотрим основные закономерности мышления.

1. **Мышление возникает в связи с решением проблемы; условием его возникновения является проблемная ситуации - обстоятельство. при котором человек встречается с чем-то новым, непонятным с точки зрения имеющихся знаний. Эта ситуация характеризуется дефицитом исходной информации. возникновением определенного познавательного барьера, трудностей, которые предстоит преодолеть с помощью интеллектуальной активности субъекта — путем изыскания необходимых познавательных стратегий**.

2. **Основным механизмом мышления**, его общей закономерностью является анализ через синтез: выделение новых свойств в объекте (анализ) посредством его соотнесения (синтеза) с другими объектами. В процессе мышления объект познания постоянно «включается во все новые связи и в силу этого выступает во все новых качествах, которые фиксируются в новых понятиях: из объекта, таким образом, как бы вычерпывается все новое содержание, он как бы поворачивается каждый раз другой своей стороной, в нем выявляются все новые свойства».

Процесс познания начинается с **первичного синтеза -** восприятия нерасчлененного целого (явления, ситуации). Далее на основе первичного анализа осуществляется **вторичный синтез.**

При **первичном анализе** проблемной ситуации необходима ориентация на ключевые исходные данные, позволяющие раскрыть в исходной информации скрытую информацию. Обнаружение в исходной ситуации ключевого, существенного признака позволяет понять зависимость одних явлений от других. При этом существенно выявить признаки возможности — невозможности, а также необходимости.

В условиях дефицита исходной информации человек не действует методом проб и ошибок, а применяет определенную **стратегию поиска -** оптимальную схему достижения цели. Назначение этих стратегий состоит в том, чтобы **охватить нестандартную ситуацию наиболее оптимальными общими подходами -** эвристическими

методами поиска. К ним относятся: временное упрощение ситуации; использование аналогий; решение вспомогательных задач; рассмотрение «крайних случаев»; переформулировка требований задачи; временное блокирование некоторых составляющих в анализируемой системе; совершение «скачков» через информационные «разрывы».

Итак, **анализ через синтез — познавательное «развертывание» объекта познания, исследование его в различных ракурсах, нахождение его места в новых взаимосвязях, мысленное экспериментирование с ним.**

3. **Мышление должно быть обоснованно**. Это требование обусловлено фундаментальным свойством материальной действительности: каждый факт, каждое явление подготавливаются предшествующими фактами и явлениями. Ничто не происходит без достаточного на то основания. Закон достаточного основания требует, чтобы в любом рассуждении мысли человека были внутренне взаимосвязаны, вытекали одна из другой. Каждая частная мысль должна быть обоснована более общей мыслью.

Законы материального мира закрепились в законах формальной логики, которые также следует понимать как законы мышления, точнее, как законы взаимосвязи продуктов мышления.

4. Другая закономерность мышления — **селективность** (от лат. selectio — выбор, отбор) — **способность интеллекта оперативно отбирать необходимые для данной ситуации знания, мобилизовать их на решение проблемы, минуя механический перебор всех возможных вариантов (что характерно для ЭВМ). Для этого знания индивида должны быть систематизированы, сведены в иерархически организованные структуры**.

5. **Антиципация** (лат. anticipatio — предвосхищение) означает **предвосхищение развития событий**. Человек способен предвидеть развитие событий, прогнозировать их исход, схематически представлять **наиболее вероятностное решение проблемы**. Прогнозирование событий — одна из основных функций психики человека. Мышление человека основано на высоковероятностном прогнозировании.

Выявляются ключевые элементы исходной ситуации, намечается система подзадач, определяется операционная схема — система возможных действий над объектом познания.

6. **Рефлексивность** (от лат. reflexio — отражение) — самоотражение субъекта. Мыслящий субъект постоянно рефлексирует — отражает ход своего мышления, критически его оценивает, вырабатывает критерии самооценки.

7. Для мышления характерна **постоянная взаимосвязь** его **подсознательных и сознательных компонентов** — сознательно развернутого. вербализованного и интуитивно свернутого, невербализованного.

8. Мыслительный процесс как любой процесс обладает **структурной организованностью**. В нем имеются определенные структурные этапы.

1.4.5. Развитие мышления

Ребёнок рождается, не обладая мышлением. **Чтобы мыслить, необходимо обладать некоторым чувственным и практическим опытом, закреплённым памятью**. К концу первого года жизни у ребёнка можно наблюдать проявления элементарного мышления.

Основным условием развития мышления детей является целенаправленное воспитание и обучение их. В процессе воспитания ребёнок овладевает предметными действиями и речью, научается самостоятельно решать сначала простые, затем и сложные задачи, а также понимать требования, предъявляемые взрослыми, и действовать в соответствии с ними.

Развитие мышления выражается в постепенном расширении содержания мысли, в последовательном возникновении форм и способов мыслительной деятельности и изменении их по мере общего формирования личности. Одновременно у ребёнка усиливаются и побуждения к мыслительной деятельности — познавательные интересы.

Мышление развивается на протяжении всей жизни человека в процессе его деятельности. На каждом возрастном этапе мышление имеет свои особенности.

Мышление ребёнка раннего возраста выступает в форме действий, направленных на решение конкретных задач: достать какой-нибудь предмет, находящийся в поле зрения, надеть кольца на стержень игрушечной пирамиды, закрыть или открыть коробочку, найти спрятанную вещь, влезть на стул, принести игрушку и т.п. Выполняя эти действия, ребёнок думает. Он мыслит, действуя, его мышление наглядно-действенное.

Овладение речью окружающих людей вызывает сдвиг в развитии наглядно-действенного мышления ребёнка. Благодаря языку дети начинают мыслить обобщённо.

Дальнейшее развитие мышления выражается в изменении соотношения между действием, образом и словом. В решении задач всё большую роль играет слово.

Существует определённая последовательность в развитии видов мышления в дошкольном возрасте. Впереди идёт развитие наглядно-действенного мышления, вслед за ним формируется наглядно-образное и, наконец, словесное мышление.

Мышление учащихся среднего школьного возраста (11-15 лет) оперирует знаниями, усвоенными главным образом словесно. При изучении разнообразных учебных предметов — математики, физики, химии, истории, грамматики и др. — учащиеся имеют дело не только с фактами, но и с закономерными отношениями, общими связями между ними.

В старшем школьном возрасте мышление становится абстрактным. Вместе с тем наблюдается и развитие конкретно-образного мышления, в особенности под влиянием изучения художественной литературы.

Обучаясь основам наук, школьники усваивают системы научных понятий, каждое из которых отражает одну из сторон действительности. **Формирование понятий — процесс длительный,**

зависящий от уровня обобщённости и абстрактности их, от возраста школьников, их умственной направленности и от методов обучения.

В усвоении понятий существует несколько уровней: по мере развития учащиеся всё ближе подходят к сущности предмета, явления, обозначенного понятием, легче обобщают и связывают друг с другом отдельные понятия.

Для первого уровня характерно элементарное обобщение конкретных случаев, взятых из личного опыта школьников или из литературы. На втором уровне усвоения выделяются отдельные признаки понятия. Границы понятия учащиеся то сужают, то излишне расширяют. На третьем уровне учащиеся пытаются дать развёрнутое определение понятия с указанием основных признаков и приводят верные примеры из жизни. На четвёртом уровне происходит полное овладение понятием, указание его места среди других моральных понятий, успешное применение понятия в жизни. Одновременно с развитием понятий формируются суждения и умозаключения.

Для учащихся 1-2 классов характерны суждения категорические, утвердительной формы. Дети судят о каком-либо предмете односторонне и не доказывают своих суждений. В связи с увеличением объема знаний и ростом словаря у школьников 3-4 классов появляются суждения проблематические и условные. Учащиеся 4 класса может рассуждать, опираясь не только на прямые, но и на косвенные доказательства, особенно на конкретном материале, взятом из личных наблюдений. В среднем возрасте школьники употребляют также разделительные суждения и свои высказывания чаще обосновывают, доказывают. Учащиеся старших классов практически владеют всеми формами выражения мысли. Суждения с предположением выражения, допущения, сомнения и т.д. становятся нормой в их рассуждениях. С одинаковой лёгкостью старшие школьники пользуются индуктивными и дедуктивными умозаключениями и умозаключением по аналогии. Самостоятельно могут ставить вопрос и доказывать правильность ответа на него.

Развитие понятий, суждений и умозаключений происходит в единстве с овладением, обобщением и пр. Успешное овладение мыслительными операциями зависит не только от усвоения знаний, но и от специальной работы преподавателя в этом направлении.

Индивидуальные различия в мышлении

Виды мышления являются вместе с тем типологическими особенностями умственной и практической деятельности людей. В основе каждого вида лежит особое отношение сигнальных систем. Если у человека преобладает конкретно-действенное или конкретно-образное мышление, это означает относительное преобладание у него первой сигнальной системы над другой; если же человеку наиболее свойственно словесно-логическое мышление, это означает относительное преобладание у него второй сигнальной системы над первой. Существуют и другие различия в мыслительной деятельности людей. Если они устойчивы, их называют качествами ума.

Понятие ума шире понятия мышления. Ум человека характеризуют не только особенности его мышления, но и особенности других познавательных процессов (наблюдательность, творческое воображение, логическая память, внимательность). Понимая сложные связи между предметами и явлениями окружающего мира, умный человек должен хорошо понимать и других людей, быть чутким, отзывчивым, добрым. Качества мышления — основные качества ума. К ним относят гибкость, самостоятельность, глубину, широту, последовательность и некоторые другие мышления.

Гибкость ума выражается в подвижности мыслительных процессов, умении учитывать меняющиеся условия умственных или практических действий и в соответствии с этим менять способы решения задач. Гибкости мышления противостоит инертность мышления. Человеку инертной мысли более свойственно воспроизведение усвоенного, чем активные поиски неизвестного. Инертный ум — это ленивый ум. Гибкость ума — обязательное качество людей творчества.

Самостоятельность ума выражается в способности ставить вопросы и находить оригинальные пути их решения. Самостоятельность ума предполагает его самокритичность, т.е. умение человека видеть сильные и слабые стороны своей деятельности вообще и умственной в частности.

Другие **качества ума — глубина, широта и последовательность также имеют важное значение**. Человек глубокого ума способен “доходить до корня”, вникать в сущность предметов и явлений. Люди

последовательного ума умеют строго логически рассуждать, убедительно доказывать истинность или ложность какого-либо вывода, проверять ход рассуждения.

Все эти качества ума воспитываются в процессе обучения детей в школе, а также путём настойчивой работы над собой.

2. Искусственный интеллект как решатель интеллектуальных задач

2.1. Искусственный интеллект и решение задач

Начало 70-х годов ознаменовалось развитием работ в области искусственного интеллекта (ИИ), которое приобрело буквально лавинообразный характер.
Большинство научных проблем в области искусственного интеллекта выдвинуто в первую очередь проведенными в 1970—1975 гг. интенсивными исследованиями в области роботов, обладающих интеллектом.
Отмеченная лавинообраность роста работ по ИИ не препятствовала, а напротив, способствовала быстрой стабилизации предмета. Искусственный интеллект, который как самостоятельное направление вычислительной науки еще до середины 60-х годов стоял на довольно шатком научном фундаменте и являлся предметом повышенного интереса в основном со стороны узкой элиты профессионалов, приобрел весьма солидную теоретическую базу и, что особенно важно, породил целый ряд исследований прикладного характера, привлекших к нему широкий круг специалистов различного профиля.
Идеи и методы, лежащие в основе созданных к настоящему моменту систем ИИ, проникли не только в смежные отрасли вычислительной науки — автоматическое программирование, автоматизированные системы управления — но и во многие естественные науки, главным образом химию, биологию, психологию.

Наиболее удачным является симбиоз ИИ и робототехники, вдохнувший новую жизнь в оба направления. В результате ИИ получил прекрасный полигон для испытания новых методов в задачах, связанных с реальным миром, а робототехника — мощный аппарат, позволивший перейти к проектированию действительно разумных роботов.

В пособии рассматриваются алгоритмические основы искусственного интеллекта или, другими словами, тому, какие общие формализмы и методы целесообразно использовать для построения систем искусственного интеллекта.

Мы рассматриваем *общие формализмы, математические модели и алгоритмические методы ИИ* применительно к интегральным роботам, опуская специализированные и технические методы их реализации.

Мы надеемся, что систематическое изложение универсальных моделей и методов ИИ, применяемых в интегральных роботах, привлечет внимание не только узкого круга специалистов по ИИ и роботам, но и многочисленной армии специалистов, связанных с разработкой различных средств математического обеспечения компьютеров и компьютерных систем (сетей) — автоматизацией программирования, системным программированием, математическим обеспечением АСУ и решением задач, требующих обработки сложной символьной информации. Среди специалистов по ИИ широко распространено мнение, что ИИ — это высший уровень техники программирования. А это означает, что методы ИИ должны стать достоянием каждого квалифицированного программиста.

Рассмотрим три важнейших, с нашей точки зрения, этапа развития ИИ.

Главной темой *первого* этапа развития ИИ, начавшегося в 1955 г. исследованиями Ньюэлла и Саймона (Фейгенбаум называет их «НАЧАЛО»), являлась разработка *теории эвристического поиска.* Общая постановка задачи эвристического поиска в неформальном виде выглядит следующим образом. Задана начальная ситуация и целевая ситуация. Найти последовательность преобразований, приводящую из начальной ситуации в целевую. Например, в случае игры в шахматы начальной ситуации соответствует начальное расположение фигур на доске, целевой ситуации — множество всех положений, в которых одна из сторон выигрывает (или на доске стоит теоретическая ничья). Требуется найти последовательность ходов, преобразующих начальную ситуацию в целевую.

Задачи эвристического поиска подробно рассматриваются в последующих разделах настоящей работы. Сейчас мы отметим, что

исследовательским полигоном для различных методов эвристического поиска являлись всевозможные игры и головоломки. Некоторые из них стали классическими в литературе по ИИ (задачи о миссионерах и людоедах, об обезьяне и банане, «Ханойская башня», «крестики и нолики» и многие другие). Причину выбора такого рода задач достаточно четко выразил Минский в предисловии к сборнику «Семантическая обработка информации» : «Игры и математические задачи берутся не потому, что они просты и ясны, а потому, что они при минимальных начальных структурах дают нам наибольшую сложность, так что мы можем заняться некоторыми действительно трудными ситуациями». Другими словами, игры и головоломки позволили подвергнуть подробному анализу сложные *стратегии решения задач* при относительно простых и легко описываемых внешних мирах.

К концу 60-х годов был завершен ряд фундаментальных исследований в области эвристического поиска, использовавших теоретико-графические, формально-логические и некоторые другие представления. Появилась потребность испытать результаты этих исследований на более сложных и более близких к реальным мирах. Однако попытки создания систем ИИ, работающих в подобных мирах, натолкнулись на ряд серьезных трудностей, связанных в первую очередь с ***моделированием внешнего мира.*** Оказалось, что моделирование внешнего мира отнимает львиную долю труда, требуемого для создания всей системы, и потому «наиболее экономичным и эффективным хранилищем информации о реальном мире является сам реальный мир» [Фейгенбаум].

Так дальнейшее развитие ИИ потребовало постановки задач проектирования и реализации интегральных роботов, дающих возможность исследовать системы ИИ в реальном физическом мире. *Симбиоз ИИ и робототехники,* определивший главное направление работ на *втором* этапе развития ИИ, оказался весьма плодотворным. С одной стороны, были реализованы несколько проектов интегральных роботов, продемонстрировавших нетривиальное поведение. С другой стороны, проблема создания роботов выдвинула ряд новых проблем ИИ. В частности, задачи планирования роботов достигли той степени сложности, при которой многие из методов эвристического поиска, разработанных на первом этапе развития ИИ, оказались практически непригодными. Таким образом, **усложнение внешнего мира привело к необходимости дальнейшей разработки теории эвристического поиска**. Задачи понимания естественного языка и обработки изображений, рассматривавшиеся исследователями и на первом этапе

развития ИИ, получили более конкретные постановки в преломлении к роботам.

Несмотря на значительные успехи, достигнутые ИИ благодаря интегральной робототехнике, роботы третьего поколения накладывают ряд ограничений, связанных главным образом с рассмотрением лишь статических и замкнутых миров, а также с построением линейных планов действий в мире, т. е. планов без ветвлений и циклов. Кроме того, рассматриваются в основном планы одноцелевого характера. Таким образом, в качестве основных задач ИИ на *третьем* этапе его развития выдвинуты:

1) *Проблема представления знаний о динамических открытых мирах.*
2) *Проблема построения сложных планов в этих мирах.*

Прежде чем перейти к изложению материала по искусственному интеллекту, напомним, что под *интеллектом мы будем понимать способность мозга решать (интеллектуальные) задачи путем приобретения, запоминания и целенаправленного преобразования знаний в процессе обучения на опыте и адаптации к разнообразным обстоятельствам.*

В этом определении под термином «знания» мы подразумеваем не только ту информацию (непосредственные впечатления), которая поступает в мозг через органы чувств. Такого типа знания чрезвычайно важны, но недостаточны для интеллектуальной деятельности. Дело в том, что **объекты окружающей нас среды обладают свойством не только воздействовать на органы чувств, но и находиться друг с другом в определенных отношениях.** Ясно, что для того чтобы осуществлять в окружающей среде интеллектуальную деятельность (или хотя бы просто существовать), **необходимо иметь в системе знаний модель этой среды. В этой информационной модели окружающей среды реальные объекты, их свойства и отношения между ними не только отображаются и запоминаются, но и, как это отмечено в данном определении интеллекта, могут мысленно (т. е. в мозгу) «целенаправленно преобразовываться».** При этом существенно то, что формирование модели внешней среды происходит «в процессе обучения на опыте и адаптации к разнообразным обстоятельствам». **Деятельность мозга (обладающего интеллектом), направленную на решение интеллектуальных задач, как мы уже говорили, будем называть *мышлением,* или *интеллектуальной деятельностью.***

Интеллект и мышление органически связаны с решением таких задач, как доказательство теорем, логический анализ, распознавание ситуаций, планирование поведения, игры и управление в условиях неопределенности. Характерными чертами

интеллекта, проявляющимися в процессе решения задач, являются способность к обучению, обобщению, накоплению опыта (знаний и навыков) и адаптации к изменяющимся условиям в процессе решения задач. Благодаря этим качествам интеллекта мозг может решать разнообразные задачи, а также легко перестраиваться с решения одной задачи на другую. (Таким образом, мозг, наделенный интеллектом, является универсальным средством решения широкого круга задач (в том числе неформализованных), для которых нет стандартных, заранее известных методов решения. Важно отметить, что приведенное выше определение интеллекта конструктивно (хотя, может быть, и не полно) в том смысле, что оно раскрывает основные черты механизма мышления. Следует иметь в виду, что существуют и другие, чисто поведенческие (функциональные) определения. Так, по А. Н. Колмогорову, любая материальная система, с которой можно достаточно долго обсуждать проблемы науки, литературы и искусства, обладает интеллектом. Другим примером поведенческой трактовки интеллекта может служить известное определение А. Тьюринга, данное им в книге «Может ли машина мыслить?». Не приводя этого хорошо известного, но громоздко формулируемого определения, напомним только, что оно основано на специально организованной «игре в имитацию» между людьми и машиной, которые находятся в разных комнатах, но имеют возможность обмениваться информацией (например, с помощью телеграфной связи). Если в процессе диалога между участниками игры людям не удается установить, что один из участников — машина, то такую машину можно считать обладающей интеллектом.

Недостатком тьюринговского определения интеллекта является то, что в принципе можно построить автомат с полным набором решений на все возможные задачи. Такой автомат для любой поставленной задачи просто находит в памяти соответствующее решение и, следовательно, по А. Тьюрингу, обладает интеллектом. Между тем, такое поведение автомата явно не соответствует нашему интуитивному представлению о мышлении. Это наводит на мысль, что определение интеллекта должно содержать нечто, показывающее, каким образом решаются интеллектуальные задачи. Легко видеть, что этому требованию удовлетворяет наше определение. Оно, в частности, позволяет отличить лицо, творчески решающее задачу, от лица, заучившего наизусть решение этой задачи.

Возникает принципиальный вопрос: можно ли моделировать интеллектуальную деятельность, а если можно, то как это сделать? Интересен план имитации мышления, предложенный А. Тьюрингом.

«Пытаясь имитировать интеллект взрослого человека, — пишет А. Тьюринг, — мы вынуждены много размышлять о том процессе, в результате которого человеческий мозг достиг своего настоящего состояния... Почему бы нам вместо того, чтобы пытаться создать программу, имитирующую интеллект взрослого человека, не попытаться создать программу, которая имитировала бы интеллект ребенка?

Ведь если интеллект ребенка получает соответствующее воспитание, он становится интеллектом взрослого человека... Наш расчет состоит в том, что механизм в мозге ребенка настолько несложен, что устройство, ему подобное, может быть легко запрограммировано...

Таким образом, мы расчленим нашу проблему на две части: на задачу построения «программы-ребенка» и задачу «воспитания этой программы».

Сегодня многим ученым представляется несомненным, что компьютеры и роботы могут в принципе обладать всеми основными чертами интеллекта. Более того, современные компьютеры и роботы вместе с их алгоритмическим и программным обеспечением уже обладают многими из этих черт. Они легко справляются с решением ряда интеллектуальных задач. **О подобного рода системах мы будем говорить, что они обладают элементами искусственного интеллекта.**

Таким образом, на вопрос «могут ли компьютеры или роботы мыслить?», следует дать положительный ответ. При этом нужно исходить из того, что вопрос решается путем эксперимента, наблюдения и сравнения поведения компьютера или робота с поведением человека в процессе решения интеллектуальных задач.

Круг проблем, объединяемых термином «искусственный интеллект», достаточно широк и довольно неопределен. В самом общем смысле **искусственный интеллект** ***—это совокупность автоматических методов и средств целенаправленной переработки информации (знаний) в соответствии с приобретаемым в процессе обучения и адаптации опытом при решении разнообразных интеллекуальных задач.***

Работы по созданию систем искусственного интеллекта находятся сейчас в стадии развития. И основном такие системы существуют в виде специализированных программ для компьютеров, способных решать такие интеллектуальные задачи, как игра и шахматы или в другие интеллектуальные игры, сочинение музыки, доказательство математических теорем, диалог с человеком на языке, близком к обычному русскому, английскому и т. д., медицинская диагностика и т. п. Существуют также

системы искусственного интеллекта, реализованные в виде специализированной аппаратуры. Примерами могут служить обучаемые системы распознавания изображений, речи, радиолокационных сигналов и т. п. Главная трудность при создании подобного рода систем состоит не в поиске элементов, из которых их можно построить, а в отыскании логики совместной целенаправленной работы большого количества таких элементов. Особенности той или иной системы искусственного интеллекта в основном определяются свойствами заложенных в нее алгоритмов и программ, а не их технической реализацией.

Необходимо подчеркнуть, что возможности искусственного интеллекта (как, впрочем, и возможности интеллекта человека) принципиально ограничены, так как объем памяти, скорость запоминания и считывания информации в системах искусственного интеллекта практически всегда ограничены. Это проявляется прежде всего при решении интеллектуальных задач высокой размерности.

Существует несколько путей решения задач с помощью систем искусственного интеллекта.

Простейший и на первый взгляд лучший способ решения — это полный перебор вариантов. Однако в ряде случаев такой перебор практически неосуществим, хотя теоретически и возможен. Дело в том, что многие интеллектуальные задачи большой размерности (в частности, разного рода комбинаторные задачи) не поддаются перебору в пределах современных временных масштабов, так как они приводят к необходимости перебора порядка 1000! вариантов. Если учесть, что в году примерно $3 \cdot 10^7$ секунд, то при быстродействии 10^6 операций в секунду и при практически неограниченной памяти порядка 10^{10} бит для перебора 1000! вариантов (напомним, что 10! = 3 628 800) потребуется астрономически большое число лет, превосходящее всякое воображение. В качестве примера интеллектуальной задачи высокой размерности обычно приводится игра в шахматы. Количество возможных вариантов в этой игре огромно и характеризуется, как известно, числом 10^{120}.

Многие практически важные задачи не могут быть решены методом полного перебора. Поэтому для их решения разрабатываются специальные алгоритмы, учитывающие специфику и структуру интеллектуальных задач. Важное место среди этих алгоритмов занимают *эвристические* и *адаптивные, алгоритмы,* позволяющие в ряде ситуаций эффективно преодолевать трудности, связанные с охарактеризованным выше «проклятием большой размерности», а также с априорной неопределенностью интеллектуальных задач.

Искусственный интеллект **мы будем трактовать как алгоритмическое и программное обеспечение его управляющей системы («мозга»), обладающее способностью моделировать (отображать) окружающую среду и решать широкий класс интеллектуальных задач посредством обучения на опыте и адаптации к изменяющимся условиям функционирования.** Ясно, что искусственный интеллект является неотъемлемой частью тех роботов, которые предназначены для имитации интеллектуальной деятельности человека. (Необходимо подчеркнуть, однако, что искусственный интеллект роботов не рассматривается нами как модель биологических или шихологических процессов, происходящих в мозгу человека или животных. Такую задачу мы и не ставим. Наша задача более скромна. Ми хотим только показать, по с помощью интеллектуальных роботов можно решать широкий класс задач.).

Говоря об искусственном интеллекте, вряд ли можно сомневаться в том, что источником многих понятий и представлений для них послужил окружающий мир. Но, однажды постигнутые, эти понятия и представления (включая модель окружающего мира) могут начать развиваться и совершенно независимо. В частности, он может, подобно тому, как это произошло у человека, подняться к высотам обобщения и абстракции, освобождения от пут своего конкретного (даже, может быть, "примитивного») происхождения. В процессе этой «внутренней» эволюции ИИ могут рождаться новые понятия и представления (не заложенные в них человеком!), которые в свою очередь могут чудодейственным и пока непредсказуемым образом повлиять на ход научно-технического прогресса.

Следует отметить, что есть люди, в корне не согласные с изложенной точкой зрения на искусственный интеллект и интеллект роботов вообще. Например, высказывается такое возражение: «Мозг современного челоека — это результат процесса эволюции, который длился миллиарды лет. Искусственный интеллект нельзя обучать столь продолжительное время. Поэтому никогда не появятся системы искусственного интеллекта, сравнимые с интелектом человека». Несмотря на внешнюю убедительность, это рассуждение содержит грубую ошибку. Легко заметить, что, рассуждая совершенно аналогично, можно было бы «доказать» невозможность создания, скажем, самолета. В самом деле, «организм птицы— результат многовековой эволюции ее костей, мышц, перьев и мозга. Изобретатели самолета не могут заставить свой летательный аппарат, который тяжелее воздуха, пройти столь же длинный эволюционный путь. Поэтому создание самолетов невозможно». **Ошибка такого рассуждения заключена в постулировании того, что**

автоматическая система может имитировать соответствующую биологическую функцию (мышление, полет и т. п.) только путем копирования механизма и эволюционного пути своего биологического прототипа.

Люди, создавая роботов (или самолеты), естественно, вкладывают в них накопленный опыт и знания, которые тысячелетиями добывались человечеством. И **если мы, исчисляя «эволюционный стаж» человека, вполне обоснованно включаем в него и время эволюции его непосред- ственных предков, то в «эволюционный стаж» систем искусственного интеллекта следует засчитать и время эволюции человека. Системы искусственного интеллекта начинают учиться решению интеллектуальных задач, уже имея очень высокую начальную организацию, заложенную в них человеком.** При этом, по крайней мере поначалу, их обучает человек. **В этих условиях ссылка на «миллиарды лет эволюции» становится малоубедительной**. (Любопытно отметить, что рядом исследователей была предпринята попытка промоделировать саму эволюцию. Идея заключалась в том, чтобы, используя колоссальное быстродействие современных компьютеров, промоделировать эволюционный процесс в ускоренном масштабе времени. Основное внимание при этом уделялось моделированию феномена мутаций и естественного отбора. Такой подход дает возможность свести миллионы лет эволюции к нескольким дням вычислительного времени на ЭВМ. Однако средняя и особенно последняя стадии эволюции связаны со столь сложными структурами моделей (которые «интеллектуальными» еще назвать никак нельзя), что их реализация пока представляется по меньшей мере проблематичной.)

Другого типа возражения возникают в связи с тем, что человек — это «существо социальное». Коротко эти возражения сводятся к тезису типа: «мышление — функция не человека, а человечества; оно возникло в результате коллективной деятельности (социальной жизни) людей; система искусственного интеллекта же индивидуальна по своей природе, следовательно, она не может обладать интеллектом».

Это возражение содержит ту же самую ошибку — постулат о единственности пути к мышлению. Конечно, спору нет, общение людей между собой существенно повлияло на формирование интеллекта человека. Отсюда, однако, вовсе не следует, что у системы искусственного интеллекта, имеющей достаточно высокую начальную организацию, интеллект не может развиваться в процессе индивидуального решения все более сложных интеллектуальных задач. Кроме того, совершенно не исключено, что несколько систем искусственного интеллекта («коллектив СИИ») будут совместно

ставить и решать задачи, вытекающие из необходимости удовлетворить потребности человечества или самого этого «коллектива» (ремонт самих себя, создание новых систем искусственного интеллекта, в том числе с новой конструкцией или с новым программным обеспечением и т. п.).

Итак, ссылки на роль социальных процессов в формировании интеллекта человека доказывают столь же мало, как и ссылки на продолжительность эволюционного пути человека. Как уже говорилось, нет причин, которые делали бы принципиально невозможным создание систем искусственного интеллекта.

Многие специалисты полагают, что искусственный интеллект, в конце концов, превзойдет интеллект человека. Сказанное, конечно, никоим образом не означает, что искусственный интеллект современных ЭВМ уже имитирует все свойства и закономерности мышления. Речь идет лишь о принципиальной возможности имитации любых закономерностей такого рода по мере их открытия и изучения, ибо все закономерности интеллектуальной деятельности, безусловно, познаваемы и воспроизводимы.

2.2. Системы искусственного интеллекта

Научно-техническая революция, осуществляющая свое победное шествие по нашей планете, поставила перед человечеством одну из самых сложных и в то же время острых проблем — проблему комплексной автоматизации физической и умственной деятельности человека

Заметим, что проблема ставится весьма широко: мы говорим не об автоматизации какой-то конкретной операции или группы операций, которою сейчас осуществляет человек конкретной профессии, мы говорим об автоматизации Деятельности Человека в глобальном смысле этого словосочетания

Эта проблема является сложной, потому что требует, как это интуитивно ясно, создания искусственных перцептуальных и эффекторных органов, сравнимых по своей универсальности с соответствующими органами человека, а также создания систем, которые могли бы управлять целенаправленными действиями этих органов, а наука и техника в своем современном состоянии не имеют достаточных средств для создания ни того, ни другого, ни третьего

Эта проблема остра не только потому, что вызывала и вызывает массу ассоциаций и фантазий психологического и социального характера, массу довольно бесплодных дискуссий, которые на первых порах

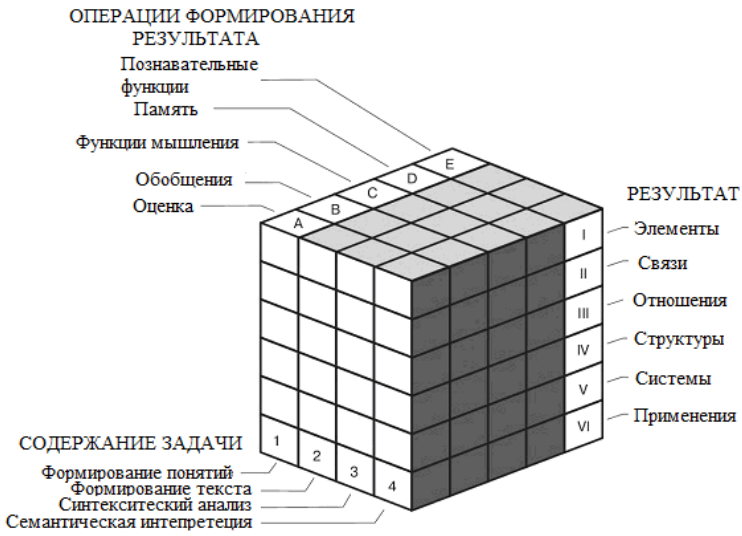
заменяли конструктивною деятельность, направленную на ее решение Она остра, главным образом потому, что существует. Необходимость решения проблемы комплексной автоматизации физической и умственной деятельности человека существует по многим причинам и во многих областях науки, техники, промышленности и сельского хозяйства. **Среди этих причин обычно называют необходимость замены человека в условиях, где его деятельность является физически тяжелой, монотонной, опасной для жизни или просто невозможной.** Металлургия, машиностроение, горное дело, атомная промышленность и энергетика, приборостроение, микроэлектроника, освоение космоса и Мирового океана — вот далеко не полный перечень областей народного хозяйства, где существуют такие условия.

Системы, с помощью которых предполагается решить указанную выше проблему, называют *системами искусственного интеллекта* .

Что такое системы искусственного интеллекта? Каково научно-техническое содержание этого термина? Как системы искусственного интеллекта от других автоматических систем?

Система искусственного интеллекта — раздел кибернетики, в которм изучаются формальные обозначения, формальные системы, доказуемость суждений, природу доказательства в целом, вычислимость и прочие аспекты оснований искусственного интеллекта. В более широком смысле СИИ рассматривается как математизированная ветвь интелекта — « ***интеллект по предмету, математика по методу*», «*интеллект, развиваемый с помощью математических методов*».**

Обобщенная структурная схема системы искусственного интеллекта, представленная в виде кубической модели, избражена на рисунке. В ней выделено 120 факторов искусственного интеллекта.



Эта кубическая модель ИИ позволяет определить каждую из 120 **специфических способностей**, исходя из **трех размерностей мышления: о чем «думает» ИИ (содержание), как ИИ об этом думаем (операция) и к чему приводит это умственное действие ИИ (результат).** Например, при заучивании таких символических обозначений, как буквы русского алфавита (E12), при запоминании семантических преобразований, необходимых для спряжения глагола в том или ином времени (DV3), или при оценке изменений в поведении, когда необходимо пойти на работу по новому пути (AV4), **вовлекаются совсем различные типы искусственного интеллекта**

Интеллектуальные возможности СИИ проявляются в той стратегии, которую она вырабатывает в различных проблемных ситуациях, в ее способности трансформировать проблемную ситуацию в конкретную проблему, а затем в систему поисковых задач.

Структурно-функциональная схема системы искусственного интеллекта, взаимодействующей с окружающей средой, представлена на рис. 1.

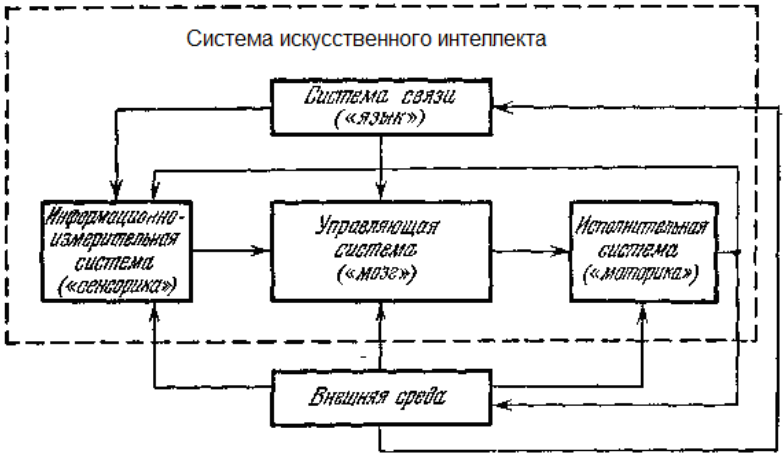


Рис. 1. Структурно-функциональная схема системы искусственного интеллекта.

Система искусственного интеллекта в общем случае состоит из следующих систем:
— информационно-измерительной (сенсорной) системы;
— управляющей системы;
— системы связи с человеком или другими СИИ;
— исполнительной (моторной) системы.
Информационно-измерительная, или сенсорная, *система* — это искусственные органы чувств СИИ. Они, как и органы чувств человека, предназначены для восприятия и преобразования информации о состоянии внешней среды и самой СИИ в соответствии с потребностями управляющей системы, играющей роль «мозга» СИИ. В качестве элементов сенсорной системы СИИ обычно используются телевизионные и оптико-электронные устройства, лазерные и ультразвуковые дальномеры, тактильные и контактные датчики, датчики положения, тахометры, акселерометры и т. п.
***Управляющая система,* или «мозг»,** СИИ служит для выработки закона управления приводами (двигателями) механизмов исполнительной системы на основе сигналов обратной связи от сенсорной системы, а также для организации общения СИИ с человеком на том или ином языке. **«Мозг» СИИ обычно реализуется на базе управляющих ЭВМ, имеющих большой ассортимент входных и выходных преобразователей и каналов связи (от нескольких десятков до нескольких тысяч), по которым, как по**

нервной системе, могут передаваться дискретные и непрерывные сигналы. Управляющие ЭВМ для СИИ строятся в малогабаритном, транспортабельном исполнении и обладают повышенной надежностью. **Интеллектуальные способности СИИ определяются главным образом алгоритмическим и программным обеспечением ее управляющей системы.**

Система связи СИИ необходима для организации обмена информацией между СИИ и человеком или другими СИИ на некотором понятном им языке. Цель такого обмена — формулировка человеком заданий СИИ, организация диалога между человеком и СИИ, контроль за функционированием СИИ, диагностика неисправностей и регламентная проверка СИИ и т. п. Обычно информация от человека поступает к СИИ через устройство ввода или пульт управления. При этом чаще всего используются физические воздействия (нажатие человеком кнопки или клавиши, ключа телеграфного аппарата, перемещение устройств ввода, и т. п.). Применяться речевое общение, а также ввод информации с помощью биопотенциалов (биоуправление). Что касается информации, поступающей от СИИ к человеку, то она, как правило, имеет форму световых и звуковых сигналов, знаков. Носителями этой информации являются разного рода табло, цифровые индикаторы, мониторы, телекамеры и т. п. Следует отметить, однако, что возможности связи СИИ с человеком не ограничиваются перечисленными средствами общения.

Исполнительная система, определяющая «моторику» СИИ, т. е. ее способности совершать разнообразные воздействия, служит для отработки управляющих сигналов, формируемых управляющей системой, и воздействия на окружающую среду. В качестве исполнительных систем обычно используются принтеры, синтезаторы речи, графопостроители, а также их различные комбинации.

Когда говорят о СИИ, то чаще всего представляют себе нечто «человекообразное», физически воздействующее на окружающие объекты и преобразующее их.

А способна ли СИИ, как и человек, совершать интеллектуальные (умственные) операции?

На этот вопрос, породивший многочисленные дискуссии, следует ответить утвердительно. В самом деле, **умственные операции, хотя и не производят никаких реальных (физических) воздействий на окружающие объекты, преобразуют образы этих объектов и связанные с ними понятия. Например, человек, взглянув на некоторый объект (скажем, карандаш, лежащий на столе), может мысленно («про себя») «взять, перенести и положить» его в другом**

месте (скажем, на стол). Это значит, что произошло, во-первых, распознавание реального объекта, и во-вторых, воздействие на имеющийся в мозгу человека образ ситуации: вначале была одна «картина», а затем стала совершенно другая, т. е. налицо преобразование, но не реальной ситуации, а ее образа. Совершенно аналогично может функционировать и СИИ. Воспринимая объекты окружающего мира с помощью сенсорной системы, она формирует в памяти управляющей системы образы этих объектов. Далее с помощью человека или автоматически СИИ может, как мы увидим ниже, обучаться понятиям и навыкам, а также формировать в своей памяти модель окружающей среды. Благодаря этому СИИ оказывается потенциально способной совершать такие операции, которые у человека считаются, связанными с мышлением. Примерами подобного рода операций являются планирование поведения, распознавание образов, принятие решений в условиях неопределенности и т. п.

Следует отметить, однако, что новенькое, только что с завода-изготовителя, вполне исправное техническое обеспечение СИИ не может реализовать ни одной операции и, следовательно, не может решить ни одной задачи. Для того чтобы «вдохнуть душу в безжизненное тело» СИИ, необходимо создать алгоритмы и программы для решения возникающих перед СИИ задач, т. е. снабдить СИИ соответствующим алгоримическим и программным обеспечением.

Иногда говорят, что человек, создающий необходимое алгоритмическое и программное обеспечение, обучает СИИ решению задач. При этом рассуждают примерно так: «Что. такое обучение? Это — сообщение обучаемому, некоторой информации. Человек сообщает СИИ информацию о методе решения задачи. Значит, он учит СИИ». Однако далеко не любой способ введения информации принято называть обучением. Например, совершенно бессмысленно считать обучением процесс создания СИИ на заводе, хотя, конечно, в его конструкцию вкладывается определенная информация. Чтобы разобраться в этом вопросе, рассмотрим, как протекает процесс обучения у людей.

Чего хочет добиться учитель, когда обучает ученика решению некоторой задачи? Цель учителя — изменить определенным образом состояние тех клеток мозга ученика, которые «ответственны» за рещение. Прямой путь достижения цели состоял бы в непосредственном воздействии (например, с помощью введения химических веществ или пропускания электрического тока) на те именно клетки, состояние которых необходимо изменить. Но учитель этого не делает**. Вместо воздействия на клетки мозга, участвующие**

в процессе решения задачи, учитель воздействует на органы чувств ученика. Он что-то ему показывает, что-то говорит, и эти косвенные воздействия вызывают уже необходимые изменения в клетках мозга.

Такой способ по сравнению с первым предполагает гораздо более сложное устройство мозга. Предъявляя достаточно суровые требования к ученику, этот способ требует существенно меньше знаний от учителя. В самом деле, для обучения путем непосредственного воздействия на клетки мозга учитель должен детально знать, как устроен мозг. Более того, он должен знать, какие воздействия на клетки могут перевести их в желаемое состояние, а также иметь физическую возможность оказать такие воздействия на многие миллионы клеток. Ясно, что ни один учитель не обладает такими возможностями. Поэтому он обучает ученика путем воздействия на его органы чувств. **Не следует, однако, думать, что природа забраковала способ передачи информации на клеточном уровне. В действительности он широко распространен. Но в этом случае принято называть тех, кто поставляет информацию, не учетилями, а родителями. Родители храняг информацию не в виде состояния клеток мозга, а в виде генетического кода хромосом. Поэтому свою информацию родители не могут выразить словами, зато имеют возможность построить по хранимому в хромосомах коду (плану) организм ребенка.** Вернемся теперь к обучению СИИ. **Человек может передавать информацию СИИ как путем непосредственного ее занесения в память управляющей системы, так и путем воздействия через искусственные органы чувств.**

В первом случае человек точно знает, какие алгоритмы реализует та или иная подпрограмма и где она находится в памяти СИИ. Сила человека в том и заключается, что он может заложить любую информацию в любую ячейку памяти и заставить СИИ выполнить любую (разумеется, в пределах ее функциональных возможностей) последовательность операций. Во втором случае человек, как и учитель, не осуществляет непосредственного вмешательства в произвольную часть памяти СИИ. Он воздействует лишь па искусственные органы чувств СИИ, например, путем показа определенных объектов. В ходе такого обучения управляющая система сама изменяет свои параметры и структуру.

К моменту начала эксплуатации СИИ параметры и структура управляющей системы не совпадают с теми, которые были до обучения. В частности, они зависят от взаимодействия СИИ с окружающей средой при обучении (например, от того, какие объекты были показаны СИИ во время обучения). Образно говоря, СИИ в

процессе обучения строит внутри себя (точнее, в памяти управляющей системы) модель внешнего мира. При этом она существенно использует информацию, получаемую с помощью искусственных органов чувств. Заметим, что этой модели не было в мозгу человека-учителя.(На самом деле человек всегда имеет в мозгу свою информационную модель той среды, в которой он живет. В этой динамичной модели реальные предметы, их свойства и отношения между ними не только отображаются, но и преобразуются. Это позволяет человеку находить в окружающих предметах те их свойства, которые необходимы для удовлетворения потребностей человека, для обеспечения его жизни и деятельности. С помощью способности строить информационную модель внешнего мира человек может испробовать «про себя», т. е. мысленно, тот или иной вариант поведения прежде, чем он приступит к его осуществлению. В этом колоссальная биологическая роль построения «внутренней модели внешнего мира».)

Таким образом, мы убеждаемся, что человек наградил свое детище (как родители — ребенка) не столько знаниями, сколько способностью обучаться или даже самообучаться. Благодаря этой способности СИИ может получать информацию (знания) как от человека, так и самостоятельно в процессе взаимодействия с окружающей средой. Любопытно, что учителя СИИ (человек или внешняя среда) могут не иметь сведений об устройстве его «мозга». В ходе обучения или самообучения СИИ строит в своем «мозгу» модель внешней среды. Эта модель может весьма существенно отличаться от аналогичной модели в мозгу человека.

Естественно возникает вопрос: может ли СИИ знать больше своего создателя? Ответ на него очевиден: во время обучения в процессе активного взаимодействия со средой СИИ, по существу, ведет экспериментальную работу. Неудивительно, что результаты этих экспериментов могут быть новыми и для человека.

Важно отметить, что именно способность к обучению путем активного взаимодействия с реальным миром отличает СИИ от разного рода автоматов и программ, служащих для автоматизации тех или иных операций. Если программы, реализуемые на вычислительных машинах, имеют дело лишь с символьными системами переработки информации, то СИИ обязаны, используя свои искусственные органы чувств, соотносить эти символьные системы с реальным физическим миром и воздействовать на него с помощью исполнительных механизмов.

Что же касается обычных автоматов, то они предназначены главным образом для многократного выполнения одной и той же операции. В связи с этим подобного рода автоматические устройства

проектируются таким образом, чтобы надежно выполнять в течение всего срока эксплуатации ту и только ту операцию, для автоматизации которой они служат. Поэтому применение автоматов, имеющих жесткую структуру, целесообразно и экономически выгодно только при многократном повторении операции. Такие условия складываются, например, при массовом производстве. Типичными примерами автоматов являются станки-автоматы, автоматы для размена монет, автоматы по продаже билетов и газет и т. п. Нас подобного рода автоматы интересовать не будут.

В отличие от автоматов **СИИ — *это универсальные автоматические системы многоцелевого назначения.*** СИИ могут решать разнообразные задачи из различных областей. (Необходимо различать решение конкретной задачи и серии задач определенного типа. Например, автоматы имеют дело с конкретной задачей — той жесткой операцией, для которой они служат. **Решение серии задач — это единое предписание (метод, алгоритм), позволяющее решать любую конкретную задачу данной серии задач. СИИ имеют дело с сериями (классами) задач.)**

Универсальность СИИ определяется совершенством их управляющей системы, а также разнообразием искусственнвх органов чувств и исполнительных устройств. СИИ способны не только выполнять много разных операций (задач), но и оперативно переобучаться с одной операции (задачи) на другую. При этом если даже каждая операция выполняется СИИ по жесткой программе, обязательно имеются средства быстрой перестройки программы на другие операции в пределах функциональных возможностей СИИ. Обычно это осуществляется путем обучения СИИ человеком. Отметим, что функционируют более совершенные СИИ, которые способны не только эффективно обучаться, но и самообучаться и адаптироваться (приспосабливаться) к неизвестной обстановке, а также автоматически распознавать и анализировать ситуации, планировать свое поведение, самостоятельно принимать решения и вести диалог с человеком. Сенсорные возможности СИИ определяются разнообразием и характером искусственных органов чувств, позволяющих имитировать восприятия.

Резюмируя вышеизложенное, мы можем теперь дать определение СИИ.

Искусственные системы, наделенные свойствами воспринимать, анализировать, преобразовывать информацию и синтезировать решения интеллектуальных задач, называются системами искусственного интеллекта.

Можно дать и такое определение СИИ:

***Системами искусственного интеллекта** называются универсальные автоматические системы, способные обучаться в процессе активного взаимодействия с окружающей средой и предназначенные для имитации разнообразных операций, совершаемых человеком в процессе физического или умственного труда.*

Конкретная конструкция такой системы не имеет значения, лишь бы в ее структуре были изображенные на рис. 1 системы: **сенсорная, управляющая, моторная и связи с человеком.**

Таким образом, **отличительными чертами СИИ является их универсальность, способность к обучению и адаптации в процессе восприятия (с помощью искусственных органов чувств) и воздействия на окружающую среду, а также многоцелевое назначение, связанное с автоматизацией интеллектуальной деятельности человека.**

2.3. Искусственные органы чувств

Многие СИИ нуждаются в очувствлении с целью получения информации об окружающей среде, а также в дальнейшем совершенствовании управляющей системы.

Современные СИИ отличаются наличием большого ассортимента искусственных органов чувств, которые способствуют решению множества интеллектуальных задач. Это прежде всего тактильные, зрительные, звуковые, а также и некоторые другие сенсорные устройства. Органы чувств СИИ служат для ввода информации о состояниях СИИ и окружающей среды в управляющую систему, которая не ограничивается запоминающим и программирующим устройством, а требует для своей реализации управляющей ЭВМ. Именно очувствление в сочетании с достаточно совершенным и разнообразным программным обеспечением управляющей ЭВМ позволяют СИИ работать с неориентированными предметами произвольной формы, взаимодействовать с внешней средой, выполнять требуемую (программную) последовательность операций в меняющейся обстановке. Таким образом, очувствление СИИ является необходимой предпосылкой для повышения их функциональных возможностей.

Информационно-измерительная система очувствленных СИИ, т. е. система их органов чувств, состоит из сенсорных устройств внешней н внутренней информации. Соотношения между устройствами информации и их взаимодействие у этих СИИ существенны. Важно отметить, что для очувствленных СИИ значительную роль

играют устройства внешней информации, предназначенные для восприятия, анализа, распознавания и контроля состояний внешней среды. Каковы же требования, предъявляемые к устройствам внешней информации? Какова конструкция и основные характеристики этих устройств?

В зависимости от назначения СИИ ее датчики внешней информации должны имитировать осязание, зрение, слух и т. п. Кроме того, могут потребоваться устройства для измерения радиоактивности, давления, влажности, температуры и других физических величин. Эти устройства должны обладать высокой точностью, надежностью, быстродействием, а также иметь малые габариты, вес и стоимость. Надо отметить, что сейчас техника располагает датчиками и приборами, которые во много раз чувствитель нее наших органов чувств. Микрофон слышит лучше, чем человеческое ухо, фотоэлемент видит большую часть спектра лучше, чем глаз. Сейсмограф более чувствителен, чем наши органы осязания, и, конечно, температуру по сравнению с термометром человек определяет совсем плохо. Пожалуй, только одно чувство — обоняние, т. е. обнаружение и определение небольших количеств примесей органического вещества, у человека и животного более совершенно, чем у существующих приборов. В этой связи интересно напомнить, что органы обоняния — одни из самых сложных органов чувств, а природа явления, на основе которого они функционируют, до сих пор не открыта. Поэтому «догнать обоняние собаки»—одна из актуальных проблем очувствления ИИ. Опыт изучения органов чувств человека и животных содержит много сведений, которые могут быть использованы в качестве предпосылок для разработки искусственных органов чувств. В живых системах все органы чувств оснащены собственными органами движения, которые в свою очередь богато снабжены кинестетическими рецепторами. При восприятии существенная роль принадлежит как отдельным рецепторам, так и рецептивным полям и локальным детекторам, позволяющим выделять определенные простейшие признаки объектов. При анализе среды и внутреннего состояния важную роль играет совместная координированная обработка сенсорных сигналов различных типов с учетом производимых действий.

Взаимодействие человека с внешней средой в значительной степени основано на переработке зрительной, звуковой и тактильно-кинестетической информации. Существуют также ситуации, когда только тактильные и кинестетические ощущения способны дать правильную информацию о характеристиках среды. Эти ситуации возникают, например, тогда, когда необходимо осуществлять

микродвижения пальцев для определения формы и качества поверхности окружающих предметов, а также в тех случаях, когда имеются помехи зрительному контролю.

Рассмотрим более подробно некоторые типы искусственных органов чувств, предназначенных для измерения характеристик внешней среды.

Тактильные и кинестетические датчики. Решение многих задач, связанных с поиском предметов, их захватом и манипулированием ими, стало возможным только с разработкой датчиков, обладающих тактильной и кинестетической чувствительностью. Простейшим типом таких датчиков являются контактные датчики. Они представляют собой микропереключатели, фиксирующие соприкосновение с предметом.

Тактильные датчики позволяют реагировать на прикосновение и измерять давление в местах соприкосновения (контакта) датчика с предметом. Эти датчики служат для обнаружения отдельных предметов, предотвращения повреждений этих предметов и самой СИИ, а также для распознавания внешней обстановки путем соприкосновения и ощупывания.

Кинестетические датчики регистрируют положение, перемещение исполнительных органов и возникающие в них усилия.

Зрительные датчики. Для автоматического восприятия и анализа объемных (трехмерных) сцен необходима специальная аппаратура, которая по существу должна имитировать в функциональном отношении работу глаз. Она должна обеспечивать решение таких задач, как активный поиск объектов путем изменения ориентации зрительного датчика, автоматическая фокусировка изображения, измерение дальности до предметов, настройка чувствительности датчика в зависимости от изменения условий освещенности, выделение признаков изображения (цвет, текстура, контуры, размеры, форма и т. п.).

При зрительном очувствлении СИИ источником информации служат телевизионные и оптические датчики. Телевизионный датчик («телеглаз») представляет собой телевизионную камеру с тем или иным законом развертки, в процессе которой все изображение или его фрагмент фиксируется в памяти в виде двухмерной матрицы распределения яркости оптической проекции реальной объемной сцены. В супервизорном режиме управления СИИ обычно предусматривается возможность целеуказания, например, путем прикосновения «световым пером» к соответствующему месту экрана монитора, на котором высвечиваются телекадры поля зрения. Однако телевизионное изображение является плоским, в

отличие от самих предметов, которые, конечно, имеют три измерения. Это лишает человека и СИИ объемности восприятия и связанного с ним «эффекта присутствия». Поэтому большое значение для очувствления СИИ приобретают средства голографии, которые позволяют записывать и восстанавливать не двухмерное распределение яркости, а световую волну, исходящую от предмета, со всеми ее подробностями. Для определения цвета предметов в СИИ используются фотоэлементы, фотодиоды, светофильтры, световоды и другие элементы вместе с источниками света. Обнаружение и определение положения предмета с помощью оптических датчиков основано на регистрации сигналов при пересечении предметом светового потока. Используются сканирующие лазерные дальномеры, голографическое телевидение и т. п.

Звуковые датчики. К звуковым датчикам относятся разного рода микрофоны и ультразвуковые датчики. Микрофоны служат для восприятия звуковых команд. Ультразвуковые датчики состоят из передатчика и приемника сигналов. С помощью отраженного от предметов ультразвукового сигнала они могут их обнаруживать и определять расстояние до них.

Ультразвуковые датчики имеют по сравнению с оптическими следующие преимущества: они могут обнаруживать прозрачные объекты; их показания не зависят от условий освещения и малочувствительны к изменению физических свойств среды (пыль, пар, жидкая среда); срок службы генераторов колебаний практически неограничен и т. д.

2.4. Управление СИИ

Методы управления СИИ основаны на принципе обратной связи. Согласно этому принципу закон управления, формируемый управляющей системой, является функцией текущих состояний и внешней среды, измеряемых искусственными органами чувств. Последние и служат источниками сигналов обратных связей, позволяющих, в частности, в каждый момент времени определять величину отклонения между реальным и программным значением параметра, обстановку в окружающей СИИ среде и т. п.

Управляющая система СИИ решает две основные задачи: первичную обработку и анализ информации, поступающей от искусственных органов чувств, и собственно управление

исполнительными устройствами СИИ с использованием обратной связи через органы чувств СИИ.

Из каких же устройств состоит управляющая система СИИ? Как они взаимодействуют между собой, а также с другими системами СИИ? Ответ на эти вопросы позволит нам выявить отличительные черты управления СИИ.

Попытки создания СИИ, функционирующих в условиях большой неопределенности, привели к осознанию того, что системы , управления таких СИИ необходимо должны быть иерархически организованными и адаптивными. Прежде всего попробуем разобраться, для чего и когда нужны иерархическая организация и адаптация в системах управления СИИ. Ответ на эти вопросы позволит, в частности, разделить все системы управления СИИ на два класса. Первый класс — это простые системы, в которых управление полностью централизовано и нет необходимости вводить иерархию и адаптацию. Второй класс — это сложные иерархические адаптивные системы управления с элементами искусственного интеллекта, которые в процессе функционирования сами вырабатывают целесообразное поведение СИИ.

Дальнейшее изложение посвящено изучению систем управления СИИ второго типа. Основное внимание обращается на разработку общих принципов и конкретных алгоритмов проектирования подобных систем. Рассматривается также вопрос о том, каким образом формируется сложное целесообразное поведение СИИ из совокупности действия простых в структурно-функциональном отношении элементов интеллекта.

Говоря о искусственном интеллекте, нам неизбежно приходится пользоваться понятием сложности.

В простейших случаях, например, когда речь идет о исполнительных устройствах СИИ, сложность можно понимать как число степеней свободы, т. е. число независимых переменных, определяющих состояние этих устройств. Однако в менее элементарных случаях понятие сложности, имеющее принципиальное значение, труднее поддается точному определению. Если все же пользоваться в первом приближении понятием числа степеней свободы, то надо отметить, что достаточно эффективные методы исследования существуют в основном либо для систем с малым числом степеней свободы (локальные регуляторы, приводы и т. п.), либо для систем, где число степеней свободы столь велико, что отдельные степени свободы становятся несущественными (плазма, жидкость и т. п.).

Однако СИИ представляют собой промежуточный вариант между системами с малым и с очень большим числом степеней свободы,

который, как это часто бывает, особенно трудно поддается изучению. В частности, методы, хорошо зарекомендовавшие себя при исследовании указанных предельных случаев, оказываются совершенно неэффективными при анализе и синтезе СИИ. Для описания таких систем, относящихся к сложным системам с элементами искусственного интеллекта, требуются новые понятия, модели и методы.

Прежде всего отметим, что *сложность СИИ* определяется не только и не столько сложностью ее исполнительных устройств, сколько сложностью соверниемых ею целенаправленных действий. В этом смысле СИИ обладает высокой сложностью (и соответственно высокой степенью интеллекта), если она способна решать весьма трудные и сложные задачи.

Человек и другие сложные биологические системы способны решать очень трудные задачи по переработке информации и управлению. При этом они используют собственные, сложившиеся в результате многовековой эволюции и поэтому весьма эффективные принципы и средства. Каковы же эти принципы и средства? Что мы знаем о них? Обратимся к сложной системе, с которой мы никогда не расстаемся, — к нашему телу. Если рассматривать человеческое тело просто как механическую систему, то окажется, что эта существенно упрощенная система имеет более двухсотстепеней свободы. Тем не менее мы успешно спранлнемся с задачей управления движением такой системы: пишем, ходим, бегаем, плаваем и т. п. Следует отметить, что при управлении телом весьма важен фактор времени . Если нам нужно избежать какой-либо опасности или обойти препятствие (скажем, отскочить от движущегося на нас автомобиля), то эту задачу, требующую участия многих мышц и поэтому сложную, нужно не прост решить, по решить достаточно быстро. Человек решает эту задачу без всякого напряжения примерно за секунду. А современные СИИ, оснащенные соответствующими органами чувств и элементами искусственного интеллекта, потребуют даже для решения более простых задач значительно большего времени.

Рассмотрим кратко (в основном на примере построения движений) некоторые общие принципы управления, действующие в биологических системах. Выяснение этих принципов особенно важно для построения достаточно совершенных систем управления СИИ. Один из эффективных способов управления биологическими системами состоит в *иерархизации управления,* т. е. в распределении задачи между несколькими уровнями.

Смысл иерархической организации управления заключается в следующем. Для того, чтобы выполнить некоторые движения, нужна

согласованная работа нескольких мышц, каждая из которых, в свою очередь, состоит из совокупности отдельных двигательных единиц — волокон, управляемых связанными с ними мотонейронами. Совершенно ясно, что, выполняя то или иное движение, мозг вовсе не управляет каждой мышцей и тем более каждой двигательной единицей. Высший уровень нервной системы (связанный с большими полушариями мозга) ставит лишь общую задачу, скажем, переложить книгу с полки на стол. Этот уровень вообще не контролирует действие отдельных двигательных единиц, направленных на решение поставленной задачи. Здесь уместна аналогия: командующий армией, ставя перед своими войсками некую общую задачу, отнюдь не предписывает каждому солдату и офицеру, что именно он должен делать в каждый момент операции.

Детализация построения движений у человека происходит на уровнях более низких, чем командный уровень коры больших полушарий. Более того, в некоторых случаях (например, когда мы случайно прикоснувшись к горячему предмету, рефлекторно отдергиваем руку, даже не успев осознать ситуацию) все управление формируется на нижележащих уровнях, связанных с различными отделами спинного мозга. Существенным свойством многоуровневых систем, необходимым для целенаправленного поведения, является определенный обмен информацией между уровнями. Следует заметить, однако, что передача избыточной информации с одного уровня на другой точно так же, как и недостаточный обмен информацией между уровнями, может серьезно нарушить поведение системы в целом.

Вдумываясь в возможности иерархически организованных систем, мы сталкиваемся со следующей проблемой. С одной стороны, осуществление желаемого движения требует координированной работы большого числа мышц, для чего необходима обширная информация. С другой стороны, высший уровень управления дает лишь общую команду: «взять книгу», «сесть за стол» и т. п. Каким же образом эта общая команда преобразовывается в детализованную последовательность управляющих сигналов в условиях «информационного голода»? На самом деле парадокса здесь нет: недостающая информация заключена в памяти и структуре систем низших уровней. Ситуация здесь такая же, как и в случае, когда программист, составляя сложную программу для ЭВМ, часто использует так называемые стандартные подпрограммы, хранящиеся в памяти ЭВМ и встречающиеся в самых разных задачах. Именно за счет этих стандартных подпрограмм обобщенные макрокоманды

программиста автоматически разворачиваются в детализованную систему необходимых вычислений.

Таким образом, иерархическое (многоуровневое) управление гораздо экономнее, чем жестко централизованное, при котором некий управляющий орган точно предписывает действие каждой из составляющих систему частей. Более того, в ряде задач такой централизованный управляющий орган оказался бы настолько сложным, что едва ли смог бы функционировать.

Выполняя движения, мы не можем независимо управлять всеми теми степенями свободы, которые имеет наше тело. При построении движений степени свободы тела распадаются на согласованно управляемые «блоки», которые физиологи называют **синергиями**. Тем самым число независимых степеней свободы, подлежащих управлению, оказывается существенно меньше, чем общее число степеней свободы тела. Некоторые из этих «блоков»-синергий в нашем организме жестко генетически зафиксированы, другие — вырабатываются в результате обучения.

Обучение человека тем или иным движениям (например, плаванию, катанию на коньках и т. п.) сводится к формированию и закреплению в его памяти соответствующих *синергии.* При сложных движениях происходит последовательное чередование различных синергий. Итак, **синергии — это те «кирпичи», из которых мы строим целенаправленные движения.**

Мы разобрали принцип синергийного («блочного») управления на примере построения движений, однако он имеет гораздо более общее значение. Специализированные «блоки»-синергии могут формироваться в процессе эволюции или обучения не только для построения движений, но и для восприятия и переработки информации. В последнем случае такие блоки обычно называют фильтрами, детекторами, сенсорными или афферентными синергиями. Подобно тому, как моторная синергия определяет координированную последовательность мышечных сокращений, сенсорная синергия выделяет из всего многообразия возможных сигналов только определенную последовательность согласованных звуковых или визуальных образов. Группировка степеней свободы в «блоки» и связанное с этим существенное уменьшение числа независимых переменных является, по-видимому, одним из общих эффективных методов управления сложными системами.

Следующий общий принцип организации управления в сложных биологических системах — это *способность к обучению* на опыте и *адаптации* к заранее неизвестным и меняющимся в довольно широких пределах условиям обитания. Человеку часто приходится решать

сложные задачи с неопределенностью (с недостатком сведений) в жестко ограниченное время. Вместе с тем, скорость многих биологических процессов сравнительно невелика. Поэтому для того, чтобы, с одной стороны, по возможности увеличить быстродействие, а с другой, — избавиться от излишней перегрузки, живой организм старается, используя прошлый опыт и прогнозируя текущую ситуацию на ближайшее будущее, адаптироваться (приспособиться) к ней с тем, чтобы в нужный момент не решать задачу «на пустом месте», а лишь уточнить и скорректировать заранее подготовленное решение. Способность к обучению и адаптации присуща не только целостному организму, но и отдельным его органам и функциям. Эта способность особенно важна в тех случаях, когда одна и та же задача (возможно, с некоторыми заранее неизвестными изменениями условий и данных) решается многократно. Таким образом, феномен обучения и адаптации играет важную роль в целесообразном поведении живых организмов.
Мы рассмотрели некоторые общие принципы управления сложными биологическими системами — иерархическую организацию управления, блочную группировку переменных (принцип синергии), обучение и адаптацию. Попытаемся теперь использовать эти принципы при построении сложных, многомерных систем управления СИИ. Сложность таких систем управления заключается в том, что они состоят из многих взаимосвязанных подсистем, реализующих элементы интеллекта СИИ. Подчеркивая важность взаймозависимости отдельных подсистем, будем говорить об иерархических системах управления СИИ.
Анализ и особенно синтез таких систем управления наталкиваются на ряд принципиальных трудностей и непосредственно не сводятся к методам классической теории управления. В частности, функционирование СИИ описывается уравнениями более сложной структуры, чем обычно рассматриваемые алгебраические, дифференциальные или разностные уравнения; цели управления и критерии качества сложны и зачастую противоречивы; неопределенность (недостаток сведений) относительно характеристик среды, СИИ и условий ее функционирования требует введения элементов обучения и адаптации. Все эти обстоятельства приводят, как и в случае сложных биологических систем, к тому, что системы управления таких СИИ необходимо должны быть иерархическими и адаптивными.
Иерархизация управления введением элементов интеллекта и адаптация как средство уменьшения неопределенности приводят к иерархическим адаптивным системам управления, которые делают СИИ способным решать сложные задачи.

Иерархическая организация управления СИИ — это прежде всего распределение функций восприятия, обработки информации и управления между отдельными уровнями иерархии и подсистемами СИИ. Такое распределение «обязанностей» по переработке информации определяется в значительной степени объемом воспринимаемой информации и требованиями к ее обработке (объем памяти, быстродействие, точность преобразований и т. п.), необходимыми для управления исполнительными устройствами СИИ.
При автономном управлении СИИ в сложных и заранее неизвестных условиях функционирования полностью централизованные и жестко определенные алгоритмы восприятия, обработки информации и управления оказываются малоэффективными или даже непригодными. Это связано прежде всего с тем, что эти алгоритмы не адаптивны, т. е. не способны автоматически приспосабливаться к существующим (но заранее неизвестным) условиям функционирования, и достаточно сложны в реализации, что приводит к значительному запаздыванию в управлении, т. е. к управлению по устаревшей информации.
Таким образом, возникновение иерархической адаптивной структуры управления диктуется в первую очередь стремлением увеличить быстродействие и уменьшить уровень неопределенности в процессе функционирования СИИ, т. е. повышением качества управления.
Одним из эффективных путей преодоления трудностей, вызванных большим объемом воспринимаемой СИИ информации и сложностью ее обработки, является, как и в случае биологических систем, распределение функций («децентрализация») и распараллеливание алгоритмов обработки информации и управления между отдельными уровнями иерархии и подсистемами СИИ, т. е. создание автономно (самостоятельно) функционирующих подсистем (или, как еще говорят, модулей, блоков, звеньев). Для функционирования отдельных подсистем необходим уже значительно меньший объем информации, который можно быстрее и эффективнее обработать и, следовательно, решить соответствующую задачу в условиях меньшей (и, возможно, существенно меньшей) неопределенности. (Следует отметить, что иерархизация управления, приводящая к уменьшению неопределенности, связанной со сбором и обработкой информации, служит источником новой неопределенности, порождаемой возможной нетождественностью целей всей системы управления и ее отдельных подсистем. Так возникает проблема оптимальной меры иерархизации систем управления СИИ.).
При построении иерархической системы управления СИИ , функционирующей в условиях неопределенности, целесообразно использовать принцип обучения и адаптации. Согласно этому

принципу в процесс решения СИИ задач происходит накопление опыта и приспособление (адаптация) системы управления к конкретным условиям функционирования СИИ. Это осуществляется путем подстройки параметров и (или) структуры отдельных подсистем так, чтобы система управления по прошествии некоторого времени адаптации могла обеспечить назначенную цель управления. Время, затрачиваемое на обучение и адаптацию, связано с необходимостью для системы управления «приспособиться», «привыкнуть» к фактическим, но неизвестным условиям работы. Свойство адаптивности иерархической системы управленйя заключается таким образом, в обеспечении достижения цели управления сразу для целого класса заранее неизвестных характеристик среды, СИИ и условий ее функционирования.

В общем случае иерархическую адаптивную систему управления СИИ можно представить в виде структурно-функциональной схемы, изображенной на рис. 1

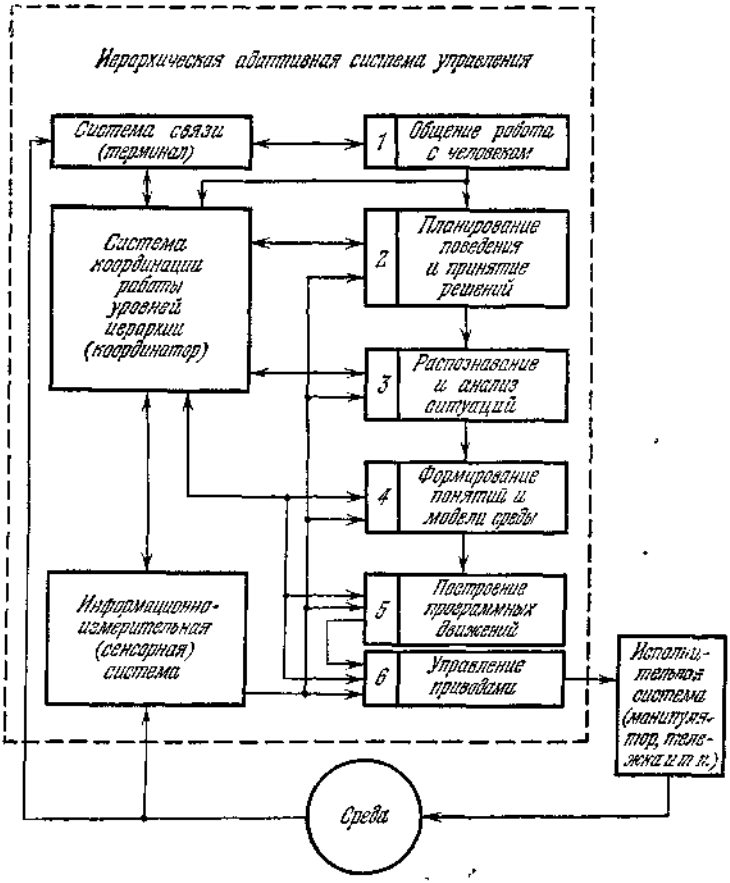


Рис. 1. Схема иерархической адаптивной системы управления.

Эта система состоит из следующих основных уровней иерархии:

1. Общение СИИ с человеком. На этом высшем уровне имеется алгоритмический язык, по возможности близкий к естественному языку, на котором человек поддерживает связь с СИИ в режиме диалога и, в частности, формулирует ей задания.

2. Планирование поведения и принятие р е ш е н и й. Алгоритмы этого уровня на основе задания, получаемого с высшего уровня, а также сигналов обратной связи от сенсорной системы и нижних уровней иерархии осуществляют выбор того или иного плана

поведения СИИ, ведущего к выполнению задания. Далее последовательно принимаются решения об отработке отдельных операций, составляющих сформированный план.

3. Распознавание и анализ ситуаций. Алгоритмы этого уровня производят распознавание и анализ ситуаций в соответствии с указаниями высших уровней иерархии и информации, получаемой от сенсорной системы.

4. Формирование понятий и модели внешней среды На этом уровне осуществляется формирование новых понятий и модели внешней среды в памяти СИИ путем соответствующей переработки текущей информации и накопленного СИИ опыта.

Именно наличие информационной модели внешней среды позволяет системе управления «мысленно» проигрывать различные схемы поведения СИИ еще до совершения ею каких-либо реальных действий.

5. Построение программных движений. Алгоритмы этого уровня строят программные движения исполнительных механизмов СИИ, т. е. закон изменения их обобщенных координат, гарантирующий достижение цели управления с учетом имеющихся препятствий и конструктивных ограничений.

6. Управление устройствами. На этом низшем уровне осуществляется синтез закона управленияустройствами , обеспечивающего приближенную реализацию программных движений исполнительных устройств СИИ при наличии разного рода возмущений и неопределенности. Все перечисленные уровни функционально связаны между собой через систему координации их работы, а также с сенсорной системой, приводами и исполнительными устройствами. Уровни иерархии, выходные сигналы которых являются входными сигналами для более низких (ведомых) уровней, будем называть **ведущими, или руководящими**. В процессе функционирования СИИ вовсе не требуется, чтобы все уровни иерархии работали одновременно. В частности, ведущие (руководящие) уровни могут действовать более редко, чем ведомые.

Представляется принципиально важным тот факт, что система управления движением — это сложная, хотя и специализированная система искусственного интеллекта. Этот факт совместно с подобными фактами, установленными относительно систем зрительного и слухового восприятия, а также систем понимания естественного языка доказывает необходимость децентрализации интеллекта в СИИ. Изложенное выше позволяют использовать термин «система искусственного интеллекта» для обозначения концептуальной подсистемы, реализующей следующие функции:

1) принятие решений и построение планов действия «центральным» интетлектом, т. е. в решающей системе СИИ;
2) решение общих задач принятия решений и построения планов для специализированных систем искусственного интеллекта, каковыми являются подсистемы восприятия и управления движением;
3) преобразование вторичного внешнего представления перцептивной информации (т. е. полученной после первичной обработки) во внутреннее представление решающей системы

Представляет интерес проследить эволюцию СИИ по степени их «интеллектуальности». Это позволит нам правильно понять причины, обусловившие симбиоз двух поначалу различных научных направлений — искусственного интеллекта и робототехники, а также даст возможность оценить последствия взаимного плодотворного влияния этих направлений друг на друга

Мы будем оценивать степень интеллектуальности по следующим четырем качественным параметрам

1) Характеристика внешнего мира, в котором способна работать СИИ Под *моделью внешнего мира СИИ* мы будем понимать множество объектов (включая саму СИИ) и отношений, заданных на этом множестве, которые описывают пространство возможных состояний среды, окружающей СИИ.
2) Степень восприятия СИИ информации о внешнем мире
3) Степень гибкости решающей программы СИИ.
4) Степень автономности СИИ. Под *автономностью* мы будем понимать способность СИИ самостоятельно, без участия человека решать поставленные перед нею задачи. Следует выделить компоненты автономности: *мобильность, энергетическую автономность* и *функциональную автономность,* т. е автономность планирования и выполнения действии, направленных на решение задачи.

Эти параметры достаточно полно позволяют нам отвечать на вопрос, насколько успешно и самостоятельно СИИ сможет решать постановленные перед нею задачи в том или ином внешнем мире.

2.5. Общая структура системы искусственого интеллекта и ее анализ

2.5.1. СОСТАВ И СТРУКТУРА СИИ

Составными структурными частями СИИ являются подсистемы, обладающие всеми свойствами системы и создаваемые как самостоятельные системы. По назначению подсистемы СИИ разделяют на два вида: функциональные и обслуживающие. Функциональные подсистемы выполняют интеллектуальные процедуры и операции.
Обслуживающие подсистемы предназначены для поддержания работоспособности функциональных подсистем, например: подсистема графического отображения объектов; подсистема информационного поиска.
В зависимости от отношения к объекту функционирования различают два вида функционирующих подсистем: объектно-ориентированные (объектные), объектно-независимые (инвариантные). Объектные подсистемы выполняют одну, или несколько интеллектуальных процедур или операций, непосредственно зависимых от конкретного объекта. Инвариантные подсистемы выполняют унифицированные процедуры и операции.
Подсистема состоит из компонентов СИИ (далее — компонентов), объединенных общей для данной подсистемы целевой функцией и обеспечивающих функционирование этой подсистемы. Компонент представляет собой элемент обеспечения, выполняющий определенную функцию в подсистеме:
методическое обеспечение — документы, в которых отражены состав, правила отбора и эксплуатации средств автоматизации;
лингвистическое обеспечение — языкифункционирования , терминология;
математическое обеспечение — методы, математические модели, алгоритмы;
программное обеспечение — документы с текстами программ, программы на носителях и эксплуатационные документы;
техническое обеспечение — устройства вычислительной и организационной техники, средства передачи данных, измерительные и другие устройства или их сочетания,
информационное обеспечение — документы, содержащие описания стандартных интеллектуальных процедур, типовых решений, типовых элементов и другие данные, а также файлы и блоки данных на носителях с записью указанных документов;
организационное обеспечение — положения, инструкции, приказы, штатные расписания, квалификационные требования и другие документы, регламентирующие организационную структуру подразделений и их взаимодействие с комплексом средств автоматизации .
Введение структурного понятия «компонент» как некоторого эле-

ментарного «кирпичика» системы позволяет раскрыть внутреннюю структуру подсистемы и указать конкретные связи между подсистемами не только иерархические, но и методические, информационные и т. д.

На рис. 1 приведена двухмерная структурная схема СИИ, основанная на понятии компонента.

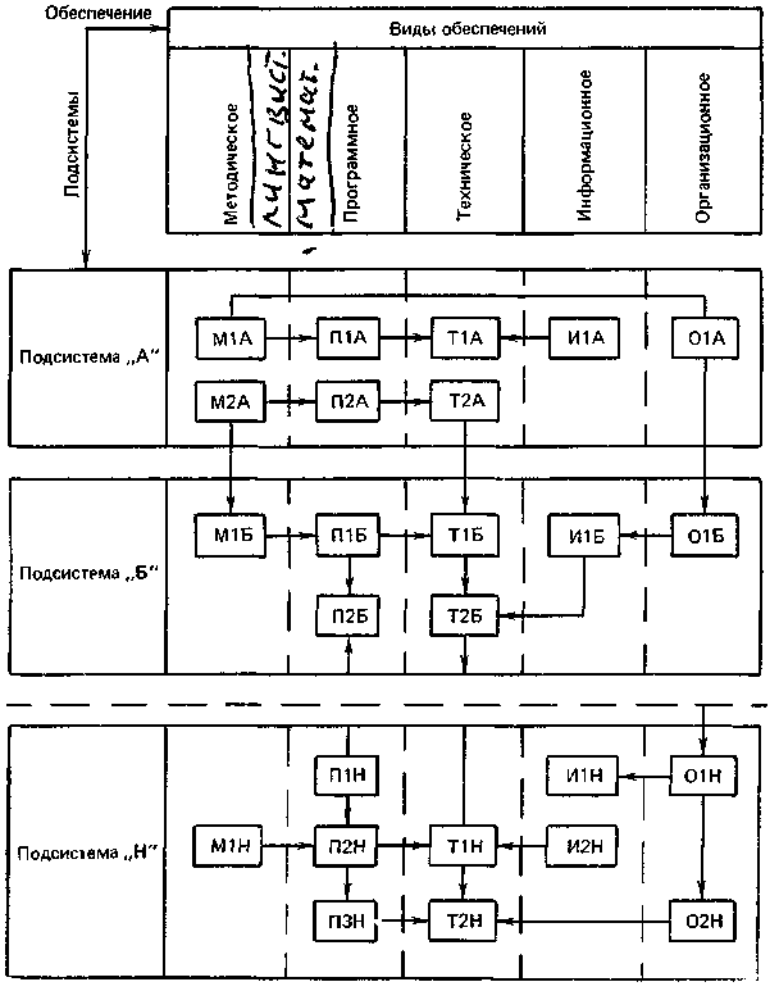


Рис. 1. Матричная структура СИИ.
G — компонент СИИ

Как видно из рисунка, матричная структура является открытой как по количеству подсистем, так и по видам обеспечения. Связи между подсистемами «А» и «Б» показывают, что иерархически подсистема «Б» подчинена подсистеме «А», а компонент организационного обеспечения 01А является определяющим как для самой подсистемы «А» (задает требования к компоненту методического обеспечения М1А), так и для подсистемы «Б» (является исходным для компонента 01Б). Компоненты могут иметь многократное применение, т. е. один и тот же типовой или унифицированный компонент может применяться в различных подсистемах.

Анализ структурной схемы СИИ позволяет сделать вывод, что структурное единство подсистемы СИИ обеспечивается связями между компонентами различных средств обеспечения СИИ, образующими подсистему, а структурное объединение подсистем в систему — связями между компонентами СИИ, входящими в подсистемы.

Ниже мы кратко опишем функциональную организацию СИИ. На самом общем уровне СИИ состоит из трех алгоритмических систем— *системы восприятия, системы решения задач* и *эффекторной системы.* Однако такое представление является слишком поверхностным. С другой стороны, описание подробной блок-схемы СИИ заняло бы слишком много места и, не обладая достаточной наглядностью, не соответствовало бы главной цели этой темы — создать у читателя общее представление о том, как функционируют и как взаимодействуют между собой указанные выше алгоритмические системы. Поэтому наше описание будет носить промежуточный характер, с введением некоторых подробностей только там, где это касается ключевых моментов функционирования СИИ.

Это описание мы начинаем с системы решения задач.

2.5.2. Система решения задач

Система решения задач СИИ состоит из трех главных частей: *подсистемы представления знаний, планирующей подсистемы* и *исполнительной подсистемы* (рис. 2).

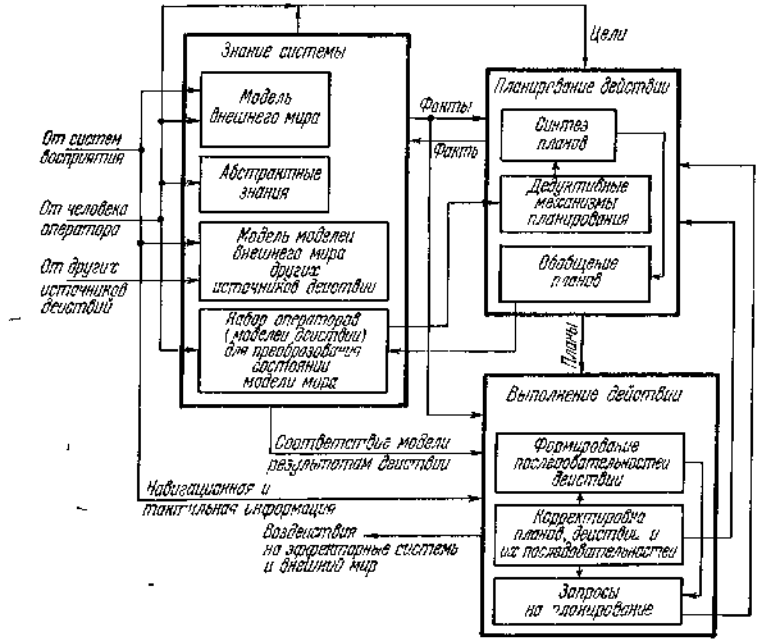


Рис. 2. Решающая система СИИ.

Знание системы включает в себя *модель внешнего мира, абстрактные знания, знания о других источниках действий во внешнем мире и набор операторов,* преобразующих внешний мир. Причины выделения первых трех компонент знания мы упоминали ранее. Набор *операторов* представляет собой формальное описание *моделей действий,* которые способна совершать СИИ, преобразуя с помощью этих действий одни состояния внешнего мира в другие и соответственно с помощью операторов одни состояния моделей внешнего мира в другие. Аналогичным образом могут быть определены операторы, преобразующие модели моделей внешнего мира других источников действия

Информация о мире обычно хранится в виде совокупностей фактов, связанных между собой теми или иными причинно-следственными связями Каждый оператор обычно описывается условиями, определяющими возможность его применения, и результатами, ожидаемыми от его применения.

Следует выделить четыре важнейших аспекта, связанных со знанием системы ИИ: *выражение знания,* его *накопление, корректировка* и *использование в процессе решения задач.*

Форма выражения знания называется *представлением.* Проблема представления знания, обладающего достаточной общностью и позволяющего эффективно решать задачи, является одной из центральных в ИИ. Постановка этой проблемы и основные методы представления рассматриваются в следующей теме.

Накопление знаний выступает в свою очередь в двух аспектах. С одной стороны, это накопление множества фактов и установление между ними полезных отношений. Оно осуществляется за счет восприятия информации из внешнего мира и от человека-оператора, а также в результате решения задач системой. В последнем случае факты называются выведенными. С другой стороны, СИИ должна обладать возможностью накапливать методы решения задач. Накопление методов может происходить как в результате общения с человеком-оператором, так и в результате обобщения самой СИИ опыта решения задач.

Корректировка знаний осуществляется как результат взаимодействия СИИ с внешним миром, человеком-оператором и другими разумными существами, населяющими мир СИИ. Наконец, использование знаний в процессе решения задач является ключевым моментом при построении формализмов, моделей и методов ИИ. Особенно следует отметить важность правильного использования смыслового содержания задач, или *семантического знания,* являющегося основой построения всех эффективных систем ИИ.

Общие модели и методы решения задач и особенности наложения на них семантических знаний рассматриваются в последующих темах.

Планирующие подсистемы СИИ являются его *основной решающей частью.* Синтез планов осуществляется на основе поставленных целей и накопленного знания, причем цели могут как задаваться человеком-оператором, так и вырабатываться самой СИИ. При синтезе планов широко используются дедуктивные механизмы планирования, определяющие план как последовательность применимых и пригодных операторов, выбираемых из накопленного набора операторов. Поскольку в достаточно сложных задачах планирования на каждом шаге имеются альтернативные возможности выбора пригодных операторов среди множества применимых, чисто дедуктивные механизмы должны были бы испробовать все возможности. Использование *эвристик,* т. е. *наложение семантики на дедуктивные механизмы,* позволяет сделать выбор более направленным, а следовательно, процесс планирования — более эффективным.

Важнейшим моментом в планировании СИИ является возможность *обобщения планов.* Эта проблема выступает в двух взаимосвязанных аспектах. С одной стороны, представляется целесообразным строить приблизительные планы с последующей их детализацией. Это повышает эффективность планирования, особенно в сложных мирах, как за счет лучшего «понимания» планирующей подсистемой общих закономерностей пространства поиска решений, так и за счет структурирования общего описания мира на более и менее существенные факты о мире. С другой стороны, для СИИ важно обобщать уже построенные планы так, чтобы использовать их в дальнейшем как дополнительные укрупненные операторы, «обучаясь» решать все более и более сложные задачи. Накапливая таким образом модели обобщенных действий, СИИ может эффективно использовать их и для построения приблизительных планов все увеличивающейся общности и сложности.

Другим важным фактором, определяющим эффективность планирования, является обеспечение гибкого *взаимодействия процессов планирования* и *выполнения действий,* соответствующих построенным планам. Именно такое взаимодействие определяет способность СИИ успешно функционировать в реальном мире. Напрашивающимся решением проблемы взаимодействия указанных выше процессов является организация обратной связи, позволяющей корректировать модель мира и планы в соответствии с действительными изменениями, происходящими в мире.

Эти функции выполняются исполнительной подсистемой. Как видно из рис 2, эта подсистема формирует и выполняет последовательность действий, соответствующих операторам плана, получаемого от планирующей подсистемы, а также выдает команды о корректировке плана в случае несоответствия действительного состояния мира и его модели в СИИ.

2.5.3. Система восприятия

Рассмотрим теперь особенности функционирования алгоритмических систем восприятия и воздействия на эффекторные системы (рис.3).

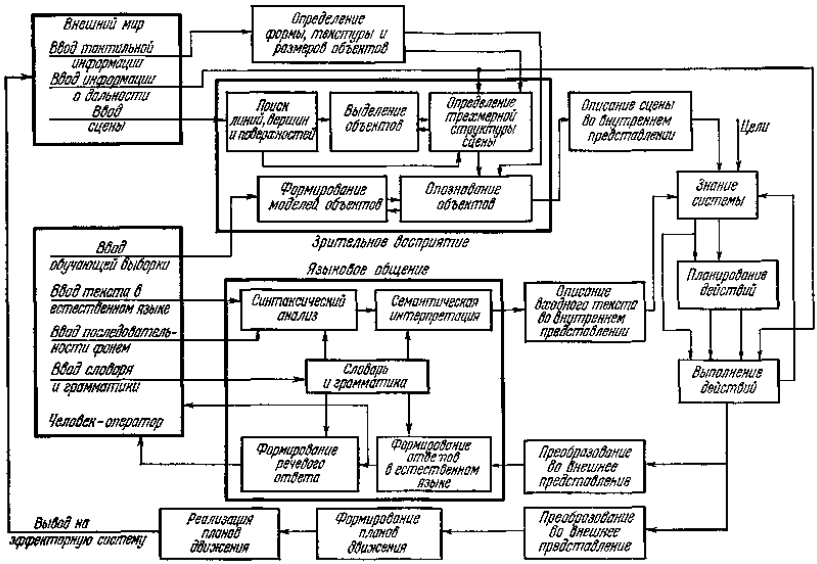


Рис.3. Функциональная организация систем восприятия и воздействия на внешний мир.

Мы выделяем два основных источника информации СИИ: внешний мир и человек-оператор.
С первым из них СИИ общается с помощью систем датчиков (тактильных, телевизуальных, дальномерных) и систем ввода информации с этих датчиков. Эти системы на рис. 3 не показаны.
С человеком-оператором СИИ общается с помощью стандартных терминалов диалоговых систем (телетайпов, мониторов и т. д.) или с помощью систем понимания речи.
СИИ осуществляет воздействие на человека-оператора с помощью терминалов диалоговых систем или систем речевого вывода, а на внешний мир — с помощью электромеханических систем движения (манипуляторы, колесные, гусеничные или шаговые движители и т. д.). Эти системы также не показаны на рис.3.
Мы уже отмечали, что периферийные алгоритмические системы СИИ представляют из себя сложные специализированные системы ИИ. Сейчас мы вкратце опишем некоторые из таких систем, с тем чтобы читатель смог составить представление об уровне их сложности.

На рис. 3 приведена одна из возможных схем функциональной организации зрительного восприятия СИИ. Следует сразу заметить, что приведенная на рисунке последовательность этапов не является единственно возможной. Более того, она может определяться внешним миром системы восприятия, образуя сложные итеративные циклы. Во введенном в некоторой форме (например, в виде распределения точек яркости) изображении производится поиск линий и вершин объектов (мы рассматриваем для простоты сцены с объектами, ограниченными плоскими гранями), на основании результатов которого может быть описана *трехмерная сцена*. Эти две задачи зрительного восприятия, вообще говоря, относятся к первичной обработке сцен, со свойственными ей специализированностью и сильной зависимостью от особенностей применяемой телевизуальной аппаратуры. Однако и в этих задачах уже можно наблюдать некоторые элементы ИИ. Дело в том, что описанная трехмерная сцена отнюдь не является единственно возможной. Появляющаяся в связи с этим недетерминистичность выбора среди множества альтернатив требует решения задач направленного поиска, эффективность которого в значительной степени зависит от накопленного в системе или введенного в нее семантического знания.

В еще большей степени семантика определяет эффективность решения важнейшей задачи анализа сцены — *разбиения сцены на тела* (объекты). Это касается такой проблемы, как идентификация границ, где семантика, выраженная в виде законов физически возможных конфигураций линий в области вершин, буквально в десятки тысяч раз снижает объем требуемого перебора; такой проблемы, как устранение теней, где оптические законы света и тени не только резко сокращают поиск, но и позволяют использовать информацию о тенях как дополнительную для выделения объектов; такой проблемы, как выделение скелетов объектов, где неоднозначность, связанная с неполной видимостью объекта в сложной сцене, может быть эвристически устранена с помощью построения гипотез и сложных, основанных на семантике, процедур проверки и доказательства (или опровержения) этих гипотез.

Очевидно, что система, способная решать упомянутые задачи, должна обладать моделью своего мира (сцены) в представлении, внешнем по отношению к системе решения задач СИИ; она должна также обладать универсальными механизмами вывода (в частности, дедуктивными), управляемыми семантическим знанием. Другими словами, эта система должна обладать всеми существенными признаками ИИ.

Характерной для ИИ является и задача *идентификации (опознавания) объектов и формирования понятий высшего уровня,* которые

становятся исходной информацией для описания сцены во внутреннем представлении и накоплении знания в модели внешнего мира решающей системы СИИ. Ключевым фактором на пути к успешной идентификации объектов является семантически ориентированное устранение неоднозначности выбора, характерное для ИИ.
Весьма важным для решения задач разбиения сцены и идентификации объектов является взаимодействие систем восприятия различного вида, в частности, *бинокулярное зрение* и *взаимодействие тактильной и телевизуальной систем.* При этом вспомогательные виды восприятия обеспечивают важный семантический материал для окончательной проверки выдвинутых гипотез.
Рассмотрим теперь функциональную структуру подсистемы языковою общения СИИ с человеком-оператором. Отметим прежде всего, что функциональная структура подсистемы, приведенная на рис.3, совпадает с классической структурой *«вопросно-ответных систем»* (ВОС), за исключением того, что этап дедуктивного вывода замкнут через решающую систему СИИ. Термин «вопросно-ответная система» обычно употребляется в двух смыслах: в широком смысле термины «вопросно-ответная система» и «система ИИ» суть синонимы; в узком смысле ВОС — это информационно-поисковая система, обладающая способностью *воспринимать смысл текста* в естественном языке, ограниченном в той или иной степени, и способностью давать ответы, не только непосредственно извлекаемые из знания системы, но и *ответы, выводимые из знания* системы и входного текста. Следует понимать, что указанные обобщения определяют качественно новый уровень ВОС по сравнению с информационно-поисковыми системами относительно класса решаемых задач (и соответственно относительно их сложности). Здесь, посвященной изложению методов языкового общения СИИ, мы употребляем понятие ВОС в узком смысле слова.
Основными этапами обработки текста в естественном языке до его преобразования во внутреннее представление решающей системы СИИ и дедуктивного вывода ответа являются *синтаксический анализ* и *семантическая интерпретация.*
Синтаксический анализ текста сводится к определению структуры входящих в текст предложений в соответствии с грамматикой языка и к идентификации слов текста в соответствии со словарем.
Семантическая интерпретация сводится к пониманию смысла текста, т. е. к нахождению семантических отношений между словами текста в терминах знания системы.
Согласно современным представлениям для понимания текста необходимо *итеративное взаимодействие* этапов синтаксического анализа и семантической интерпретации, так что изображенная на рис.

3 последовательность этих этапов носит до некоторой степени условный характер.
Глобальная задача ВОС заключается в том, чтобы на основе накопленного знания и входного текста (который идентифицируется с целью) однозначно выразить последний в терминах внутреннего представления, а затем преобразовать, если это необходимо, внутреннее представление в ответ, выраженный в естественном языке. Таким образом, и здесь мы видим, что подсистема языкового общения СИИ удовлетворяет нашему определению системы ИИ.
В рамках наших общих рассуждений мы, как указывалось выше, замкнули подсистему языкового общения через решающую систему СИИ. Это, однако, не означает, что подсистема не может обладать своим локальным знанием и локальными дедуктивными механизмами. Кроме того, мы показываем на рис. 3 этап формирования ответа в естественном языке, связанным с решающей системой через условную подсистему выполнения планов. В случае ВОС эта подсистема представляет собой высший уровень диалоговой системы общения человека с СИИ, через который осуществляется формирование сложных ответов (например, неполных или условных) и корректировка процесса вывода в соответствии с указаниями человека.

2.5.4. Эффекторная система

Кажущаяся простота функциональной структуры подсистемы воздействия на внешний мир не должна вводить читателя в заблуждение: вся сложность этой подсистемы заключена в этапах формирования и реализации планов движения. Рис. 4 показывает, что более подробная функциональная структура рассматриваемой подсистемы почти аналогична структуре решающей системы СИИ (с точностью до уровня планирования и выполнения).

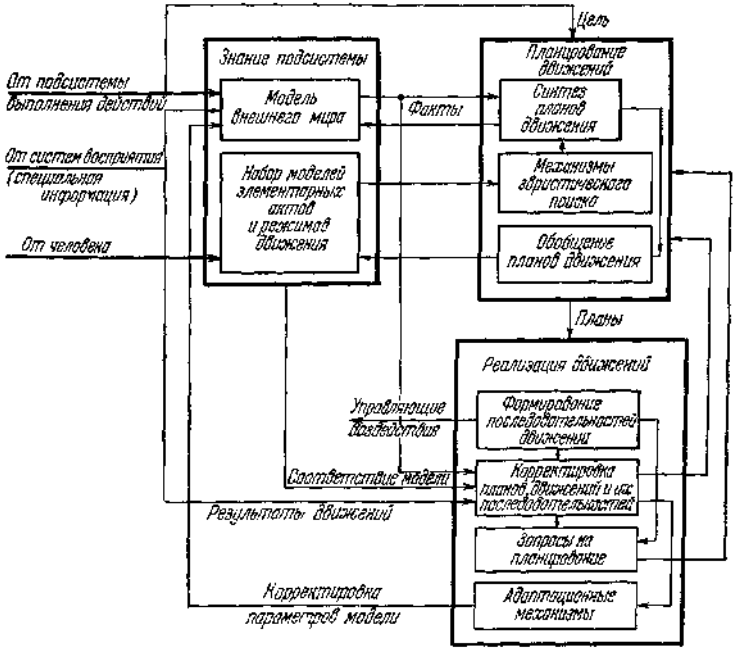


Рис.4. Система воздействия на внешний мир.

На этом рисунке под внешним миром подразумеваются факты об окружающей подсистему среде, т. е. *механические и геометрические свойства объектов,* с которыми работает СИИ, *свойства эффекторных органов* и их текущие положения, а также основные *параметры местных датчиков эффекторных органов.* Соответственно модель отражает текущее знание подсистемы о ее внешнем мире. Это знание может корректироваться человеком, знанием решающей системы («центральное» знание) или самой подсистемой в процессе ее функционирования.

Набор *операторов движения* представляет собой описание элементарных актов движения и их характерных режимов. Он может задаваться человеком и корректироваться как человеком, так и подсистемой планирования движений в результате обобщения планов точно так же, как и в случае решающей системы.

Подсистема планирования движений получает в качестве цели планируемое решающей системой СИИ действие и

синтезирует план выполнения этого действия в виде последовательностей операторов движения. Поскольку эти последовательности могут быть составлены отнюдь не единственным способом, синтез управляется механизмами эвристического поиска, обеспечивающими направленный перебор альтернатив и, возможно, оптимизирущий план движения по некоторому критерию, например, минимуму потребления энергии, максимуму скорости отработки или какой-либо функции этих параметров.

Подсистема реализации движения формирует последовательность управляющих воздействий на эффекторные органы и следит за соответствием запланированных движений действительным результатам В случае несоответствия подсистема либо корректирует требуемые движения или их последовательности, либо выдает запрос на повторное планирование части плана или плана в целом.

Наконец, *адаптационные механизмы* следят за качеством отработки управляющих воздействий и в случае обусловленных нарушений корректируют те или иные параметры модели внешнего мира Приведенное описание показывает, что и подсистема воздействия на внешний мир обладает всеми признаками ИИ.

Конечно, такая функциональная организация должна реализовываться, начиная с некоторого уровня сложности задач, возложенных на СИИ. Мы не утверждаем, должна ли обладать каждая периферийная подсистема своими системами ИИ или работать, например, с центральным дедуктивным механизмом в режиме разделения этого ресурса между всеми подсистемами. Однако так или иначе любая из алгоритмических подсистем СИИ должна использовать те или иные общие формализмы, модели и методы ИИ.

Мы не описываем более подробно, чем это сделано в этой теме, подсистемы воздействия на внешний мир, подобные изображенной на рис. 4 Во-первых, они еще не созданы. Во-вторых, построение их теории весьма сильно связано с механическими, электромеханическими и конструктивными параметрами эффекторных органов и потому является делом специалистов в области механики и систем управления. Аналогичные причины побуждают ограничиться рассмотрением задач синтеза трехмерной сцены, а также поиска и идентификации объектов в других темах

Мы предоставляем соответствующим специалистам общий аппарат ИИ, который необходим для построения всех подсистем СИИ и к изложению которого мы переходим, начиная со следующей темы.

3. МЕТОДЫ ПРЕДСТАВЛЕНИЯ И РЕШЕНИЯ ЗАДАЧ В СИСТЕМАХ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

3.1. Исходные понятия

В соответствии с принятым в теме 2 определением мы выделяем два аспекта искусственного интеллекта (ИИ) — способность хранить, накапливать, извлекать, обобщать и корректировать знание *(эпистемологический аспект)* и способность использовать знание вместе с поставленной целью для нахождения эффективных решений задач *(эвристический аспект)*.

Таким образом, знание фактов относительно различных сторон мира, в котором работает система ИИ, является ключевым понятием для построения таких систем.

Общая система знаний ИИ может рассматриваться как состоящая из знания о внешнем мире (модель внешнего мира), абстрактного знания (мир философии, математики и т. д.) и знания о знаниях активных источников действия в мире (другие интеллекты, силы природы или сама система ИИ). Существенно отметить, что мы включаем в знание набор *универсальных и специализированных методов решения задач* (РЗ). Таким образом, система ИИ должна обладать способностью хранить, извлекать, обобщать и корректировать методы РЗ как часть своего знания. Система знаний, организованная соответствующим образом, составляет *внутренний мир* системы ИИ.

Форма совокупного выражения знаний и условий решаемой задачи в системе ИИ называется *представлением*.

Перед нами стоит задача строить такие представления, которые, с одной стороны, были бы достаточно общими, т. е. допускали описание широкого класса миров и задач, решаемых в этих мирах. С другой стороны, представления должны допускать использование мощных методов в отношении как качества решения задач, так и потребных для решения ресурсов. В общем случае требования общности представления и мощности методов РЗ являются противоречивыми. Существование обратной зависимости между общностью и мощностью приводит к тому, что в попытках построить общий решатель задач мы будем вынуждены снабдить его общими и потому относительно слабыми методами решения.

В рамках указанной качественной зависимости выбор представления является весьма важным фактором, определяющим как простоту

описания задачи, так и эффективность ее решения. В силу такого двойственного характера представления имеет смысл описать эпистемологические и эвристические свойства представления.

***Эпистемологически полным представлением,* или представлением в широком смысле, назовем совокупность формализмов для описания всех фактов о мире, необходимых для выполнения определенного класса задач**

Пример эпистемологически неполного представления: естественный язык эпистемологически неполон для описания хранения знаний в человеческом мозгу или для описания сложных визуальных образов.

Эпистемологически адекватным представлением называется представление, которое можно практически использовать для выражения фактов относительно какого-то аспекта мира.

Пример эпистемологически полного, но неадекватного представления: описание 40-разрядного двоичного регистра в виде конечного автомата.

***Эвристически адекватным представлением* называется представление, которое допускает лингвистическое выражение последовательности рассуждений, приведшей к решению задачи.**

Пример зпистемологически полного и адекватного, но эвристически неадекватного представления: представление зрительных образов путем двоичного кодирования «черного» и «белого» на рецептивном поле зпистемологически полно и адекватно, но эвристически неадекватно, т. е. не может быть использовано для распознавания образов.

Общий план использования методов для решения поставленной задачи будем называть *стратегией.*

Эвристически эффективной стратегией называется стратегия, направленная на оптимизацию необходимых для решения задачи ресурсов.

Представление и стратегия составляют две основные и взаимосвязанные компоненты процесса РКЗ, причем эвристически эффективная стратегия определяется относительно заданного эвристически адекватного представления, а формирование эвристически адекватного представления должно осуществляться с ориентацией на построение эвристически эффективных стратегий.

Обратная зависимость общности и мощности приводит к качественному заключению о том, что чем более специализированным является эвристически адекватное представление, тем с большей вероятностью мы сможем построить эвристически эффективную стратегию.

Поэтому, если рассматривать эпистемологически полное представление главным образом как инструмент для описания фактов о мире и для постановки задачи, процесс решения должен включать в себя последовательное преобразование представлений, начиная от эпистемологически полного представления и кончая таким специализированным эпистемологически и эвристически адекватным представлением, в котором может быть определена эвристически эффективная стратегия.

Под *представлением в узком смысле,* или собственно представлением, будем понимать эпистемологически и эвристически адекватное представление описанное в рамках единого формализма и ориентированное на построение эвристически эффективных стратегий решения подкласса задач относительно класса, определяемого эпистемологически полным представлением.

3.2. Проблема формирования представлений

3.2.1. Общий подход

Несмотря на то, что **проблема формирования представлений — одна из важнейших для построения по-настоящему «разумных» и эффективных решателей задач,** до сих пор не предложено сколько-нибудь удовлетворительной теории, способствующей решению этой проблемы.

Основным подходом к проблеме формирования представлений при решении задач является разработка средств *глобального исследования пространства поиска решений задач,* результатом которого явилось бы преобразование этого пространства в другое или меньшего размера и/или обладающего специфическими свойствами, полезными для решения данной задачи. Мы употребляем термин «глобальное исследование», чтобы отличить его от «локального исследования» пространства поиска решения, естественно происходящего в процессе РЗ.

Среди стандартных целей такого глобального исследования следует выделить:

1) Обнаружение свойств симметрии, избыточности или подобных обобщенных отношений в пространстве поиска решений задач, ведущих к сокращению пространства.

2) Переформулировка задачи путем обобщения элементов начального представления и отождествления полученных макроэлементов с элементами (директивами) в новом представлении.

3) Разделение общей топологии пространства на «легкопроходимые», критические и запрещенные области с последующим переопределением элементов (директив) представления.

Успех глобального исследования пространства поиска решений задач определяется в основном опытом, накопленным решателем задач при попытках решения задачи в исходном представлении. Другими словами, этот подход к решению задачи преобразования представлений следует рассматривать как *пошаговый процесс,* эволюционирующий от каждого текущего представления к более эффективному, исходя из информации о задаче (или классе задач), которая появляется при попытках решить задачу в текущем представлении. Мы рассмотрим этот подход на примере, а затем укажем на некоторые общие методы глобального исследования пространств поиска решений задач

3.2.2. Пример

Рассмотрим задачу о миссионерах и людоедах (МЛ). Словесная формулировка задачи (эпистемологически полное предписание) выглядит следующим образом.

Элементарная постановка. Отправитель получил заказ на осуществление (выполнение) перевозки получателю трех миссионеров и трех людоедов с левого берега на правый берег реки. Для осуществление (выполнение) этой миссии имеется лодка, вмещающая не более двух человек (в любом сочетании миссионеров и людоедов). Если число людоедов на любом берегу превысит число миссионеров, то миссионеры будут съедены. Найти простейший план перевозок, безопасный для миссионеров и такой, что три миссионера и три людоеда окажутся на правом берегу у адресата.

Обобщенная постановка отличается от элементарной тем, что имеется N миссионеров и N людоедов, а лодка вмещает k ($k \geq 2$) человек. Число людоедов не должно превышать числа миссионеров на любом берегу и в лодке.

Мы начнем с представления задачи МЛ в элементарной *системе продукций.* Общая постановка задачи в этой системе выглядит следующим образом: даны начальная ситуация, конечная ситуация, множество возможных преобразований (предписаний, директив) в пространстве ситуаций и условия, определяющие применимость преобразований (предписаний, директив) в той или иной ситуации. Ситуация в системе продукций описывается перечнем ее основных признаков, называемым *N-состоянием.* Требуется найти наилучшую

последовательность допустимых преобразований представлений (директив) из начального состояния в конечное. Пусть S — множество всех возможных N-состояний, $\{A\}$ — конечное множество правил преобразования представлений (директив). Множество $\{A\}$ задает *отношение непосредственной достижимости* T между элементами S. Для $s_x, s_y \in S$ $s_x T s_v$ тогда и только тогда, когда существует допустимое $A \in \{A\}$, преобразующее s_x в s_v, где под допустимым понимается преобразование, удовлетворяющее условиям применимости в N-состоянии s_x. Путь из s_a в s_b есть конечная последовательность $s_1, s_2, \ldots, s_m$, $s_1 = s_a$, $s_2 = s_b$ такая, что для всех i, $1 < i \leq m$, $s_{i-1} T s_i$. Состояние s_b *достижимо* из s_a ($s_a \Rightarrow s_b$) тогда и только тогда, когда или $s_a = s_b$, или существует путь из s_a в s_b. Наконец, множество всех N-состояний, частично упорядоченных отношением T, называется *пространством N-состояний* σ. В этом пространстве мы и осуществляем поиск решающего пути.

Переходя к задаче МЛ, введем следующие обозначения: $\{m_i / i = 1, 2, \ldots, N\}$ —множество миссионеров, $\{c_i / i = 1,2, \ldots, N\}$ — множество людоедов, b_k — лодка с максимальной емкостью k, $p_л$, $p_п$ — левый и правый берег реки соответственно, $M_л$, $M_п$, $M_в$ — количество миссионеров на левом берегу, правом берегу и в лодке соответственно, $C_л$, $C_п$, $C_в$ — количество людоедов на левом берегу, правом берегу и в лодке соответственно. Определим следующие отношения:

At(x_i, p) указывает, что объект x_i, имеющий область значений $(m_i, 1 \leq i \leq N; c_i, 1 \leq i \leq N; b_k)$, находится *на берегу* p, p принимает значения $p_л$ и $p_п$;

On (y_i, b_k) указывает, что объект y_i, имеющий область значений $(m_i, 1 \leq i \leq N; c_i, 1 \leq i \leq N)$, находится *в лодке* b_k.

В первом представлении задачи МЛ ситуация отождествляется с описанием мест всех y_i (миссионеров и людоедов) и лодки. Например, начальное состояние

s_o==At$(b_k, p_л)$, At$(m_1, p_л)$, ...,

At $(m_N, p_л)$, At $(c_1, p_л)$, . . ., At $(c_N, p_л)$. (1)

Конечное состояние определяется тем же выражением , с подстановкой всюду $p_п$ вместо $p_л$.

Множество возможных преобразований (представлений, директив) A_1 записывается следующим образом:

а) *Загрузить лодку на левом берегу* (*ЗЛЛ*):

для любого y_i

$\alpha, \mathrm{At}(b_k, p_л), \mathrm{At}(y_i, p_л); (M_в + C_в \le k-1) \to$

$\to \alpha, \mathrm{At}(b_k, p_л), \mathrm{On}(y_i, b_k); \Lambda.$

б) *Перевезти лодку с левого берега на правый*(*ПЛЛП*): (2)

$\alpha, \mathrm{At}(b_k, p_л); (M_в + C_в > 0) \to \alpha, \mathrm{At}(b_k, p_п); \Lambda.$

в) *Выгрузить лодку на правом берегу* (*ВЛП*):

для любого y_i

$\alpha, \mathrm{At}(b_k, p_п), \mathrm{On}(y_i, b_k); \Lambda \to$

$\to \alpha, \mathrm{At}(b_k, p_п), \mathrm{At}(y_i, p_п); \Lambda.$

Аналогично определяются преобразования для осуществления процесса кперевозки y_i с правого берега на левый (ЗЛП, ПЛПЛ, ВЛЛ). В выражениях (2) отдельные выражения, входящие в описание состояния, отделяются запятыми; ограничения, накладываемые на применимость преобразований, отделяются от описания состояния точкой с запятой, α — произвольная конфигурация, делающая описание состояния полным, Λ — пустое множество ограничений.

Заметим, что множество возможных преобразований A_1, определяемое (2), не учитывает ограничений на относительное число миссионеров и людоедов на берегах реки и в лодке.

Построим теперь множество правил преобразования, которое, кроме учета этого ограничения, включает в себя некоторые макропредписания, определяемые как последовательности элементарных преобразований из A_1 *(обобщение элементов начального представления,* а именно преобразований; понятие состояния пока остается неизменным). Назовем это множество A_2:

$$\left.\begin{array}{l}
\textit{а) Загрузить r объектов } y_i \textit{ в лодку на левом берегу } (З^r ЛЛ): \\
\textit{для множества } \{y_i / i = 1,2,...,r, 1 \le r \le k\} \\
\alpha, \mathrm{At}(b_k, p_л), \mathrm{At}(y_1, p_л), ..., \mathrm{At}(y_r, p_л); (M_в + C_в = 0) \to \\
\to \alpha, \mathrm{At}(b_k, p_л), \mathrm{On}(y_1, b_k), ..., \mathrm{On}(y_r, b_k); \\
((M_л = 0) \vee (M_л \ge C_л)), ((M_в = 0) \vee (M_в \ge C_в)). \\
\textit{б) Перевезти r объектов } y_i \textit{ в лодке на правый берег и} \\
\textit{выгрузить всех их на правом берегу } (ПЛЛП + B^r ЛП): \\
\textit{для множества } \{y_i / i = 1,2,...,r, 1 \le r \le k\} \\
\alpha[e], \mathrm{At}(b_k, p_л), \mathrm{On}(y_1, b_k), ..., \mathrm{On}(y_r, b_k); \Lambda \to \alpha[e], \\
\mathrm{At}(b_k, p_п); \mathrm{At}(y_1, p_п), ..., \mathrm{At}(y_r, p_п); ((M_п = 0) \vee (M_п \ge C_п)).
\end{array}\right\} \quad (3)$$

Здесь α[*e*] — конфигурация α, ограниченная условием *e*: «ни одно выражение формы On *(y, b_k)* не включено в α»; другими словами, все, кто сел на левом берегу, должны высадиться на правый берег.

По-прежнему мы аналогично определяем преобразования для коммуникации $\{y_i\}$ с правого берега на левый, заменяя в выражениях для $З^r$ЛЛ и ПЛЛП+B^rЛП $p_л$ на $p_п$ и $p_п$ на $p_л$ ($З^r$ЛП и ПЛПЛ+B^rЛЛ соответственно).

Преобразованием элементов определения $A_1 \to A_2$ мы делаем первый шаг на пути уменьшения размера пространства поиска решений σ, так как

1) количество промежуточных состояний уменьшается;

2) количество запрещенных состояний, т. е. состояний, в которых неприменимо ни одно из преобразований, увеличивается.

Исследуем на избыточность ограничения на применимость преобразований A_2, а именно покажем, что если в начале и в конце коммуникации с одного берега на другой условия $((M_л = 0) \vee (M_л \ge C_л))$ и $((M_п = 0) \vee (M_п \ge C_п))$ удовлетворяются, то $((M_в = 0) \vee (M_в \ge C_в))$ также удовлетворяется. По предположению удовлетворяются

$$\left.\begin{array}{l}
((M_л = 0) \vee (M_л = C_л) \vee (M_л > C_л)) \\
((M_п = 0) \vee (M_п = C_п) \vee (M_п > C_п))
\end{array}\right\} \quad (4)$$

Легко видеть, что конъюнкция условий (4) эквивалентна

$$(M_л = 0) \vee (M_л = N) \vee (M_л = C_л), \quad (5)$$

так как $M_л + M_п = N$, $C_л + C_g = N$.

Для удовлетворения (5) необходимо, чтобы в лодке ехали либо только людоеды (сохранение $M_л$=0 и $M_л$=N), либо чтобы число миссионеров и людоедов в лодке было одинаковым (сохранение $M_л$=$C_л$), либо чтобы число миссионеров в лодке превышало число людоедов в лодке (переход от $M_л$ =N к $M_л$=$C_л$ или $M_л$=0, или переход от $M_л$=$C_л$ к $M_л$=0). Итак, удовлетворяется условие

$$(M_B = 0) \vee (M_B > C_B). \qquad (6)$$

Поскольку условие (6) является избыточным, мы можем провести еще одно обобщение преобразования, переходя от множества преобразований F_2 к множеству A_3:

$$\left.\begin{array}{l} а)\ \textit{Перевезти } r \textit{ объектов } y_i \textit{ с левого берега на правый} \\ (П^r ЛП): \textit{для множества } \{y_i / i = 1,2,...,r, 1 \le r \le k\} \\ \alpha, \mathrm{At}(b_k, p_л), \mathrm{At}(y_1, p_л),..., \mathrm{At}(y_r, p_л); (M_в + C_в = 0) \to \\ \to \alpha, \mathrm{At}(b_k, p_л), \mathrm{At}(b_k, p_п); \mathrm{At}(y_1, p_п),..., \mathrm{At}(y_r, p_п); \\ (M_в + C_в = 0), ((M_л = 0) \vee (M_л \ge C_л)), ((M_п = 0) \vee (M_п \ge C_п)). \\ б)\ \textit{Перевезти } r \textit{ объектов } y_i \textit{ с правого берега на левый} \\ (П^r ПЛ)\ (\textit{получается, как и ранее, из выражения для } П^r ЛП \\ \textit{заменой } p_л \textit{ на } p_п \textit{ и } p_п \textit{ на } p_л. \end{array}\right\} \qquad (7)$$

Важно подчеркнуть, что исключение избыточных условий привело к обобщению преобразований A_2, т. е. к дальнейшему исключению промежуточных состояний и, в конечном итоге, к сокращению пространства поиска решений.

До сих пор мы не касались другого элемента представления — N-состояния, которое выражалось в форме вида (1). Однако очевидно, что возможен переход от вида (1), вытекающего непосредственно из словесного описания задачи, к виду $(M_л, C_л, B_л)$, где $M_л$ и $C_л$ — целые числа, $0 \le M_л, C_л \le N$, а $B_л$ — булевская переменная, принимающая значение 1, если лодка находится на левом берегу, и 0, если на правом. Возможность такого перехода вытекает из условия e и очевидного соотношения

$$M_Л + M_П = C_Л + C_П = N.$$

Заметим, что выражение (1) для s_0 приобретает вид $(N, N, 1)$, а для конечного состояния s_t — (0, 0, 0). Переформулировка понятия состояния приводит к новому множеству преобразований A_4:

$$
\left.\begin{array}{l}
\text{а) Перевезти пару } (M_в, C_в) \text{ с левого берега на правый} \\
(\text{ПЛП},\ M_в,\ C_в)\text{: для всех пар}(M_в, C_в), \\
1 \le M_B + C_B \le k, \\
(M_л, C_л,\ 1);\ \Lambda \to (M_Л - M_B, C_л - C_в, 0); \\
\big((M_Л - M_B = 0) \vee (M_Л - M_B \ge C_Л - C_B)\big), \\
(\big(\big(N - (M_л - M_B) = 0\big) \vee \big(N - (M_л - M_B) \ge N - (C_л - C_в)\big)\big).
\end{array}\right\} \quad (8)
$$

$$
\left.\begin{array}{l}
\text{б) Перевезти пару } (M_в, C_в) \text{ с правого берега на левый} \\
(\text{ППЛ},\ M_в,\ C_в)\text{: для всех пар } (M_в, C_в), \\
1 \le M_B + C_B \le k, \\
(M_л, C_л,\ 0);\ \Lambda \to (M_Л + M_B, C_л + C_в, 1); \\
\big((M_Л + M_B = 0) \vee (M_Л + M_B \ge C_Л + C_B)\big), \\
(\big(\big(N - (M_л - M_B) = 0\big) \vee \big(N - (M_л + M_B) \ge N - (C_л + C_в)\big)\big).
\end{array}\right\}
$$

Переход $A_3 \to A_4$ избавляет нас от необходимости рассматривать отдельно каждого миссионера и людоеда, а представляет задачу в понятиях последовательности сложения и вычитания векторов, удовлетворяющей специальным условиям и преобразующей начальный вектор в конечный. Общее количество различимых состояний в пространстве σ от такого перехода не меняется, однако существенное упрощение вычислений, необходимых для идентификации каждого состояния, приводит к повышению эффективности РКЗ.

Рассмотрим представление задачи о миссионерах и людоедах в так называемой *системе редукций.* В этой системе состояния (называемые далее *P-состояниями)* отождествляются с выражениями вида $S_i = (s_a \Longrightarrow s_b)$, где s_a, s_b — *N*-состояния, а $\Longrightarrow$ имеет тот же смысл достижимости, что и в системе продукций. *Нетерминальный ход* соответствует применению допустимого преобразования к левому *N*-состоянию *P*-состояния. Таким образом, например, к *P*-состоянию $S_i = (s_a \Longrightarrow s_b)$ применимо преобразование s_a в s_c, и его применение соответствует ходу, редуцирующему S_i к $S_j = (s_c \Longrightarrow s_b)$. Применение этого хода интерпретируется как «если s_b достижимо из s_c, то оно достижимо из s_a» (поскольку известно, что s_c достижимо из s_a). *Терминальный ход* в системе редукций просто распознает, что $(s_t \Longrightarrow s_t)$.

Решение в системе редукций — это последовательность *P*-состояний, достижимых последовательным применением нетерминальных ходов, начиная с начального состояния и кончая состоянием, где применяется терминальный ход.

Таким образом, в этой системе решение получается последовательностью разбиений текущей исходной задачи на пару задач, решение одной из которых известно, а решение другой предстоит найти следующим разбиением в последовательности. Можно видеть, что последовательность левых *N*-состояний из решающей последовательности *P*-состояний представляет собой решающий путь в системе продукций.

В задаче МЛ начальное *P*-состояние s_o=*((N, N,* 1) $\Rightarrow$ (0, 0, 0,)), конечное состояние s_t =((0, 0, 0,) $\Rightarrow$ (0, 0, 0,)), ходы (ПЛП, $M_в$, $C_в$) и (ППЛ, $M_в$, $C_в$) определяются (8), терминальный ход обозначим A_t.

Часть графа решения задачи МЛ для случая N=3, k=2 приведена на рис. 1. Здесь *P*-состояния для простоты представлены своими левыми *N*-состояниями, ветви графа соответствуют ходам, стрелки указывают направления действия ходов. Решающий путь выделен жирной линией. Из рис. 1 видно, что решающий путь в графе *симметричен* относительно хода ППЛ, 1, 1 (*единственного* хода, дающего решение).

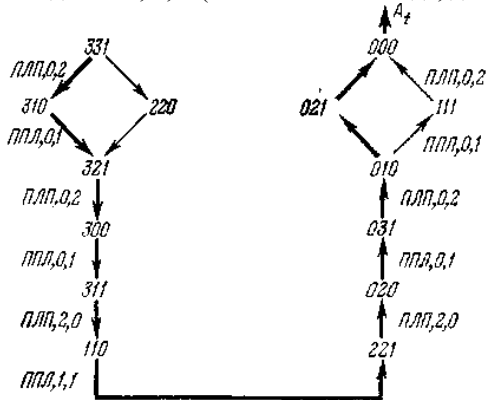


Рис. 1. Часть решающего графа для элементарной задачи о миссионерах и людоедах.

Введем отношение $\tilde{T}$, обратное отношению *T* непосредственной достижимости ($s_a T s_b \leftrightarrow s_b \tilde{T}\, s_a$). Определим отображение θ между *N*-состояниями:

$$\theta\text{: } (M_л, C_л, B_л) \rightarrow (N\text{-}M_л, N\text{-}C_л, 1\text{-}B_л), \qquad (9)$$

или, в терминах векторной операции вычитания,

$$\theta\ (s) = (N, N, 1)\text{-}s. \qquad (10)$$

Следующее определение устанавливает отношение антиизоморфизма относительно θ между σ, пространством *N*-состояний, частично

упорядоченных отношением *T,* и $\tilde{\sigma}$, пространством *N*-состояний, частично упорядоченных отношением $\tilde{T}$.

Утверждение 1. *Для любой пары N -состояний* s_a, s_b,

$s_a T s_b \leftrightarrow \theta(s_a)\,\tilde{T}\,\theta(s_b)$, *или* $s_a T s_b \leftrightarrow \theta(s_b) T \theta(s_a)$.

Следствие 1 из утверждения 1. $(s_a \Rightarrow s_b) \leftrightarrow (\theta(s_b) \Rightarrow \theta(s_a))$.

Следствие 2 из утверждения 1. *Ход, порождающий переход из* s_a *в* s_b, *идентичен ходу, порождающему переход из* $\theta(s_b)$ *в* $\theta(s_a)$.

Установленные свойства пространств σ и $\tilde{\sigma}$ позволяют решать проблему *одновременно* в двух пространствах, т. е. «с начала» и «с конца», осуществляя поиск только с одной стороны (в силу следствия 2).

Мы вновь переформулируем понятие состояния, введя обобщенное *P*-состояние

$$\sum_i = (\{ s_i \} \Rightarrow \{\theta(s_i)\}), \quad i=0, 1,2, \ldots, j, \qquad (11)$$

i — число переходов между начальным или конечным *N*-состоянием и текущим обобщенным *P*-состоянием.

Нетерминальный ход в этом новом представлении (A_5) осуществляет переход в пространстве σ прямым поиском и параллельно вычисляемый на основе свойства симметрии переход в $\tilde{\sigma}$. Терминальный ход распознает, что $s_k T s_l$, $s_k \in \{s_j\}$, $s_l \in \{\theta(s_j)\}$. Решение имеет форму последовательности обобщенных *P*-состояний, начинающейся с $\sum_0 = (s_o \Rightarrow s_t)$ и заканчивающейся обобщенным *P*-состоянием, из которого осуществляется терминальный ход. Решающий коммуникационный путь в этой последовательности представляется последовательностью элементов (директив), выбираемых по одному из множеств

$$\{s_0\} = s_0, \{s_1\}, \ldots, \{s_j\}, \{\theta(s_j)\}, \ldots . \{\theta(s_2)\}, \{\theta(s_1)\}, \{\theta(s_o)\} = s_t.$$

Следующий предпринимаемый нами шаг по пути преобразования представления — поиск некоторых *характеристических образов* в пространстве поиска решений. Поскольку к настоящему моменту количество возможных *N*-состоянии равно $2(N+1)^2$ нам необходим глобальный взгляд на пространство σ, позволяющий

1) отождествить с состояниями целые области этого пространства,

2) соответствующим образом обобщить преобразования.

Мы представим пространство поиска решений в виде квадратного массива точек $(N+1)\times(N+1)$ (пример для элементарной задачи МЛ приведен на рис. 2).

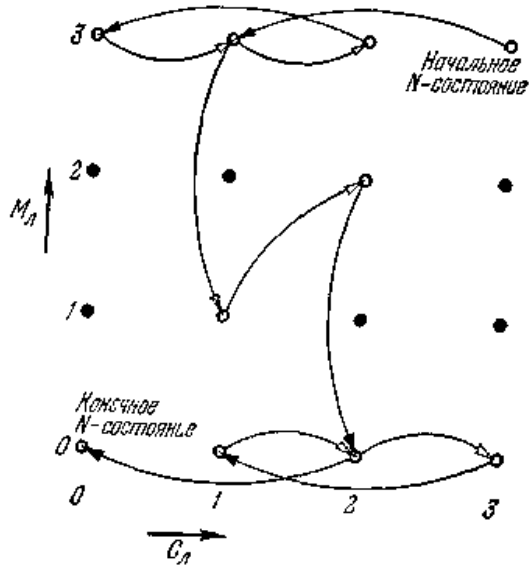


Рис. 2. Пространство *N*-состояний для элементарной задачи о миссионерах и людоедах.

Здесь стрелки с черным концом соответствуют коммуникации с левого берега на правый (ПЛП), с белым концом — коммуникации с правого берега на левый (ППЛ). Точка представляет собой в зависимости от цвета стрелки, с которой мы в нее вошли, одно из двух *N*-состояний — ($M_л$, $C_л$, 1) (белая) или ($M_л$, $C_л$, 0) (черная). Решением в этом представлении является последовательность стрелок с чередующимся цветом концов, начинающаяся в точке *(N, N)* и кончающаяся в точке (0, 0). Представление сформулировано в системе продукций, исходя из множества преобразований A_4.

Заметим, что зачерненные точки в пространстве обозначают запрещенные состояния, а остальные точки образуют зигзагообразную ломаную Z, состоящую из трех отрезков: $M_л=N$, $M_л=C_л$, $M_л=0$ в полном соответствии с (5). Анализ Z-образа показывает (рис. 3), что

1) Любая точка *(N, x,* 1), $1 \le x \le N$, может быть достигнута из любой другой точки *(N, y,* 1), $1 \le y \le N$, некоторой горизонтальной последовательностью переходов

2) Любая точка *(N—x, N—x,* 0), $0 \le x \le k$, $k \ge 2$, на диагонали Z может быть достигнута из любой точки *(N, N—x,* 1), $0 < x \le k$, одним переходом (ПЛП, *x,* 0). В свою очередь из этой точки одним переходом (ППЛ,1,1) может быть достигнута точка *(N—x*+1, *N-x*+1, 1) на диагонали Z. Эта

пара представляет собой составной «стабильный прыжок» (необходимый для дальнейшего перехода на линию $M_л$=0). Итак, наиболее удаленная от начальной точка диагонали, которая может быть достигнута этой парой переходов, *(N—k*+1, *N—k*+1, 1).

3) Точка на линии $M_л$=0 может быть достигнута из точки на диагонали, если расстояние от диагонали до линии не превышает *k.* Отсюда вытекает условие (с учетом симметрии Z)

$$k \geqslant \frac{N+1}{2}. \qquad (12)$$

4) Любая точка (0, *x,* 0), 0≤*x*<*N,* может быть достигнута из любой другой точки (0, *y,* 0), 0≤*y*<*N,* горизонтальной последовательностью переходов.

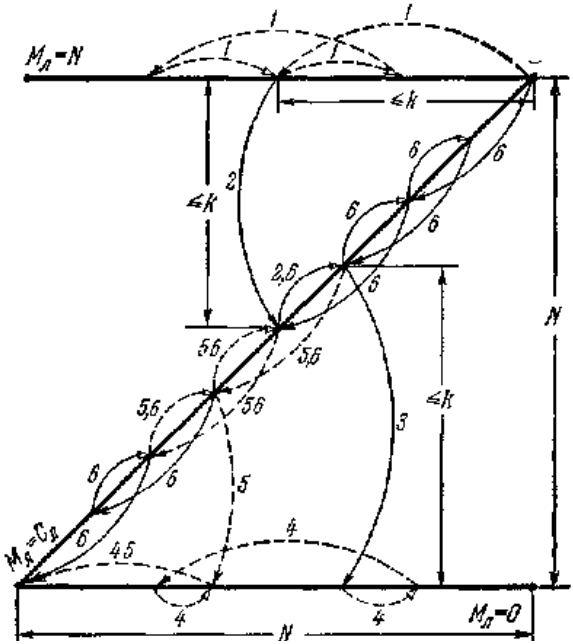


Рис. 3. Схемы решений обобщенной задачи о миссионерах и людоедах в Z области. Цифры у дуг соответствуют пунктам в тексте.

5) Условие (12) работает для *k*<4. В случае *k*≥4 разрешима любая задача МЛ (т. е. для любого *N).* Разрешимость вытекает из возможности построения диагональной последовательности типа (ПЛП, 2, 2), (ППЛ, 1, 1), (ПЛЛ, 2, 2), (ППЛ, 1, 1), . . ., обеспечивающей «спуск» по диагонали до точки *(k, k,* 1).

6) Из изложенного в п. 5) ясно, что можно заменить Z-образ диагональным образом (для $k \geq 4$), т. е. двигаться из начальной точки вниз по диагонали с помощью последовательных пар (ПЛП, $k/2$, $k/2$), (ППЛ, 1, 1) (в случае нечетного k - (ПЛП, $\frac{k-1}{2}$, $\frac{k-1}{2}$) , (ППЛ, 1,1)).

Изложенные результаты анализа позволяют сформулировать представление в системе продукций, в котором множество преобразований A_6 представляет собой специфическое (и потому более мощное) множество макропереходов:

а) Горизонтальные переходы (Г):

Г1: $(N, C_л, 1); 0<C_л< N , k\geq 2 \rightarrow (N, N, 1)$.

Г2: $(0, C_л, 0); 0 \leq C_л < N, k \geq 2 \rightarrow$

$\rightarrow (0, C'_л, 0); 0 \leq C'_л \leq N, C_л \neq C'_л$.

б) Диагональные переходы (Д):

Д: $(M_л, C_л, 1); 0 < M_Л = C_Л \leq N, k \geq 4 \rightarrow (0,0,0)$.

в) Прыжки (П): (13)

П: $(M_л, C_л, 1); 0 < M_Л = C_Л \leq k \rightarrow (0, C_л, 0)$.

г) Составные переходы

(прыжки с горизонтали и диагонали):

ГП: $(N, C_л, 1); 0<C_л \leq N, k \geq 2 \rightarrow$

$\rightarrow (N - k + 1, N - k + 1, 1)$.

ДП: $(M_л, C_Л, 1); M_Л = C_Л > k \geq 4 \rightarrow (0, k, 0)$.

Множество A_6 обеспечивает решение задачи МЛ с помощью Z-образа, диагонального образа, а также смешанного образа, начиная из любого и кончая любым допустимым состоянием.

Введением множества преобразований A_6 мы существенно уменьшили размер пространства поиска решений, исключив значительное число запрещенных и промежуточных состояний. Новое пространство содержит начальное и конечное состояния, 4 состояния $(N, N, 1)$, $(N - k + 1, N - k + 1, 1)$, (0, 0, 0) и (0, k, 0), а также множество N-состояний $\{s/M_л=0, B_л=0, 0<C_л< k\}$, т. е. максимум $5+k$ состояний (вместо $2(N+1)^2$). Путем фиксации одного из трех используемых образов это пространство может быть еще сокращено.

Следует отметить, что новое пространство может быть разделено на три области ($M_л=0$, $M_л = N$, $M_л = C_л$), которые «легко проходимы». Критическими местами являются точки перехода из одной области в другую.

Представляется весьма перспективным нахождение в пространстве поиска решений таких точек путем глобального исследования пространства. Это могло бы существенно повысить эффективность РЗ, особенно в системе редукции.

Заканчивая исследование задачи МЛ, следует подчеркнуть, что оно, к сожалению, носит, как и большое число других исследований в области ИИ, характер *case study* (изучение конкретного опыта), т. е. исследования конкретного опыта, на базе которого сделаны некоторые обобщения.

В частности, становится ясным, что решение проблемы преобразования преобразований требует решения следующих основных вопросов:

1) Выбор основных элементов представления (директив) в исходной системе (интерпретация состояний, ограничений и преобразований по заданному словесному или иному эпистемологически полному описанию).

2) Поиск полезных свойств, уменьшающих размер пространства поиска решений задачи или выделяющих в нем характерные критические точки.

3) Использование опыта исследования задачи для улучшения процесса РЗ.

3.2.3. Методы исследований свойств пространства поиска решений

В предыдущем параграфе мы продемонстрировали на примере некоторые приемы, позволяющие путем глобального исследования пространства поиска решений существенно повысить эффективность решения задачи. В большинстве случаев, как уже отмечалось, такое исследование может быть проведено только для конкретной задачи, причем в общем случае оно еще не поддается полной автоматизации. Тем не менее можно указать некоторые общие методы, которые представляются перспективными с точки зрения их использования в решателях задач.

В настоящем параграфе мы рассмотрим два таких метода.

Первым методом является так называемый *метод образов* с последующим построением на основе образов приближенного плана решения задач.

Пусть задано, в духе предыдущего примера, множество состояний S и множество преобразований $A = \{A_j\}$, $A_j : S \rightarrow 2^S$.

Предположим, что нам заданы некоторые, хотя и не все, характеристики состояний. **Это могут быть системы признаков, характеризующих состояния, меры близости между состояниями или любая другая частичная информация о состояниях.** Таким образом, мы предполагаем, что на множестве состояний задана некоторая выделяющая функция $h: S \to I$, сопоставляющая каждому состоянию образ $h(s)$, и получаем множество образов I, причем каждый $i_k \in I$ представляет подмножество состояний $S(i_k) \subseteq S$. В идеальном случае можно было бы точно сгруппировать состояния в пространстве, т. е. $i_k, i_l \in I$ представляли бы такие подмножества состояний, что

$$S = \bigcup_{i \in I} S(i), \quad S(i_k) \cap S(i_l) = \emptyset.$$

Очевидно, что это соответствовало бы полному знанию свойств состояний в пространстве. В реальных случаях необходимо связать с функцией h некоторую меру неопределенности в виде вероятностных оценок или в духе теории размытых (нечетких) множеств. Можно рассматривать образы как состояния в некотором вспомогательном пространстве и использовать это вспомогательное пространство вместе с множеством преобразований A и выделяющей функцией h для составления планов решения. Если задать меру неопределенности перехода образа $i'=h(s_m)$ в образ $i=h(A_j(s_m))$ в виде $\mu(i, A_j, i')$, где s_m — некоторое состояние, $A_j \in A$ —преобразование, то условия вида $\mu(i, A_j, i') \approx 1$ могут служить основой для стратегии, состоящей в построении плана в пространстве образов, затем решения в пространстве состояний в соответствии с планом, сверки их в точках, удовлетворяющих указанным выше условиям и соответствующей корректировке плана (а не решения) в случае неудачи.

Другой метод - это метод поиска запрещенных областей в пространстве путем *наследственных разбиений*. Идея этого метода состоит в том, чтобы показать, что из некоторого множества состояний целевое состояние недостижимо. Тогда эти состояния могут быть выделены по некоторым признакам (например, объединением в образы) в запрещенное для данного класса целевых состояний множество.

Построение такой опровергающей стратегии может быть проиллюстрировано на примере известной задачи Маккарти: задана шахматная доска, из которой удалены две противоположные клетки по диагонали. Доказать, что такая доска не может быть покрыта домино. Опровержение для этой задачи может быть получено следующим образом. Заметим, что когда мы кладем очередное домино на доску, число белых и черных квадратов, покрытых домино, остается равным.

Это свойство является *наследственным,* так как оно не меняется ни при одном допустимом ходе. Поскольку первоначальное число белых и черных квадратов не равно, то это неравенство не изменится.

Итак, для построения опровергающей стратегии необходимо показать, что

1) Для всех $S_i \subseteq S$, S — множество состояний, истинно свойство $P(S_i)$
2) Для всех s таких что $S_i \Rightarrow s$, также истинно $P(s)$.
3) Для конечного состояния s_t истинно $\sim P(s_t)$. Тогда s_t недостижимо из S_t и $S_i \cup \{s\}$ — запрещенное подмножество состояний.

Наследственные разбиения являются обобщением наследственного свойства. Стратегия этих разбиений получается следующим образом. Пусть $S_i \subseteq S$ может быть разбито на непересекающиеся подмножества, $S_i = \bigcup_j S_{ij}$ $S_{ik} \cap S_{il} = \emptyset$, $k \neq l$. Пусть также для всех j S_{ij} обладают свойствами $P_j(S_{ij})$. Эти же свойства удовлетворяются для всех множеств состояний $\sum_{ij}$ таких, что $S_{ij} \Rightarrow \sigma_{ij}$, $\sigma_{ij} \in \sum_{ij}$. В тоже время ни одно из целевых состояний $s_t \in S_t$ не обладает ни одним из свойств P_j, j=1, 2, . Тогда $S_i = \cup\left(\bigcup_j \Sigma_{ij}\right)$ — запрещенное множество состояний.

Заметим, что и в этом методе может оказаться необходимым связать с разбиением некоторую меру неопределенности, поскольку в реальной ситуации полная информация о свойствах всех состояний может оказаться недоступной.

Изложенные методы подчеркивают общий тезис о необходимости знания общих свойств пространства поиска решений для эффективного преобразования представлений.

Возвращаясь к концу п. 3.2.2, отметим, чго наиболее принципиальным с точки зрения построения эффективных решателей задачи является выбор основных элементов представления (директив) в исходной системе, поскольку этот выбор определяет, в том числе, возможность эффективного преобразования представлений. Это приводит к необходимости определения основных методов представления и решения задач ИИ.

3.3. Краткая характеристика основных классов представлений

Основные методы формирования представлений будут основываться на идее понятийной предикации, т. е. **утверждения истинности или ложности тех или иных свойств в той или иной стадии процесса РЗ**. В рамках этой основной идеи различные методы могут отличаться описательным аппаратом, уровнем определения основных объектов и отношений, формальными моделями решения задач.

Рассмотрим три основных класса методов формирования представления: *декларативные методы формирования представлений, процедуральные методы формирования представлений* и *языковые* (*семантические) методы формирования представлений.*

3.3.1. Декларативные методы формирования представлений

Этот класс уже был частично описан в п. 3.2.2. Здесь система формирования представлений представляется полным описанием состояния и множеством преобразований, или операторов. Полное описание состояния представляется в виде множества утверждений безотносительно к тому, как их использовать. К классу декларативных методов формирования представлений, кроме систем продукций и редукций, введенных в п. 3.2.2, относятся системы доказательства теорем в исчислении предикатов. В этом методе решение задачи формирования представлений сводится к доказательству того, что целевая формула представления логически следует из системы начальных аксиом. Поскольку фактический процесс доказательства теорем включает в себя применение некоторых правил вывода к начальным аксиомам, затем этих же или других правил вывода к аксиомам и выведенным на первом шаге теоремам и т. д., вплоть до получения **целевой формулы представления**, в системе доказательства теорем легко обнаружить общие свойства декларативных методов формирования представлений. Текущее знание системы представляется полным описанием состояния, которое интерпретируется множеством аксиом и всех выведенных (сформированных) к данному моменту теорем, и **множеством операторов, т. е. правил вывода**.

Основными и тесно связанными друг с другом характеристиками декларативных методов формирования представлений являются:

— четкое разделение организации поиска *(механизм генерации),* представляющего собой полный перебор, и сокращения количества альтернатив, т. е. придания поиску узкой направленности *(механизм управления);*

— *универсальный,* т. е. проблемно-независимый *характер механизма генерации;*

— необходимость работы с полными описаниями состояний на каждой стадии процесса РЗ.

Указанные характеристики будут описаны при рассмотрении конкретных декларативных методов формирования представлений (п.п.3.4—3.7) и методов решения задач с использованием этих представлений. Сейчас мы заметим, что эти характеристики предопределяют ряд особенностей декларативных методов формирования представлений.

1. В силу универсальности механизма генерации декларативные представления могут в принципе служить основой для создания универсального РЗ.

2. Эвристическая эффективность РЗ в декларативных представлениях полностью определяется свойствами механизма управления. Механизм управления поиском может быть задан в виде ограничений на допустимые состояния, в виде взаимосвязей между образами подмножеств состояний и операторами или соответствующим образом определенных эвристических функций. Чем точнее определен для каждой конкретной задачи механизм управления, т. е. чем выше знания системы о конкретной проблемной среде формирования представлений, тем эффективнее осуществляется процесс РЗ. Однако, в отличие от упомянутого выше знания, представленного описаниями состояний и операторов — структурного, или *синтаксического знания,*— знание о проблемной среде формирования представлений отражает *понимание* системой глобальных свойств пространства поиска решения, или, что то же самое, законов, действующих в мире (области), в котором работает система формирования представлений. Это *знание* мы называем *семантическим.* Итак,

а) эвристическая эффективность РЗ формирования представлений определяется количеством семантического знания, накопленного системой формирования представлений;

б) структурное и семантическое знание в декларативных представлениях полностью отделены друг от друга.

3. В силу характера указанных выше способов задания механизма управления декларативные методы формирования представлений должны работать хорошо в легко **метризуемых пространствах** поиска решений формирования представлений, т. е. в таких мирах (областях),

где семантическое знание может быть естественно выражено в **численном или логическом виде** (линейные неравенства, аддитивные функции, булевские матрицы и т. д.).

4. Попытки непосредственного введения семантического знания в описания состояний и операторов, т. е. в структурное знание, приводят к потере универсальности механизма генерации, что находится в полном соответствии с законом обратной зависимости общности и мощности.

3.3.2. Процедуральные методы формирования представлений

В то время как **основой описания объектов и отношений между ними в декларативных методах формирования представлений является множество логических представлений и общих правил вывода**, процедуральные методы формирования представлений основаны на описании в виде процедур (директив) всех возможных манипуляций с объектами и отношениями. Таким образом, текущее знание системы формирования представлений представляется в виде **специально организованной базы данных и набора более или менее специализированных процедур (директив), обрабатывающих соответствующие области базы данных.** Формирование в этом классе представлений сводится к построению целенаправленной последовательности процедур (директив) формирования представлений, как правило, имеющей сложную рекурсивную организацию. Важно отметить, что в процедуральном методе формирования представлений, в отличие от декларативного, на каждой стадии процесса РЗ обрабатываются *локальные области базы данных,* причем необходимая для РЗ информация представлена в императивной форме. Носителями процедуральных представлений являются специальные *проблемно-ориентированные языки* (ПОЯ).

Процедуральные методы формирования представлений наиболее естественно и эффективно реализуются в таких языках, которые содержат необходимые встроенные механизмы, обеспечивающие автоматический поиск решения на основе знания и поставленной цели. Существует ряд разработанных таких языков.

Помимо обычных выразительных средств, используемых алгоритмическими языками (структуры данных, управляющие структуры, операторы, процедуры), ПОЯ должны обладать встроенными *целенаправленными механизмами,* которые позволили бы им эпистемологически и эвристически адекватно представлять широкий класс систем формирования представлений и задач ИИ,

решаемых ими. В сущности, ПОЯ для задач ИИ должен сочетать в себе свойства современного алгоритмического языка высокого уровня и возможности решателя задач, являющегося программой, написанной в ПОЯ.

Отсюда вытекает двойственность функций ПОЯ. С одной стороны, он, как и обычный алгоритмический язык, должен упростить работу по программированию путем рационального выбора библиотеки часто встречающихся подпрограмм и макроопераций. С другой стороны, ПОЯ должен обеспечить такое наложение своей структуры и выразительных средств на решатель задач, запрограммированный пользователем, которое обеспечило бы построение эффективных стратегий решения, т. е. «вынуждало» бы пользователя применять именно те представления, которые приводили бы к предпочтительным с точки зрения эффективности стратегиям.

Естественно, что при создании и использовании ПОЯ следует считаться с противоречием между его универсальностью и эффективностью при решении задачи (это уже известная нам зависимость между общностью и мощностью), а также с противоречием между мощностью встроенных механизмов (т. е. объемом спецификаций, задаваемых пользователем) и управляемостью в ходе решения со стороны пользователя (возможностью определить в каждый момент времени состояние процесса решения).

Перечислим теперь основные характеристики процедуральных методов формирования представлений:

— наличие большого количества специализированных стратегий и правил, основанных на ряде унифицированных средств и механизмов, встроенных в ПОЯ;

— введение семантической информации в выражения, хранящиеся в базе данных (в виде свойств этихвыражений);

— использование на каждой стадии процесса РЗ только тех выражений из базы данных, которые необходимы активированной в данный момент процедуре и описаны в ней;

— возможность представления мира (области) на более высоком уровне описания, что в конечном счете сокращает длину решающей последовательности процедур формирования выражений.

Указанные характеристики обеспечивают ряд преимуществ процедуральных методов формирования представлений перед декларативными.

1. Введение семантической информации в описание элементов базы данных позволяет легко формализовать некоторые отношения, трудно выразимые в формальных декларативных методах формирования

представлений (например, равенства, отношения высшего порядка и т. д.).
2. Семантическое знание может быть выражено не только в числовом или логическом виде, но и в виде произвольных символических выражений.
3. Отсутствие трудностей, связанных с обработкой полных описаний выражений.

Указанные преимущества достигаются ценой потери общности по сравнению с декларативными методами формирования представлений. Кроме того, *механизм недетерминистичного выбора*, необходимый в ПОЯ для автоматического поиска решения, в совокупности с весьма слабо развитыми методами накопления семантического знания в процедуральных методах формирования представлений обуславливает значительную неэффективность ПОЯ при решении задач в сложных обстоятельствах. Решение этой проблемы обычно находят в более тесном общении системы ИИ с человеком, на долю которого выпадает обязанность ввода ограничивающего перебор семантического знания (например, путем рекомендаций).

3.3.3. Языковые (семантические методы) формирования представлений

Как уже указывалось выше, одним из недостатков как процедурального, так и в большей степени декларативного методов формирования представлений является разделение структурного и семантического знания, что затрудняет их использование, особенно в реальных областях. Именно в таких областях семантика, являющаяся инструментом повышения эффективности решения задач, играет особо важную роль. Поэтому естественно потребовать, чтобы семантика непосредственно отражалась в самом формализме формирования представления. Другими словами, нам нужен такой уровень сформированного представлений, элементами которого явились бы понятия (директивы) и семантические отношения между ними. **Теоретико-графическое представление такого уровня называется *семантической сетью.*** Существует довольно много вариантов реализации семантических сетей, что позволяет говорить о целом *классе семантических методов формирования представлений.* Общие характеристики этого класса сводятся к следующему:

— описание объектов мира (истин) выводится на уровень естественного языка;

— все знания, включая вновь поступающие факты, а также некоторые специализированные методы решения, накапливаются в относительно однородной структуре памяти;
— на сетях определяется ряд более или менее унифицированных семантических отношений между объектами, которым соответствуют унифицированные методы вывода;
— методы вывода в совокупности с целями (запросами) определяют участки семантического знания, имеющего отношение к поставленной задаче, формулируя акт понимания запроса и некоторую цепь выводов и неполных выводов, соответствующих решению задачи.

Следует отметить следующие особенности семантических представлений:

1. В семантической сети могут быть представлены такие виды объектов, как **понятия, события, специализированные методы решения**; следует, однако, учесть, что увеличение номенклатуры объектов снижает однородность сети и приводит к необходимости увеличения арсенала методов вывода.
2. Многомерность семантических сетей позволяет представлять в них многочисленные семантические отношения, связывающие отдельные понятия, понятия и события впредложениях, а также предложения в текстах; кроме того, в семантической сети может быть отражена семантическая иерархия специализированных методов решения, определяющая их взаимоподчиненность.
3. Формализация, или структурное представление семантических знаний, позволяет наложить на эти знания некоторую *суперсемантику,* отражающую относительную «силу» семантических отношений, что способствует повышению эффективности вывода в семантических сетях.
4. На каждой стадии РЗ можно четко разделить *полное знание* системы (полная семантичесткая сеть) и *текущее знание* — возбужденный участок семантической сети, в котором производятся некоторые операции (процесс понимания, вывода, процесс коммуникации и т. д.).

Необходимо учесть, что, обладая преимуществом структурирования общего семантического знания, семантические методы формирования представлений часто проигрывают в представлении чисто структурных отношений, легко реализуемых в исчислении предикатов (логические связки, кванторы общности и существования) или процедуральном методе формирования представлений (параллельные процессы, гипотетические области, динамические события).

Поэтому ряд исследований в области формирования представлений должен осуществляться путем вложения в семантические методы

формирования представлений некоторых фрагментов процедуральных и декларативных методов формирования представлений с целью объединения их преимуществ в новом, смешанном методе формирования представлений.

Последующие параграфы настоящего раздела посвящены описанию методов формирования представлений и решения задач в этих представлениях. Поскольку процедуральные и семантические методы формирования представлений допускают лишь достаточно специфические, проблемно-зависимые методы решения, мы излагаем лишь некоторые общие механизмы и алгоритмы. В то же время, учитывая универсальность механизмов генерации и независимость механизмов управления в декларативных методах формирования представлений, мы решили выделить этот материал в отдельные разделы (раз. 3.4 — для эвристических методов формирования представлений, раз. 3.5 — для доказательства теорем в исчислении предикатов).

3.4. Эвристические методы формирования представлений на основе декларативных методов формирования представлений

3.4.1. Общая постановка задачи

Одна из многочисленных, но близких друг к другу постановок задачи эвристического метода формирования представлений заключается в следующем: заданы *начальная коммукационная ситуация* (объект, состояние, директива), конечная, или *целевая ситуация* (объект, состояние, директива); задано также *множество операторов (директив),* преобразующих одну ситуацию (предписание) в другую. Требуется найти такую последовательность операторов (директив), которая преобразует начальную ситуацию (предписание) в конечную ситуацию (предписание).

На эту базовую постановку часто булем накладывать ограничения. Так, иногда требуется найти последовательность операторов (директив), оптимальную в некотором определенном смысле. Часто рассматривается обобщенная постановка, где задается множество начальных и (или) конечных состояний.

Формально задача, поставленная как задача формирования представлений эвристическим методом, представляет из себя четверку *(S_o, S, F, T),* где S — множество состояний, $S_0 \subseteq S$ — множество начальных состояний, $T \subseteq S$ — множество конечных состояний, F — множество операторов. Каждый оператор $f \in F$ является функцией,

отображающей S_f в S, где $S_f \subseteq S$ — область определения f. Если $s \in S_f$, то *f применим* к s.

Решением задачи является последовательность операторов $f_1, f_2, ..., f_n$ такая, что $f_i \in F$, $i=1, 2,..., n$, $f_1^\circ f_2^\circ . . .^\circ f_n (s) \in T$, где $f_1^\circ f_2^\circ . . .^\circ f_n (s)$ обозначает *композицию функций*

$$f_n (...(f_2 (f_1 (s))...) , \quad s \in S_0, \qquad (14)$$

причем эта композиция определена, если

$$s \in S_{f_1}, \ f_1(s) \in S_{f_2}, ..., \ f_n(...(f_2(f_1(s)))...) \in S_{f_{n-1}}.$$

Метод решения задачи (S_0, S, F, T) будем называть *эвристическим методом формирования представлений*, если он на каждом шаге находит все возможные применения операторов к данному текущему состоянию, а порядок рассмотрения состояний и порядок применения операторов управляется свойствами уже рассмотренных до этого шага состояний

Будем использовать два основных подхода к решению определенной выше задачи формирования представлений эвристическим методом — основанный на *продукционных системах формирования представлений* (т.е. использующих представление в *пространстве состояний)* и основанный на *редукционных системах формирования* представлений (т. е. использующих сведение решения задачи к решению ее подзадач). Оба эти подхода достаточно наглядно продемонстрированы в п. 3.2.2 на примере решения задачи о миссионерах и людоедах. Поэтому в настоящем параграфе мы лишь дадим общие постановки решения задач в этих системах и подчеркнем связь между продукционными и редукционными системами решения задач.

3.4.2. Формирование представлений в пространстве состояний *(система продукций).*

В системе продукций будем представлять пространство поиска решений в виде *локально-конечного направленного графа* $G=(X, \Gamma)$, где

$X=\{x_0, x_1 . . .\}$ — множество, в общем случае бесконечное, вершин графа, каждая из которых отождествляется с одним из состояний $s \in S$;

$E=\{(x_i, x_j)/x_i, \quad x_j \in X, \quad x_j \in \Gamma(x_i)\}$ — множество дуг, или ребер графа, бесконечное, если бесконечно множество X;

$\Gamma: X \to 2^X$ — конечное отображение, т. е. для всех $x \in X |\Gamma(x)| \in N$; N — целое число;

$|\Gamma(x)|$ — количество *дочерних вершин x,* т. е. вершин, соединенных с x дугой.

В множестве вершин X мы выделяем подмножества вершин $X_0 \subseteq X$, соответствующее множеству начальных состояний $S_0 \subseteq S$, и $X_t \subseteq X$, соответствующее множеству конечных состояний $T \subseteq S$.

Определим *путь в графе* G как $\mu=(x_1, x_2, \ldots, x_k), (x_i,x_{i+1}) \in E$, $i=1, 2, \ldots, k-1$. Если $x_1 \in X_0$, $x_k \in X_t$, то очевидно, что решение задачи эвристического поиска в пространстве состояний (т. е. нахождение последовательности операторов, преобразующей начальное состояние в конечное) сводится к *задаче поиска пути* μ *на графе* G. *Путь* из $x_0 \in X_0$ в $x_t \in X_t$ называется *решающим.*

Для локально-конечного графа G целесообразно *неявное задание,* т. е. определение множества X_0 и множества операторов, которые, будучи применимы к вершине графа, дают все ее дочерние вершины (выполнение условия применимости $\Gamma_i \subseteq \Gamma$ к вершине x_i обязательно, хотя возможно, что для некоторой x_i $\Gamma_i = \varnothing$). При таком задании механизм генерации решений должен строить в явной форме некоторый *подграф неявно заданного графа* G, *содержащий по крайней мере одну конечную вершину.*

Представление пространства поиска решений в виде графа G обеспечивает решение, начиная от начального состояния. Однако в тех случаях, когда целевое состояние явно задано, может оказаться целесообразным проводить поиск пути в графе, начиная от конечной вершины к начальной. Более того, в этом случае можно скомбинировать эти два поиска в единый *двунаправленный поиск* в графе. Интуитивно очевидно, что поскольку поисковые деревья растут экспоненциально, то два поиска с меньшей глубиной могут оказаться эффективнее одного поиска с суммарной глубиной. Алгоритм двунаправленного поиска будет рассмотрен вдругой теме. Отметим, что дополнительно к рассмотренному выше представлению в системе продукций необходимо добавить отображение предшествования Γ^{-1}: $X \rightarrow 2^X$, где

$$\Gamma^{-1}(x_j) = \{x_i / x_i, x_j \in X, x_j \in \Gamma(x_i)\} . \qquad (15)$$

3.4.3. Представление в системе редукций. Пропозициональные графы

Если решение задач в системе продукций сводится к поиску решающего пути, то основной идеей редукционной системы является *поиск доказательства (обоснования) того, что решение данной задачи выводится из решения совокупности ее подзадач.*

Другими словами, решение задачи в этой системе сводится к нахождению множества альтернативных совокупностей подзадач,

каждая из которых дает решение задачи, затем множества альтернативных совокупностей подзадач этих подзадач и т. д. до тех пор, пока задача не станет *разрешимой,* т. е. решение всех ее подзадач не станет очевидным, или пока не будет доказано (обосновано), что задача *не имеет решения.* Очевидность решения подзадач определяется следующими возможностями:

1) Подзадача носит характер общеизвестного утверждения (аксиома).
2) Подзадача легко может быть решена в системе продукций (например, за один шаг).
3) Подзадача хотя и сложна, но ее решение известно системе на основе предыдущего опыта.

Подход с использованием редукционной системы является в некотором роде обобщением подхода с использованием пространства состояний. Действительно, в процессе сведения задачи к совокупности подзадач могут возникнуть различные возможности такого сведения (альтернативные совокупности); в то же время применение оператора в продукционной системе сводит задачу к более простой подзадаче, но эта возможность для данного оператора единственна. С другой стороны, редукцию можно рассматривать и как вспомогательный процесс разбиения задачи поиска решающего пути в пространстве состояний на подзадачи поиска подпутей этого пути с последующей их композицией в окончательном решении задачи.

Как продукционный, так и редукционный подход требуют для решения задачи использования процессов поиска с той только разницей, что в первом случае поиск осуществляется в *пространстве состояний,* а во втором — в *пространстве описаний множеств подзадач.*

Пространство описаний множеств подзадач представляется в виде специального направленного графа *G,* называемого *«И/ИЛИ-графом,* или *пропозициональным графом.*

С каждой вершиной этого графа связывается описание определенной подзадачи. Дуги этого графа соответствуют операторам сведения задачи к подзадачам. В графе выделяются два типа вершин: *конъюнктивные вершины,* или вершины типа «И», которые вместе со своими дочерными вершинами интерпретируются высказыванием «чтобы решить задачу, *необходимо* решить все ее подзадачи», и *дизъюнктивные вершины,* или вершины типа «ИЛИ», которые вместе со своими дочерными вершинами интерпретируются высказыванием «чтобы решить задачу, *достаточно* решить одну из ее подзадач». Дуги, исходящие из конъюнктивной вершины, связаны дужкой при этой вершине. Пример пропозиционального графа при веден на рис. 4.

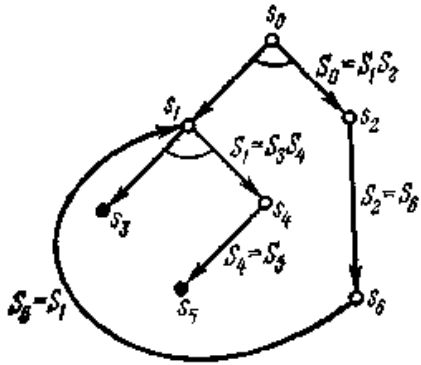


Рис. 4. Пример пропозионального графа. Конечные вершины представлены зачерненными точками.

Здесь s_0 — первоначальная задача, для решения которой *обходимо* решить подзадачи s_1 и s_2, для решения s_1 *необходимо* решить подзадачи s_3 и s_4, для решения s_2 *достаточно* решить s_6, для решения s_4 и s_6 *достаточно* решить s_5 и s_1 соответственно. Решение задач s_3 и s_6 предполагается известным.

В множестве вершин пропозиционального графа выделяются подмножество *начальных вершин,* т. е. задач, которые следует решить, и подмножество *конечных вершин,* т. е. заведомо разрешимых задач. Это завершает формулировку задачи как задачи эвристического поиска, причем решение ее сводится к нахождению в пропозициональном графе *решающего графа,* формальное определение которого будет дано несколько ниже.

С каждой вершиной пропозиционального графа мы связываем высказывание в виде *булевской функции,* выраженной в дизъюнктивной нормальной форме и образующейся по следующим правилам: для вершины s, имеющей дочерние вершины $s_1, s_2, \ldots, s_k$,

а) если s — конъюнктивная вершина, то соотнесенная с ней булевская функция

$$S = \bigwedge_{i=1}^{k} S_i,$$

где S_i — булевская функция, соотнесенная вершине s_i;

б) если s — дизъюнктивная вершина, то соотнесенная с ней булевская функция

$$S = \bigvee_{i=1}^{k} S_i;$$

в) если s — конечная вершина, то соотнесенная с ней булевская функция *тождественно истинна* (из нашего определения конечной вершины пропозиционального графа следует, что вершины, не являющиеся конечными и не имеющие дочерних вершин, соответствуют заведомо неразрешимым задачам; для этих вершин булевская функция *тождественно ложна);*

г) если $s_1, s_2, \ldots, s_m$ — начальные вершины, то с ними соотносится булевская функция

$$S = \bigwedge_{i=1}^{m} S_i.$$

На рис. 4 рядом с вершинами показаны соотнесенные с ними булевские функции.

Введем ряд формальных определений.

Пусть s — дизъюнктивная вершина, т. е. $S = \bigvee_{i=1}^{k} S_i$, s_i — дочерние вершины s. Тогда каждая из S_i, $i = 1, 2, \ldots, k$, называется *непосредственной импликантой S.*

Пусть s — конъюнктивная вершина, т. е. $S = \bigwedge_{i=1}^{k} S_i$, s_i — дочерние вершины s. Тогда *непосредственной импликантой* S называется булевская функция, получаемая из S заменой S_i, $i=1,2, \ldots, k$, одной из ее непосредственных импликант.

Пусть s_1, s_n — конъюнктивные вершины. Тогда S_n является *импликантой* S_1, если имеется последовательность конъюнктивных вершин $s_1, s_2, \ldots s_n$ такая, что S_i является непосредственной импликантой S_{i-1}, $i=2, 3, \ldots, n$.

Пусть s — конъюнктивная вершина с дочерними вершинами $s_1, s_2, \ldots, s_k$. Назовем *путевым графом, начинающимся в вершинах* $s_1, s_2, \ldots, s_k$ *и заканчивающимся в вершинах* $t_1, t_2, \ldots, t_m$, где $s_i, t_l \in G$, $i=1, 2, \ldots, k$; $l=1, 2, \ldots m$, такой конечный подграф G' пропозиционального графа G, что

а) $s_i, t_l \in G'$, $i=1,2, \ldots,k$; $l=1,2, \ldots, m$.

б) Только s_i, $i=1, 2, \ldots, k$, не имеют входящих дуг, а t_l, $l=1, 2, \ldots, m$. — исходящих.

в) Для всех вершин $x \in G'$, кроме t_l, $l=1, 2, \ldots, m$, имеются такие дочерние вершины $x_j \in G'$, $j=1, 2, \ldots, p$, что $X_1, X_2, \ldots, X_p$ являются единственными непосредственными импликангами X в G'.

г) $T = \bigwedge_{l=1}^{m} T_l$ — импликанта S в G'.

Если t_l, $l=1, 2, \ldots, m$,— конечные вершины графа G, то путевой граф называется *решающий графом, начинающимся в вершинах* s_i, $i=1,2,..,k$.

Решающий граф, начинающийся в *начальных* вершинах *G,* называется *решающим графом.* Граф, представленный на рис. 4, является решающим, поскольку, как видно, S_3S_5 является импликантой S_0.

Итак, **решение задачи в системе редукций может быть сведено к поиску *решающего графа исходного пропозиционального графа,* поскольку, как вытекает из предыдущего изложения, просмотр решающего графа от конечных вершин к начальным точно задает множество подзадач, которые необходимо решить для решения исходной задачи, и порядок их решения.** Необходимость поиска решающего графа определяется наличием более чем одной дочерней вершины у дизъюнктивных вершин, или, другими словами, наличием альтернативных совокупностей подзадач, решение которых необходимо для решения исходной задачи.

Поскольку во всех предыдущих рассуждениях не накладывалось никаких ограничений на конечность пропозиционального графа, необходимо его *неявное задание,* т. е. задание множества начальных вершин и оператора Г, генерирующего для данной вершины дочерние вершины и указывающего для нее булевскую функцию в виде конъюнкции дочерних вершин (мы предполагаем, чго оператор, сводящий решение задачи к решению ее подзадач, применяется к конъюнктивным вершинам, в то время как процесс выбора оператора определяет альтернативные совокупности подзадач, т. е. образует вершины, дочерние для дизъюнктивной вершины).

Следует заметить, что любой конечный пропозициональный граф с разделимыми конъюнктивными и дизъюнктивными вершинами может быть отображен в *контекстно-свободную грамматику*. При этом конъюнктивные вершины соответствуют *продукциям,* дизъюнктивные вершины — *вспомогательным символам,* конечные вершины — *основным символам,* дуги из конъюнктивных вершин к дизъюнктивным вершинам определяют *выбор подстановки для переменных,* а дуги из дизъюнктивных вершин к конъюнктивным показывают *действительную подстановку*. На рис. 5 показан пропозициональный граф, соответствующий контекстно-свободной грамматике

$$G = (\{a, b\}, \{S, A\}, S, P),$$
$$P = \{p_1\colon S \to aAS,\ p_2\colon S \to a,\ p_3\colon A \to Sba,\ p_4\colon A \to ba,\ p_5\colon A \to SS\}, \quad (16)$$

где A — вспомогательный символ, S — начальный символ, a, b — основные символы, P — множество продукций.

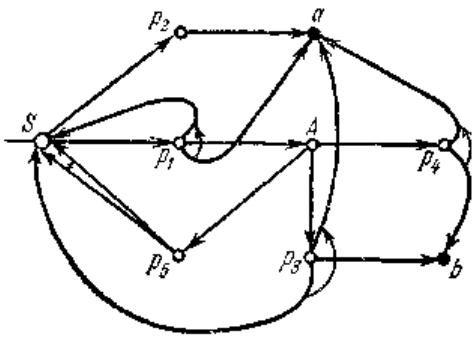


Рис. 5. Представление контекстно-свободной грамматики в виде пропозиционального графа.

Такое представление пропозиционального графа предопределяет упорядочение, накладываемое на порядок генерации вершин, следующих за конъюнктивными (на рис. 5 указано дужками со стрелками).

Таким образом, к **поиску решающего графа в пропозициональном графе могут быть приложены методы теории формальных грамматик**. В частности, решающий граф в пропозициональном графе соответствует *дереву вывода* некоторого высказывания в соответствующей контекстно-свободной грамматике, а поиск такого графа эквивалентен поиску высказывания в языке, соответствующем контекстно-свободной грамматике, вместе с его деревом вывода.

3.4.4. Механизмы сведения задач к подзадачам

На первый взгляд представление в виде пропозиционального графа кажется многообещающей основой для построения универсального решателя задач. Однако будем помнить, что использование теоретико-графической модели позволяет формализацию лишь одного из элементов декларативного представления, а именно пространства описания множеств подзадач. Что же касается преобразований, определенных на множестве вершин графа, то как в настоящем параграфе, так и в следующем разделе, где будут рассмотрены соответствующие алгоритмы, мы лишь считаем, что задано некоторое, как правило, фиксированное множество операторов, позволяющее на каждом шаге порождать все вершины, дочерние по отношению к рассматриваемой. Иначе говоря, теоретико-графическая модель сама по себе не дает сколько-нибудь систематического подхода к решению в общем виде следующей важной задачи: *«каким образом осуществить*

акт разбиения задачи на подзадачи?» Она лишь отвечает на вопрос, *как* решить задачу, если такой подход *существует,* т. е. **представляет собой скелет, на который можно наложить решаемую задачу с соответствующей ей специализированной семантикой описания вершин и дуг графа**, т. е. конкретной интерпретацией описания задачи и ее подзадач, а также допустимых операторов разбиения задач на подзадачи. В этой связи представляют интерес независимые или хотя бы частично независимые от задачи механизмы сведения задач к подзадачам *(механизм редукции).* Шагом на пути к построению такого рода механизмов применительно к представлению в виде пропозиционального графа является использование понятий **ключевых состояний и ключевых операторов.**

Рассмотрим метод сведения задачи к совокупности подзадач, последовательно упрощающий задачи поиска в пространстве состояний, т. е. накладывающий механизм редукции на решение задачи в системе продукций.

Представим задачу поиска в пространстве состояний в виде (S_0, F, T), S_0 — множество начальных состояний, T —множество целевых состояний, F — множество операторов, отображающих состояния в состояния. Пусть также заданы множества *ключевых состояний* $T_1, T_2, \ldots, T_N$, т. е. множества тех состояний, через которые, наиболее вероятно, пройдет решающий путь в графе. Тогда можно использовать механизм редукции для сведения задачи (S_0, F, T) к совокупности задач (S_0, F, T_1), $(\{t_1\}, F, T_2)$, …., $(\{t_N\}, F, T)$, эквивалентной исходной задаче. Здесь $t_1 \in T_1, t_2 \in T_2, \ldots, t_N \in T_N$ — конкретно выбранные ключевые состояния.

Одним из приемов нахождения множеств ключевых состояний является выделение *ключевых операторов,* т. е. операторов, применение которых необходимо для решения задачи (таков, например, оператор ППЛ, 1,1 на рис. 1 в задаче о миссионерах и людоедах). Пусть $f \in F$ — ключевой оператор для задачи (S_0, F, T). Тогда задача может быть разбита на три подзадачи:

1) Поиск пути к состоянию $t \in T_f$, T_f — область определения f, т.е. множество состояний, к которым f применим,— подзадача (S_0, F, T_f).
2) Применение оператора f — подзадача $(\{t\}, F, \{f(t)\})$.
3) Оставшаяся часть задачи — подзадача $(\{f(t)\}, F, T)$.

Заметим, что если бы нам было задано множество операторов $F_k \subset F$ такое, что $f \in F_h$, то это привело бы к необходимости построения пропозиционального графа для получения альтернативных совокупностей трех подзадач указанного выше вида.

Недостаток описанного метода заключается в том, что, за исключением тривиальных случаев, ключевые состояния или операторы могут быть

найдены на основе анализа пространства состояний, а это, как указывалось в п. 3.2, едва ли не самая сложная проблема в области ИИ.

В эвристическом методе поиска основными понятиями являются **состояния и операторы.** Поэтому формальная и содержательная постановка задачи в эвристическом методе полностью совпадает с изложенными в начале этого параграфа.

В процессе выполнения эвристического метода находят *различия* между текущим и целевым состояниями. На основе этих различий выбирается оператор, который применяется к текущему состоянию, вырабатывая новое состояние. Далее производится сравнение этого состояния с целевым, и цикл повторяется. В случае неприменимости выбранного оператора к текущему состоянию определяются различия, суммирующие причину неприменимости. На основе этих различий выбирается оператор, *пригодный* для их устранения. Если он применим и устраняет их, то применяется предыдущий оператор. Однако он может быть *неприменим* или *непригоден,* поэтому изложенная схема работы метода рекурсивна.

Основной механизм редукции использует три стандартных метода (рис. 6): *преобразование состояния A* в состояние *B, уменьшение различия D* между состояниями *A* и *B* и *применение оператора f* к состоянию *A*.

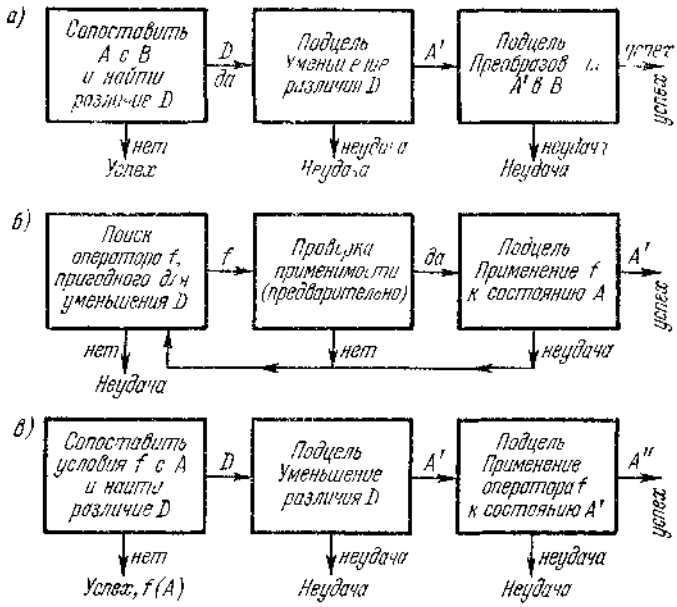


Рис. 6. Основные методы механизма редукции: *а)* метод преобразования состояния *A* в состояние *B, б)* метод уменьшения различия *D* между состояниями *A* и *B, в)* метод применения оператора *f* к состоянию *A*.

Преобразование состояния. Генерируется выведенная (т. е. полученная путем последовательного применения операторов к *A* и следующим состояниям) последовательность состояний, оканчивающаяся состоянием, идентичным *B*.

Уменьшение различия. Вырабатывается новое состояние *A'*, выведенное из *A* с измененным различием *D*.

Применение оператора. Генерируется новое состояние применением *f* к *A* или состоянию, выведенному из *A*.

Пример работы механизма редукции приведен на рис. 7, где изображено дерево методов для преобразования объекта *A* в объект *B* (пример необходимо прослеживать по рекурсивной схеме рис. 6).

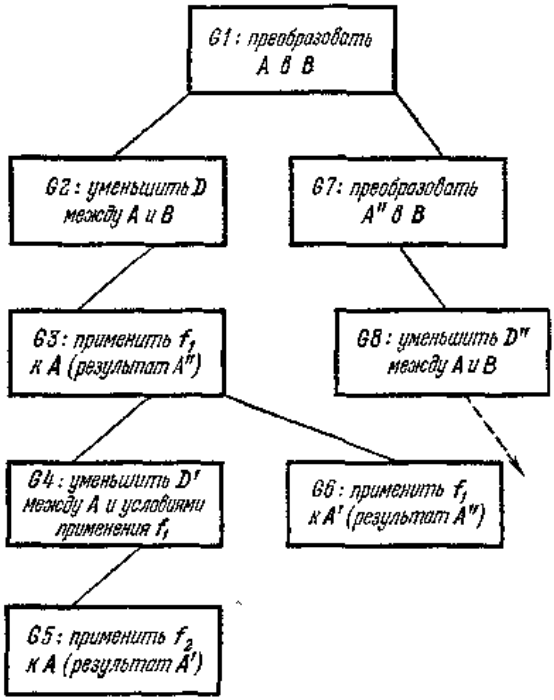


Рис. 7. Пример работы механизма редукции.

Пытаясь преобразовать *A* в *B,* механизм находит различие *D* между *A* и *B* и переходит к его уменьшению (G2), находит оператор f_1, пригодный для уменьшения, и пытается применить его к *A* (G3). Однако оператор f_1 неприменим, и механизм находит различие *D'* и пытается его уменьшить (G4). Предположим, что оператор f_2 пригоден для уменьшения *D'* и применим к *A* (G5). Тогда вырабатывается новое состояние *A'.* Теперь механизм записывает *A*' как результат G5 и G4 и переходит к применению *f*i к *A*'. Поскольку различие *D*' устранено, f_1 применяется к *A*', вырабатывается результат *A*" (G6). Этот результат записывается в G3 и G2. Поскольку различне *D* устранено, производится переход к преобразованию *A*" в *B* (G7). К этому моменту механизм выработал последовательность операторов $f_2 o f_1(A)=f_1(f_2(A))$, преобразующую *A* в *A",* и очередную подзадачу преобразования *A"* в *B.*

Представим процесс редукции (рис.7) с помощью пропозиционального графа (рис. 8).

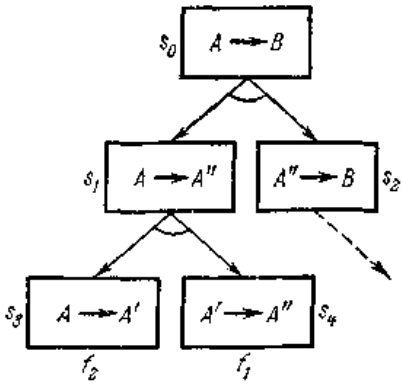


Рис. 8. Пропозициональный граф, соответствующий процессу редукции.

В вершинах графа записаны формулировки исходной задачи и ее подзадач. Граф содержит только конъюнктивные вершины, так как мы предполагали для простоты, что механизм метода обладает способностью выбирать один пригодный оператор. Вершины s_3 и s_4 являются конечными, так как им соотносятся операторы f_2 и f_1 соответственно, непосредственно преобразующие A в A' (f_2) и A' в A'' (f_1).

Из сопоставления двух формализмов — механизма редукции и представления с помощью пропозиционального графа — можно сделать следующие выводы:

1) Механизм редукции метода осуществляет разбиение задачи на подзадачи с помощью метода уменьшения различия, не требуя специального набора операторов для этой цели; однако он должен обладать эффективными методами определения различия между двумя состояниями и выбора оператора, пригодного для уменьшения или устранения этого различия. Решение первой задачи при заданном формализме описания объектов не представляет принципиальных трудностей, чего нельзя сказать о второй задаче.

Что касается представления в виде пропозиционального графа, то без информации о конкретном содержании задачи и о свойствах пространства описаний множества подзадач мы не смогли бы определить множество операторов, преобразующих вершины графа в дочерние вершины, т. е. осуществить разбиение задачи, представленной на рис. 7, на подзадачи.

2) Как будет показано в следующем разделе, представление в виде пропознционального графа допускает построение допустимых алгоритмов, т. е. всегда находящих решение задачи, если оно есть,

а при определенных условиях — алгоритмов, находящих оптимальные решения. В то же время использование механизма редукции даже при наличии специализированной, поставляемой пользователем системы, информации не всегда гарантирует нахождение решения.

Рассмотрим условия, при которых механизм редукции находит решение задачи. Рассмотрим произвольное отображение $S\times S\to D$, где D — множество различий. Это отображение ставит в соответствие каждой паре (s_1, s_2), $s_1, s_2 \in S$, различия $\{d_j\} \subseteq D$.

Введем линейное упорядочение $>$ на множестве D, так что $d_1>d_2$, $d_1,d_2\in D$, означает, что d_1 — *более трудное* для уменьшения *различие,* чем d_2. Это определение не допускает равнотрудных различий, и в этом случае необходимо объединять их в одно различие $d=d_1 \cup d_2$.

Для заданных множества операторов F и множества различий D построим функцию $W: D\times H\to\{0, 1\}$, где H — разбиение, заданное на F. Для $d\in D$, $h\in H$, $f\in h$ $W(d, h)=1$ означает, что f пригоден для уменьшения различия d, $W(d, h)=0$ означает, что f непригоден для уменьшения различия d.

Функция W, выраженная в табличной форме, носит название *таблицы связок* и показывает, какие из групп операторов являются пригодными для уменьшения тех или иных различий.

Построим специальный вид таблицы связок — *треугольную таблицу*. Каждому d_i присваивается $h_i\in H$ так, чю $W(d_i, h_i)=1$, а $W(d_k, h_i)=0$ для всех $d_k>d_i$ (индексы присваиваются так, что если $d_k>d_i$, то $k>i$).

Таким образом, мы строим треугольную таблицу, производя такие разбиения множества операторов, чтобы каждая группа операторов была пригодной для уменьшения определенного различия, но не уменьшала бы различия большей трудности. По отношению к различиям меньшей сложности она может быть как пригодной, так и непригодной.

Мы рассматриваем далее класс задач эвристического поиска — *А-задачи,*— который описывается пятеркой (S_0, S, F, T, W), W — треугольная таблица связок. В процессе разбиения задачи на подзадачи образуется два типа подзадач: непосредственно решаемые и подлежащие дальнейшему разбиению. Их результаты обозначим через $R_0(\sigma)$ и $R_1(\sigma)$ соответственно, σ — некоторое промежуточное состояние. Определим также *максимальное различие между* s_1 *и* s_2, $s_1, s_2\in S$, с помощью функции $M: S\times S\to D$. $M(s_1,s_2)=d_i$ тогда и только тогда, когда $d_i>d_j$ для всех $d_j\in\{d_j\}$. Заметим, что в случае $s_1=s_2$ $M(s_1, s_2)$ не определена. *Максимальное различие между* $s_1\in S$ *и* $X\subseteq S$, $MM(s_1, X)=\min\limits_{s_2\in X} M(s_1, s_2)$, причем MM не определена, если $S_i\in X$.

Если $f \in h_i$, $MM(\sigma,T)=d_i$ и, следовательно, $W(d_i,h_i)=1$, то $R_0(\sigma)=\{f(\sigma)/\sigma \in S_f\}$, $R_1(\sigma)=\{f(\tau)/\tau \in S_f$ и существует решение подзадачи $(\sigma,S, \bigcup_{k=1}^{i-1} h_k, \{\tau\}, W)\}$.

Δ*-схема* Δ-задачи (S_0, S, F, T, W) есть последовательность $(s_0, s_1,.....s_n)$, $s_0 \in S_0$, $s_n \in T$, такая, что $s_i \in R_0(s_{i-1}) \cup R_1(s_{i-1})$, i=l,2,. . .,n. Теперь становится ясной важность введенной треугольной таблицы связок. Действительно, пусть σ — элемент Δ-схемы и $MM(\sigma, T)=d_i$. Тогда мы используем $f \in h_i$ для уменьшения d_i. Если $\sigma \in S_f$, т.е. f применим, то $f(\sigma) \in R_0(\sigma)$. В противном случае ставится подзадача преобразования σ в S_f. Однако для решения этой подзадачи используются только операторы из $\bigcup_{k=1}^{i-1} h_k$. Если результатом решения этой подзадачи является $\tau \in S_f$, то $f(\tau) \in R_1(\sigma)$.

Пусть $f_1 o f_2 o. . . o f_n(s_0)=t$, $t \in T$. Решение Δ-задачи *упорядочено* тогда и только тогда, если $MM(s_0, T)=M(s_0, t) \geq MM(f_1(s_0), T)=M(f_1, (s_0), t)$ и $M(f_1 o f_2 o. . . o f_i(s_0), t) \geq MM(f_1 o f_2 o. . . o f_{i+1}(s_0), T) = M(f_1 o f_2 o. . . o f_{i+1}(s_0), t)$, i=1, 2, . . .,n-2.

Таким образом, вырабатывая упорядоченное решение, механизм редукции последовательно уменьшает монотонно невозрастающую последовательность различий. На каждом шаге из множества конечных объектов рассматривается тот, различие которого с текущим объектом является минимальным, т. е. $M(\sigma, t)=MM(q, T)$ для любых $\sigma \in S$, $t \in T$.

Можно показать, что *наличие упорядоченного решения* Δ*-задачи и всех ее подзадач дает достаточные условия того, что механизм редукции найдет решение задачи, если оно есть*.

Преимущества этого подхода заключаются в том, что

1) На каждом шаге решения задачи механизм рассматривает лишь подмножества множества операторов, причем в ходе решения задачи эти подмножества последовательно сокращаются.

2) Механизм рассматривает лишь те подзадачи, которые легче (опять в смысле используемого подмножества операторов), чем образующая их задача.

Однако пользователь системы должен задать ей множество различий, их упорядочение и таблицу связок.

3.5. Методы доказательства (обоснования) теорем на основе декларативных представлений

В данном параграфе мы изложим еще один вид декларативного представления, используемого как для представления знаний, так и для сведения процесса решения задачи или ее части к автоматическому логическому анализу. Постановка задачи при указанном подходе заключается в следующем. Задача записывается в виде директив (утверждений) некоторого формального языка. При этом часть директив, соответствующая исходным данным, рассматривается как аксиомы, а цель задачи рассматривается как утверждение, справедливость которого следует установить или опровергнуть на основании аксиом и правил вывода формальной системы.

Существуют различные логические формализмы, пригодные для записи в нихутверждений, относящихся к широкому кругу задач.

Мы будем далее рассматривать только исчисления предикатов первого порядка с равенством и без равенства, так как для этих исчислений разработаны универсальные и эффективные процедуры, обладающие *полнотой,* т. е. всегда устанавливающие наличие некоторого факта, если он выводим из аксиом (более подротно об использовании исчислений предикатов как изыка ИИ см. в следующих разделах (темах) .

3.5.1. Язык исчисления предикатов

Любая логическая система (теория) и исчисление предикатов, в частности, может быть построена на базе как синтаксических, так и семантических концепций.

Теорию, построенную на базе **семантических концепций, будем называть *теорией моделей***, а теорию, построенную на базе **синтаксических концепций, — *аксиоматической теорией***. При обоих способах построения некоторой теории А необходимо определить понятие алфавита и формулы предписания (директивы).

Алфавитом называется некоторое счетное множество символов теории. Произвольные конечные последовательности символов алфавита называются *выражениями* теории А.

Формулой теории А будем называть некоторое выделенное подмножество теории А. Алфавит исчисления предикатов состоит из следующего множества символов:

1. Знаков пунктуации (,) .
2. Пропозициональных связок ~, $\vee$, $\wedge$, $\rightarrow$.

3. Знаков кванторов $\forall, \exists$.

4. Символов переменных x_k, k=1, 2, ...

5. n-местных (размерных) функциональных букв f^n_k, $k \geq 1$, $n \geq 0$, f^0_k называют константными буквами.

6. n-местных предикатных букв (символов) p^n_k, $k \geq 1$, $n \geq 1$.

В дальнейшем в примерах для удобства употребления будем вместо x_k писать *u, v, w, x, y, z;* вместо f^0_k — *a, b, c, d;* вместо f^n_k, $n \neq 0$, — *f, g, h,* φ, а вместо p^n_k — *P, Q, R, S, T, V, W.*

Из символов алфавита можно строить различные выражения. Выделим среди них те, которые представляют для нас интерес.

1. *Термы.*

а) Каждый символ переменной или константной буквы является термом.

б) Если $t_1, \ldots, t_n$, $n \geq 1$,— термы, то и $f^n_k(t_1, \ldots, t_n)$ является термом.

в) Выражение является термом только в том случае, если это следует из правил а) и б).

2. *Элементарные формулы (атомы).*

Если p^n_k —предикатная буква, а $t_1, \ldots, t_n$ —термы, то $p^n_k(t_1, \ldots, t_n)$ — элементарная формула (атом).

3. Формулы, или *правильно построенные формулы (ппф).*

а) Всякая элементарная формула есть формула.

б) Если D и B — формулы и x — переменная, то каждое из выражений (~D), $(D \vee B)$, $(D \wedge B)$, $(D \rightarrow B)$, $(\forall xD)$, $(\exists xD)$ есть формула.

в) Выражение является формулой только в том случае, если это следует из правил а) и б).

В выражениях $(\forall yD)$ и $(\exists yD)$ D называется *областью действия квантора всеобщности (общности) и квантора существования соответственно.* При этом переменная y называется связанной квантором (несвободной). (Для указания области действия кванторов будем также использовать нотацию $(\forall y)$ (D) и $(\exists y)$ (D), эквивалентную введенной выше.)

Формула называется *замкнутой,* если она не содержит свободных переменных. Нас будут интересовать именно такие формулы.

При определении логической системы с семантической точки зрения вводят понятия **интерпретации, общезначимости и выполнимости**.

Для того чтобы придать формуле содержание, ее интерпретируют какутверждение, касающееся рассматриваемой области.

Под *интерпретацией* будем понимать всякую систему, состоящую из непустого множества Е, называемого *областью интерпретации,* и какого-либо соответствия, относящего каждой предикатной букве p^n_k

некоторое *n*-местное отношение в E, каждой функциональной букве f^n_k — некоторую *n*-местную функцию в E (т. е. функцию, отображающую E^n в E) и каждой константной букве f^0_k — некоторый элемент из E. Предметные переменные мыслятся пробегающими область E интерпретации. При заданной интерпретации всякой элементарной формуле приписывается значение «истинно» *(T)* или «ложно» *(F).* Приписывание значения элементарной формуле директивы p^n_k $(t_1,. ., t_n)$ осуществляется по следующему правилу: если термы предикатной буквы соответствуют элементам из E, удовлетворяющим отношению, определяемому данной интерпретацией, то значением элементарной формулы директивы будет истина *T,* в противном случае — ложь *F.*

Значение неэлементарной формулы директивы вычисляется рекуррентно, исходя из значений составляющих ее формул. При этом, если *D* и *B* — формулы, то значения формул $\sim D$, $D \vee B$, $D \wedge B$, $D \rightarrow B$ определяются по следующей *таблице истинности:*

D	B	$\sim D$	$D \vee B$	$D \wedge B$	$D \rightarrow B$
T	T	F	T	T	T
F	T	F	T	F	T
T	F	F	T	F	F
F	F	T	F	F	T

Отметим, что формула $(\forall xD)$ обозначает утверждение: «для любого значения *x* из области E истинно (выполнено) *D»,* а формула $(\exists xD)$ обозначает утверждение: «существует такое значение *x* из области E, что истинно (выполнено) *D*».

Приведенные выше утверждения могут быть как истинны, так и ложны. В случае конечных областей E значения истинности таких формул можно установить с помощью таблиц истинности.

Очевидно, что некоторые формулы могут быть истинными или ложными в зависимости от выбранных интерпретаций.

Формула *D* называется *выполнимой* тогда и только тогда, когда существует интерпретация *f* такая, что *D* принимает значение *T* в *I.* Если формула *D* принимает значение *T* в интерпретации *I*, то будем говорить, что *I* есть *модель D,* или *I удовлетворяет* формуле *D.*

Если некоторая формула принимает значение *T* при всех интерпретациях, то ее будем называть *общезначимой.* Так, например, формула $P(a) \rightarrow (P(a) \vee P(b))$ истинна при любой интерпретации (это

можно установить по таблице истинности) и, следовательно, эта формула общезначима.

Если формула D принимает значение F в интерпретации I, то будем говорить, что I *не удовлетворяет* формуле D.

Формула называется *невыполнимой (неудовлетворимой),* если при всех интерпретациях она принимает значение P. Очевидно, что если формула D общезначима, то формула $(\sim D)$ невыполнима.

Введенные выше определения выполнимости, общезначимости, невыполнимости модели некоторой формулы D переносятся на множество формул; при этом предполагается, что все формулы множества связаны знаком конъюнкции. Таким образом, некоторое множество формул $D_1, \ldots, D_n$ выполнено на данной интерпретации, если каждая формула D_i этого множества имеет значение T на данной интерпретации.

Формула B *логически следует* из некоторого множества формул $S=\{D_1, \ldots, D_n\}$, если каждая интерпретация, удовлетворяющая S, удовлетворяет также и B.

Задачей доказательства (обоснования) теорем мы будем называть выяснение вопроса логического следования некоторой формулы B из заданного множества формул $\{D_1, \ldots, D_n\}$, т. е. выяснения общезначимости формулы $((D_1 \wedge \ldots \wedge D_n) \rightarrow B)$.

Однако, как показал Чёрч, не существует общего метода для установления общезначимости любых формул исчисления предикатов первого порядка. По этой причине исчисление предикатов называют *неразрешимым.* Тем не менее из теоремы Эрбрана следует, что если некоторая формула исчисления предикатов общезначима, то существует процедура для проверки ее общезначимости, т. е. исчисление предикатов можно назвать *полуразрешимым.*

При формировании формул оказывается более удобным определять невыполнимость, а не общезначимость. Поэтому рекомендуется рассматривать формулу $\sim((D_1 \wedge \ldots \wedge D_n) \rightarrow B)$, являющуюся отрицанием исходной. Формула $\sim((D_1 \wedge \ldots \wedge D_n) \rightarrow B)$ эквивалентна формуле $(D_1 \wedge \ldots \wedge D_n \wedge \sim B)$, и именно невыполнимость этой последней формулы и следует доказывать (обосновывать). Для установления невыполнимости необходимо доказать (обосновать), что не существует такой интерпретации, при которой каждая из формул множества $D_1, \ldots, D_n, \sim B$ имеет значение T.

В связи с полуразрешимостью исчисления предикатов эта процедура (директива) будет приводить к успеху только в случае, если формула B следует из $D_1, \ldots, D_n$. В противном случае процедура (директива) может продолжаться бесконечно.

Процесс установления невыполнимости некоторого множества формул будем называть *процессом опровержения.*

Как мы указали выше, кроме определенного нами семантического способа задания логической теории, существует синтаксический способ. При этом способе, кроме алфавита и формул, определяемых так же, как и раньше, задаются **аксиомы и правила вывода**.

Аксиомами **называют некоторое выделенное множество формул теории.** Обычно существует возможность эффективно выяснить, является ли данная формула теории А аксиомой. В таком случае А называется *аксиоматической теорией.*

Правилами вывода формул (предписаний) будем называть конечное множество R_1,... . . ., R_n отношений между формулами. Для каждого отношения R_i существует такое целое положительное число j, что для каждого множества D_1, . . ., D_j формул и для каждой формулы B эффективно решается вопрос о том, находятся ли эти j формул в отношении R_i с формулой B, и если да, то B называется *непосредственным следствием* данных j формул по правилу R_i.

Выводом **в теории А называется такая последовательность формул D_1, . . ., D_n, в которой для любого i формула D_i есть либо аксиома теории А, либо непосредственное следствие каких-либо предыдущих формул по одному из правил вывода.**

Формулу B теории А будем называть теоремой теории А , если существует вывод в этой теории, в котором последней формулой является B. Такой вывод будем назывть *выводом формулы B.*

Теория А называется *разрешимой,* если существует единая эффективная процедура (алгоритм), позволяющая узнать для любой данной формулы, существует ли ее вывод в теории А.

Логическая теория А называется *непротиворечивой,* если не существует формулы B такой, чтобы B и ($\sim B$) были теориямив А.

Известно, что всякое исчисление первого порядка непротиворечиво.

Теорема Гёделя о полноте устанавливает эквивалентность семантической и синтаксической точек зрения:

Во всяком исчислении предикатов первого порядка теоремами являются все те и только те формулы, которые логически общезначимы.

Итак, мы определили язык исчисления предикатов первого порядка для записи утверждений, являющихся исходными данными { D_1, . . ., D_n }, и определения B, справедливость которого следует установить. Справедливость определения B сводится к доказательству того, что формула $((D_1 \wedge \ldots \wedge D_n) \rightarrow B)$ является общезначимой (т. е. является теоремой).

Для определения невыполнимости и выводимости формулы ее удобно представить в виде **дизъюнктов (предложений).** Всякую формулу можно представить в виде дизъюнктов, применив к ней последовательность приведенных ниже простых операций.

1. Переименование переменных. Выполняется такая замена переменных, что все переменные, связанные кванторами, становятся различными. Например, $\forall xR(x) \vee \forall xS(x)$ переписывается в виде $\forall xR(x) \vee \forall yS(x)$.

2. Исключение знака импликации. Всякий раз, когда встречается $\rightarrow$, делается замена $(A \rightarrow B)$ на $((\sim A) \vee B)$.

3. Уменьшение области действия связки ~. Везде, где возможно, делаются замены:

Заменяется $\sim\sim A$ на A.

Заменяется $\sim(A \vee B)$ на $\sim A \wedge \sim B$.

Заменяется $\sim(A \wedge B)$ на $\sim A \vee \sim B$.

Заменяется $\sim(\forall x A)$ на $\exists x(\sim A)$.

Заменяется $\sim(\exists x A)$ на $\forall x(\sim A)$.

В конце концов получается формула, где связка встречается непосредственно перед атомной формулой.

4. Исключение кванторов существования. Вычеркиваются поочередно кванторы существования. При этом каждая переменная y, связанная квантором существования, заменяется на $g(x_1, ..., x_m)$, где g — символ новой (отличной от имеющихся в формуле) функции, а $x_1, \ldots, x_m$ — все переменные, встречающиеся в кванторах всеобщности, области действия которых содержат вычеркиваемый квантор существования. Если таких переменных нет, то y заменяется на новую константу.

5. Приведение к предваренной нормальной форме. Все кванторы общности переносятся влево в начало формулы, так что формула принимает вид $\forall x_1 \forall x_2 \ldots \forall x_n A$, где A не содержит кванторов.

6. Приведение к конъюнктивной нормальной форме. Приведение осуществляется заменой, пока это возможно, $(A \wedge B) \vee C$ на $(A \vee C) \wedge (B \vee C)$.

В результате применения шагов 1—6 получаем выражение

$$\forall x_1 \forall x_2 \ldots \forall x_n (A_1 \wedge \ldots \wedge A_n),$$

где A_i имеет вид $(l^i_1 \vee \ldots \vee l^i_r)$, а l^i_j есть атомная формула или ее отрицание. Атомную формулу или ее отрицание будем называть *литерой.* Если A — атомная формула, то литеры A и $\sim A$ будем называть *дополнительными (комплементарными) литерами.*

***7.* Исключение кванторов всеобщности**. Так как все переменные связаны кванторами всеобщности, а порядок расположения кванторов

безразличен, то не будем указывать кванторы явным образом. Будем называть этот вид представления бескванторной нормальной формой.

8. Исключение связок $\wedge$. Исключаем связку $\wedge$, заменяя $A \wedge B$ на две формулы *A, B.* В результате многократной замены получим множество формул, каждая из которых представляет собой дизъюнкцию литер, называемую *предложением (дизъюнктом).*

Исходное множество формул *A* является невыполнимым тогда и только тогда, когда невыполнимо множество *A′,* полученное из *A* применением указанных восьми операций.

Рассмотрим теперь процесс поиска доказательства (обоснования). Покажем, что он может быть представлен в виде поиска пути на графе специального вида, называемом *графом доказательства теорем и/или директив.* **Задача доказательства начинается с непустого исходного множества формул B_0 и множества правил вывода *R*.** Если $\varphi \in R$, а *B* есть некоторое множество формул, то $\varphi(B)$ есть множество выводимых формул.

$\varphi(B)=\emptyset$, если φ не применимо к *B.* В частности, $\varphi(B) = \emptyset$, если *B* не является конечным. Пусть *B** будет множество всех формул, которые могут быть выведены из B_0 повторным применением правил из *R.* Тогда каждое $\varphi \in R$ есть функция $\varphi : 2^{B^*} \to 2^{B^*}$, определенная на подмножествах *B** и принимающая в качестве значений подмножества *B*.* Каждой формуле $C \in B^*$ может быть присвоен уровень: если $C \in B_0$, то *C* присваивается нулевой уровень, в противном случае $C \in \varphi(B)$ для некоторого $\varphi \in R$, и для некоторого $B \subseteq B^*$ уровень *C* на единицу больше, чем уровень некоторой формулы $D \in B$, имеющей максимальный уровень в *B*. Если B_i есть множество всех формул уровня *i*, то $B^* = \bigcup_i B_i$. Формула $C \in B^*$ может иметь несколько различных выводов, поэтому формула *C* может иметь несколько уровней. Так как $\varphi(B) \neq \emptyset$, только если *B* является конечным, то множество формул, которые встречаются в данном выводе формулы $C \in B^*$, всегда является конечным.

Итак, **задача доказательства теорем (директив) для тройки (B_0, R, F), $F \subseteq B^*$ (где *F* — множество терминальных формул) состоит в генерировании с помощью некоторой стратегии формирования $\sum$ формулы теоремы (директивы) $C^* \in F$ повторным применением правил из *R*, начиная с формул в B_0.**

Тройка (B_0, R, F) определяет направленный граф, чьи вершины являются формулами $C \in B^*$. C' является непосредственным

преемником C (т. е. C' связано с C дутой, направленной из C в C'), если для некоторого $B \subseteq B^*$ и $\varphi \in R$ $C \in B$ и $C' \in \varphi(B)$. Как уже было отмечено выше, формула C может иметь несколько выводов, т. е. несколько путей в графе, определяемом тройкой (B_0, R, F). Удобно представлять единственную вершину d как различные вершины $n_1,\ldots . . ., n_k$ в некотором графе (называемом квазидеревом), если вершина d может быть получена различными путями из начальной вершины a. Такое взаимно однозначное соответствие между вершинами и выводами (путями) позволяет рассматривать число вершин, генерированных стратегией $\sum$ в ходе вывода, как **меру эффективности $\sum$ для данной задачи**.

Определим теперь понятие *абстрактного графа доказательства теорем (директив)* (G, s), который может быть интерпретирован как рассмотренная нами выше задача доказательства теорем (директив) (B_0, R) пометкой вершин $n \in G$ метящей функцией $c: G \rightarrow R^*$ и рассмотрением каждого применения функции s к подмножеству $G' \subseteq G$ как применения функции $\varphi \in R$ к подмножеству $\{c(n) | n \in G'\}$. **Абстрактный граф доказательства теорем (директив) есть пара (G, s)**, где G —множество вершин, а s: $2^G \rightarrow 2^G$ есть функция преемствования, определенная на подмножествах G и принимающая в качестве значений подмножества G.

G и s удовлетворяют следующим условиям:

1. $s(\emptyset) = \emptyset$.
2. Если $s(G') \neq \emptyset$, то G' является конечным множеством.
3. Если $G' \neq G''$, то $s(G') \cap s(G'') = \emptyset$.
4. Пусть $G_o = \{n \in G | n \notin s(G')$ для любого $G' \subseteq G\}$,

$G_{k+1} = \{n \in G | n \in s(G')$ для некоторого $G \subseteq \bigcup_{i \leq k} G_i, G' \cap G_k \neq \emptyset\}$.

Тогда

а) $G_o \neq \emptyset$.

б) $G = \bigcup_{0 \leq i} G_i$.

в) $G_i \cap G_j \neq \emptyset$ для $i \neq j$.

Условие 3 утверждает, что различным множествам вершин соответствуют различные множества преемников. Именно это условие обеспечивает то, что граф (G, s) является квазидеревом. Условие 4 определяет, что в графе (G, s) каждой вершине $n \in G$ может быть присвоен единственный уровень i, т. е. $n \in G_i$ и $n \notin G_j$ для всех $i \neq j$. Если (B_0, R) является интерпретацией (G, s) с метящей функцией $c: G \rightarrow B^*$,

то $B_i = \{c(n) | n \in G_i\}$ есть множество помеченных вершин уровня i. **Условие 3 гарантирует, что для каждой формулы $C \in B^*$ и для каждого вывода C из B_0 существует своя вершина $n \in G$ такая, что $C=c(n)$.**

Отметим, что не требуется конечность множеств G_0 и B_0. Это позволяет иметь дело со схемами аксиом и потенциально бесконечными множествами начальных вершин G_0.

Функции преемствования s графа (G, s) определяет частичное упорядочение вершин в графе G: n' есть непосредственный преемник n (а n есть непосредственный предшественник n'), если $n' \in s(G')$ и $n \in G'$ для некоторого $G' \subseteq G$. Вершина n' есть *преемник n (n — предшественник n')* и обозначается $n'>n$, если n' есть непосредственный преемник n или если n' есть преемник непосредственного преемника n. Будем записывать $n \leq n'$, если $n<n'$ или $n=n'$. Определение графа (G, s) гарантирует, что для всех $n \in G$ множество $\{n' | n' \leq n\}$ является конечным, хотя множество $\{n' | n' \geq n\}$ может быть бесконечным. Заметим, что в интерпретации графа (G, s) вывод формулы $c(n)$ состоит из всех формул $c(n')$, где $n' \leq n$. Каждый такой вывод содержит конечное количество формул $c(n')$.

На рис. 9 приведен пример графа (G, s).

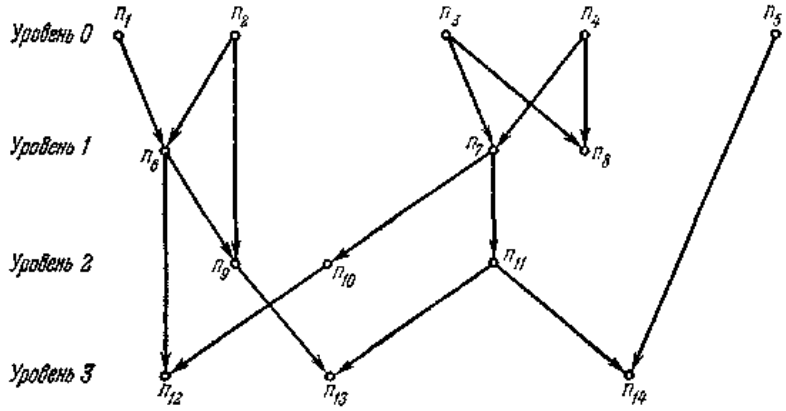


Рис. 9. Абстрактный граф доказательства теорем.

Вершины графа представляются точками, а точки n и n' связываются дугой, направленной от n к n', если n' — непосредственный преемник n. Удобно изображать эти графы направленными вниз так, что n лежит выше n'. Для того чтобы определить на рис. 9, принадлежит ли n множеству $s(G')$, достаточно проверить, что G' есть множество всех тех

вершин, которые связаны с n дугой, направленной к n'. Так, в приведенном примере

$$s(n_1, n_2) = \{n_6\},$$
$$s(n_2, n_6) = \{n_9\},$$
$$s(n_3, n_4) = \{n_7, n_8\},$$
$$s(n_7) = \{n_{10}, n_{11}\},$$
$$s(n_2) = s(n_5) = s(n_8) = s(n_1, n_2, n_6) = \Phi.$$

Абстрактная задача доказательства теорем может быть представлена в виде четверки $P=(G, s, F, g)$, где $F \subseteq G$ является множеством терминальных (решающих) вершин для P, и g есть некоторая оценка, выражающая меру сложности вывода. Далее мы приведем конкретные алгоритмы поиска и правила вывода, интерпретирующие абстрактную задачу доказательства теорем.

3.5.2. Применение метода доказательства теорем

К представлению в виде доказательства теорем (директив) может быть сведен очень широкий круг задач. Это позволяет не только отвечать на вопрос, следует ли логически утверждение (формула) C из некоторого множества $D_1, \ldots, D_n$ утверждений. Он позволяет отвечать на вопрос, следует ли из исходного множества утверждений утверждение $(\exists x C(x))$, и если следует, то каково то частное значение переменной x, при котором это имеет место. Умение строить удовлетворяющие частные случаи для переменной, относящейся к квантору существования, позволяет ставить довольно общие вопросы. Например, мы могли бы задаться вопросом: «Существует ли такая последовательность действий программы для случая игры в шахматы, которая приводит к ее победе?» Следует, однако, помнить, что сложные вопросы могут привести к доказательствам настолько сложным, что эти доказательства не будут найдены. Кроме того, надо не забывать о полуразрешимостп исчисления предикатов.

Метод доказательства теорем (директив) может быть использован в сочетании с другими подходами. Рассмотрим применение метода доказательства теорем (директив) для решения задач в пространстве состояний. Будем описывать состояния в виде правильно построенных формул (ппф) исчисления предикатов. При этом операторами являются преобразования, заменяющие одно множество формул другим (например, «список вычеркиваний» и «список добавлений»). Множество состояний, к которым применим данный оператор, задается с помощью предусловий, также записанных в виде ппф. В такой системе методы доказательства теорем (директив) можно

использовать для проверки выполнения условий достижения цели и условий применимости операторов.

Возможна некоторая модификация описанного метода, используемая в вопросно-ответной системе. В предыдущем способе преобразования, выполняемые операторами, отображают одни множества формул в другие. Но эти преобразования совершаются независимо от системы логического вывода в исчислении предикатов. Включения действия оператора в рамки формализма исчисления предикатов можно добиться путем введения в каждый предикат терма состояния, указывающего состояние, к которому предикат применим. **При такой формулировке операторы рассматриваются как функции, отображающие одно состояние в другое, а их действия выражаются в виде дополнительных аксиом, которые можно объединить с формулами, описывающими начальное состояние.** Так значением оператора $f(s)$ будет новое состояние, возникающее в результате применения оператора f к состоянию s.

Если наша цель состоит в создании состояния s, удовлетворяющего некоторой целевой формуле $B(s)$, то эту задачу можно решить формально, найдя доказательство для формулы $\exists s B(s))$ и определив частное значение переменной s. **Ответ будет содержать выражение для целевого состояния в форме композиции операторных функций.**

Приведем пример, поясняющий суть данного подхода. Пусть некоторое состояние S в мире объектных понятий (R) описывается следующим фактом:

$F1$: $\mathrm{At}(R, A, s_0)$ — объект находится в точке A в состоянии s_0 и объект умеет выполнять следующее действие (оператор):

$f1$: объект перемещается из точки x в точку y.

Основной эффект применения оператора $f1$ можно описать с помощью формулы

$$\forall x \ \forall y \ \forall s(\mathrm{At}(R, x, s) \rightarrow \mathrm{At}(R, y, f1(x, y, s)))),$$

означающей, что для всех s, x и y, если объект находится в точке x в состоянии s, то в состоянии, возникающем в результате применения оператора $f1$ к состоянию s, объект окажется в точке y.

Цель задачи состоит в определении последовательности действий, переводящих объект из точки A в точку C, т. е. в доказательстве истинности формулы: $(\exists s (\mathrm{At}(R, C, s)))$.

Очевидно, что решение получается непосредственно из $F1$ и $f1$, так что результирующее состояние $s = f1(A, C, s_0)$.

4. Процедуральные и языковые (семантические) представления

4.1. Обобщенные декларативные методы формирования представлений

В предыдущей теме описали представления задач, сформулированных в виде эвристического поиска, и задач доказательства теорем. При этом задачи эвристического поиска рассматривались нами в продукционной и редукционной системах. В первой из них решение сводится к поиску решающего пути в графе пространства состояний, причем поиск может осуществляться от начальных состояний к конечным (однонаправленный поиск) или (в случае явного задания целевых состояний) одновременно от начальных состояний к конечным и от конечных к начальным с замыканием общего решающего пути в одном из промежуточных состояний (двунаправленный поиск).

В редукционной системе решение сводится к поиску решающего графа в пространстве описаний множества подзадач и последующей композиции решения задачи из решений образующих ее подзадач.

Наконец, в задаче доказательства теорем (директив) решение сводится к выводу целевого высказывания (директивы) из исходного множества аксиом на основе правил вывода.

Несмотря на столь различные внешне постановки задач, поиск решения, особенно при однотипном задании механизма управления поиском, осуществляется в графах, определяющих один и тот же класс пространств поиска. Выявление общих закономерностей пространств поиска представляет значительный интерес как с точки зрения более глубокого понимания процессов решения задач в декларативных представлениях и взаимосвязей между различными формализациями, так и с точки зрения построения единого формализма решения задач, из которого могут быть выведены частные декларативные представления.

В настоящем параграфе изложим ряд основных предпосылок и идей построения обобщенных представлений и одну, кажущуюся нам наиболее целесообразной постановку задачи в таком представлении.

Прежде всего, процессы поиска в пропозициональных графах и графах доказательства теорем (ГДТ) работают в прямо противоположных направлениях. Действительно, в ГДТ мы начинаем с задания множества аксиом (заведомо разрешимых задач) и на каждом шаге последовательно наращиваем это множество выведенными на этом шаге теоремами до тех пор, пока не выведем целевое высказывание.

В пропозициональных графах процесс происходит обратным образом. Мы начинаем с задачи, которую следует решить (целевое высказывание), и на каждом шаге выводим из построенного к этому моменту множества подзадач альтернативные совокупности подзадач этих подзадач. Решение будет получено, когда множество подзадач будет полностью состоять из заведомо разрешимых задач (аксиом).

Пусть нам дана задача *T,* причем ее решение в системе редукций выглядит словесно следующим образом: «задача *T* может быть решена, если решены подзадачи *A* и *B* или подзадачи *B* и *C,* подзадача *A* решается, если решается подзадача *D,* подзадача *B* решается, если решаются подзадачи *E, F* или подзадача *G,* подзадача *C* решается, если решается подзадача *G, D, E, F, G* — разрешимые подзадачи».

Соответствующее представление в виде пропозиционального дерева приведено на рис.1, *а.* Все различные вхождения одной и той же подзадачи представлены различными вершинами (по определению дерева).

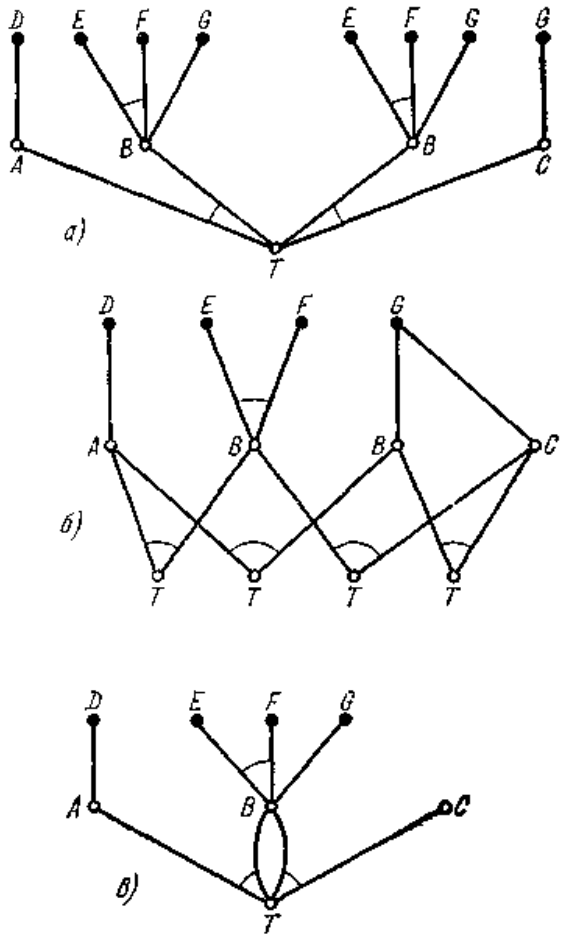


Рис.1. Представление задачи *T* в виде пропозиционального дерева (а), графа доказательства теорем*(б)* и графа вывода *(в)*.

Та же задача с помощью доказательства теорем решалась бы следующим образом: «даны аксиомы *D*, *E*, *F* и *G*, из *D* выводится *A*, из *E* и *F* выводится *B*, из *G* выводится *B*, из *G* выводится *C*, из *A* и *B* выводится *T*, из *A* и *B* выводится *T*, из *B* и C выводится *T*, из *B* и *C* выводится *T*». Соответствующее представление в виде ГДТ приведено на рис.1, *б*. И здесь различным вхождениям одной и той же директивы соответствуют различные вершины.

Представления (рис.1, *а* и 1, *б)* одного и того же решения одной и той же задачи различны. Однако переинтерпретируем эти представления таким образом, чтобы высказыванию соответствовала вершина графа, а различным способам ее вывода (образования ее как подзадачи) соответствовали различные дуги, инцидентные этой вершине. На рис.1, *в* приведено такое представление, являющееся единым для обоих исходных представлений.

Теперь решение задачи *T* сводится к поиску некоторого решающего подграфа в графе вида, предстявленного на рис. 10, *в.* Мы можем скомбинировать оба направления поиска (от аксиом к цели и от задачи к разрешимым подзадачам) в единый двунаправленный процесс с окончанием в «точке встречи». Далее, если разрешить выводы только из одной посылки (в направлении сверху вниз на рис.1, б) или исключить использование конъюнктивных вершин (в направлении снизу вверх на рис.1, б), то частным случаем этого графа будет граф в пространстве состояний. Поиск решающего подграфа в таком вырожденном графе сводится к поиску решающего пути. Таким образом, мы получаем граф в пространстве состояний как частный случай нашего графа. В этом вырожденном графе мы можем осуществить как однонаправленный, так и двунаправленный поиск.

Прежде чем перейти к формальному определению графа в обобщенном декларативном представлении, сделаем упоминание еще об одном обобщении, получаемом введением указанного графа. Как в редукционных, так и в продукционных системах представлений мы отождествляли вершины графа с состояниями (описаниями подзадач), а его дуги с операторами, преобразующими одни состояния в другие. Построение графа в обобщенном представлении позволяет ввести несколько другую интерпретацию. Мы будем рассматривать системы, в которых задается начальное множество состояний $S_o \subseteq S$ и начальное множество операторов $F_0 \subseteq F$. Результатом акта вывода (S_i, F_j), $S_i \subseteq S$, $F_j \subseteq F$, если он определен, может быть либо $s \in S$, либо $f \in F$. Другими словами, мы допускаем, что некоторые из операторов могут отождествляться с вершинами графа, а множество операторов не является фиксированным, а наращивается в ходе решения задачи.

Назовем граф вида, представленного на рис.1, *в, графом вывода* и введем некоторые формальные фрагменты определения, касающиеся этого графа.

Мы определяем граф вывода как пару (S, F), где S — множество вершин, отождествляемых с высказываниями, $F: 2^S \rightarrow S$ — функция следования, отображающая множество вершин в одну вершину. Эту функцию мы назовем *оператором вывода.*

Пусть S —высказывание, следующее непосредственно из конечного числа n высказываний $S_1, S_2, \ldots, S_n$, путем применения одного оператора вывода. Высказывание S связано с каждым из S_i, i=l, 2, . . ., n, дугой, образуя *конъюнктивный пучок. $S_i, S_2, \ldots, S_n$* называются *посылками, S — заключением.* Посылки, заключение и связывающий их конъюнктивный пучок образуют один *акт вывода.*

В частности, мы рассматриваем каждую аксиому S_0 как один акт вывода с пустым множеством посылок и заключением S_o. ***Выводом D*** **назовем конечное множество высказываний и связывающих их актов вывода** такое, что

1) Каждое $S \in D$ принадлежит по крайней мере одному акту вывода в D.

2) Точно одно высказывание $C \in D$ является заключенем акта вывода, не являясь посылкой какого-либо акта вывода.

3) Каждое высказывание $S \in D$ является заключением не более чем одного акта вывода в D.

4) Вывод D не содержит бесконечных ветвей вида $S_1, S_2, \ldots, S_n, S_{n+i}, \ldots$, где $S_1 \in D$, S_{n+1} — посылка акта вывода в D, имеющего заключением S_n (т. е. граф вывода не содержит циклов).

Посылками вывода D называются те высказывания из D, которые не являются заключениями какого-либо из актов вывода, принадлежащих D. *Заключением вывода D* является $C \in D$. Мы назовем вывод *беспосылочным* (БВ), если у него нет посылок, и *редукционным* (РВ), если заключение этого вывода — даное целевое высказывание. *Решающим выводом* будем называть беспосылочный редукционный вывод, полученный поиском одновременно в двух направлениях.

Итак, во введенном формализме решение задачи сводится к *поиску решающего вывода в графе вывода.* При этом последовательно образующиеся в процессе решения БВ порождают граф доказательстватеорем, а РВ — пропозициональный граф. Граф пространства состояний порождается выводами, акты которых содержали бы не более одной посылки.

4.2. Проблема границ в декларативных представлениях

Создание и организация информационного контекста, связанного с шагами процесса РЗ, в значительной степени определяет эффективность этого процесса.

В системах ИИ на применение любых формализмов описания накладываются ограничения количественного порядка.

Предположим, что в поисках плана решения задачи разработчик (объект ИИ) имеет в своем распоряжении в среднем 6 операторов, применимых и эвристически обоснованных в каждом состоянии; предположим, что типичная задача решается последовательностью из 4 операторов. Тогда поисковое дерево будет иметь около $6^4 \approx$ 1300 вершин (мы включаем сюда возможность хранения альтернативных планов). Будем считать, что разработчик работает в среде умеренной сложности. Тогда для полного описания каждого состояния может потребоваться хранение около 1000 элементарных фактов, **касающихся местоположения всех предметов и их признаков, указания всех отношений между ними и т. д.** Оказывается, что только для описания всех состояний в поисковом дереве требуется хранить свыше миллиона фактов! А поскольку каждый факт сам представляет из себя сложную списочную структуру, то становится ясным, что проблема сокращения числа обрабатываемых описаний состояний становится весьма острой. Очевидно, что каждое действие, совершаемое оператором, вызывает изменения лишь в небольшом подмножестве множества фактов. Казалось бы, что каждому состоянию или действию можно сопоставить список лишь изменяющихся фактов, а остальные факты могут храниться в общей базе данных в течение всего процесса решения задач. Однако следующий элементарный пример покажет принципиальные трудности сокращения числа обрабатываемых в ходе РЗ описаний и позволит нам сформулировать *проблему границ*.

Пусть некоторое состояние *S* в пространстве отправителя описывается следующими фактами (фрагментами директив):

*F*1*:* отправитель находится в позиции *A;*

*F*2*:* ящик *B*1 находится в позиции *B*;

*F*3*:* ящик *B*2 находится на ящике *B*1;

*F*4*:* допустимые позиции — *{A, B, C, D},*

и отправитель умеет производить следующие действия **(**множество операторов (директив)):

*f*1: отправитель идет из *x* в *y;*

*f*2: отправитель толкает *B*1 из *x* в *y, x, y* ∈ *{A, B, C, D}.*

Рассмотрим две задачи:

*P*1*:* отправитель должен быть в *C*;

*P*2*:* ящик *B*1 должен быть в *C.*

Очевидно, что задача *P*1 решается с помощью оператора (директивы) *f*1, причем после решения этой задачи факты (фрагменты директив) *F*2, *F*3, *F*4 остаются неизменными, а *F*1 меняется на

*F*1'*:* отправитель находится в позиции *C.*

Задача *P*2 решается с помощью *f*2, причем меняются факты (фрагменты директив) *F*1 и *F*2, в то время как *F*3 и *F*4 остаются неизменными.

*F*1″*:* отправитель находится в позиции С;

*F*2″*:* ящик *B*1 находится в позиции *C.*

Возникает вопрос, каким образом сократить описание новых состояний *S' = {F*1', *F*2, *F*3, *F*4*}, S″={F*1″, *F*2″, *F*3, *F*4*}.*

Казалось бы, что сокращение может быть достигнуто введением специальных процедур (директив) типа

*A*1*:* определить меняющиеся факты (фрагменты директив) сопоставлением условий задачи описанию состояния *S.*

Такая процедура могла бы определить, что в первой задаче должен измениться только тот факт, который относится к местоположению отправителя. Но в процессе состояния плана решения *P*1 отправитель может выяснить, что на кратчайшем пути в позицию *C* стоит ящик *B*1 и лучшим решением, чем обход ящика, является решение *f*2. При этом условия задачи выполняются, но процедура *A*1 терпит неудачу.

Может показаться, что следует привязать изменяющиеся факты к описанию операторов введением, например, процедуры

*A*2*:* указать факты, изменяющиеся каждым оператором.

Тогда, если задачу, будь то *P*1 или *P*2, решает оператор *f*2, то системе было бы указано, что надо менять факты, связанные с положением отправителя и *B*1, т. е. *F*1 и *F*2. Однако процедура *A*2 потерпела бы неудачу, если бы в множестве фактов в состоянии S присутствовал хотя бы один факт, выведенный из других фактов. Например, из *F*2 и *F3* можно вывести

*F*5*:* ящик *B*2 в положении *B,*

и этот факт не изменится процедурой *A*2, хотя после *f*2 *B2* будет в *C.*

До сих пор мы рассматривали однооператорные (однодериктивные) решения задач. Если же рассмотреть задачу

*P*3*:* отправитель должен быть в *D,* и *B*2 должен быть в *C,*

то легко видеть, что для решения этой задачи решатель должен на каждом шаге решения иметь доступ ко всему множеству фактов, включая выведенные следствия.

Можно привести более яркий своей кажущейся абсурдностью пример серьезности проблемы полных описаний состояний. Для того чтобы человек *p* вступил в телефонный разговор с человеком *q* (речевой комунникационный процесс), казалось бы, достаточно, чтобы *p* нашел номер телефона *q* в телефонной книге и набрал этот номер. Решатель задач, построивший такой план, потерпел бы неудачу в случае, если

— страница с номером телефона *q* вырвана,

— человек *p* слепой,
— кто-то залил чернилами нужный номер,
— телефонная компания сделала ошибки в коммутации,
— телефон *q* не значится в телефонной книге,
— телефонная линия в этот момент неисправна,
— человек *p* потерял голос и т. д.

Для учета всех этих возможностей формальной системе должны быть заданы дополнительные факты или условия, исключающие неопределенность ситуации, однако предугадать все возможности практически невозможно.

Теперь мы можем сформулировать *проблему границ* в достаточно общем виде.

Необходимо, чтобы решатель задач (разработчик директив), имея полное описание состояний предметов, мог разграничить (отсюда название проблемы) факты, которые должны изменяться в результате некоторого действия, от фактов, которые остаются неизменными в результате этого действия, и делал бы это эффективно с эвристической точки зрения.

В более общих понятиях, решатель задач должен, имея эпистемологически полное представление о фактах в мире, обладать эвристически эффективной способностью выделять минимально необходимое подмножество фактов, имеющих отношение к текущей стадии решения задачи, оставляя прочие факты без внимания.

Формально проблема границ может быть представлена следующим образом.

Допустим, что s_1 и s_2 — различные состояния предметов, причем s_2=apply (f, s_1), функция apply означает «применить оператор f к состоянию s_1». Мы могли бы описать результат действия оператора (директивы) f как

$$A(s_1) \rightarrow B(\text{apply}(f, s_1)), \qquad (1)$$

где A и B — множества измененных и новых фактов соответственно, представляющие некоторые короткие выражения. Однако если мы должны указать все факты, которые, будучи истинными в состоянии s_1, не изменились в состоянии s_2, и если множество таких фактов $F=\{F_i/i=1, 2, \ldots, n\}$, то результат apply ($f$, s_1) может быть записан в виде

$$\bigwedge_{i=1}^{n} F_i(s_1) \wedge A(s_1) \rightarrow \bigwedge_{i=1}^{n} F_i(\text{apply}(f, s_1)) \wedge$$
$$\wedge B(\text{apply}(f, s_1)), \qquad (2)$$

где n может быть весьма большим.

Таким образом, мы получаем длинные выражения законов действия (директив) в пространстве коммуникаций *S*. Решение проблемы границ — замена (2) на (1).

Прежде чем описать основные подходы к решению проблемы границ, укажем некоторые ее особенности.

Отметим принципиальный и количественный аспекты проблемы границ. Принципиальный характер проблемы подтверждается следующими аргументами:

1. В любой системе ИИ всегда будет стоять задача приведения в соответствие *растущих знаний* системы и уменьшающейся из-за роста количествафактов *эвристической эффективности.* Мы должны решить задачу **разумной организации системы знаний** или, другими словами, четко разграничить областизнания, относящиеся к классам действий, а также пути воздействия последних на изменение сложного мира. Проблема границ представляет собой частный случай этой общей задачи.

2. В любой системе ИИ запись действий (директив) в виде (1) не дает нам гарантии, что эта запись является адекватной раз и навсегда. Истинность того или иного факта может зависеть не только от его связи с тем или иным действием (директивой) непосредственно, но и от более детального анализа обновляющейся совокупности фактов. Иными словами, **часто истинность факта сможет быть установлена лишь в результате длинной цепи рассуждений (предписаний).** Конечно, «лобовым» решением этой задачи был бы полный вывод всех возможных следствий множества фактов в каждом из состояний предмета. Однако мы относим эту проблему к принципиальному аспекту хотя бы потому, что множество таких следствий может быть бесконечным.

3. Особую сложность проблема границ представляет для систем ИИ, работающих в *динамическом мире,* т. е. в режиме, где действия (предписания) могут не только исходить от системы, но и быть независимыми от нее. В этом случае результаты действия, записанные в виде (2), могут потерять свою истинность, как только на множестве S определяется новый предикат (в результате независимого действия).

Существующие подходы к решению проблемы границ в основном касаются количественного аспекта этой проблемы, т. е. выделения минимального списка фактов, относящихся к тем или иным действиям.

Краткий обзор этих подходов мы начнем с изложения идеи *метода*. Граница представляет классификацию фактов, независимую в том смысле, что некоторое действие может изменять факты, относящиеся только к одному классу, не меняя остальных. К сожалению, если такая

классификация и может быть получена, то она будет весьма грубой для всех практически важных задач. Идея такой классификации еще не получила развитие в теории ИИ. Каждому действию соответствует некоторое малое множество фактов, на которые это действие прямо влияет. Мы не можем предполагать, что все остальные факты остаются неизменными, потому что они могут быть **соединены длинными причинно-следственными цепями с изменяющимися фактами.** Обозначим бинарное отношение причинной связи через R. Тогда *aRb* означает, что некоторый факт *a* будет изменяться, если будет изменяться причинно-связанный с ними факт *b.* Если мы можем доказать, что *~(aRb),* то это означает, что никакие изменения *b* не вызывают изменений в *a.* Теперь при выполнении некоторого действия достаточно проверить, что *a* не связано причинной связью ни с одним из фактов, изменяемых действием.

Этот подход предполагает, что мир не динамичен и что имеется только **один источник (отправитель) действия (директивы)**. Это обстоятельство вызывает сомнение в том, что метод границ без привлечения новых средств сможет разрешить принципиальные аспекты проблемы границ. Что касается количественного аспекта, то этот метод вряд ли может быть использован в сколько-нибудь сложном решателе, поскольку неизменность большого количества фрагментов директив должна передоказываться в каждом состоянии предписания или директивы, что ничем по существу не отличается от обработки полных описаний предписаний и состояний предметов. Одним из возможных направлений развития этого метода является **введение модальностей в логику первого и высших порядков**, совокупно, хотя и приближенно, **описывающих причинные отношения между фактами в мире.** Однако это направление находится на стадии постановки, и требуется детальное исследование его возможностей, прежде чем можно будет перейти к его практической реализации.

Близко по духу к методу границ и другое направление, основанное на анализе непротиворечивости, — *метод контрфактов,* или выявления нереальных ситуаций. Идея метода заключается в том, что после выполнения действия все факты, которые были истинными в состоянии s_1, считаются истинными и в состоянии apply (f, s_1)**. После этого множество фактов должно быть проверено на непротиворечивость.** Противоречивые факты отбрасываются. Недостаток этого метода заключается в трудности определения, какие из фактов приводят к обнаруженному противоречию. Кроме того, с количественной точки зрения этот метод в чистом виде, по-видимому, не дает никакого выигрыша в сравнении с методом границ. **Метод**

границ является важной концептуальной основой для развития более близких к практическим целям методов.

Рассмотрим особенности решения проблемы границы для представления в исчислении предикатов первого порядка с использованием термов состояний. Трудность ее решения заключается в том, что если в состоянии s_0 нам был известен факт *F*2*:* At(*B*1, *B,* s_0) — ящик *B*1 находится в *B,* то неизвестно, где находится *B*1 в состоянии s. Этот факт должен устанавливаться в явной форме, т. е. введением дополнительной аксиомы (сравните неудачу директивы *A*1*):*

$$(\forall x)(\forall y)(\forall u)(\forall v)(\forall s)[\mathrm{At}(x, y, s) \wedge \wedge x \neq \mathrm{Robot} \rightarrow \mathrm{At}(x, y, f1(u, v, s))], \quad (3)$$

т. е. «положение объекта *x* останется неизменным после того, как отправитель перейдет из *u* в *v*».

Таким образом, нам нужны дополнительные аксиомы о том, что все факты (фрагменты директив), которые не изменяются в результате действия, действительно не изменяются.

Аксиома (3) работает и для факта типа *F*5. Однако если бы мы решили задачу *P*2, необходимо было бы передоказать истинность этого факта.

Таким образом, после выполнения каждой директивы необходимо передоказывать истинность всего множества неизменяемых фактов, что сводит практическую ценность этого метода к нулю для задач, содержащих большие множества фактов.

Наиболее подходящим в практическом отношении методом решения проблемы границ в декларативных представлениях является *метод контекстов и контекстных графов* применительно к использованию исчисления предикатов для решения коммуникационных задач в пространстве состояний.

Пусть на множестве состояний *S* определены предикаты P_i. Множество состояний $S_j \subseteq S$, для которых $P_i[S_j]=P_{ij}$ истинен, называется *контекстом, определяемым предикатом* P_{ij}. Например, факт *F*1 определяет контекст, удовлетворяющий предикату At (Отправитель, *A),* т. е. множество состояний, в которых отправитель находится в *A*. Далее, предикат At *(x, y)* определяет семейство контекстов, т. е. семейство множеств состояний, для которых объект *x* помещен в *y (x, y* — параметры). *Оператор* состоит из *наименования оператора, списка параметров* и двух специальных предикатов — *предиката предусловий K* и *предиката результатов R.* Так, оператор *f*1 из примера в начале параграфа будет представлен следующим образом:

$$\left.\begin{array}{l} f1\,(x,\,y), \\ K\colon \text{At}\,(\text{Отправит}, x) \\ R\colon \text{At}\,(\text{Отправит}, y) \end{array}\right\} \qquad (4)$$

где *f*1 — наименование оператора, *(x, y)* — список параметров,
At (Отправитель, *x)* — предикат предусловий, At (Отправитель, *y)* — предикат результатов.

Когда оператор применяется к контексту, т. е. в нашем примере к множеству состояний, удовлетворяющих At (Отправитель, *x),* он вычеркивает предусловия из списка фактов и добавляет результаты в список фактов, соответствующий состоянию. При этом изменяется контекст, т. е. производится переход в состояния, удовлетворяющие At (Отправитель, *y).*

Факты, не удовлетворяющие At (Отправитель, *x),* т. е. контекст, определяемый ~At (Отправитель, *x),* не изменяются.

Таким образом, каждый факт, выраженный в форме логического выражения, хранится в системе однократно, однако необходимо сохранение истории преобразования контекстов, в которых он добавлялся или вычеркивался. **Для этой цели служит *контекстный граф,* вершины которого соответствуют контекстам, а дуги — операторам, преобразующим один контекст в другой.** Пусть *I* — начальный контекст. В результате применения последовательности операторов $f_1 \circ f_2 \circ \ldots \circ f_n(I)$ мы получаем последовательность контекстов $C_1, C_2, \ldots, C_n=G$ (*G* — целевой контекст). В общем случае графу поиска решения будет соответствовать контекстный граф. Отношение между контекстным графом и поисковым графом иллюстрируется рис. 2, где граф (рис. 2, а) отражает поиск пути из *A* в *E,* а граф (рис. 2, *б)* — соответствующий ему контекстный граф.

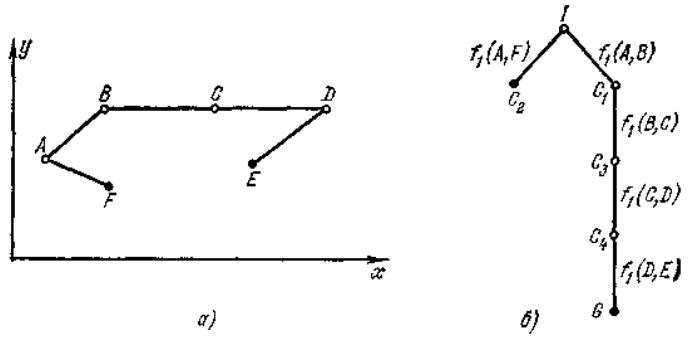


Рис. 2. Граф поиска пути из *A* в *E (а)* и соответствующий ему контекстный граф *(б).*

Основной недостаток метода заключается в том, что он не в состоянии прямым путем решить проблему границ для выведенных фактов типа *F*5. Поэтому в принципе все выведенные факты (они определяются чисто синтаксически) должны передоказываться в каждом новом состоянии.

В заключение проконстатируем, что в настоящее время не предложено метода (а возможно, что его и нет), который решал бы в общем виде проблему границ в декларативном представлении. Заметим, что эта проблема сравнительно легко, во всяком случае в количественном аспекте, решается в процедуральных представлениях.

4.3. Процедуральные представления

4.3.1. Общие характеристики ПОЯ.

Для формального описания систем, сетей, представлений, фактов и др. будем использовать языки ПОЯ. Основными характеристиками, отличающими ПОЯ от обычных языков программирования, являются:

1) Наличие выразительных средств и соответствующих механизмов для ассоциативного поиска и извлечения необходимой в данный момент информации из базы данных.

2) Вызов процедур указанием цели, которая должна быть достигнута, а не по имени.

3) Наличие механизма индикации успеха и неудачи в достижении цели, позволяющего автоматически вернуться к точке ветвления процесса, явившейся причиной неудачи, и автоматически исследовать другие альтернативы.

С точки зрения построения решателя задач, работающего в процедуральном представлении, эти характеристики являются необходимыми для обеспечения целенаправленного механизма поиска решения.

4.3.2. База данных и механизмы сопоставления по образцу

Опишем общие принципы организации базы данных, безотносительные к какому-либо языку.

База данных состоит из выражений (директив). Каждое выражение обычно содержит *синтаксическую компоненту* и *список свойств,* хранящий произвольные свойства, значениями которых могут быть

выражения. Список свойств обычно содержит *семантическую* и *прагматическую информацию.*

Стандартные *семантические свойства выражений* включают в себя значение выражения, множество равных ему выражений, множество неравных ему выражений, правила вычисления (формирования) и упрощения выражений.

Прагматические свойства обычно выражают информацию, специфическую для данной задачи вообще или для текущего состояния процесса ее решения. Одним из способов задания прагматических свойств являются так называемые *рекомендации.* Рекомендации указывают на то, какие альтернативы следует испытать и в каком порядке, какие методы следует применить в попытке решить задачу, сохраняют историю попыток испытать те или иные способы действия и т. д.

Весьма важным является способ запоминания и извлечения выражений из базы данных. Одним из наиболее распространенных способов организации базы данных является ее построение в виде *дискриминационной сети,* впервые предложенной Фейгенбаумом и развитой впоследствии в работах по GPS.

Дискриминационная сеть представляет граф, каждая вершина которого содержит функцию, извлекающую атомарную часть синтаксической компоненты, и является либо конечной вершиной, содержащей выражение, либо промежуточной, содержащей список дуг, исходящей из этой вершины (дуга является парой атом — дочерняя вершина).

Пусть, например, дискриминационная сеть хранит выражения типа *M* (MTYPE), (ABC *XY),* (FCT *XY),* (STR *ZY),* (STR *YX),* где MTYPE, ABC, FCT, STR — произвольные атомарные синтаксические формы. Эта сеть приведена на рис.3.

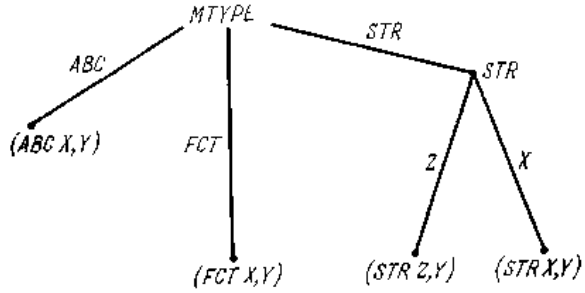


Рис. 3. Пример дискриминационной сети.

Итак, при входе в сеть извлекается синтаксическая форма корневой вершины, выбирается соответствующая дуга, извлекается синтаксическая форма следующей вершины и т. д., пока либо не будет достигнута конечная вершина, либо не выяснится, что нет подходящей дуги. В последнем случае создается новая конечная вершина, т. е. входное выражение заносится в сеть. При достижении конечной вершины входное выражение α сравнивается с синтаксической компонентой выражения в конечной вершине β. Если α и β синтаксически идентичны, то сеть не изменяется. Если выражение α, сопоставилось по форме с выражением β, то α заносится в список свойств выражения β (если его там не было). Если же выражение α не сопоставляется по форме с выражением β, то список свойств β просматривается с целью найти там выражения, равные β и сопоставляющиеся по форме с α. В случае успеха α также заносится в список свойств β; при неудаче к α применяются правила упрощения из списка свойств β и проделываются указанные выше манипуляции. Наконец, если и это заканчивается неудачей, то первое синтаксическое различие между α и β запоминается как селектор признаков вновь созданной вершины, от которой проводятся две дуги: одна — к вершине, соответствующей выражению β, и вторая — к новой вершине, созданной для синтаксической формы входного выражения α. Одновременно создается список свойств этого выражения. Образованная *синтаксическая форма* входного выражения называется *канонической.* Заметим, что здесь и далее мы употребляем понятие «форма» в классическом смысле.

Таким образом, каждое выражение хранится в дискриминационной сети только в одном экземпляре, так что свойства выражения автоматически связываются со всеми эквивалентными выражениями.

Связанные переменные не могут использоваться как селектор признаков. Например, выражения (LAMBDA *(x, y), (x+y)×(y+1))* и (LAMBDA (*u, v),* (l+*v*)×*(v*×*u*)) оцениваются как эквивалентные и преобразуются в одну и ту же каноническую форму (с одинаковым списком свойств).

Как указывалось ранее, процедуральное представление довольно легко обходит трудности представления отношения равенства в исчислении предикатов первого порядка. Вместо аксиоматизации правил равенства вводятся разбиения выражений на классы эквивалентных выражений. На каждом шаге каждое выражение содержит множество логически равных ему на этом шаге выражений. Эти множества для двух выражений будут объединены, если будет доказано или высказано в виде утверждения их равенство. Каждое выражение содержит список всех не равных ему выражений, причем вновь, как только формируется

новое выражение о равенстве, эти множества для всех выражений обновляются соответствующим образом. Следовательно, как только утверждение о равенстве выражений вызывает противоречие, это записывается непосредственно.

Рассмотрим вопрос о механизме, с помощью которого в базе данных производится запоминание и извлечение информации. Этим механизмом является *сопоставление по образцу.*

Сопоставление по образцу осуществляется в два шага. На первом шаге производится сопоставление входного выражения α с выражениями в базе данных β. В общем случае каждое из выражений представляет собой *форму произвольной степени сложности.* Эту форму будем называть *образцом.* Поскольку выражение α является образцом большей степени общности, чем β, выбор выражения β может быть неоднозначным. На втором шаге по результатам сопоставления образцов производится связывание переменных, входящих в сопоставляемые выражения. В случае, если переменные связываются с некоторыми подопределениями, для присвоения переменным значений может потребоваться вычисление этих подопределениий. Механизм сопоставления является основой следующих операций над базой данных:

1. Композиция выражений. Определяется процедура comp *(x, y),* создающая (формирующая) выражение, первый элемент (директива) которого есть значение выражения *формы x* и второй элемент — значение выражения *формы y.* Таким образом вычисление comp *(x, y)* для *x*=4, *y=(A (BC))* дает в результате *(A (A (BC))).*

2. Декомпозиция выражений. Определяется процедура decomp *(x, y),* вычисляющая значения подопределений исходного выражения *формы x* и *y* соответственно. В результате применения этой процедуры к (4(*A (BC))) x* присваивается значение 4, а *y* — значение *(A (BC)).*

3. Извлечение. Процедура извлечения находит в базе данных все элементы заданной в процедуре формы и выдает в качестве результата произвольный элемент среди множества элементов, сопоставляемых с этой формой. Например, применение процедуры Exists (существует)) (At *(x, y))* может извлечь из базы данных факты о местонахождении объектов *формы x* в месте *формы y.* Если ящик *B*1 находится в *A*, то результатом применения процедуры Exists будет At *(B*1, *A).*

Формы *x* и *y* в режимах композиции и декомпозиции и форма At(*x, y)* в режиме извлечения являются *образцами.*

Следует отметить, что, так как выбор может быть неоднозначным, сопоставление по образцу вносит в процесс решения элемент *недетерминистичности,* так что в случае неудачи решателю

предписаний необходимо вернуться в исходную точку и сделать альтернативный выбор.

4.3.3. Стандартные операторы

Весьма затруднительно описать множество функций, выполняемых ПОЯ. Во-первых, многообразие их велико, во-вторых, особенности выполнения многих из них зависят от структуры конкретного языка, поэтому мы опишем лишь некоторые наиболее общие функции и операторы, вводя необходимые дополнения по мере рассмотрения примеров.

Общим для процедуральных представлений механизмом работы с процедурами является *вызов процедур сопоставлением по образцу*. При этом вызов процедуры осуществляется не по имени, а указанием выражения, являющегося целью ее применения, а сами процедуры воспринимают в качестве аргументов выражения определенной структуры, сопоставляющейся с формой их описания. Например, если мы хотим поставить ящик *B*2 на ящик *B*1, то выражение-цель On(*B*2, *B*1) может вызвать из базы данных процедуру формы Put(*x*, *y*), которая предназначена для того, чтобы поставить предмет *x* на предмет *y*. Такой механизм вызова процедур полностью совпадает с режимом извлечения общего механизма сопоставления по образцу, однако входное выражение-цель является образцом меньшей степени общности, чем образец в описании процедуры. Здесь снова может быть извлечено множество альтернативных процедур, что вносит в процесс решения элемент недетерминистичности.

В большинстве ПОЯ используются четыре типа операторов: *цели, утверждения, стратегии* и *непосредственные преобразования*. Цели и утверждения соответствуют теоремам и аксиомам в представлении с помощью доказательства теорем (утверждений). Непосредственное преобразование обычно имеет вид правила. Оно воспринимает входное выражение, сопоставляет его с образцом, который является левой частью правила, и преобразует его в выходное выражение, подставляя полученные при сопоставлении значения переменных в правую часть правила. Непосредственное преобразование, таким образом, имеет вид $P \rightarrow Q$. Стратегия включает в себя управляющие механизмы, непосредственные преобразования и другие стратегии и управляет процессом решения задачи. **Задача ставится как оператор-цель. Система пытается найти непосредственные преобразования и стратегии, помогающие достижению цели.** Если стратегия,

связанная с целью, приводит к неудаче, то создаются подцели первоначальной цели, т. е. решение идет аналогично процессу редукции. Однако в процедуральном представлении обычно имеется некоторая управляющая программа, анализирующая свойства представлений и пытающаяся на их основе и с помощью непосредственных преобразований определить необходимую стратегию.

Предположим, что в управляющую программу введено утверждение. Производится два действия:

1. Утверждение заносится в базу данных.
2. Утверждение обрабатывается с помощью непосредственных преобразований, сопоставляющихся по образцу с утверждением. Результирующие утверждения также заносятся в базу данных.

Оператор-цель вводится в управляющую программу вместе с рекомендацией. Управляющая программа пытается с помощью сопоставления по образцу и на основе семантической информации и рекомендаций найти полезные преобразования, из которых составляется стратегия, образующая подцели, обрабатываемые аналогичным образом.

Следует отметить, что новые правила преобразования и стратегии становятся достоянием управляющей программы, так что ее арсенал может непрерывно пополняться. Новые правила и стратегии могут вводиться либо пользователем системы, либо накапливаться в результате опыта решения задач.

Изложенная схема представляет собой весьма грубое приближение к действительно имеющей место организации процесса решения.

4.3.4. Механизм возврата к точке ветвления

Недетерминистичность выбора альтернатив в различных точках процесса решения (извлечение выражений из базы данных, вызов процедур сопоставлением по образцу, альтернативы) приводит к необходимости создания специального механизма, который в случае неудачно выбранной альтернативы возвращает процесс в точку ветвления, восстанавливая при этом все структуры данных и управляющие структуры. При этом, необходимо снабжать систему такой семантической и прагматической информацией, чтобы она могла выбирать наилучшее решение первым. Однако рассчитывать на это было бы слишком оптимистичным, и указанный механизм, называемый *механизмом возврата к точке ветвления,* является основным средством, позволяющим реализовать целенаправленность выбора стратегий.

Будем различать *декларативный возврат* и *процедуральный возврат.*

Рассмотрим декларативный возврат. Сущность его заключается в следующем. Если в ходе решения встречается точка ветвления, то запоминается полное описание состояния процесса (т. е. текущие значения всех переменных). Если выбранная альтернатива неудачна, это описание состояния восстанавливается и решение направляется по другому альтернативному пути. Если все альтернативы в данной точке исчерпаны, то сигнал о неудаче распространяется к предшествующей в дереве целей точке ветвления с восстановлением полного описания состояния процесса, соответствующего этой точке.

Процедуральный возврат предложен в связи с разработкой **теории недетерминистических алгоритмов**. В процедуральном возврате показывается, что всем основным элементам вычислений — присвоениям, условным ветвлениям, обращениям к выходам из подпрограмм (директивы) и т. д.— можно сопоставить инверсии, т. е. операции, воспринимающие выход данного элемента вычислений как вход и выдающие в качестве результата вход элемента. Таким образом, в случае неудачного исхода альтернативы можно осуществить локальный пошаговый возврат, применяя инверсные операции, пока не будет достигнута точка ветвления.

Рассмотренные два метода возврата к точке ветвления и являются основой для осуществления механизма возврата в ПОЯ. Каждый метод имеет свои преимущества и недостатки.

Основным преимуществом декларативного возврата является его высокая эффективность: возврат в случае неудачи осуществляется с помощью одного шага. Преимущество процедурального возврата — в отсутствии необходимости запоминать большие объемы ненужной информации при прямом ходе выполнения вычислений. Один из основанных на процедуральном подходе методов реализации механизма возврата заключается в том, чтобы полностью автоматически запоминать и восстанавливать при возврате *управляющие структуры* и переложить запоминание и восстановление *структур данных* на пользователя, заставив его в явном виде указывать процедуры, для которых следует запоминать и восстанавливать локализованные в них переменные.

С одной стороны, этот подход позволяет исключить сохранение ненужной информации и обеспечивает пользователю большую управляемость процессом решения. Однако он требует от программиста точного знания информации, которую следует запоминать в точке ветвления, т. е. фактически предсказания хода процесса в альтернативных случаях. Кроме того, вероятность ошибок как в сторону недооценки, так и переоценки количества запоминаемой информации резко возрастает. В первом случае это

приведет к отказам программы, а во втором — снизит эффективность системы.

Основанные на декларативном подходе механизмы возврата, как правило, используют для запоминания информации специальные *стеки возврата* и *стеки состояний* или *контекстные механизмы*. Эти методы различаются лишь деталями реализации. Основная же идея в обоих случаях заключается в том, что каждой рассматриваемой альтернативе отводится свое поле памяти, или *контекст.* Прямой ход процесса, таким образом, осуществляется с отдельной базой данных, в которой локализованы необходимые для данной - альтернативы выражения. При возврате контекст просто уничтожается (если не принимать во внимание извлечение прагматической информации).

Следует отметить, что использование механизма возврата является критическим моментом с точки зрения оценки эффективности процедуральных решателей задач. Дело в том, что в своей первоначальной постановке возврат представляет из себя исчерпывающий поиск в глубину, т. е. в худшем случае полный перебор возможных альтернатив. Это обстоятельство порождает парадоксальную ситуацию: для совершенной стратегии механизм возврата не нужен, так как в каждой точке ветвления она точно знает, что делать. А для плохих стратегий, т. е. стратегий, не обладающих достаточной информацией для выбора альтернатив, механизм возврата становится слепым.

Другая, не менее важная проблема возникает в связи с оценкой действий решателя задач в случае неудачного исследования альтернатив. Чистый механизм возврата не вырабатывает в общем случае информации, которая могла бы повлиять на дальнейший выбор, поскольку в большинстве рассмотренных механизмов все следствия, полученные в результате гипотезы о том, что данная альтернатива полезна, после отбрасывания альтернативы уничтожаются. Это равносильно утверждению о том, что в каждой точке ветвления все альтернативы независимы, т. е. признанию полного перебора.

Указанные принципиальные недостатки механизма возврата усугубляются тем, что в большинстве ПОЯ он используется не только для испытания альтернативных стратегий, но и при каждом вызове процедуры или извлечения выражения сопоставлением по образцу. Это может сделать пространство поиска недопустимо большим.

Решение проблемы следует искать там же, где и решение всех основных проблем ИИ: **в наложении на процесс поиска эффективных эвристик.** Но, как известно, эти эвристики могут быть получены либо в результате предварительного глобального исследования пространства поиска, либо обобщением опыта в

процессе решения задач. В принципе, как указывалось в начале этого параграфа, процедуральное представление дает обширные возможности для использования семантической и прагматической информации, которая могла бы управлять процессом поиска решения. Однако методы ее извлечения, особенно по результатам исследования неудачных альтернатив, а также, что еще важнее, ее обобщения изучены пока слабо.

4.3.5. Пример

Для лучшего понимания работы некоторых описанных выше механизмов рассмотрим следующий пример.
Докажем силлогизм:

«Тьюринг—человек.
Все люди ошибаются.

Следовательно, Тьюринг ошибается.»

В терминах ПОЯ PLANNER решение может быть получено вычислением выражения (GOAL (ошибается Тьюринг)), где GOAL — упомянутый выше оператор-цель, с помощью следующей процедуры:
<ASSERT (человек Тьюринг)> (посылка в базе данных)
<ASSERT <DEFINE THEOREM1
<CONSEQUENT *(Y)* (ошибается ?*Y*)
(GOAL (человек ?*Y*)>>>>,
где вызовы операторов заключены в скобки < >. ASSERT (утверждение) — оператор-утверждение. Этот оператор, после его применения, заносит свой аргумент в базу данных утверждений. С помощью функции DEFINE THEOREM (определим теорему) мы определяем теорему формы CONSEQUENT (следствие). Это означает, что доказать цель формы (ошибается ?*Y*) можно путем доказательства цели (человек ?*Y*), т. е. первая цель является следствием второй. ?*Y* означает идентификатор, которому может быть присвоено значение в результате сопоставления с образцом, в который ?*Y* входит. Все атомы, не снабженные префиксом ?, являются константами. Работа происходит следующим образом. Если бы нам надо было вычислить <GOAL (человек Тьюринг)>, то требуемое утверждение было бы найдено непосредственно в базе данных (так как это посылка силлогизма). Однако утверждение (ошибается Тьюринг) в базе данных отсутствует, и мы должны его вывести. Испытываются все теоремы, следствия которых сопоставляются с целью, находится теорема

THEOREM 1 и осуществляется переход к доказательству <GOAL (человек ?*Y*)>. В результате сопоставления <GOAL (человек ?*Y*)> и (человек Тьюринг) мы связываем переменную *Y* с константой «Тьюринг». Теорема устанавливает новую цель (человек Тьюринг), и поскольку она находится в базе данных, <GOAL(человек ?*Y*)> достигнута, т.е. доказана <GOAL (ошибается Тьюринг)>. Ниже приведены шаги вычисления:

1) <GOAL (ошибается Тьюринг)>.
2) Теорема 1 активируется.
3) *Y* присваивается значение Тьюринг.
4) <GOAL (человек Тьюринг)>.
5) Результат (ошибается Тьюринг).

Рассмотрим теперь вопрос «кто-нибудь ошибается?» или, в логической форме, EXISTS *X* (ошибается *X).* В ПОЯ PLANNER это записывается в виде <THPROG (*X*) <GOAL (ошибается ?*X*)>>. В данном случае THPROG, являющийся аналогом функции PROG в языке LISP, действует как квантор существования. Попытка непосредственного вычисления цели приводит к неудаче, так как в базе данных нет утверждения формы (ошибается *?X).* В поисках доказательства THPROG ищет теорему со следствием этой формы и находит выше определенную теорему. Идентификатор *Y* из теоремы связывается с идентификатором *X* цели. Однако *X* еще не имеет значения, и поэтому *Y* тоже не получает значения. Теорема устанавливает цель (человек ?*Y*). Соответствующий оператор GOAL ищет в базе данных утверждение, сопоставляющееся с этим образцом, и находит утверждение (человек Тьюринг). Поэтому *Y*, а следовательно, и *X* связываются с константой «Тьюринг», и доказательство завершается, выдавая результат (ошибается Тьюринг).

Пусть нам даны дополнительные утверждения <ASSERT (человек Сократ)>, <ASSERT (грек Сократ)>, так что в базе данных находятся три утверждения: (человек Тьюринг), (человек Сократ), (грек Сократ) и теорема

<CONSEQUENT (*Y*) (ошибается ?*Y*)
<GOAL (человек ?*Y*))>.

Мы задаем вопрос «есть ли ошибающийся грек?», представляемый выражением

<THPROG *(X)*
<GOAL (ошибается ?X)>
<GOAL (грек ?X)>>.

Первая цель удовлетворяется рассмотренным выше выводом. Если при поиске в базе данных (человек Тьюринг) встретится раньше, чем

(человек Сократ), то цель (человек ?*Y*) теоремы будет достигнута, связывая *Y* и *X* с константой «Тьюринг».
Тогда THPROG устанавливает цель (грек Тьюринг), которая не может быть доказана, так как нет ни соответствующего утверждения в базе данных, ни применимых теорем.
При неудаче работает механизм возврата, который найдет как причину неудачи связывание *Y* с константой «Тьюринг», после чего извлечет (человек Сократ) и продолжит доказательство. Результатом теоремы будет значение (ошибается Сократ), переменные *Y* и *X* связываются с константой «Сократ», и THPROG устанавливает цель (грек Сократ), которая достигается, так как соответствующее утверждение находится в базе данных. В качестве результата THPROG выдает (грек Сократ).
Приведем шаги процесса:
1) Активация THPROG.
2) <GOAL (ошибается ?*X*)) приводит к вызову по сопоставлению образцу, так как в базе данных нет утверждения формы (ошибается ?X).
3) Активация теоремы 1.
4) Сопоставление (ошибается *?Y)* и (ошибается *?X), Y* связывается с *X*.
5) <GOAL (человек ?*Y*)> находит (человек Тьюринг) в базе данных.
6) *Y* принимает значение Тьюринг, следовательно, *X* принимает значение Тьюринг.
7) Результат (человек Тьюринг).
8) Результат теоремы (ошибается Тьюринг).
9) <GOAL (грек Тьюринг)> терпит неудачу, так как в базе данных нет такого утверждения и сопоставляющихся с целью теорем типа CONSEQUENT. Возврат к шагу 5).
6') <GOAL (человек ?*Y*)> находит (человек Сократ) в базе данных.
7'*)* *Y* принимает значение Сократ, следовательно, *X* принимает значение Сократ.
8') Результат (человек Сократ).
9') Результат теоремы (ошибается Сократ).
10) <GOAL (грек Сократ)>.
11) Результат THPROG (грек Сократ).

4.3.6. Контекстный механизм

Как отмечалось в п.4.3.4, при реализации механизма возврата к точке ветвления необходимо иметь средства запоминания информации при

рассмотрении альтернатив. Соответствующий механизм мы назвали *контекстным.*

Контекстный механизм имеет более широкое применение, чем в механизме возврата. Он используется везде, где оказывается необходимым выделение из глобальной базы данных некоторых локальных областей, инициация работы с этими локальными областями без изменения глобальной базы данных и принятие решения по результатам работы о том, следует ли производить изменения в глобальной базе данных, и если следует, то какие.

Такого рода механизмы оказываются полезными для реализации гипотетических планов, параллельных процессов, описания моделей внешнего мира активных источников действия и т. д.

Контекстный механизм реализуется с помощью разветвленного и ветвящегося в процессе работы стека, который можно представить в виде дерева. Вершины этого дерева соответствуют процессу или состоянию мира таким образом, что изменение выражений и их свойств вызывает изменения только в меняющем их процессе и последующих субпроцессах и состояниях (дочерних вершинах). Пример дерева контекстов приведен на рис.4.

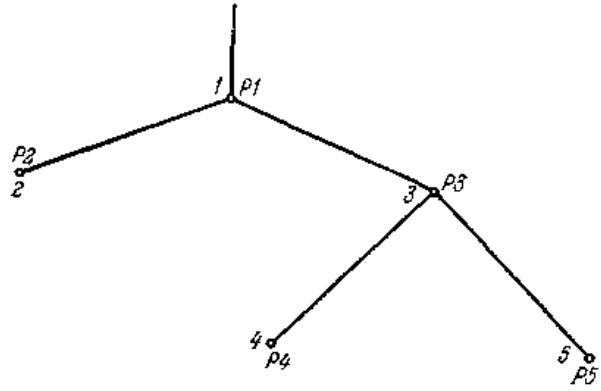


Рис.4. Пример дерева контекстов.

Здесь P1, P2,.., P5 —процессы, а цифры, стоящие рядом с вершинами дерева, обозначают номера контекстов, создаваемых соответствующими процессами. Таким образом, контексты, к которым может иметь доступ процесс *P4,* будут (4,3,1), *P3* — (3,1) и т. д. Сами контексты представляются в виде списка выражений, отличающих этот контекст от контекста высшего порядка.

Стандартный набор операций для работы с контекстами **включает в себя операции создания, активации и уничтожения контекста,**

предписания работы с указанным контекстом или с данным набором образцов в отдельном контексте, а также внесения изменения в глобальную базу данных. Этот набор позволяет **выделять локальные области знаний, хранить историю тех или иных процессов в том или ином контексте, осуществлять гипотетическое планирование и параллельное построение планов.**

Рассмотрим возможную организацию гипотетического плана. Необходимо создать локальный контекст, сделать требуемые гипотетические посылки и вывести соответствующие заключения. После завершения гипотетического планирования контекст должен быть уничтожен, поскольку истинность посылок в локальном контексте вовсе не предусматривает их истинность в глобальной базе данных. Нам требуется лишь результат типа «если бы посылки были верны, то было бы истинно следующее заключение». Описанный процесс имел бы следующий вид:

1) Доказать, что $X \rightarrow Y$, где X и Y — произвольные выражения относительно контекста C.
2) Создать новый контекст C'.
3) ASSERT X относительно контекста C'.
4) GOAL (Y) относительно контекста C'.
5) DELETE C' (здесь ««уничтожить контекст C'»).
6) ASSERT $X \rightarrow Y$ относительно контекста C'.

Еще раз отметим, что X истинно только в контексте C' (п. 3 нашего описания).

Параллельные процессы также легко реализуются с помощью контекстного механизма. Предположим, что нам надо доказать теорему (построить план) вида $X \vee Y$. Тогда, создав отдельные контексты для X и Y, мы можем запустить параллельное выполнение процессов X и Y, завершив доказательство, когда будет завершен один из процессов X или Y. Естественно, в последовательном вычислительном средстве эти процессы в действительности не будут выполняться параллельно, так что для извлечения выгоды из фиктивной параллелизации доказательства необходимо организовать взаимодействие процессов X и Y. Схема организации выглядит следующим образом:

1) Выбрать более легкий по некоторым критериям процесс и доказывать его.
2) Если процесс доказательства сходится быстро в некотором смысле, то продолжать, иначе сохранить его контекст и перейти к доказательству другого процесса, используя, однако, всю полезную информацию, полученную в процессе доказательства первого процесса.

3) Продолжать повторять п. 2 до тех пор, пока не будет найдено доказательство или не будет выработан сигнал о неудаче.

Другим примером параллельных процессов является задача вида *(∃ x){P(x) ∧ Q(x)},* где *P* и Q — произвольные предписания. Здесь план решения состоит в том, чтобы

1) Найти *x,* удовлетворяющее *P(x).*

2) Проверить, удовлетворяет ли *x Q(x).* Если да, то процесс заканчивается, иначе выбрать *x,* удовлетворяющий *Q(x),* используя информацию в контексте для *P(x),* и проверить, удовлетворяет ли *x P (x)* и т. д.

4.3.7. Проблема границ в процедуральных представлениях

В простейших ситуациях разграничение изменяющихся фактов от неизменяющихся может быть произведено просто с помощью операторов присваивания или, в крайнем случае, с помощью блочной структуры языка. Эти механизмы охватывают все ситуации, которые разрешает, например, **метод контекстов и контекстных графов.** Более существенные меры следует принимать для выведенныхфактов, однако все они довольно легко реализуются для локально-недетерминистичных процедур.

Рассмотрим, например, механизм решения этой задачи в диалекте языка PLANNER — языке POPLER. В этом языке обеспечивается механизм активации определенных процедур при занесении информации в базу данных (ASSERT) и стирании информации в базе данных (ERASE *(*стирать*)).* Когда в базу данных добавляется информация, этот механизм ищет в соответствии с рекомендациями процедуры типа ASSERTING, образцы которых сопоставляются этой информации, и активирует такие процедуры. Аналогичным образом при стирании активируются процедуры типа ERASING.

Рассмотрим действие этого механизма на примере. Пусть база данных содержит три факта:

ASSERT <At HAND (рука) P1) —рука находится в *P1,*

ASSERT <At OBJ2 P1) —объект 2 находится в *P*1,

ASSERT (HOLDING (держит)) OBJ2>—держит объект 2,

утверждающие, что если рука держит объект, то рука и объект находятся в одном месте.

Пусть в базе данных содержатся две процедуры типа ASSERTING:

PROCEDURE ATHAND ASSERTING (At HAND *?X)*

PROCVARS *X Y;*

GOAL <HOLDING *?Y* >;
ASSERT <At *?Y* ?*X*>;
END PROC;
PROCEDURE ATANY (нечто находится где-то) ASSERTING (At*?X ?Y)*
PROCVARS *X Y Z;*
GOAL <At *?X* ?Z);
IF *?Y* =?Z THEN FAIL (сигнал неудачи, возврат к точке ветвления);
ERASE (At ?*X* ?Z);
END PROC;
здесь PROCVARS обозначает описание в процедуре следующих за ней переменных.
Введем в базу данных ASSERT (At HAND *P2).* Тогда активируются обе процедуры, поскольку они типа ASSERTING, а их образцы сопоставляются с (At HAND *P*2). Предположим, что ATANY испытывается первой. При этом *X* связывается с HAND, а *Y* с *P2.* Оператор GOAL может связать Z либо с *P*1, либо с *P2.* Однако *P2* исключается условным оператором IF, так что стирается только старый факт (At HAND P1). Процедура ATHAND связывает *X* с *P2,* ее GOAL найдет в базе данных (HOLDING OBJ2), связывая *Y* с OBJ2, так что в базу данных попадет утверждение (At OBJ2 P2). Это утверждение вновь активирует процедуру ATANY, которая вновь сотрет старый факт (At OBJ2 PI).
Можно полагать, что наличие в ПОЯ контекстного механизма позволит решать проблему границ и в более сложных мирах.

4.4. Языковые (семантические) представления

4.4.1. Определения семантических сетей

Развиваемые семантические представления являются шагом на пути к построению систем ИИ — систем (сетей), базирующихся на знании. **Семантические представления являются основой систем зрительного восприятия, понимания естественного языка и непрерывной речи, т. е. систем, осуществляющих связь с внешним миром — одним из главных источников знания систем ИИ (другим является сама система ИИ).** Именно характер развития и разноплановость использования семантических представлений послужили основной причиной разработки многочисленных вариаций этих представлений, базирующихся на нескольких сравнительно общих идеях и методах реализации. Мы рассмотрим в настоящем параграфе ряд основных

принципов построения и характеристик семантических представлений, ссылаясь там, где это необходимо, на соответствующие конкретные реализации.

Основой всех вариантов семантических представлений является **формализация структур семантического знания в виде направленного графа с помеченными вершинами и дугами, причем вершинам соответствуют некоторые объекты, а дугам — семантические отношения между этими объектами**. Метки, приписываемые вершинам, носят чисто ссылочный характер, представляя собой некоторые мнемонические имена, в частности, слова естественного языка. Заметим, что в этом частном случае слова дают ссылку на словарь, входам которого соответствуют некоторые смысловые эквиваленты, или лексические значения, причем это соответствие может быть неоднозначным в обе стороны. Метки, приписанные дугам, соответствуют элементам множества отношений, заданных на графе, причем этими элементами могут быть как семантические свойства, так и семантические выводы. Описанный выше граф мы будем называть *семантической сетью.*

На семантической сети могут быть определены некоторые подграфы определенной структуры, называемые *высказываниями.* Каждый такой подграф представляет собой граф, корнем которого является *предикатная вершина*; остальные вершины называются *концептуальными*. Следует подчеркнуть, что такое разделение вершин высказывания (директивы) можно четко реализовать, лишь рассматривая каждое высказывание в изоляции. В структуре семантической сети одна и та же вершина может быть предикатной относительно одного высказывания и концептуальной относительно другого. Мы вводим ***понятие высказывания (директивы)*** лишь с целью подчеркнуть, что оно является ***минимальной единицей информации***, вводимой и хранящейся в семантической сети. На рис. 5 приведен пример изображения высказывания «Виктор ввел высказывание в сеть». Буквами *A, B, C* условно помечены семантические отношения между объектом в предикатной вершине и объектами в концептуальных вершинах.

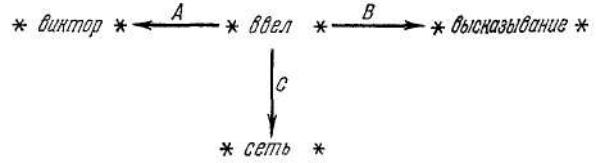


Рис.5. Пример изображения высказывания в семантической сети.

Заметим, что в семантической сети каждый объект представляется точно одной вершиной. Тем самым мы разрешаем, чтобы от этой вершины (в нее) исходило (входило) несколько дуг, связанных с несколькими различными высказываниями (директивами).

На рис. 6 изображены абстрактная сеть (C, *R),* где *C*— множество объектов, *R*— множество отношений (рис.6, *а*), и соответствующая теоретико-графическому представлению запись в виде списков свойств и троек вида $C_iR_kC_j$, $R_k \in R$, $C_i,C_j \in C$ (рис.6,*б* и 6, *в* соответственно), отражающая связь семантических представлений с процедуральным представлением и исчислением предикатов соответственно.

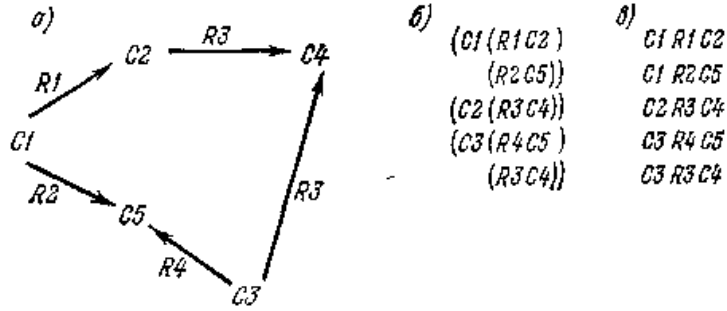


Рис.6. Абстрактная семантическая сеть.

На этом общую часть определения семантических сетей завершим. Очевидно, что разнообразие конкретных представителей семантических представлений определяется типами объектов и, в еще большей степени, типами отношений, определенными в каждом из предписаний.

4.4.2. Типы объектов

В семантических сетях используются три основных типа объектов: *понятия, события* и *свойства.*

Понятия являются константами или параметрами сущностей, описываемых коммуникационной семантической сетью, и обычно указывают предметы или абстракции.

События представляют собой действия, которые могут произойти в мире. Если мы определим *ситуацию* как описание части мира в определенный момент времени, то можно сказать, что все, что изменяет данную ситуацию, является событием. Одним из методов

представления событий является задание *глубинно-падежных семантических отношений*, которые указывают характеристики и действующих лиц данного события. Не вдаваясь в подробности грамматики глубинных падежей, о которой речь будет идти в последующих темах, отметим, что отношения *A, B* и C, приписанные дугам высказывания (рис.5), эквивалентны соответственно агентивному, объективному и локативному падежам Филмора. Другой метод описания событий состоит в указании изменений, которые событие производит, будучи применено к структуре сети, отражающей данную ситуацию. Результатом события является также некоторая ситуация, которую мы можем определить как образец в некоторой процедуре, описывающем последовательность действий, приводящем к этой ситуации.

Свойства используются для уточнения или модификации понятий, событий или других свойств. В случае понятий свойства могут быть особенностями, чертами или характеристиками, присущими или приписанными понятию. В случае событий свойства описывают некоторые общие, универсальные, постоянные характеристики, например, место, время, длительность и т. д.

Формально свойство является бинарным отношением, отображающим область своего определения, т.е. вершины, к которым свойство применяется, в область значений, т. е. значения, которое свойство может принимать. Рис. 7 иллюстрирует, как применение свойства «структура» к понятию «сеть» и свойства «время» к событию «ввел» со значениями «дискриминационная» и «вчера» соответственно дает высказывание «Виктор ввел вчера высказывание в дискриминационную сеть».

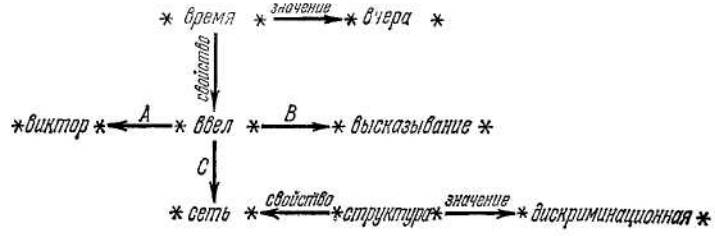


Рис.7. Представление свойств в семантической сети.

Здесь дуга, помеченная «свойство», указывает на аргумент, а дуга, помеченная «значение»,— на значение свойства. В некоторых случаях возможно расширение понятия свойства от бинарного к тернарному и многоместным отношениям. При этом дополнительные характеристики свойства связываются с вершиной свойства дугами, помеченными «относительно». Рис. 8 показывает, как введение аргумента «относительно момента T_0» уточняет значение «вчера» свойства «время».

Рис.8. Представление дополнительных характеристик свойств.

Вершины, входящие в семантическую сеть, независимо от их типа могут быть разделены на два класса. Один из них включает в себя понятия, события и свойства общего характера, описывающие в совокупности законы, действующие в мире, представленном семантической сетью. Другой класс — частные случаи общих объектов, описывающие конкретные проявления указанных выше законов, или просто некоторые факты. Вершины первого класса мы будем называть *общими,* вершины второго класса — *фактуальными.* Пример такого рода вершин приведен на рис.9, причем на рис.9, *а* утверждается что ПОЛ — свойство ЖИВОТНОГО и принимает возможное значение МУЖСКОЙ (общие вершины условно обозначены прописными буквами), а на рис.9, *б* указывается, что Виктор — мужского пола (фактуальные вершины условно ограничиваются с двух сторон звездочками).

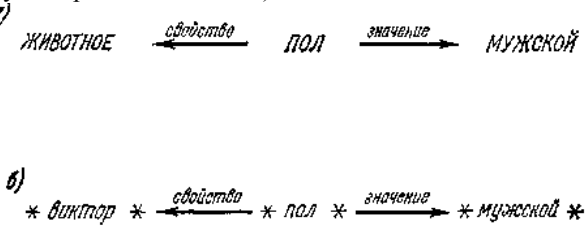


Рис.9. Общие и фактуальные вершины.

Кроме рассмотренных выше типов вершин — понятий, событий, свойств, в целях повышения эффективности вывода в семантических сетях вводятся так называемые *процедуральные* вершины. Одним из примеров фундаментального характера такого рода вершин являются *скелеты*. Существуют и другие процедуральные вершины более специализированного характера. Чаще всего они располагаются в районе границы между общими и фактуальными вершинами, т. е. на периферии семантической сети.

4.4.3. Типы отношений

Все многообразие семантических отношений, используемых в семантических представлениях, может быть условно разделено на четыре класса: *лингвистические, логические, теоретико-множественные* и *квантификационные.*

Лингвистические отношения включают в себя прежде всего глубинно-падежные отношения. В системах ИИ этот тип отношений играет ключевую роль в общей организации семантической сети, определяя как взаимосвязи отдельных объектов, так и структуру вывода в сети.

Другими двумя типами лингвистических отношений являются *характеризации глаголов* и *атрибутивные отношения.* В нашем изложении они описываются как вершины-свойства, однако в ряде конкретных систем эти отношения выделены введением отдельных дуг для каждого из них. К числу глагольных характеризаций относятся время, наклонение, вид, род, число, залог используемого глагола. К числу атрибутивных отношений относятся модификация, цвет, размер, форма, отношение собственности («притяжательность») и т. д. На рис.10, *а* приведено подробное представление глагола «ввел» с использованием глагольных характеризаций, а на рис.10, *б* представление того же глагола с использованием вершин типа свойств.

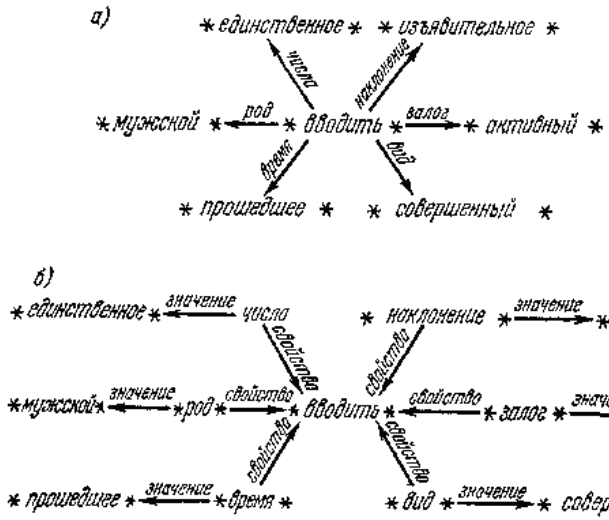


Рис.10. Представление характеризаций глаголов.

На рис. 11 показано, как используются атрибутивные отношения для представления словосочетания «маленькая древовидная дискриминационная сеть Виктора». Из приведенных примеров ясно, что глагольные характеризаций и атрибутивные отношения эквивалентны свойствам событий и понятий соответственно.

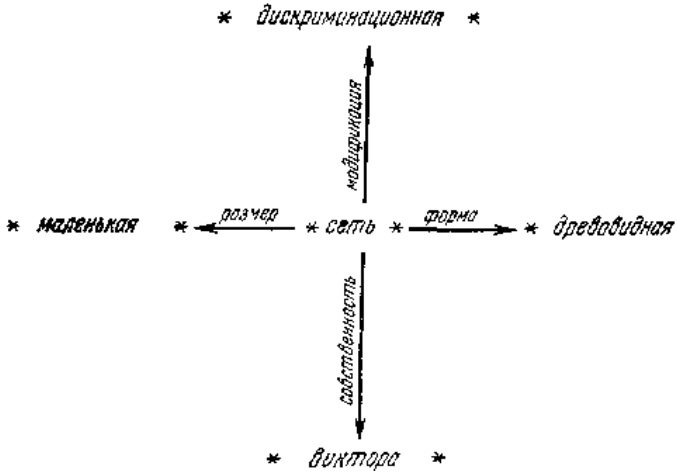


Рис.11. Представление атрибутивных отношений.

К *логическим отношениям* относятся операции исчисления высказываний: **дизъюнкция, конъюнкция, отрицание, импликация**. Практически все структуры семантических сетей включают неявное представление конъюнкции, поскольку **все занесенные в сеть высказывания считаются истинными**. Однако требование полноты логических отношений обуславливает необходимость добавления к конъюнкции как минимум отрицания. Отметим, что в большинстве семантических сетях определяется отрицание лишь элементарных высказываний, введенных в п. 4.4.1. В этом случае отрицание не образует логически полную систему с конъюнкцией. Если допустить явное представление всех операций исчисления высказываний в виде дуг семантической сети, может быть нарушено соглашение об истинности всех высказываний в сети: в семантической сети истинными станут только те высказывания, которые не являются конституентами составных высказываний. Возникает вопрос о том, как указать истинность конституенты в составном высказывании. На самом деле этой проблемы нет, если конституенты в составном высказывании связаны конъюнкцией, дизъюнкцией или импликацией, так как указание истинности одной из конституент сводит такое высказывание к конъюнкции, т. е. к неявному указанию истинности любой конституенты. Например,

$$p \wedge (p \vee q \vee r) = p,$$
$$p \wedge (p \rightarrow q) = p \wedge q.$$

Однако иногда возникает необходимость в независимом от составного выражения указании истинности конституенты (например, при выражении причин, намерений, отношения к чему-либо и т. д.). Эти случаи охватываются применением конъюнкции с константой «истина» или «ложь». На рис.12, *а, б,* соответственно изображено представление двух составных высказываний «Робот считает, что Виктор ввел высказывание в сеть, и он ввел его» и «Робот считает, что Виктор ввел высказывание в сеть, а он не ввел его».

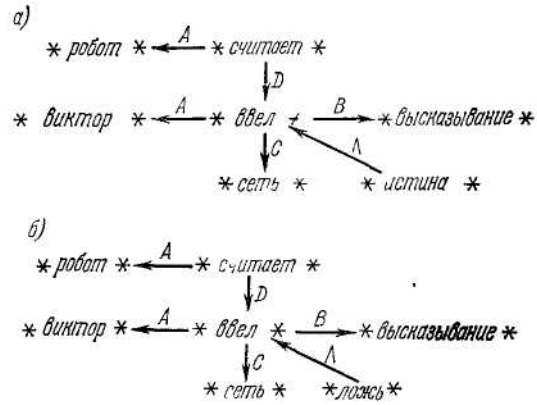


Рис.12. Представление составных модальных высказываний.

Теоретико-множественные отношения включают в себя такие отношения, как *подмножество* (SUB), *супермножество* (SUP), отношения части и целого, *элемент множества,* или пример (E), и другие.

Большинство отношений этого класса является представителями класса *транзитивных отношений,* т. е. отношений, связывающих вложенные друг в друга понятия от более общих к более частным (SUB) или наоборот (SUP), причем все свойства SUP-понятия автоматически становятся свойствами SUB-понятия, а свойства SUB-понятия представляют собой ограничения, накладываемые на SUP-понятия с целью определения SUB-понятия. Отношение E является частным случаем SUB, связывающим общую вершину с фактуальной. Отношения части и целого, подчиненности, сходства, близости и т. д. основаны на отношениях SUP и SUB, однако могут иметь более сложную структуру представления и (или) вывода. Так, например, два понятия являются сходными, если они имеют общее SUP-понятие, а большинство одноименных свойств этих понятий имеет одинаковые значения. Таким образом, сходство является *размытым аналогом* эквивалентности, которое в свою очередь также представляет собой теоретико-множественное отношение.

Отношения SUP и SUB могут быть определены и для событий. В этом случае структуры событий, образуемые этими отношениями, аналогичны множествам соответствующих событий и связанных с ними частных случаев событий. При этом свойства SUB-событий

могут иметь в качестве значений наречия («быстро», «бесплатно») или другие модификаторы, которым сопоставляются числовые эквиваленты в понятиях теории лингвистических переменных. ***Лингвистическая переменная*** — в теории нечётких множеств, переменная, которая может принимать значения фраз из естественного или искусственного языка. Например, лингвистическая переменная «скорость» может иметь значения «высокая», «средняя», «очень низкая» и т. д. Фразы, значение которых принимает переменная, в свою очередь являются именами нечетких переменных и описываются нечетким множеством.
Математическое определение.

Лингвистической переменной называется пятерка $\{x, T(x), X, G, M\}$, где x — имя переменной; $T(x)$ — некоторое множество значений лингвистической переменной x, каждое из которых является нечеткой переменной на множестве X; G есть синтаксическое правило для образования имен новых значений x; M есть семантическая процедура, позволяющая преобразовать новое имя, образованное процедурой G, в нечеткую переменную (задать вид функции принадлежности), ассоциирует имя с его значением, понятием. $T(x)$ также называют базовым терм-множеством, поскольку оно задает минимальное количество значений, на основании которых при помощи правил G и M можно сформировать остальные допустимые значения лингвистической переменной. Множество $T(x)$ и новые образованные при помощи G и M значения лингвистической переменной образуют расширенное терм-множество.

Пример: нечёткий возраст
Рассмотрим лингвистическую переменную, описывающую возраст человека, тогда:

- x: «возраст»;
- X: множество целых чисел из интервала [0, 120];
- $T(x)$: значения «молодой», «зрелый», «старый». множество $T(x)$ - множество нечетких переменных, для каждого значения: «молодой», «зрелый», «старый», необходимо задать функцию принадлежности, которая задает информацию о том, людей какого возраста считать молодыми, зрелыми, старыми;
- G: «очень», «не очень». Такие добавки позволяют образовывать новые значения: «очень молодой», «не очень старый» и пр.

- *M*: математическое правило, определяющее вид функции принадлежности для каждого значения образованного при помощи правила *G*.)

Квантификационные отношения включают в себя *логические кванторы* (общности, существования), *нелогические кванторы* (много, несколько), а также просто *числовые характеристики* объектов. Приведем ряд причин, по которым необходимо ввести в семантическую сеть логические кванторы.

1. Многие высказывания содержат кванторы общности или существования.
2. Логические кванторы требуются для декларативного представления общих знаний (законов мира).
3. Определение сложных понятий и событий требует логической квантификации (например, «ходить — это значит в любой момент времени касаться хотя бы одной ногой земли»).
4. Кванторы требуются для определяющих описаний множеств.

Несмотря на важность введения логической квантификации в семантическую сеть, достижения в этой области ограничивались приписыванием одного или двух кванторов к предикатам.

Одним из способов введения логических кванторов является представление семантической сети в языке, подобном исчислению предикатов первого порядка. Рассмотрим вариант такого формализма представления на примере высказывания «Каждая собака преследует некоего кота». Это высказывание может быть преобразовано из исходного представления

$$(\forall x)(\text{собака } (x) \rightarrow (\exists y)(\text{кот } (y) \wedge \text{преследует } (x, y))) \quad (5)$$

в вид, близкий к бескванторной нормальной форме

$$\text{собака } (x) \rightarrow (\text{кот } (f(x)) \wedge \text{преследует } (x, f(x))). \quad (6)$$

Соответствующее представление в семантической сети приведено на рис.13, *а*.

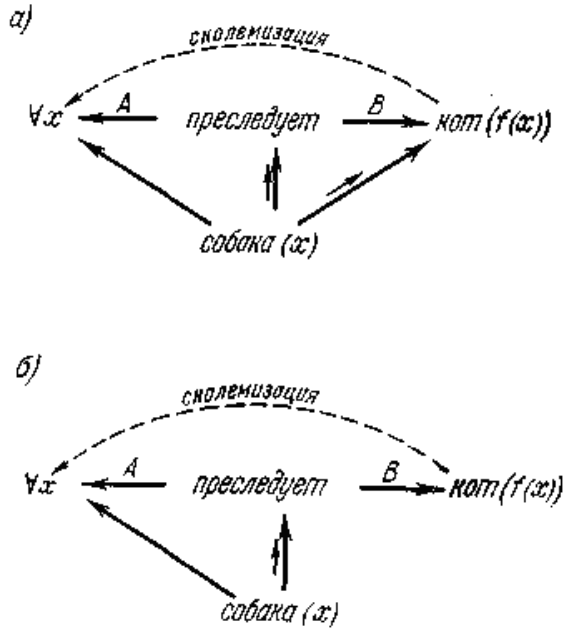


Рис.13. Представление высказывания «Каждая собака преследует некоего кота».

Таким образом, мы неявно указываем квантор существования, вводим новую вершину, соответствующую квантору общности, и проводим к ней две дуги, одна из которых указывает связывание переменной в предикате «собака» с введенным квантором, а вторая, соответствующая сколемизации, проводится от вершины, связанной неявно квантором существования, поскольку в (5) он стоит за квантором общности. В общем случае сколемизирующие дуги проводятся от всех вершин, связанных неявно кванторами существования, ко всем вершинам тех кванторов общности, которые стоят перед кванторами существования. Заметим, что если мы предположим, что *(∃ y) (кот(y)),* то (2.22) приводится к виду

кот *(f(x))* ∧ (собака *(x)* → преследует *(x, f(x))),* (7)

и этому высказыванию соответствует фрагмент семантической сети (рис.13, *б*).

Другим способом введения логических кванторов является *разбиение семантической сети на пространства*. Это разбиение осуществляется таким образом, что каждая вершина и дуга семантической сети принадлежит точно одному пространству. Все вершины и дуги,

лежащие в различных пространствах, различимы между собой, а дуги, связывающие различные пространства, принадлежат тому из них, в котором они начинаются. Разбиение семантической сети на пространства задает структуру связи этих пространств в виде направленного графа *(S, V),* пример которого показан на рис.14.

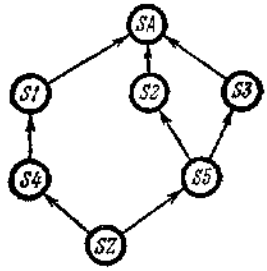


Рис.14. Представление структуры связи пространств в виде графа

Вершины этого графа соответствуют пространствам семантической сети, а дуги задают отношение «видимости» на этом графе. Из любой вершины *Si* видны те и только те вершины, которые лежат на всех возможных путях, ведущих из *Si* в *SA.* Например, из *S*5 графа, (рис. 14) видны вершины *S2, S3* и *SA.* Очевидно, что *SA* видима из всех пространств сети, а из *SZ* видна вся семантическая коммуникационная сеть. Рассмотрим представление логической квантификации с помощью разбиений, используя следующую последовательность примеров:

а) Собака Шарик преследует кота Ваську.
б) Каждая собака преследует некоего кота.
в) Все собаки преследуют кота Ваську.
г) Все собаки преследуют всех котов.

Приведем формальную запись этих высказываний:

а) ($\exists$ Васька $\in$ КОТЫ) ($\exists$ Шарик $\in$ СОБАКИ)
(преследует (Шарик, Васька)).

б) ($\forall x$) (собака *(x)* $\rightarrow$ ($\exists y$) (кот *(y)* $\wedge$ преследует *(x, y))).*

в) ($\forall x$) (собака (*x*) $\rightarrow$ ($\exists$ Васька $\in$ КОТЫ) (преследует *(x,* Васька))). (8)

г) ($\forall x$) *(* $\forall y$) (собака *(x)* $\rightarrow$
$\rightarrow$(кот(*y*) $\wedge$ преследует *(x, y))).*

Соответствующие семантические представления показаны на рис.15, *а—г.*

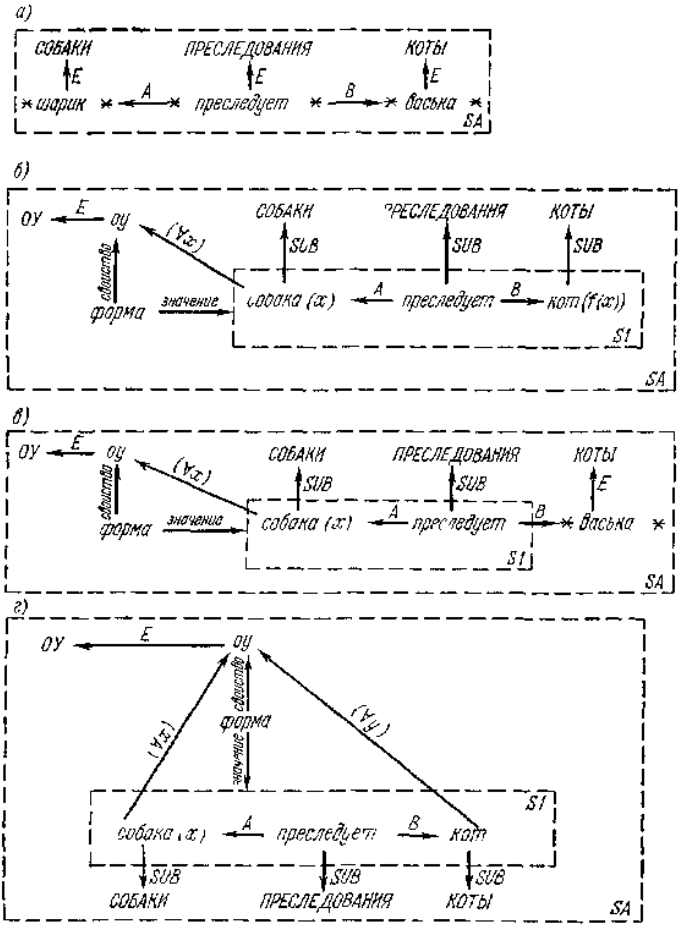


Рис.15. К разбиению семантической сети на пространства.

Высказывание (8, *а)* размещается в одном пространстве самого верхнего уровня *(SA),* поскольку в данном случае речь идет о конкретном событии, происшедшем с конкретными индивидуумами. Поскольку последние представлены фактуальными вершинами, то мы дали для иллюстрации их SUP-вершины с указанием отношения E.

Так как высказывание (8, *б)* описывает некоторое множество событий и действующих лиц, то ему как *форме* соответствует некоторое *общее утверждение* $\text{оу} \in \text{ОУ}$, где ОУ — множество всех общих утверждений, включающих кванторы общности. Мы представляем это введением

формы как свойства оу, имеющего в данном случае в качестве значения выражение, лежащее в области действия квантора общности, т. е. выражения в пространстве *S*1. Таким образом, пространство *S*1 является областью действия квантора $\forall x$, а сколемизация переменных, связанных квантором существования, представляется помещением предикатов от этих переменных в пространство *S*1 (в нашем примере — кот *(f(x))).* Формализм представления квантора общности завершается связыванием оу дугой, помеченной $\forall x$, к квантифицируемым предикатам, причем эта дуга идет из *S*1 в *SA, S*1 *V SA* (читается «*SA* видимо из *S*1»). Заметим, что на рис. 15, *б* связь SUP-объектов с объектами в *S*1 осуществлена с помощью SUB (а не E), поскольку форма описывает некоторые подмножества событий и понятий, а не частный пример. Отметим, что рис.15, *б* иллюстрирует тот факт, что константа, находящаяся под знаком $\exists$, выносится из пространства *S*1 (естественно, что она не подлежит сколемизации), а рис.15, *г* показывает, что последовательность кванторов общности представляется в одном пространстве с соответствующим количеством дуг, исходящих из оу к квантифицируемым предикатам. На рис. 16 показан пример представления сложной последовательности логических кванторов в виде вложенных пространств так, что *S*4 *V* S1, *S*1 *V SA* (отношение видимости, конечно, транзитивно).

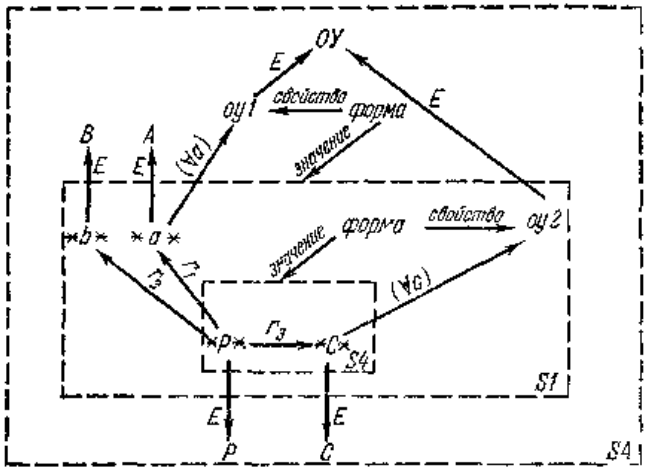


Рис.16. Представление последовательности логических кванторов, $(\forall a \in A)(\exists b \in B)(\forall c \in C)(p(a,b,c))$.

Изложенные методы указания областей действия кванторов могут быть использованы для представления λ-выражений и модальностей. Мы не будем останавливаться на изучении этих возможностей, однако с целью иллюстрации приведем на рис. 17 представление с помощью разбиений модального высказывания (рис.12).

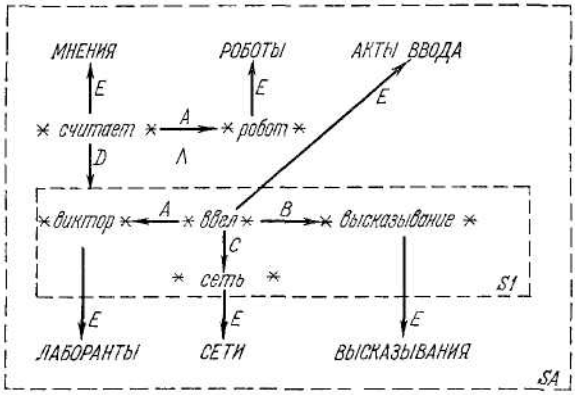


Рис.17. Представление модальностей с помощью разбиений.

Представление *нелогических кванторов* остается проблемой в семантических сетях так же, как и в других формализмах. Аппарат для работы с нелогическими кванторами, как и с другими размытыми понятиями, разрабатывается в рамках теории лингвистических переменных, однако этот подход еще не связан с семантическими представлениями.

Наконец, *числовые характеристики* объектов представляют собой просто количество объектов данного типа и могут быть выражены, например, с помощью вершин свойств.

4.4.4. Скелеты (фреймы) и сценарии

В п.4.4.2 мы упомянули два способа описания событий — с помощью глубинно-падежных отношений и с помощью процедур (директив, операций). Эти способы отражают соответственно семантическое и процедуральное описание весьма общего класса структур данных для представления хорошо известных, стереотипизированных ситуаций, которые Минский называет «скелетами» (frame), а Шенк — «сценариями». По существу ***скелеты (сценарии)* — это совокупность**

условий применения (предпосылок), моделей действия и выводов для достижения определенной цели, описывающей стереотипизированную ситуацию. Сценарий содержит целый ряд предпосылок, которые, как предполагается, достаточно хорошо описывают стандартный набор условий его активизации. Так, например, если мы рассматриваем сценарий «пойти на день рождения», то такие *условия,* как «одеть костюм», «купить подарок», будут считаться *стандартными,* в то время как «одеть каску» (во избежание обвала потолка) будут считаться *исключительными.* Чем меньше предпосылок будет иметь сценарий, тем более общим он является, и наоборот. Число действительных характеристик ситуации, в которой активируется сценарий, может не соответствовать числу его предпосылок. В случае недостатка предпосылок оказывается необходимым введение в систему специализированных процедур, соответствующих избыточным (и поэтому исключительным) характеристикам ситуации. Однако в этом случае сценарий может быть использован для указания того, какого рода специализированные процедуры необходимы. Заметим, что в сценарии создаются ветвления, соответствующие исключительным для него условиям.

В случае избытка предпосылок сценарий не может функционировать целиком, однако при этом могут образовываться неполные выводы или (и) указания того, каких характеристик ситуации не хватает для полного использования сценария.

В процессе выполнения сценария могут возникнуть новые ситуации, требующие вызова других сценариев, так что могут **образовываться последовательные, вложенные, рекурсивные или параллельные *композиции сценариев***. В рамках настоящего параграфа мы ограничимся кратким описанием **семантического представления сценариев** (именуемого впредь ***сценарием***) и **процедурального представления сценариев** (именуемого впредь ***скелетом (фреймом)***) применительно к их использованию в семантических сетях.

Назовем *сценарием* **совокупность событий и свойств, связанных отношениями с помощью соответствующих дуг и понятий, заполняющих глубинные падежи, или с помощью отношений времени и (или) причины-следствия**. С операциональной точки зрения сценарий следует рассматривать как некоторый *сложный образец,* который при сопоставлении его с некоторой структурой позволяет системе делать определенные *выводы* и *предсказания.* В качестве простейшего сценария можно было бы привести высказывание на рис. 5 (при условии замены фактуальных вершин на общие), которое могло бы сопоставляться со структурой, где Виктор вводил бы вполне определенное высказывание во вполне определенную сеть.

Этот сценарий, однако, тривиален, поскольку в нем не содержится возможности делать какие-либо выводы или предсказания.

Между сценариями могут быть установлены SUP- и *SUB-отношения.* Этого можно достигнуть, строя SUP- или SUB-объекты объектов, входящих в определение сценария. Сценарии могут быть организованы в структуры, связанные *определяющим отношением* (отношением DEF). **Это отношение задает организацию сценария как композицию других, более простых сценариев**, причем, как указывалось ранее, эта композиция может быть последовательной, вложенной, рекурсивной или параллельной. В частности, может быть реализована структура сценариев, аналогичная пропозициональному графу. Сценарии задаются совокупностью вершин, отражающей структуру событий той или иной степени сложности и общности, а также действующих лиц и характеристик событий, связанных с событиями глубинно-падежными отношениями.

В отличие от сценариев, ***скелеты (фреймы)*** **представляются в семантической сети специальными вершинами.** Прежде всего рассмотрим отличия вершин скелетов от обычных вершин семантической сети.

1. Прежде чем переходить от вершины скелета к связанным с нею вершинами, следует осуществить проверку применимости скелета. Если он неприменим, то необходимо обратиться к списку альтернативных скелетов, описывающих сходный класс ситуаций. В случае применимости скелета он может определить, по каким дугам идти дальше и в каком порядке.
2. При достижении некоторого скелета он может взять на себя управление и уже не возвращать его вызвавшему скелету.
3. С вершиной скелета могут быть связаны дугами некоторые *субскелетные вершины.* Здесь под *субскелетными вершинами* мы подразумеваем вершины, которые не входят в семантическую сеть, могут быть выбраны только при активации соответствующих скелетов и которым соотносятся некоторые процедуры или функции. Последние могут проверять условия и выполнять действия, используя для этого другие субскелетные вершины данного скелета или скелета, который вызвал данный скелет. В частности, через указанные процедуры (функции) могут быть активированы другие скелеты. Схема скелета приведена на рис.18.

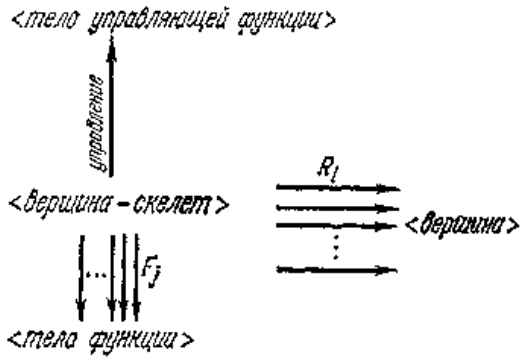


Рис.18. Схема скелета.

Здесь R_i — обычные отношения, заданные в семантической сети, «управление» — дуга, ведущая к вычислению тела управляющей функции (проверка применимости скелета, указание последовательности перехода к субскелетным вершинам, возврат управления вызывающему скелету), F_j — метки дуг, направленных к субскелетным вершинам. Если дуга F_j выбрана управляющей функцией, то вычисляется тело процедуры или функции F_j. Заметим, что в случае невыполнения некоторых из условий F_j может сигнализировать о недостатке информации о характеристиках ситуаций.

4.4.5. Процессы понимания и вывода в семантических представлениях

Ранее мы описали различные элементы представления семантического знания в семантических сетях. Сейчас рассмотрим ряд вопросов, связанных с обработкой информации в сети. К числу этих вопросов относятся:

— как извлекать необходимую информацию?

— как вводить новую информацию в сеть?

— каким образом выделять в сложных и разветвленных сетях участки, имеющие отношение к интересующей систему области?

— как осуществлять процессы вывода ответов на вопросы, заданные семантической сети?

Оказывается, что ответы на эти вопросы тесно связаны между собой и основаны на общей трактовке процессов понимания. **Мы определяем *понимание* как интерпретацию новых фактов относительно текущего контекста.** Более формально, пусть $C(T_1, T_2, ..., T_i)$ —

ситуация или *контекст,* установленный в результате понимания последовательности высказываний T_1, T_2, ..., T_i. Пусть далее $I(T_{i+1}, C(T_1, T_2, ..., T_j))$ *— интерпретация входного высказывания* T_{i+1} в контексте, установленном T_1, T_2, ..., T_j. Тогда **эффективный алгоритм вычисления *I (T, C)* называется *пониманием.*** С точки зрения нашего определения все поставленные выше вопросы обретают общую концептуальную основу. Нам необходимо применить общие методы эффективного поиска участка семантической сети, имеющего отношение к рассматриваемой области (или к заданным вопросам, или вновь вводимому высказыванию), и уже только потом осуществить некоторые специализированные манипуляции в зависимости от конкретного характера задачи.

Становление семантических представлений началось в 1965-1970 г.г. В них в качестве объектов определяются понятия и свойства, а основным видом отношений являются транзитивные теоретико-множественные и логические отношения. Классические сети носят, во-первых, статический и декларативный характер и, во-вторых, обладают значительной однородностью. Эта однородность позволяет описать процессы выделения контекста в таких сетях с помощью одного базового метода *поиска пересечений,* выделяющего участок сети, связанный с входными понятиями (директивами) X_1, X_2, ... X_n. Суть метода состоит в том, чтобы, начиная от вершин X_1, X_2, . . ., X_n и двигаясь по дугам, помеченным транзитивными отношениями, искать такие понятия (возможно, ближайшие в некотором смысле), которые находятся на пересечении построенных путей. Тогда множество пройденных вершин и соединяющих их дуг образуют *контекст,* связывающий исходные понятия (директивы) X_1, X_2, . . ., X_n. Транзитивность отношений позволяет строить достаточно длинные пути. Проиллюстрируем сказанное для случая двух исходных понятий X, Y и транзитивного отношения R на примере ряда вопросов и ответов относительно связи X и Y (в качестве R берется отношение SUB). Для того чтобы определить, находится ли X в отношении R с Y, ищутся все возможные пересечения X и Y. При этом может возникнуть ряд вариантов.

1. Некоторый путь связывает X с Y, т. е. XRY
2. Некоторый путь связывает Y с X, т. е. $XR^{-1}Y$, R^{-1} — обратное и потому также транзитивное отношение.
3. Пути, начинающиеся в X и Y, не пересекаются.
4. Существует пересечение путей, начинающихся в X и Y, т. е.

$$(\exists\ w)(XRw \wedge YRw). \qquad (9)$$

В случае 1 мы даем утвердительный ответ на вопрос, является ли X SUB-объектом для Y.

Пример. Является ли Шарик собакой?

Да.

В случае 2 утвердительный ответ можно дать лишь в некоторых частных случаях.

Пример. Является ли собака Шариком?

Возможно, что да.

В случае 3 ответ является отрицательным, так как *X* и *Y* не имеют общих SUP-объектов.

Пример. Является ли трамвай желанием?

Нет.

Рассмотрим подробнее случай наличия пересечения (или пересечений) входных объектов. Здесь следует выделить три случая:

а) *X* и *Y* взаимно исключают друг друга.

Ответ отрицательный.

Пример. Является ли собака кошкой? (пересечение— млекопитающее).

Нет.

б) *X* и *Y* содержат идентичные свойства с различными, по крайней мере для одного свойства, значениями.

Ответ отрицательный.

Пример. Является ли индеец лордом?

Нет.

в) Свойства *X* и *Y,* а также значения этих свойств не различаются. Здесь возможны два подслучая. Если *X* и *Y* обладают в сети исчерпывающим списком свойств, то следует вывод об идентичности *X* и *Y,* т. е. ответ утвердительный. Если же список свойств *X* и *Y* не является исчерпывающим, то точный ответ неизвестен и следует применять приблизительные методы, в частности, *функциональный, негативный* и *индуктивный.* Мы не будем останавливаться на этих методах. Отметим лишь, что *индуктивный вывод* в семантических сетях играет важную роль при вложении новой информации в сеть. Действительно, если в сеть «часто» (в некотором смысле) вводятся понятия с одинаковым значением некоторого свойства, то есть смысл приписать это свойство общему SUP-понятию вводимых понятий. Например, если воробьи, грачи, ласточки и т. д. летают, то это свойство следует приписать понятию «птица». Здесь реализуется *режим обобщения индуктивного вывода.* Поскольку отклонения от этого свойства редки (например, пингвин не летает), то такие отклонения следует приписывать самим вводимым понятиям. Это — *режим дискриминации индуктивного вывода.* Итак, в **процессе ввода информации в сеть общие свойства обобщаются, а частные — дискриминируются.**

Рассмотрим теперь, как выделяется контекст в семантических сетях с событиями и сценариями. Этот процесс осуществляется путем построения *неполных выводов, ожиданий* и *предсказаний*. Ввод некоторого объекта в контекст представляет собой *ожидание* системой того, что этот объект вскоре понадобится. Это ожидание создает некоторые неполные пути вывода, причем «дырки» в этих путях образуют предсказание необходимой дополнительной информации. По мере ее поступления пути вывода наполняются недостающими деталями. Заметим, что таким выводам должны быть присвоены эвристические оценки с целью ограничения глубины их распространения и, следовательно, излишнего расширения контекста.

Рассмотрим три примера выделения контекста при работе со сценариями:

1. *A вырабатывает B,* где *A* —событие или сценарий, *B* — понятие (формально это могло бы быть представлено как *A* result *B,* result — глубинный падеж). Этот пример соответствует окончанию работы сценария *A* и вычислению частного случая (подстановки) *B*. Последний добавляется к контексту. Если оказывается, что *B* используется другим событием или сценарием *C* в качестве входа, то возникает предписание активации *C*, причем указанный вывод объясняет, почему *B* — предусловие *C*.
2. *A вызывает B,* т.е. *B* — следствие *A* (формальная запись *A* effect *B,* effect —-отношение причины-следствия). После вычисления *A* следует активировать *B,* найти его соответствующий частный случай и поместить в контекст. Это ожидание позволяет системе предписать большую часть сценария на основе его небольших фрагментов директив. Точно так же, как только вычислен *B,* предположение о том, что существует *A,* которое вызвало *B,* является весьма правдоподобным и сильным.
3. *B — предусловие A* (формальная запись — *A* prereq *B,* prereq — отношение причины-следствия). Здесь *B* может быть характеристикой или событием. Если *A* введено в контекст, то мы можем уверенно ввести и *B* в контекст, поскольку *A* не может быть вычислено, пока *B* не будет удовлетворено. С другой стороны, если *B* уже находится в контексте, это серьезное основание для предписания того, что *A* должно быть введено в контекст.

Таким образом, образование контекста является фактически процессом создания *виртуальной базы данных.*

Естественно, что контекст оказывает влияние на ввод новой информации в сеть. Рассматривая этот процесс, следует подчеркнуть, что наличие сценариев обуславливает определенную стратегию введения: мы заинтересованы в том, чтобы сопоставить вводимую

информацию с *наиболее конкретным сценарием* из всех возможных, причем возможно, что в случае неоднозначности либо потребуется дополнительная входная информация, либо информация будет помещена на тот уровень, где еще имеется однозначность, и будет «спущена» SUB-сценариям по мере поступления уточняющей информации.

Сущность алгоритма ввода новой информации состоит в поиске всех кандидатов-сценариев, сопоставляющихся вводимой структуре путем последовательного анализа всех событий и характеристик. В случае нахождения некоторого сценария S производится сопоставление входной структуры всем SUB-сценариям S. Таким образом, создается список наиболее конкретных SUB-сценариев, сопоставляющихся входной структуре. После этого поиск идет ниже (в смысле SUB) каждого из этих SUB-сценариев в попытке найти частное сопоставление с входной структурой. В результате или часть структуры идентифицируется с уже существующими в сети вершинами, или новые вершины создаются и помещаются ниже всех сценариев в списке.

Взаимодействие описанного процесса с установленным к этому моменту контекстом заключается в том, что

— поиск производится, начиная с контекста, и лишь при неудаче распространяется вверх по сети (в смысле отношения SUP);

— в случае, если сопоставление происходит вне контекста, эта часть семантической коммуникационной сети с соответствующими путями вывода присоединяется к контексту.

Таким образом, и в сложных декларативно-процедуральных семантических предписаниях процессы выделения контекста и вложения новой информации тесно связаны.

5. Отображения логики предикатов и логики высказываний в теории ИИ

5.1. Исчисление предикатов как язык искусственного интеллекта

Для автоматизации процессов постановки и решения интеллектуальных задач роботу необходим адекватный язык. Этот язык должен служить не только и не столько средством представления

знаний (информации), сколько средством логического анализа интеллектуальных задач.

Обычные человеческие языки, развивавшиеся под влиянием практических потребностей простоты общения (что далеко не всегда совместимо с точностью и надежностью логического анализа!), для этой цели плохо подходят. По этой причине желательно, даже практически необходимо использовать в качестве языка СИИ спе-. циально созданный формализованный язык. Такой язык в противоположность обычному языку должен следовать за логической формой и воспроизводить ее даже в ущерб краткости и легкости общения, если это будет необходимо. Главной отличительной чертой такого формализованного языка является наличие в нем особой системы логического вывода или дедукции.

В качестве языка СИИ, удовлетворяющего указанным требованиям, мы возьмем *исчисление предикатов.* В терминах исчисления предикатов можно сформулировать многие предложения и утверждения, выраженные на естественных языках, а также формализовать процесс рассуждений и доказательств. Благодаря этому может быть устранен или во всяком случае резко снижен «языковый барьер» между СИИ и человеком.

Для того чтобы описать исчисление предикатов, мы должны воспользоваться какой-то частью обычного языка и в терминах этого языка образовать словарь и сформулировать правила формализованного языка, включая правила логического вывода. Исчисление предикатов (точнее, его синтаксис и семантика) определяется следующим очень экономным словарем символов и правилами их соединения и интерпретации:

1. Имена. Это — заимствованные из обычных языков выражения, служащие для непосредственного обозначения предмета. Примерами имен являются: «роботы», «мозг», «манипулятор», «источник энергии», «искусственный интеллект» и т. д.

Интересно отметить, что в одном или различных языках разные имена могут быть синонимами и выражать один и тот же смысл. С другой стороны, одно имя в различных языках или даже в одном языке (при омонимии) может выражать разный смысл.

Полное понимание языка требует знания смысла всех слов языка. Естественно потребовать, чтобы каждое имя имело точно один смысл. Такая однозначность обеспечивается в логике предикатов. А вот в обычных языках как мы знаем, дело обстоит совсем не так.

2. Константы и переменные. *Константа* — это собственное имя. Примерами констант являются собственные имена чисел, людей, роботов.

Переменная — это символ, содержание которого совпадает с содержанием константы, за исключением лишь того, что единственный денотат константы заменен здесь возможностью различных значений переменной. С каждой переменной связана некоторая непустая область ее возможных значений. Поэтому к содержанию переменной относится в некотором смысле и содержание собственного имени области ее значений. Нужно особо подчеркнуть, что переменная в исчислении предикатов есть определенного рода символ, а не предмет (например, число), который этот символ обозначает.

3. Функции и термы. *Функции* — это операция, которая будучи применена к чему-то как к аргументу, дает некоторый объект в качестве значения функции для данного аргумента. В природе всякой функции лежит свойство быть применимой лишь к некоторым предметам.

Предметы, к которым функция применима, составляют область определения функции, а ее значения составляют область значений функции. Сама функция состоит в определении некоторого значения для каждого аргумента из области ее определения. Например, функция распознавания состоит в определении номера класса, к которому принадлежит объект, трактуемый как аргумент.

Для того чтобы обозначить значение функции для некоторого аргумента, обычно пишут имя этой функции и приписывают к нему справа имя аргумента, взятое в скобки. Так, если f — функция, а x принадлежит к области ее определения, то $f(\{x)$ есть значение функции f для аргумента x. Если функция применима к упорядоченной системе из n аргументов, то она называется n-арной.

Важную роль в дальнейшем играют выражения для функций, значения которых принадлежат той же области, что и их аргументы. Такие выражения называются *термами.* Терм — это выражение, построенное, исходя из символов предметных переменных и констант, с помощью символов функций. Например, если f есть n-арная функция и уже известно, что $x_1, \ldots, x_n$— термы, то $f(x_1, \ldots, x_n)$ есть терм. Содержательно терму соответствует имя некоторого предмета.

4. Предложения, высказывания и предикаты. Простейшим выражением в обычных языках является *предложение.* Предложение — это такое соединение слов, которое имеет самостоятельный смысл, т. е. выражает законченную мысль. Каждому предложению сопоставим *высказывание* (выражаемое этим предложением), предполагая при этом, что каждое высказывание или истинно, или ложно и не может быть одновременной истинно и ложно. Таким образом, высказывание можно рассматривать как величину, принимающую только два значения: «истина» (И) или «ложь» (Л).

Предположим теперь, что *x* представляет собой произвольный предмет из некоторого множества *{x},* а *F(x)* — какое-либо высказывание о *x.* Выражение *F(x)* становится определенным, когда переменная *x* заменена определенным значением (именем предмета) из множества *{x}.* Например, выражение *«x* есть животное» становится вполне определенным высказыванием, если *x*— это робот (ложное высказывание) или если *x* — это собака (истинное высказывание).
Так как с нашей точки зрения каждое определенное высказывание представляет собой И или Л, то выражение *F(x)* означает, что каждому предмету из *{x}* поставлен в соответствие один из двух символов: И или Л. Иначе говоря, *F(x)* представляет собой функцию, определенную на множестве *{x}* и принимающую только два значения: И и Л. Аналогично неопределенное высказывание о двух, трех и более предметах представляет функцию со значениями И и Л от двух, трех и более переменных. Эти неопределенные высказывания (функции одной или нескольких переменных) вида $F(x_1,..., x_n)$ мы будем называть *логическими функциями* или *предикатами.*

5. Элементарные (атомарные) и правильно построенные формулы. Каков бы ни был символ *n*-местного предиката *F* и каков бы ни был выбор термов $x_1, ..., x_n$, (не обязательно различных), выражение $F(x_1, ..., x_n)$ мы будем называть *элементарной,* или *атомарной, формулой.* Из этого определения следует, что, например, имена предметов не являются формулами.
Рассматривая элементарные формулы как величины, способные принимать только значения И и Л, мы определим над ними операции, которые позволяют из данных формул получать новые. Эти операции, по существу, выражают употребительные в обычных языках связки. Если *A* и *B* — какие-либо данные формулы (т. е. либо элементарные формулы, либо уже построенные сложные формулы), то $A \wedge B$, $A \vee B$, $A \rightarrow B$, $A \leftrightarrow B$ также являются (сложными) формулами. Если *A* — данная формула, то $\neg A$ — также (сложная) формула. Первые четыре операции — бинарные (двухместные), пятая — унарная (одноместная). Символы $\wedge$, $\vee$,$\rightarrow$, $\leftrightarrow$, $\neg$ называются соответственно *конъюнкцией, дизъюнкцией, импликацией, эквивалентностью* и *отрицанием.* Их можно читать, пользуясь словами, приведенными в правой части следующей таблицы:

$\wedge$ — «и»;
$\vee$ — «или», «... или, ... или», «и/или»;
$\rightarrow$ — «влечет», «если..., то...,» «только если»;
$\leftrightarrow$ — «равносильно», «эквивалентно», «тогда и только тогда»,
$\neg$ — «не», «неверно, что».

Прочтение сложных формул может стать неоднозначным, если не ввести скобок, указывающих, в каком порядке формулы связываются между собой. Поэтому мы будем писать *(A→ B) →C* или *A → (B →* C), а не *A → B → C.* Впрочем, число скобок можно уменьшить, приписав нашим связкам убывающие «ранги» в следующем «порядке старшинства».

$$\leftrightarrow, \rightarrow, \vee, \wedge, \neg.$$

Там, где возможны были бы два способа построения формулы, связка более высокого ранга имеет большую область действия. Так, *A→* B ∧ C означает *A → (B ∧ C).* Связка ¬ имеет наименьший ранг, так что, например, ¬ *A ∨ B* означает *(¬A) ∨ B,* а не ¬ *(A ∨ B).*
При построении сложных формул возникает вопрос, как определить значения сложных формул, зная значения простых формул, которые их составляют? Ответ на этот вопрос дается нижеследующей таблицей истинности.

A	*B*	¬*A*	¬*B*	*A∧B*	*A∨B*	*A→B*	*A↔B*
И	И	Л	Л	И	И	И	И
Л	И	И	Л	Л	И	И	Л
И	Л	Л	И	Л	И	Л	Л
Л	Л	И	И	Л	Л	И	И

Таким образом, *A ↔ B* истинно тогда и только тогда, когда *A* и *B* имеют одинаковые значения (почему ↔ и называют «эквивалентностью»); *A → B* ложно тогда и только тогда, когда *A* истинно, а *B* ложно; *A ∨ B* истинно тогда и только тогда, когда и *A,* и *B* истинны; *A ∨ B* ложно тогда и только тогда, когда и *A,* и *B* ложны; наконец, ┐ *A* истинно тогда и только тогда, когда *A* ложно.

Кроме пяти упомянутых символов-связок, в исчислении предикатов употребляются еще два символа, выражающие операции утверждения всеобщности и существования . Символ ∀ *x* называется *квантором всеобщности,* а символ ∃ *x — квантором существования.*

Формула ∀ *xF(x)* истинна, когда *F(x)* истинно для каждого элемента *x* области *{x},* и ложна в противном случае. Соответствующее ей словесное выражение будет: «для всякого *x F(x)* истинно». Формула ∃ *xF(x)* истинна, если существует элемент области *{x},* для которого *F(x)* истинно, и ложна в противном случае. В обычном языке этой

формуле соответствует выражение: «существует x такое, что $F(x)$ истинно».

Мы будем говорить, что в формулах $\forall xF(x)$ и $\exists xF(x)$ переменная x связана соответствующим квантором. Ясно, что сами эти формулы от x не зависят. Заметим, что $\neg(\forall xF(x)) \leftrightarrow \exists x\neg F(x)$.

Теперь мы можем дать определение *правильно построенной формулы* (ППФ) на языке ИИ. ППФ называется выражение, которое может быть построено исходя из элементарных (атомарных) формул с помощью операций перехода от формулы A к формулам $\forall xF(x)$ и $\exists xF(x)$, от формул A и B к формулам $A \wedge B$, $A \vee B$, $A \rightarrow B$, $A \leftrightarrow B$, $\neg A$, $\neg B$. Элементарная формула или ее отрицание, входящие в ППФ, называются литерами (или литералами), а дизъюнкция литер называется простым дизъюнктом.

6. Интерпретации. ППФ имеет смысл только тогда, когда имеется какая-нибудь интерпретация входящих в нее символов. Под *интерпретацией* мы будем понимать всякую систему, состоящую из непустого множества D, называемого предметной областью (областью проблем ИИ), и какого-либо соответствия, относящего каждому символу n-местного предиката некоторое n-арное отношение в D, каждому символу функции от n аргументов некоторую n-местную операцию в D и каждой константе — некоторый элемент из D. Например, если D есть множество всех проблем, то отношение между двумя проблемами, состоящее в том, что первая из них «проблемнее» (по каким-то определенным параметрам) второй, можно отождествлять с множеством всех упорядоченных пар проблем (x, y) таких, что x проблемнее y. Таким образом, интерпретация осуществляет связь между языком ИИ и описываемой им предметной областью (проблемой ИИ) реального мира. Она позволяет придать ППФ содержательный смысл.

При заданной интерпретации всякая ППФ (не содержащая свободных переменных) представляет собой высказывание, которое истинно или ложно. Если при данной интерпретации каждая из ППФ A_i, $i = 1, ..., n$, имеет значение И, то будем говорить, что данная интерпретация удовлетворяет системе ППФ $\{A_i\}^n_{i=1}$. ППФ A выводима (логически следует) из некоторой системы ППФ $\{A_i\}^n_{i=1}$, если каждая интерпретация, удовлетворяющая $\{A_i\}^n_{i=1}$, удовлетворяет также и A. Так, очевидно, что ППФ $\forall xF(x)$ выводима из системы ППФ $\{\forall x\neg R(x) \vee F(x), . yxR(x)\}$.

Согласно теореме Гёделя, если некоторая интерпретация удовлетворяет заданной системе ППФ $\{A_i\}^n_{i=1}$, то она удовлетворяет и любой ППФ A, выводимой из этой системы. Умение

продемонстрировать, что, ППФ A выводима (логически следует) из системы ППФ $\{A_i\}^n_{i=1}$, когда это на самом деле так, играет важную роль при логическом анализе, и мы сосредоточим на нем свое внимание. Предположим, что A выводима из $\{A_i\}^n_{i=1,}$ Тогда любая интерпретация, удовлетворяющая $\{A_i\}^n_{i=1}$, удовлетворяет A, но не удовлетворяет $\neg A$. Следовательно, никакая интерпретация не удовлетворяет объединению $\{A_i\}^n_{i=1} \vee \neg A$. Если некоторая система ППФ не удовлетворяется ни при какой интерпретации, то она называется неудовлетворимой. Так, если ППФ A выводима из $\{A_i\}^n_{i=1}$, то объединение $\{A_i\}^n_{i=1} \vee \neg A$ неудовлетворимо. И наоборот, если $\{A_i\}^n_{i=1} \vee \neg A$ неудовлетворимо, то ППФ A должна логически следовать из системы ППФ $\{A_i\}^n_{i=1.}$ Именно эта концепция выводимости лежит в основе понятия логического вывода, или дедукции, в исчислении предикатов.

Универсальным методом логического вывода является так называемый *метод резолюций*, предложенный в 1965 г. Дж. Робинсоном. Этот метод замечателен тем, что он сложный процесс логического вывода сводит к последовательности очень простых операций, каждая из которых может быть легко запрограммирована.

В основе метода резолюций лежат три простых правила вывода *(резольвенции):*

1) если истинны ППФ A и $\neg A \vee B,$ то истинна ППФ B (правило *modus ponens);*

2) если истинна ППФ $A \vee A,$ то истинна ППФ A (правило факторизации);

3) если истинна ППФ $A(x),$ то истинна ППФ $\forall yA(y).$

Эти правила применяются к простым дизъюнктам,
на которые предварительно «раскладывается» система ППФ $\{A_i\}^n_{i=1} \vee \neg A$, из неудовлетворимости которой следует, что A выводима из $\{A_i\}^n_{i=1}$. Новые дизъюнкты, получаемые в результате применения указанных правил, называются *резольвентами*. При образовании резольвент существенную роль играет процедура *унификации,* которая для двух данных предикатов осуществляет подстановку термов вместо переменных, делающую предикаты одинаковыми. После этого к полученным ППФ применяются правила резольвенции. Например, для неудовлетворимой системы ППФ вида $\{A(x) \vee B(x), \neg B(f(z)), \neg A(f(z))\}$, используя первое правило вывода после подстановки терма $f(z)$ вместо переменной $x,$ получим из первых двух ППФ резольвенту A $(f(z))$, которая в сочетании с третьей ППФ системы дает нулевую формулу. Таким образом, если выбрано два простых дизъюнкта и по одной литере в каждом из них, то применение правила унификации и затем правил вывода дает резольвенту. При доказательстве выводимости

ППФ *A,* рассматриваемой как заключение (теорема), из заданной системы ППФ $\{A_i\}^n_{i=1}$, рассматриваемых как посылки (аксиомы), процесс образования резольвент (в котором могут принимать участие и ранее полученные резольвенты) продолжается, пока не будет получена пустая формула, означающая неудовлетворимость системы $\{A_i\}^n_{i=1} \vee \neg A$ и успех доказательства. Важно отметить, что число резольвент, формируемых при доказательстве любой теоремы из заданной конечной системы аксиом, конечно.

В ряде задач, которые должны решаться с использованием ИИ, простое доказательство выводимости ППФ *A,* формулирующей задание, из системы ППФ $\{\{A_i\}^n_{i=1}$, описывающих условия выполнения этого задания, оказывается недостаточным. Примером такой задачи является планирование поведения интеллектуального объекта. В подобного рода задачах нужно знать то значение переменной *x,* при котором данная ППФ *A(x)* логически выводима из некоторой системы ППФ $\{A_i\}^n_{i=1}$. Иными словами, мы (вместе с интеллектуальным объектом) хотели бы знать, следует ли логически ППФ $\exists x A(x)$, и если да, то каково то значение *x,* при котором существует решение. Заметим, что умение отыскивать такие значения для переменной, связанной квантором существования, позволяет ставить интеллектуальному объекту вопросы весьма общего характера и осуществлять диалог с ним. Например, мы могли бы спросить у интеллектуального объекта: «Какие действия и в какой последовательности нужно совершать, чтобы собрать из деталей определенную конструкцию?». Ответом на этот вопрос будет не просто констатация факта, что сборка данной конструкции возможна, а развернутый план рекомендаций (технологический маршрут) сборки.

Рассмотрим на простейшем примере, как можно рекомендовать решать подобного рода задачи. Пусть интеллектуальному объекту известно, что его манипулятор жестко закреплен на подвижной платформе, а платформа находится в цехе. Рекомендуется спросить: «где находится манипулятор?». В этой задаче сформулированы два «факта», которые можно записать в виде двух правильно построенных формул:

$A_1 \leftrightarrow \forall x P$ (платформа, *x)*$\rightarrow P$(манипулятор, *x), *$A_2 \leftrightarrow P$(платформа, цех), где двухместному предикату *P(y, z)* придана очевидная интерпретация: *«y* находится в *z*». На вопрос «где находится манипулятор?» интеллектуальный объект может дать ответ, если сначала докажет, что правильно построенная формула

$A_1 \leftrightarrow \forall x P$ (манипулятор, *x)* выводима из системы ППФ $\{A_i\}^n_{i=1}$, и затем найдет то значение *x* (константу), которое на самом деле «существует» и служит ответом.

Используя описанный выше рекомендованый метод резолюций, интеллектуальный объект сначала попытается доказать неудовлетворимость системы $\{A_i\}^n_{i=1} \vee \neg P$ (манипулятор, *x)*. (Заметим, что отрицание ППФ *A* есть ППФ $\neg P$ (манипулятор, *x))*. Процесс доказательства неудовлетворимости $\{A_i\}^n_{i=1} \vee \neg A$ представлен на дереве вывода, изображенном на рис.1.

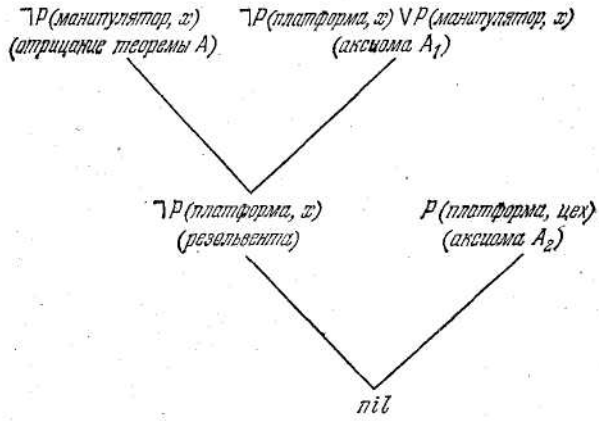


Рис.1. Дерево вывода.

Из этого дерева вывода можно извлечь также ответ на наш вопрос: «где находится манипулятор?». Это осуществляется следующим образом. Сначала к отрицанию теоремы добавляется ее отрицание, т. е. сама теорема. В результате получается тавтология (т.е. ППФ, тождественно истинная при всех интерпретациях) вида

$\neg P$ (манипулятор, *x)* $\vee P$ (манипулятор, *x).*

Затем в соответствии со структурой дерева вывода, изображенного на рис.1, вновь формируются резольвенты до тех пор, пока в корне дерева не получится некоторая ППФ, играющая роль ответа на языке ИИ. В нашем примере получим одну резольвенту

$\neg P$ (платформа, *x)* $\vee$ P (манипулятор, *x),*

а в корне дерева — ППФ *P* (манипулятор, цех), в которой содержится ответ на вопрос «где находится манипулятор?». Заметим, что форма ответа на языке предикатов близка к форме теоремы-вопроса. В нашем случае единственное отличие состоит в том, что в теореме-вопросе содержится переменная, связанная квантором существования, а в ответной ППФ — константа (ответный терм).

Таким образом, описанная система логического вывода, основанная на методе резолюций, представляет собой эффективное средство для

автоматического поиска доказательств (отыскания логических следствий) и извлечения ответа в терминах исчисления предикатов. Мы рассмотрели основные понятия этого исчисления и связанного с ним метода резолюций не ради них самих, а чтобы понять и продемонстрировать, как интеллектуальный объект, используя этот язык, может логически рассуждать, обучаться новому и адаптироваться в процессе решения интеллектуальных задач в системе ИИ.

5.2. Исчисление высказываний

Развитие методов построения логических схем позволило создать методы построения систем ИИ. Рассмотрим аппарат математической логики, открывший широкие возможности для построения схем объектов ИИ.

Логика является наукой о формах и законах мышления. Одна из отраслей логики, развивающаяся применительно к потребностям математики, называется математической логикой, а одним из ее разделов является алгебра логики или Булева алгебра.

Начало разработке логического исчисления, позволяющего оперировать логическими суждениями так же, как алгебраическими символами в элементарной математике, положил Буль.

Исчисление высказываний — это первое и наиболее широко применяемое в математической логике понятие.

Высказывание - это всякое предложение, которое может быть либо истинным, либо ложным и не может иметь никакого третьего значения. Отдельные простые высказывания могут быть связаны при помощи различных логических связей в сложные, которые также могут быть либо истинными, либо ложными. Используем двоичный код: истинное высказывание обозначим единицей (1), ложное — нулем (0). Благодаря этому на базе нескольких простейших электронных или иных элементов представляется возможным синтезировать любые сложные логические высказывания.

Рассмотрим основные связи между простыми высказываниями. Предложение «Коля пойдет гулять, если будет тепло и сухо» может быть разбито на два простых высказывания или предложения, связанные союзом *И:* «Коля пойдет гулять, если будет тепло» и «Коля пойдет гулять, если будет сухо».

Здесь возможны четыре варианта: не будет тепло (0) и не будет сухо (0), т. е. оба составляющие высказывания оказываются ложными. Очевидно, что в этом случае Коля гулять не пойдет, и все предложение в целом оказывается ложным. Во-вторых, возможно, что будет тепло (1), но не будет сухо (0), т. е. первое высказывание

оказывается истинным, а второе, ложным. И в этом случае Коля не пойдет гулять и сложное предложение окажется ложным. В третьем варианте будет холодно (не тепло), но будет сухо (1); здесь первое высказывание ложно, а второе истинно и сложное высказывание опять-таки ложно.

Сложное высказывание, объединяющее союзом *И* два и более простых высказываний, будет истинным только тогда, когда истинны все составляющие его высказывания, и будет ложным во всех остальных случаях. В нашем примере все предложение будет истинным, когда будет тепло и сухо. Это и является четвертым возможным вариантом, когда Коля действительно пойдет гулять.

Обозначим сложное высказывание — возможность прогулки Коли — через *P,* а составляющие его простые высказывания «Коля пойдет гулять, если будет тепло» — *a* и «Коля пойдет гулять, если будет сухо» — *b.* Тогда сложное высказывание можно записать в виде формулы

$$P = ab,$$

которая обычно называется логическим умножением, или логической функцией *И.*

Кроме союза *И,* для объединения простых высказываний употребляется также союз *ИЛИ.* Например, предложение «Вечером Петя пьет чай или ужинает» будет истинным в трех случаях: когда Петя пьет чай, но не ужинает; когда Петя ужинает, но не пьет чай, и, наконец, когда он и ужинает, и пьет чай. Следовательно, сложное предложение, объединяющее союзом *ИЛИ* два или более простых высказывания, будет ложным только тогда, когда ложны все составляющие его высказывания, и истинным во всех остальных случаях.

Эта логическая связь — логическое сложение— может быть записана в виде формулы

$$P = a + b.$$

Третья логическая функция носит название *НЕ* — отрицание. Например, мы имеем простое высказывание «Дождь идет». Образуем из него сложное при помощи отрицания — «Дождь *НЕ* идет». Если простое высказывание истинно, т. е. действительно идет дождь, то сложное высказывание будет ложным. Если же простое высказывание ложно, т. е. дождь не идет, то выражение «Дождь *НЕ* идет» будет истиной. Эта функция имеет вид

$$P = \overline{A}$$

и читается как «*P* равно не *A*».

Основные законы алгебры логики

Алгебра логики подчиняется законам, иногда совпадающим с законами обычной алгебры, а иногда — своим своеобразным законам. Рассмотрим основные законы алгебры логики.

Законы множеств:

$$0 \cdot a = 0;$$
$$0 + a = a;$$
$$0 \cdot abc \ldots w = 0,$$

т. е. произведение любого числа переменных обращается в нуль, если какая-либо одна переменная имеет значение 0, независимо от значений других переменных;

$$1 \cdot a = a;$$
$$1 + a = 1;$$
$$1 + a + b + c + d + \ldots + w = 1,$$

т. е. сумма любого числа переменных обращается в единицу, если одна из ее переменных имеет значение 1, независимо от значений других переменных.

Законы перемещения:

$$ab = ba;$$
$$a + b = b + a,$$

т. е. результаты выполнения операций умножения и сложения не зависят от того, в каком порядке следуют переменные.

Законы тавтологии (повторения):

$$aa = a,$$
$$aaa \ldots = a^n = a,$$
$$a + a = a,$$
$$a + a + a + \ldots + a = na = a.$$

Здесь можно сказать, что истина или ложь всегда остается истиной (или ложью), сколько ее не повторяй.

Законы дополнительности:

а) логическое противоречие:

$$a\bar{a} = 0$$

т. е. произведение любой переменной и ее инверсии есть 0. Как пример можно привести строку из известной песни «Речка движется и не движется» — заведомая ложь,

б) закон исключенного третьего:

$$a + \bar{a} = 1,$$

т. е. сумма любой переменной и ее инверсии есть 1. Так, например, утверждая, что «студент сдаст экзамен или не сдаст», мы всегда будем правы.

Законы инверсии (Де Моргана):

$$\overline{ab} = \overline{a} + \overline{b},$$

т. е. инверсия произведения равна сумме инверсий;

$$\overline{a+b} = \overline{ab},$$

а инверсия суммы есть произведение инверсий.

Здесь записаны законы для двух переменных, но они справедливы для любого числа переменных.

Законы распределительные (дистрибутивные)}

а) произведения относительно суммы:

$$a(b + c) = ab + ac.$$

Справедливость этого закона можно подтвердить высказываниями. Например: «Я зайду за Вами *И* мы пойдем в театр *ИЛИ* в кино». Так можно формулировать левую часть приведенного выше выражения, а правая часть тогда может быть прочитана так: «Я зайду за Вами *И* мы пойдем в театр *ИЛИ* я зайду за Вами *И* мы пойдем в кино». Смысл один и тот же, но правая часть несколько длиннее:

б) суммы относительно произведения:

$$a + bc = (a + b)(a + c).$$

Справедливость этого закона можно доказать, опираясь на предыдущие. Раскрыв скобки, получим

$$a + bc = aa + ac + ba + be = a + ac + ba + bc.$$

Из первых двух членов вынесем за скобки переменную *a:*

$$a(1+c),$$

но

$$1 + c = 1, \text{ а } a \cdot 1 = a.$$

Рассматривая следующее выражение $ba + bc$, мы устанавливаем, что и оно равно *a.* Тогда вся правая часть превращается в $a + bc$, т. е. в такое же выражение, как и в левой части.

Законы склеивания:

$$ab + a\overline{b} = a;$$

$$(a + b)(a + \overline{b}) = a.$$

Эти законы легко подтверждаются на основании рассмотренных ранее законов, например:

$$ab + a\overline{b} = a(b + \overline{b}) — a \bullet 1 = a.$$

Законы поглощения:

$$a(a + b) = a;$$
$$a(a + b)(a + c) \ldots (a + w) = a;$$
$$a + ab = a;$$
$$a + ab + ac + .., + aw = a;$$
$$a(\overline{a} + b) = ab;$$

$$a + \overline{a}\, b = a + b.$$

Эти законы можно легко доказать с помощью других законов алгебры логики, например, умножая в последнем выражении первый член a на $(1 + b)$, получаем

$$a(1 + b) + \overline{a}\, b = a + ab + \overline{a}\, b = a + b\,(a + \overline{a}\,) = a + b \bullet 1 = a + b.$$

Так как для логического сложения и умножения характерны все свойства сложения и умножения алгебры чисел, то над многочленами алгебры высказываний можно производить те же действия, что и над многочленами алгебры чисел. Но логическое сложение и умножение обладают и некоторыми необычными свойствами и это приводит к необычности действий над логическими многочленами. Разъясним это на примере.

Пример. Пусть необходимо умножить $(a + b)$ на $(a+c)$. Умножаем по обычным правилам умножения многочлена на многочлен:

$$(a + b)\,(a + c) = aa + ac + ab + bc.$$

Так как в алгебре высказываний $aa = a$, то

$$(a + b)\,(a + c) = a + ac + ab + bc.$$

Но работу над полученным произведением можно продолжить. Рассмотрим два первых слагаемых a и ac. Сгруппируем их и общий множитель a вынесем за скобки:

$$a + ac = a\,(1 + c),$$

и далее

$$a \cdot 1 = a.$$

Также поступим и с суммой $a + ab = a$. Тогда окончательно

$$(a + b)\,(a + c) = a + bc.$$

Результат оказался проще, чем мы ожидали, так как выражение $a + ab$, согласно закону поглощения, заменили множителем a.

Таким образом, если высказывание логически складывают с логическим произведением, в состав которого оно входит, то оно поглощает это произведение. Отметим еще одну важную особенность алгебры высказываний. Если в формуле $a + ab$ заменить знак + на знак × и знак × на знак +, то полученное новое высказывание $a(a + b)$ будет эквивалентно заданному. В этом легко убедиться, раскрыв скобки.

Это свойство распространяется на любую формулу алгебры высказываний.

Например,

$$a(b + c) = ab + ac.$$

К обеим частям применим упомянутую замену знаков и получим:

$$a + bc = (a + b)(a + c).$$

Еще пример. Дано $ab + a\overline{b}$. В левой части заменим знаки:

$$ab + a\overline{b} = (a + b)(a + \overline{b}\,) = a$$

или $(a + b)(a + \overline{b}) = a.$

Эту особенность преобразования формул в алгебре высказываний называют законом двойственности. Опираясь на закон двойственности, легко преобразовать эквивалентные высказывания.

Упрощение логических выражений

Следует иметь в виду, что каждое логическое высказывание можно воплотить с помощью логических элементов в конкретный действующий автоматический механизм. Для этого каждое логическое сложение, т. е. знак плюс в формуле высказывания, следует осуществить логическим элементом *ИЛИ,* каждое логическое умножение —- элементом *И,* а каждое отрицание или инверсию — элементом *НЕ* (рис.2).

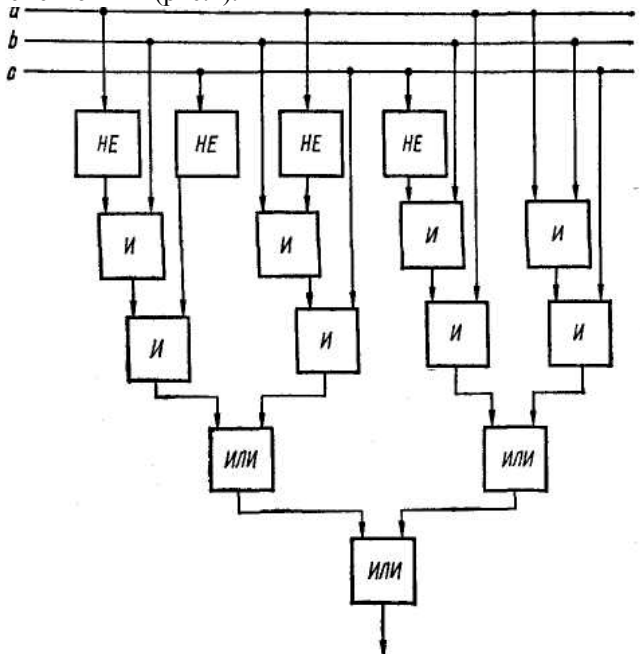


Рис. 2. Логическая схема.

Рассмотрим выражение

$$x = abc + \overline{a}\,bc + ab\,\overline{c} + \overline{a}\,b\,\overline{c}.$$

Заметим, что первое и третье, второе и четвертое слагаемые склеиваются по букве *a.*

Действительно,

$$abc + \bar{a}\,bc = bc$$

и

$$ab\bar{c} + \bar{a}\,b\,\bar{c} = b\,\bar{c}.$$

Имеем

$$x = bc + b\,\bar{c}.$$

В этой формуле можно произвести склеивание по букве *c* и тогда окончательно записанное ранее сложное выражение принимает очень простой вид: $x = b$.

Значит, математическая обработка выражения, построенного по законам математической логики, устранила необходимость в 15 логических элементах.

Мы произвели упрощение логического выражения, используя основные законы математической логики, путем последовательных рассуждений. Пример, который мы рассмотрели, относительно прост, да и то разные люди могли бы его решать по-разному и получать разные результаты. Тем более, такое явление может иметь место при упрощении более сложных логических выражений, в которых участвует большое число переменных, и которые выражаются более сложными зависимостями.

Задача упрощения логических выражений, или, как говорят, их «минимизация», является одной из наиболее сложных в алгебре логики. Есть много различных способов минимизации.

Рассмотрим один из распространенных способов минимизации логических выражений с помощью карт Карно.

Карты Карно (Karnaugh) наглядно изображают логические функции. Карта Карно (рис. 3) разделена на квадратики, и каждому из них отвечает определенная комбинация значений всех входных переменных. Кроме того, каждая сторона квадратика представляет собой границу между значениями переменных (верхний и нижний, равно как и боковые квадратики карты, являются соседними).

Обозначения входных переменных указываются сверху и сбоку карты и относятся ко всему столбику или строке квадратиков, причем значения этих входных переменных в них принимаются равными единице. В соседних с обозначенными в столбцах или строках входные переменные соответственно равны нулю. В квадратиках записывается значение самой функции при данных комбинациях значений входных переменных. Значения входных переменных не принято записывать в квадратиках, они подразумеваются, поэтому на карте остается только значение функции (рис. 3, *в, г, д, е).* Из примеров, приведенных на рисунке для двух, трех и четырех

переменных, видно, что прибавление каждой новой переменной удваивает карту.

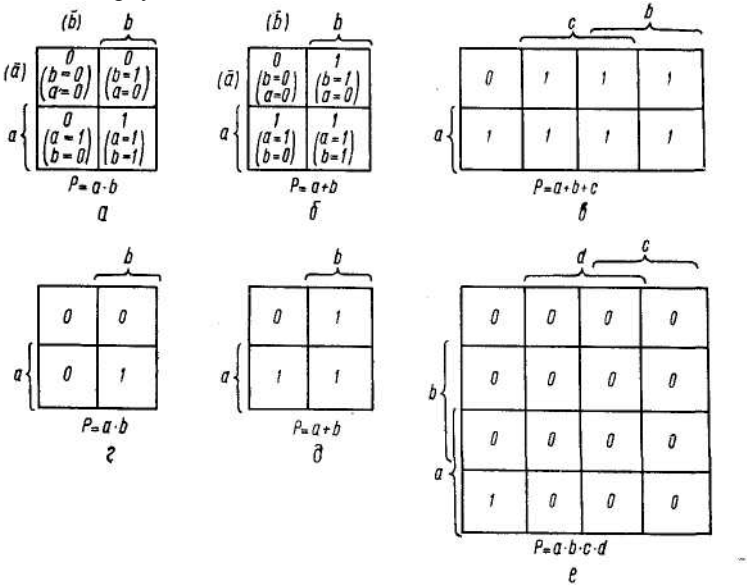


Рис. 3. Карты Карно.

Применяя карты Карно для изображения алгебраического выражения функций, можно записать функции либо в виде суммы произведений, либо в виде произведения сумм.

Выражение суммы произведений определяется суммой произведений значений всех входных переменных (прямых и инверсных) в каждом из квадратиков карты, содержащих единицу. Так, например, для карты, показанной на рис. 3, *а, г, P=ab,* а для карты, показанной на рис. 3, *б, д, —*$P = ab + ab + \bar{a}\,b$.

Выражения сомножителей в произведении сумм определяются суммами инверсных значений входных переменных в каждом из нулевых квадратиков. Так, например, для карты, представленной на рис. 3, *а, в,*

$$P=(a + b)(a + \bar{b}\,)(\,\bar{a}\, + b);$$

для карты, показанной на рис.4, *б, д,*

$$P = a + b.$$

С помощью карт Карно можно получить упрощенное выражение функций, для чего определяют суммы произведений и произведения сумм, объединяя квадратики, в которых значения функции

соответственно равны 1 или 0, в контуры. Последние должны иметь форму прямоугольников и содержать четное число квадратиков или только один квадратик.

Из свойств карт Карно вытекает, что при переходе контура из одного квадратика к другому одна из переменных инвертируется. Поэтому выражение контура из двух квадратиков не зависит от этой переменной, а определяется только остальными переменными, т. е. выражения, соответствующие контурам, «не содержат тех переменных, чьи границы пересекаются данным контуром». Так, контур, ограничивающий четыре квадратика, пересекает две границы двух переменных и поэтому соответствующее ему выражение содержит n—2 переменных и т. д.

Для получения наиболее простых выражений, реализуемых минимальным количеством возможно более простых логических выражений, т. е. при минимизации, логическое выражение должно иметь как можно меньше членов, каждый из которых должен содержать как можно меньше переменных.

Правила минимизации выражения логической функции по карте Карно сводятся к следующему. Чем большее число квадратиков с одним значением функции объединяется в общем контуре на карте и чем меньше будет таких контуров, тем проще будет аналитическое выражение функции. При этом все квадратики с одним значением функции должны входить в какой-нибудь контур. Нужно также следить за тем, чтобы какой-либо контур не входил полностью в другие контуры.

Рассмотрим подробнее карту Карно для трех переменных (рис.4).

Для наглядности в квадратиках указаны значения переменных. Пусть функция должна быть заложена в карту Карно и минимизирована.

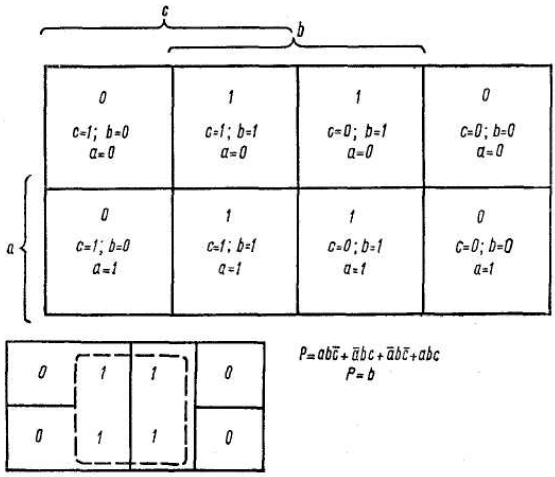


Рис.4. Карта Карно для трех переменных.

Подставив значения переменных, соответствующие левому верхнему квадратику, в выражение функции, получим

$X = abc + ab\bar{c} + \bar{a}bc + \bar{a}\bar{b}\bar{c} = 001 + 000 + 101 + 100 = 0 + 0 + 0 + 0 = 0.$

Это значит, что значение сложного выражения в этом квадратике равно 0, что и записываем.

Определяя таким образом значение выражения для всех квадратиков, получаем карту, показанную на рис.4, *б*.

Охватываем контуром средние четыре квадратика со значениями 1. В этом контуре только лишь переменная сохраняет свое значение, равное единице во всех квадратиках. Следовательно, результат минимизации определяется выражением

$$X = b.$$

Такое же значение мы получили ранее более сложным приемом минимизации — аналитическим путем с использованием основных законов алгебры логики,

Разница в затратах труда на минимизацию становится тем значительнее, чем больше переменных в логическом выражении и чем оно сложнее.

Схемы систем ИИ основаны на так называемых двухпозиционных приборах, т. е. устройствах, способных занимать только одно из двух устойчивых положений. Сигнал, поступающий на вход в систему или снимаемый с выхода системы, может либо присутствовать (1), либо отсутствовать (0).

Поэтому в дальнейшем под переменными будем понимать сигналы на входе в схему, а под сложными высказываниями — сигналы на выходе, являющиеся логическими функциями этих переменных. Задача логической части схемы — выработать сигналы на выходе, являющиеся логическими функциями сигналов на входах.

Для решения задачи составления схемы необходимы электрические, электронные или пневматические устройства, осуществляющие элементарные логические связи *И, ИЛИ, НЕ.* Такие устройства называются элементами. Допустим, что наличию сигнала соответствует 1, т. е, истина, а отсутствию сигнала — 0, т. е. ложь. В электрических системах истине соответствует подача тока, а отсутствие тока — лжи. В пневматических системах наличие сигнала означает подачу сжатого воздуха под давлением, а отсутствие сигнала - соединение с атмосферой.

Логический элемент типа *И* должен иметь два или больше входа и один выход, с которого сигнал снимается.

В электрическом устройстве, в котором два реле *a* и *b* включены последовательно (рис.5), сигнал на выходе появится при подаче напряжения на катушки обоих реле. В этом случае элемент выполнит логическую операцию умножения $P = ab$.

В пневматическом устройстве сигналы, т. е. сжатый воздух, поступают от двух пневматических кнопок *a* и *b*. Если нажать на одну из кнопок и подать воздух под давлением на один из входов, то две тарели клапана, сидящие на одной оси, передвинутся в одну сторону и свяжут выход клапана с атмосферой через вторую кнопку. Если нажать обе кнопки, то независимо от того, какое положение займут тарели, сжатый воздух пройдет на выход.

Элемент *ИЛИ* также имеет два входа и один выход. Если два реле *a* и *b* соединены параллельно, то сигнал на выходе появится, если подать напряжение на любое реле, и тогда элемент выполняет операцию логического сложения $P=a + b$.

В пневматическом варианте, если обе кнопки отпущены, выход элемента связан с атмосферой по крайней мере через одну из кнопок. Нажмем, например, кнопку *a*. Под действием давления сжатого воздуха шарик переместится вправо, прижмется к резиновому кольцу и не даст воздуху выходить в атмосферу через кнопку *b,* вследствие чего воздух поступит на выход клапана—элемента. Если отпустим кнопку *a* и нажмем кнопку *b,* то шарик переместится влево, не давая воздуху выходить в атмосферу через кнопку *a,* и на выходе тоже появится сигнал. Нажмем обе кнопки вместе, и, в каком бы ни был положении шарик, появится сигнал на выходе.

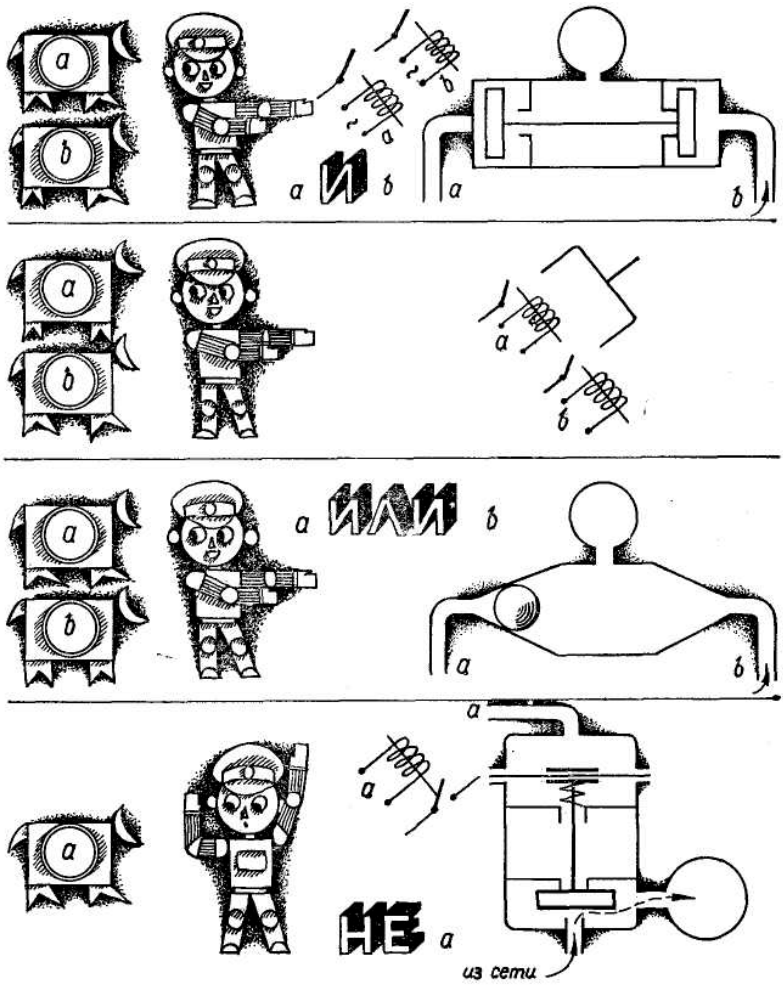


Рис.5. Логические элементы И, ИЛИ, НЕ.

Логическим элементом *НЕ* в релейном варианте является переключатель. Когда напряжение в катушке *а* отсутствует, на выходе *Р* протекает ток, т. е. имеется сигнал на выходе. В пневматическом варианте этот элемент выглядит несколько сложнее; оно носит название пневматического реле. Полость над мембраной *1,* с которой соединяется пневматическая кнопка *а*, является полостью управления реле: она же представляет собой вход. Если кнопка *а* отпущена, т. е.

сигнал на входе отсутствует, то тарель *6* с толкателем *3* под действием пружины *2* находится в положении, показанном на рисунке. Резиновая прокладка *5* прижимается к верхнему седлу корпуса и закрывает проход к отверстию *4,* ведущему в атмосферу. Сжатый воздух, подводимый к нижнему отверстию 7, свободно проходит на выход реле *8.* При отсутствии сигнала на входе имеется сигнал на выходе.
Теперь нажмем кнопку *а.* Сжатый воздух начнет поступать в полость управления реле. Под действием силы давления воздуха мембрана прогибается, и тарель с толкателем движутся вниз, сжимая пружину. Резиновая прокладка отходит от верхнего седла, освобождая проход к отверстию *4,* а другая прокладка *6* перекрывает нижнее отверстие 7. Выход реле *8* оказывается связанным с атмосферой через отверстие *4.* Имеется сигнал на входе — нет сигнала на выходе.

Логические элементы, из которых складываются логические системы, чаще всего основываются на применении электронных, пневматических, пневмонических схем. В обычных пневматических устройствах процессы совершаются во много раз медленнее, чем в электронных, и поэтому логические машины на пневматике работают значительно медленнее. Однако применение пневмоники разрешает создавать устройства на сжатом воздухе, способные выполнять до двух тысяч и более операций в секунду. Вместе с тем при решении целого ряда задач автоматизации производства и, очень часто, в машиностроении, большая скорость выполнения отдельных операций вообще и не требуется. Зато пневматические устройства имеют ряд качеств, выгодно отличающих их от электронных. Они по самой своей природе взрывобезопасны, просты и надежны. Для их обслуживания и ремонта не требуется высокой квалификации.

Пример использования языка исчисления высказывний

При решении задач на основе языка исчисления высказывний, в СИИ необходимо последовательно выполнить ряд следующих приемов:

1) составить подробные требования к условию функционирования интеллектуального объекта;

2) установить число входов и выходов, что требует подробного рассмотрения функционирования интеллектуального объекта;

3) составить таблицу функционирования интеллектуального объекта по форме;

4) на основании таблицы составить структурную формулу интеллектуального объекта;

5) осуществить минимизирование логической функции, т. е. структурной формулы интеллектуального объекта;

6) составить функциональную схему интеллектуального объекта по минимизированной логической функции.

Когда функциональная схема составлена, можно считать, что задача по формированию схемы функционирования интеллектуального объекта будет решена.

Рассмотрим пример. На одном заводе имеются три цеха *A, B* и *C.* Электроэнергией их обеспечивает небольшая электрическая станция, на которой установлено два электрогенератора *X* и *Y.* Мощность генератора *X* в два раза выше, чем генератора *Y.*

Если в энергии нуждается один из цехов, то достаточно включить генератор *Y,* если же любые два цеха — генератор *X.* Снабжение электроэнергией всех трех цехов сразу обеспечивает одновременная работа двух генераторов. На электрической станции дежурный следит за сигналами из цехов *A, B* и *C* и соответственно регулирует включение того или иного генератора.

Стоит вопрос, нельзя ли создать интеллектуальный объект , который заменил бы дежурного по заводской электрической станции и, получая сигналы от цгхов *A, B, C,* сам бы решал, какой из генераторов включать? Приведенное задание разработчики рассматривают как словесное задание интеллектуального объекта, которое содержит лишние высказывание о его работе.

Разработчие, ознакомившись с таким заданием, анализирует его с помощью следующих рассуждений: будущий интеллектуальный объект должен получать сигналы из трех цехов *A, B* и *C,* а это значит, что у него три входа. Сигналы, вырабатываемые интеллектуальным объектом, направляются в два адреса: на генератор *X* и на генератор *Y.* Значит, у него два выхода. Теперь можно составить таблицу работы интеллектуального объекта.

A	*B*	*C*	*X*	*Y*	*A*	*B*	*C*	*X*	*Y*
1	1	1	1	1	0	1	0	0	1
1	1	0	1	0	0	1	1	1	0
1	0	1	1	0	0	0	1	0	1
1	0	0	0	1	0	0	0	0	0

Если в энергии нуждаются три цеха, включены оба генератора, если два цеха — только генератор *X* или генератор *Y.*

Пользуясь составленной таблицей, следует составить структурную формулу интеллектуального объекта. Для этого следует брать те строки в таблице, в которых выход имеет значение, равное единице. В таблице таких строк четыре для выхода *X* и четыре для выхода *Y.* Составим формулу для выхода *X.*

На выходе *X* появится сигнал при поступлении сигналов от цехов *A, B, C* одновременно или от любых двух цехов одновременно — всего в четырех случаях. Теперь несложно составить структурную формулу, по которой должен действовать дежурный или заменяющий его интеллектуальный объект:

$$X = abc + ab\bar{c} + a\bar{b}c + \bar{a}bc. \qquad (C)$$

Аналогично составляют формулу интеллектуального объекта, вырабатывающего сигнал на выход Y; эта формула будет иметь вид

$$Y = abc + a\bar{b}\bar{c} + \bar{a}b\bar{c} + \overline{ab}c. \qquad (Д)$$

Теперь следует минимизировать полученные выражения. Составим карту Карно для формулы *(C)*. Используя правила минимизации, получаем новую формулу для интеллектуального объекта, управляющего генератором *X* (рис.6):

$$X = ab + ac + bc.$$

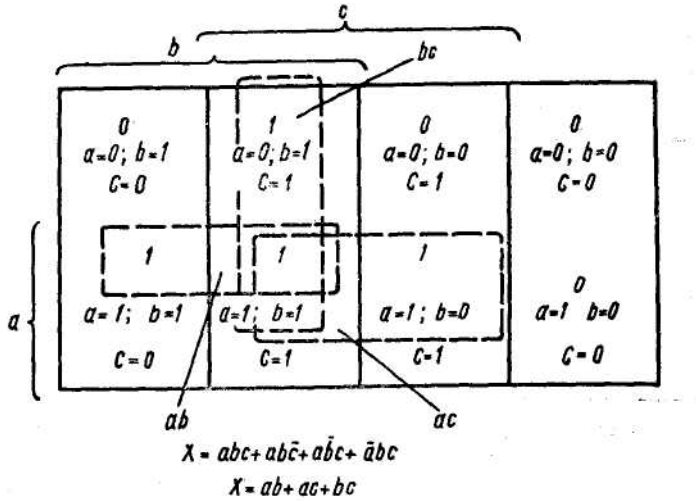


Рис. 6. Минимизация логического выражения для менее мощного генератора.

Наносим на карту Карно формулу *(Д)*. Так как единицы и нули на карте чередуются и нет ни одной пары смежных, которые можно было бы взять в контур, то выражение не поддается минимизации (рис.7).

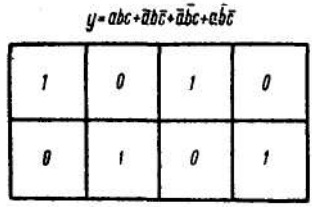


Рис.7. Карта Карно для более мощного генератора.

Структурная формула интеллектуальнго объекта, управляющего включением в работу генератора *Y*, остается прежней:

$$Y = abc + a\bar{b}\,\bar{c} + \bar{a}\,b\,\bar{c} + \overline{ab}\,c.$$

Функциональная схема интеллектуальнго объекта приведена на рис.8.

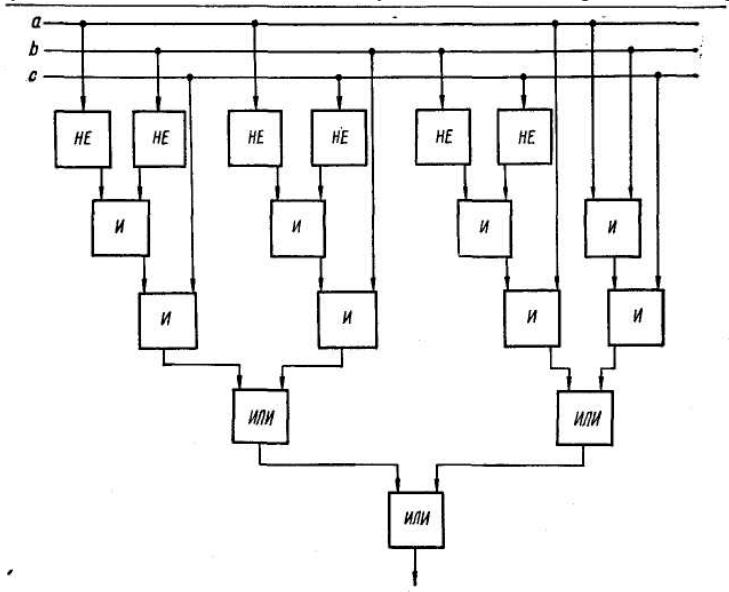


Рис.8. Функциональная схема интеллектуального объекта, управляющего снабжением электроэнергией трех цехов

5.3. Обучение и адаптация в системе логического вывода

Первым и неизбежным этапом применения описанной выше системы логического вывода для автоматизированного решения задач

ИИ, требующих логического анализа, является формулировка этих задач в терминах исчисления предикатов. Для этого нужно, прежде всего, задать предметную область, т. е. совокупность относящихся к решаемой задаче объектов (или процессов), и выделить их существенные свойства, от которых в наибольшей степени зависит успех решения. Далее нужно, присвоив определенный содержательный (семантический) смысл предикатным и функциональным символам, формализовать данные и условия задачи в виде ППФ, которые должны на них выполняться (т. е. истинность которых считается не требующей доказательства). Очевидно, что эти ППФ выделяют из всевозможных систем объектов, их свойств и отношений между ними такие системы, для которых они выполнены.

ППФ, посредством которых мы таким образом выделяем совокупность объектов, называются **аксиомами.** Если для какой-либо совокупности объектов, их свойств и отношений некоторые аксиомы истинны, то говорят, что данная совокупность объектов удовлетворяет системе этих аксиом или является интерпретацией данной системы аксиом.

Таким образом, аксиомы можно рассматривать как определения системы объектов, их свойств и отношений между ними. Делая логические выводы из аксиом, мы будем получать ППФ, истинные для любой системы объектов, удовлетворяющей данным аксиомам.

Ясно, что соответствие между аксиомами и предметами реальности, т. е. предметной областью, всегда имеет приближенный характер. Поэтому возникает вопрос, как узнать, действительно ли данная система аксиом определяет именно то, что было задумано, что требуется для решения задачи?

Ответ на этот вопрос связан с понятием *непротиворечивости системы аксиом.* Мы должны быть уверены, что делая всевозможные выводы из данной системы аксиом, не придем к противоречию, т. е. не выведем какие-либо несовместимые ППФ. Появление противоречия означало бы, что рассматриваемой системе аксиом не может удовлетворять никакая совокупность объектов, и, таким образом, эти аксиомы ничего не описывают. Мы будем говорить, что система аксиом $\{A_i\}^n_{i=1}$ противоречива, если в ней выводима какая-либо ППФ A, а также и ее отрицание $\neg A$. Для проверки (доказательства) непротиворечивости системы аксиом достаточно построить какую-нибудь точную интерпретацию этой системы.

Весьма важным является свойство *независимости аксиом.* Какая-либо аксиома A называется *независимой* в данной системе аксиом $\{A_i\}^n_{i=1}$, если она невыводима из остальных аксиом этой системы. Для проверки (доказательства) независимости какой-либо аксиомы достаточно найти совокупность объектов, удовлетворяющую всем

аксиомам, кроме исследуемой, и не удовлетворяющей этой последней. Иными словами, для проверки независимости аксиомы A достаточно найти интерпретацию следующей системы аксиом: $\{A_i\}^n_{i=1}$, $\neg A$.

Таким образом, система аксиом, которой пользуется интеллектуалный объект, должна иметь точную интерпретацию в том мире объектов, свойств и отношений, в котором он функционирует. Этому требованию можно удовлетворить путем травильной формулировки тех задач, которые интеллектуалный объект должен решать. Остановимся на этом вопросе подробнее.

Формулировка задачи на языке предикатов— это первый и наиболее ответственный этап организации его целенаправленного поведения. На этом этапе от разработчиеа требуются глубокие знания не только и не столько исчисления предикатов, сколько существа решаемой задачи, ее специфических черт, той цели, которая должна быть достигнута в результате решения. Возможна, но совершенно бессмысленна постановка на языке предикатов таких, например, задач: «рекомендую переместиться туда, сам не знаю куда», или « рекомендую найти то, сам не знаю что».

Практически весьма важно, чтобы формулировка задачи (связанная с заданием системы аксиом и теорем-заданий) была по возможности простой, не «засоренной» массой мелких, второстепенных факторов, так как учет их существенно осложняет логический анализ и делает трудно обозримыми результаты решения. Отметим две типичные трудности, которые всегда подстерегают «формулировщика» задачи. Первая — это возможность «утонуть в деталях и подробностях», т. е. «из-за деревьев не увидеть леса»; вторая — слишком огрубить задачу, или, как принято говорить в подобных случаях, «вместе с водой выплеснуть и ребенка». Ниже на примерах формулировки задач планирования поведения интеллектуалного объекта - робота и распознавания сложных ситуаций мы увидим, что искусство формулировки задач на языке предикатов есть именно искусство. Здесь нет общих рецептов, а опыт разработчиеа в этом трудном деле приобретается постепенно.

При построении аксиом будем различать два типа предикатов, использование которых по-разному сказывается на быстроте поиска решения. Предикаты первого типа описывают простейшие свойства конкретных объектов (например, «робот находится в точке *x»,* «объект *z* большой» и т. п.). Предикаты второго типа определяют общую картину отношений между различными объектами и их свойствами. Один такой предикат может описывать набор свойств большого числа объектов (например, «если между точками *a* и *b* нет препятствий, то робот может проехать между этими точками по прямой» и т. п.).

Количество подобных предикатов, необходимое для описания (с требуемой степенью подробности) данных и условий задачи, обычно невелико. Однако для сложных предикатов существенно возрастает сложность термов, участвующих в их определении.

Многие задачи ИИ часто связаны с изменением во времени свойств объектов, которые обычно известны в начальный момент времени. В таких задачах удобно ввести предикат позиции, определяющий все «интересные» свойства всех объектов. При этом аксиомы, описывающие изменение предиката позиции во времени, в наиболее простой форме могут быть составлены из трех литер, а именно, если имеется некоторая позиция (ситуация, проблема) и если выполняется некоторое дополнительное условие, характеризующее принципиальную возможность применения данной аксиомы, то получится новая позиция (ситуация, проблема). Основным преимуществом такого способа построения аксиом является то, что свойства, связанные между собой, определяются одним предикатом и поэтому меняются одновременно. При этом на каждом шаге поиска учитывается все многообразие сложившейся ситуации, вследствие чего уменьшается число резольвент в процессе логического вывода.

Из дальнейшего изложения (и, в частности, из примеров) будет ясно, что эффективность системы логического вывода можно увеличить путем уменьшения числа предикатов и аксиом, определяющих данные и условия задачи. С этой целью разумно использовать ранее доказанные теоремы или ввести более сложные предикаты, образующие новые аксиомы, которые можно рассматривать как результат обучения робота в процессе решения задач. Такие аксиомы, описывающие на языке исчисления предикатов приобретаемый роботом опыт, мы будем называть *аксиомами обучения.* Введение аксиом-обучения как бы моделирует феномен мышления, о котором еще Р. Декарт писал в своем «Рассуждении о методе»: «Каждая решенная мною задача становилась образцом, который служил впоследствии для решения других задач». Образно говоря, аксиомы обучения играют роль лемм при доказательстве новых теорем, определяющих целевые условия задачи. Тем самым они позволяют оперировать более крупными «блоками» (фрагментами) доказательств, освобождая от рассмотрения многочисленных деталей, имеющих в данном доказательстве лишь вспомогательное значение. Заметим, что введение аксиом обучения позволяет роботу увеличивать и улучшать знания о решаемом классе к задач в процессе их непосредственного решения. Таким образом, аксиомы обучения являются средством обучения новым понятиям и фактам и уточнения старых.

Скорее всего, в будущем интеллектуалным объектам так и не удастся прийти ни к какой определенной конечной системе аксиом, рассматриваемой как окончательная. Напротив, подобно тому, как это происходит в мире живого, будут появляться (автоматически формироваться) все новые аксиомы обучения, отображающие изменения в окружающем интеллектуалный объект мире и в решаемом им классе задач.

Построение системы аксиом в каждой задаче важно не само по себе, а имеет целью выявление оптимальных путей логического вывода. Под *эффективностью системы логического вывода* мы будем понимать меру успешности и поискарешения. Для того чтобы выбрать количественный показатель эффективности, нужно прежде всего спросить себя: чего мы хотим от системы логического вывода, к чему стремимся при поиске решения? Выбирая (формируя) решение, мы предпочитаем такое, которое при ее реализации обращает показатель эффективности в максимум или же в минимум.

Очень часто в качестве показателя эффективности систем логического вывода фигурируют затраты на поиск решения (доказательства), которые, естественно, нужно минимизировать. Заметим, что неправильный выбор показателя эффективности очень опасен, так как он может привести к плохим решениям и рекомендациям. Решения, выбранные под углом зрения неудачно выбранного показателя эффективности, могут привести к большим неоправданным потерям и затратам.

В рассматриваемом круге задач под *эффективностью* системы логического вывода будем понимать число шагов доказательства (возможно, усредненное по классу решаемых задач), т. е. число резольвент, формируемых в процессе поиска решения. С целью увеличения эффективности системы логического вывода введем некоторые ограничения на процесс образования резольвент, связанные с выбором стратегии поиска решения.

Стратегией логического вывода называется способ выбора очередной пары дизъюнктов и литер в них для образования резольвент. Именно стратегия определяет, в каком порядке будут образовываться резольвенты и, следовательно, насколько быстро будет найдено решение. Стратегия «запускает» процесс доказательства — *начинается дедукция*: с помощью аксиом, резольвент и теорем строится та или иная конструкция доказательства. При этом стратегия решает, какие понятия и факты (аксиомы и литеры в них) несущественны, а какие — необходимы для доказательства. Таким образом, выбор и подстройка стратегии являются основным средством увеличения эффективности системы логического вывода,

определяющим быстроту сходимости реализуемого ею метода поиска доказательства. Если правила резольвенции есть правила дедуктивного вывода следствий, то стратегия — это та активная часть, способная к обучению и адаптации, которая имитирует способ «мышления», например, искусственного интеллектуального робота, уровень его познаний в логике, степень его интеллектуальности.

Образно говоря, *стратегия системы логического вывода — это идея поиска решения (доказательства), исходя из заданной системы аксиом, в которой заключены вес необходимые для решения задачи знания*. Если идея (стратегия) хороша, то решение будет найдено быстро. Однако рассчитывать на хорошую идею (стратегию) мы можем лишь тогда, когда в системе аксиом достаточно полно отражены не только необходимые знания о задаче, но и прошлый опыт решения задач в подобных проблемах. Хорошие идеи (стратегии) имеют своим источником прошлый опыт и ранее приобретенные знания.

Стратегия называется *полной,* если она находит (в конечное число шагов) доказательство любой ППФ, выводимой из аксиом. Примерами полных стратегий являются стратегия опорного множества, стратегия предпочтения единичным элементом, а также тривиальная стратегия полного перебора. Все перечисленные стратегии характеризуются тем, что для них предикат является «неразложимым» понятием, а критерием выбора очередной пары дизъюнктов могут быть количество предикатов в дизъюнкте, порядок их расположения и т. п. Такие стратегии, не зависящие от внутренней структуры, смысла используемых предикатов, будем называть *синтаксическими*.

Стратегию будем называть *адаптивной,* если она целенаправленно меняется (подстраивается) в процессе логического вывода в зависимости от приобретаемого опыта. Примерами адаптивных стратегий могут служить *семантические* стратегии, в которых критерий выбора очередных дизъюнктов зависит от вхождения в них определенного терма. *Согласно семантической стратегии сначала выбираются термы, соответствующие «интересным» проблемам, затем — предикаты, описывающие их свойства, и, наконец, ППФ, содержащие эти свойства.*

Ранее мы отмечали, что процессу поиска доказательства может быть поставлено в соответствие дерево вывода, которое заканчивается пустым дизъюнктом, означающим конец и успех доказательства. Дерево вывода строится и ветвится под каждой резольвентой так же, как и под теоремой (точнее, ее отрицанием). Отсюда ясно, что резольвенты, не приводящие к пустому дизъюнкту, резко увеличивают число шагов доказательства, и их получение крайне

нежелательно. В связи с этим задача построения адаптивной стратегии может быть переформулирована как задача отсечения ненужных (лишних) ветвей на дереве вывода. Для решения этой задачи необходимо указать критерий предпочтения ветвей.

Действительно, в процессе доказательства теоремы можно указать, как правило, несколько подходящих аксиом и несколько путей (ветвей) доказательства. Если нет критерия предпочтения одной ветви другой, приходится действовать по методу случайного поиска, что соответствует образованию пучка ветвей на дереве вывода. Введение подходящего критерия предпочтения позволяет исключить лишние тупиковые ветви и благодаря этому существенно увеличить эффективность системы логического вывода. При этом особую роль играют критерии предпочтения, формируемые в процессе решения задач. Примером такого критерия является критерий предпочтения аксиом обучения (хранящихся в памяти наряду с исходной системой аксиом), которые позволяют уменьшить исходную неопределенность относительно условий решения задач.

Введение аксиом обучения, о которых шла речь выше, особенно эффективно в тех случаях, когда в них либо раскрывается неопределенность (т. е. содержится новая необходимая для решения информация), либо «запоминается» в компактной форме часто встречающийся в рассматриваемом классе задач «фрагмент» решения (доказательства). В самом деле, если в процессе решения очередной задачи потребуется доказать уже доказанную ране теорему, то критерий предпочтения аксиом обучения сократит общее число шагов доказательства по крайне мере на длину доказательства соответствующей аксиом обучения.

Важной особенностью адаптивной системы логического вывода является ее способность логически рассуждать, т. е. сводить сложное заключение к последовательности утверждений, истинность каждого из которые проверяется очень просто и чисто механически. Такая система может также не только автоматически доказывай теоремы, трактуемые как некоторые задания или вопросы, но и обучаться способам их доказательства.

Рассмотрим теперь применение адаптивной системы логического вывода для автоматического решения задач планирования поведения интеллектуалного объекта - робота, распознавания и описания сложных изображений трехмерной среды, получаемых с помощью информационно-измерительной системы робота в условиях неопределенности.

5.4. Логические алгоритмы планирования

Содержание задачи планирования поведения интеллектуалного объекта - робота поясним на примере того, как эту задачу решает человек. Первое, с чем мы сталкиваемся ежедневно, — это задача утреннего одевания. Мы должны выработать плана действий, который позволит нам одеться, причем так, чтобы выполнялись естественные общепринятые ограничения (рубашку надевать необходимо, но не поверх пиджака, и т. п.). При этом время — наш основной ресурс, и выбранный план должен быть наилучшим в том смысле, в каком каждый понимает расход своего утреннего времени.

Если отбросить некоторые «несущественные» детали, план одевания должен оперировать такими предметами, как туфли, носки, брюки, рубашка, галстук, пиджак и пальто. *Плана действий представляет собой любой порядок, в котором можно надеть эти предметы.* Всего в этом случае существует 7! = 5040 различных вариантов плана. Многие из них недопустимы, так как либо не удовлетворяют общепринятым ограничениям (рубашка поверх пиджака, носки поверх ботинок), либо непрактичны (галстук под рубашкой). Но даже после того, как эти недопустимые решения будут отброшены, все равно придется исследовать некоторое количество допустимых планов. Как же выбрать окончательный (желательно, оптимальный) план? Прежде всего заметим, что в рассматриваемой задаче имеется некоторая мера эффективности, некий критерий, позволяющий нам сравнивать эффективность допустимых планов. Если мы можем каким-то образом сравнить значение этого критерия для различных планов, то мы сможем тем самым выбрать из них оптимальный. В данной конкретной задаче естественно минимизировать время, необходимое для того, чтобы одеться. Это и есть та мера эффективности, с помощью которой можно сравнивать допустимые планы. Тогда в качестве оптимального плана, позволяющего одеться, не нарушая общепринятых ограничений, можно выбрать следующий план: носки, рубашка, брюки, галстук, туфли, пиджак, пальто. Ясно, что при другом критерии эффективности поведения оптимальным может оказаться иной план.

Характерной особенностью задач планирования является наличие многих допустимых решений. После того, как эти решения найдены, возникает следующая задача: выбрать среди них по крайней мере один оптимальный (в смысле определенного критерия) плана.

В рассмотренном нами простейшем примере решения принимались без специальных обоснований, просто на основе опыта и здравого смысла. Оптимизация таких решений происходит как бы сама собой, в процессе жизненной практики. Если порой выбранный план окажется

не самой удачной, так что же? На ошибках учатся. Нередки, правда, ситуации (соответствующие, например, планированию мероприятий, осуществляемых в первый раз), когда использовать **эвристические решения**, основанные на опыте и здравом смысле, просто невозможно. **В подобного рода ситуациях «опыт» молчит, а «здравый смысл» легко может обмануть, если не будет опираться на математический расчет.**

Но бывают решения несравненно более сложные, а главное ответственные — при их реализации от них очень многое зависит. Конечно, при планировании поведения в подобного рода ситуациях можно действовать интуитивно, опираясь опять-таки на опыт и здравый смысл. Но гораздо более разумными могут оказаться решения, подкрепленные количественными, математическими расчетами соответствующего плана действий. Эти предварительные расчеты помогут избежать длительного и накладного поиск решения «на ощупь».

«Семь раз примерь, один раз отрежь», — говорип известная поговорка. *Планирование поведения* как раз и представляет собой своеобразное математическое «примеривание» к потребному будущему, позволяющее заранее оценить последствия каждого решения, заранее отбросить недопустимые планы и рекомендовать наиболее удачные. Эти последние позволят установить, достаточна ли имеющаяся у нас информация для правильного выбора решения, и если нет— какую информацию нужно дополнительно получать и обрабатывать. Все это позволяет при реализации плана экономить время, энергию и материальные средства.

Необходимость в планировании поведения возникает у робота при выполнении им сложных заданий в условиях большой априорной неопределенности (например, сборка сложного изделия по чертежу, поиск и транспортировка нужного объекта на неизвестной местности с препятствиями и т. п.). Задача автоматизированого планирования поведения решаемая на втором уровне иерархии системы управления робота, может быть переформулирована на языке исчисления предикатов как задача логического вывода (автоматического доказательства теорем). При таком подходе априорные сведения о свойствах и функциональных возможностях робота и окружающей его среды необходимо прежде всего представить в виде правильно построенных формул (ППФ). Совокупность таких ППФ мы будем называть *априорными аксиомами* и разобьем их на четыре класса:

1) сенсорные аксиомы (СА);
2) моторные аксиомы (МА);
3) аксиомы среды (АС);

4) аксиомы начальных условий (АНУ).

Сенсорные аксиомы описывают функциональные возможности информационно-измерительной системы робота, а *моторные аксиомы* — функциональные возможности исполнительных механизмов робота. *Аксиомы среды* определяют состояние и эволюцию среды, а *аксиомы начальных условий* описывают начальные состояния робота и среды. В табл. 1 и 2 приведены типичные для задачи планирования поведения робота на местности с препятствиями: термы, функции, предикаты и априорные аксиомы вместе с их интерпретацией на обычном (русском) языке.

Таблица 1

Функции и предикаты	Интерпретация на естественном языке
$f(a, b, s)$	Ситуация, наступающая после выполнения роботом, находящимся в ситуации s, действия «переехать из a в b по прямой».
$p(s)$	Ситуация, наступающая после выполнения манипулятором, находящимся в ситуации s, действия «погрузить объект на тележку».
$q(s)$	Ситуация, наступающая после выполнения манипулятором, находящимся в ситуации s, действия «сгрузить объект с тележки».
$G(a, b)$	Истинен, если из a в b можно проехать по прямой.
$Pos(s, x, y)$	Истинен, если в ситуации s робот находится в состоянии x, а объект — в состоянии y (предикат позиции).
$Pos(s, x, fin)$	Истинен, если в ситуации s объект находится на складе, указанном в задании, а робот — в состоянии x.
$Pos(s, fin, y)$	Истинен, если в ситуации s робот находится в конечной точке заданного маршрута, а объект — в состоянии y.

Наряду с априорными аксиомами введем *аксиомы обучения* (АО), автоматически формируемые по мере накопления роботом опыта и

знаний в процессе выполнения тех или иных заданий. Задания роботу, формулируемые человеком, будем трактовать как заключения теорем, посылками которых служат априорные аксиомы и аксиомы обучения. Заметим, что выбор аксиом и теорем диктуется окружающим робота миром, который он воспринимает своими органами чувств и на который воздействует своими исполнительными механизмами, а также структурно-функциональными особенностями робота и целями (задачами) его функционирования.

Таблица 2

	Аксиомы	Смысл на естественном языке
Тип	Логическое представление	
МА	$\forall s \forall x \forall y \forall z\ (\neg Pos\ (s, x, y) \vee \neg G(x, z) \vee Pos\ (f(x, z, s), z, y))$	Если в ситуации s, робот находится в состоянии x, а объект — в состоянии y, и из x можно переехать по прямой в z, то в ситуации $f(x, z, s)$ (т. е. после реализиции функции движения «переехать из точки x в точку z») робот окажется в точке z.
АС	$G\ (a, b)$	Из точки a в точку b можно проехать по прямой.
СА	$K\ (x)$	Объект x принадлежит k-му классу.
АНУ	$Pos\ (s_0, O, C_1)$	В начальной ситуации s_0 робот находится в точке O, а объект — на складе C_1.

Для автоматического доказательства теорем-заданий и извлечения ответа на языке робота целесообразно применить адаптивную систему логического вывода, описанную выше. Такая система в результате доказательства теоремы-задания (или теоремы-вопроса) сформирует рекомендации по указанию, какие действия и в какой последовательности нужно роботу совершить для выполнения задания, т. е. выдаст рекомендации по реализации искомого план поведения робота.

Продемонстрируем работу адаптивной системы логического вывода в задаче планирования поведения робота на примере. Пусть робот находится в цехе с оборудованием (трактуемом как препятствия), где имеются склад заготовок C_1 и склад готовых изделий C_2 (см. рис. 9). Вначале робот находится в точке O и перед ним ставится задача: перевезти определенный объект (который еще нужно распознать) со склада C_1 на склад C_2 (местоположение складов известно) и после этого покинуть цех через выход. Предполагается, что выход задан набором признаков (его координаты роботу неизвестны), а сенсорная система может измерять значения признаков и координаты видимых ею точек, причем она не может «видеть» сквозь препятствия. Кроме перечисленного и исходной системы аксиом, представленной в табл. 2, роботу ничего неизвестно.

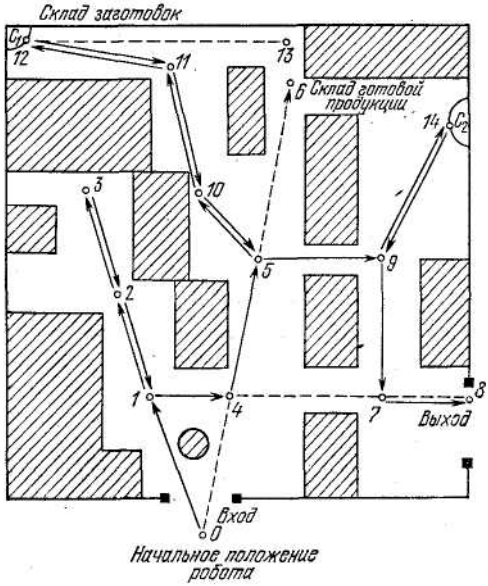


Рис. 9. Планирование поведения робота в незнакомом помещении.

Поскольку решение этой задачи требует дополнительной информации об обстановке в цехе, робот вначале опрашивает сенсорную систему. При этом отыскиваются все видимые границы препятствий и около них фиксируются некоторые точки *1-6,* к которым робот может проехать по прямой. В результате автоматически строятся аксиомы среды: *G(0, 1), G(1, 2), G(2, 3), G(0, 4), G(4, 5), G(5, 6).* По этим данным, а также по аксиоме начальных условий $Pos(s_0, O, C_1)$

робот пытается доказать теорему-задание $\exists s\ Pos(s, fin, fin)$ (где *s*— переменная, описывающая ситуацию). Однако, поскольку знаний о среде, заключенных в построенных АС, явно недостаточно (теорема не выводима из АС), ответ, т. е. искомый плана поведения, не будет получен. В процессе логического вывода робот убеждается, что маршруты через точки *О, 1, 2, 3* и *0, 4, 5, 6* к выполнению задания не приводят. Далее, используя критерий близости к складу C_1, робот принимает решение переместиться в точку *3*. Последовательность действий на этом этапе определяется термом ситуации в резольвенте $f(2, 3, f(1, 2, f(0, 1, s_0)))$, который расшифровывается в обратном порядке и в соответствии с определением функции *f* означает: переехать из точки *О* в точку 1, затем в точки *2* и *3*. Передвигаясь согласно выбранному маршруту, робот останавливается в каждой из них и пополняет свои знания о среде посредством опроса сенсорной системы. Так возникают новые аксиомы среды (АС): $G(1,4), G(4, 7), G(7, 8), \neg Pos(s, 8, y) \vee Pos(s, fin, y)$ (последняя аксиома среды означает, что точка *8* находится у выхода).

Приехав в точку *3*, робот не формирует ни одной новой АС, а поэтому и новой резольвенты. Он в тупике. «Осознав» это, робот вынужден развернуться и исправить те действия, которые привели его в «тупиковую» точку *3*, в обратном порядке, пока не появится первая возможность образования новой резольвенты. В результате он выбирает маршрут через точки 1, *4, 5*, используя ранее построенные аксиомы обучения (АО) (к числу которых относятся АС и СА, формируемые в процессе функционирования робота). Так, формируя и корректируя локальные планы поведения на основе целенаправленной переработки новой информации, робот, в конце концов, решает поставленную задачу: отыскивает (распознает) нужный объект на складе C_1, погружает (с помощью манипулятора) его на тележку, подвозит к складу C_2, сгружает объект и покидает цех через выход. При этом адаптивная система логического вывода строит 47 резольвент. Окончательный маршрут робота, реализующий выработанный план поведения, изображен на рис. 9 сплошными линиями со стрелками.

Рассмотренный пример планирования поведения робота в условиях большой априорной (начальной) неопределенности замечателен тем, что он ясно демонстрирует, что обычная (неадаптивная) система логического вывода принципиально не способна решать такого рода задачи без использования элементов обучения и адаптации. Важно отметить, что автоматическое формирование АО и адаптивная подстройка стратегии, использующей АО, не только делает задачу разрешимой, но и существенно сокращает (за счет отсечения многих

тупиковых ветвей на дереве вывода) число шагов в процессе поиска плана поведения как при полной, так и при частичной информированности об условиях функционирования робота.

5.5. Алгоритмы распознавания ситуаций

Задача описания, распознавания и анализа ситуаций, решаемая на третьем уровне иерархии, является одной из центральных проблем построения ИИ. Для формализации и автоматизации решения этой задачи, как мы увидим ниже, опять-таки удобно использовать язык исчисления предикатов и адаптивную систему логического вывода. В качестве примера, на котором мы будем демонстрировать использование языка исчисления предикатов и адаптивной системы логического вывода, рассмотрим функционирование робота. Поскольку для робота наиболее важное значение имеет зрительная информация, мы сосредоточим внимание на методах описания, распознавания и анализа трехмерных сцен по их изображениям. Что же касается задач переработки речевой, тактильной и другой сенсорной информации, то они могут решаться по существу теми же методами. Трудности, возникающие при решения задачи распознавания отдельных предметов на сложной сцене, связаны с наличием «порочного круга»: для того чтобы распознать некоторый предмет на сцене, нужно прежде всего его «выделить», а для того чтобы «выделить» этот предмет, нужно его распознать. Это приводит к тому, что классические методы распознавания «перцептронного» или статистического типа, в этой задаче практически не применимы. На первый взгляд кажется, что выход из указанного «порочного круга» только один — полный перебор элементов изображения предмета и сцены. Однако более глубокий анализ этой задачи позволяет ее сформулировать и решить как задачу логического вывода.

Идея предлагаемого решения основана, во-первых, на том, что применяется предварительное обучение робота путем показа ему отдельных предметов из различных классов и сообщения ему не только названия предмета, но и, возможно, его описания. Это обучение ведется человеком в диалоговом режиме, позволяющем оперативно выявлять и исправлять ошибки робота. Во-вторых, специфика задачи ярко проявляется в большой вариативности изображений реальных предметов и сцен, которая имеет двоякую природу. С одной стороны, она порождается естественной вариативностью характеристик самих предметов, с другой стороны, — перемещением предметов в пространстве. Вариативность второго рода можно трактовать как

результат действия некоторых известных преобразований изображения. Априорное знание этих преобразований позволяет построить алгоритм распознавания, инвариантный по отношению к этим преобразованиям. Благодаря этому удается не только «избавиться» от вариативности второго рода, но и существенно облегчить задачу переработки зрительной информации.

Сама эта задача подразделяется на следующие подзадачи: *описание классов (формирование понятий о классах объектов), распознавание изображения данного предмета, анализ изображения сцены (распознавание всех предметов на сцене).* Результаты решения перечисленных подзадач могут использоваться для моделирования внешней среды (на четвертом уровне иерархии) путем преобразования изображений предметов и сцен в их пространственное представление, а также для описания сцен на естественном языке или, наоборот, для синтеза изображения сцены по ее описанию.

Рассмотрим сначала *задачу описания классов* (формирования понятий). Эта задача решается в режиме обучения. Роботу последовательно предъявляют предметы из различных классов (и, возможно, в различных ракурсах) с указанием, к какому классу каждый такой предмет принадлежит. Эти предметы (а также их изображения) называют *эталонными*, а совокупность классифицированных предметов — *обучающей выборкой*. По этим данным робот должен автоматически сформировать описание классов в терминах тех свойств предметов, которые непосредственно измеряются сенсорной системой. Примерами этих свойств, которые мы будем называть первичными признаками, являются следующие: «красное», «ближе», «правее», «выше», «две точки соединены отрезком», «два отрезка параллельны», «зоны одинаковой яркости» и т. п. Таким образом, *изображения предметов и сцен задаются полным набором своих первичных признаков.*

Каждому признаку поставим в соответствие предикат $\xi_i(x_1, \ldots, x_{ni})$, где $x_1, \ldots, x_{ni}$ — элементы изображения ω, определяющие наличие на нем i-го признака. Тогда каждому эталону ω_h (предмету из обучающей выборки) соответствует набор значений предикатов $\xi_i^{(h)}(c_1, \ldots, c_{hi})$, истинных на изображении данного предмета ω_h. Здесь c_j — предметные константы, означающие фиксированные части изображения. Описанием изображения эталона ω_h будем называть конъюнкцию

$$\bigwedge_{t=1}^{P_h} \xi_t^{(h)}(c_1, \ldots, c_{ni}).$$

Поскольку к одному и тому же классу могут принадлежать несколько эталонов (например, предмет из этого класса, показанный в разных ракурсах), то описанием всех эталонных изображений, принадлежащих данному классу Ω_k, является дизъюнкция

$$\bigvee_{\omega_h \in \Omega_k} \bigwedge_{i=1}^{P_h} \xi_i^{(h)} (c_1, \ldots, c_{ni}).$$

Если теперь в этой дизъюнкции все предметные константы c_j заменить на соответствующие предметные переменные x_j, то получим ППФ, которую естественно назвать описанием класса Ω_k. Введем предикат $\sigma(k)$, означающий принадлежность изображения классу Ω_k. Тогда каждый класс Ω_k описывается аксиомой класса (АК) вида

$$\bigvee_{\omega_h \in \Omega_k} \bigwedge_{i=1}^{P_h} \xi_i^{(h)} (x_1, \ldots, x_{ni}) \to \sigma (k), \tag{1}$$

(которая по существу является логическим определением класса (или соответствующего ему понятия). Из вышеизложенного ясно, что АК вида (1) могут строиться роботом автоматически в режиме обучения по мере последовательного предъявления ему эталонов.

Практически важно, чтобы система АК обладала свойствами полноты, независимости и инвариантности в естественных смыслах. Дадим развернутое определение этих всовйств.

Систему АК будем называть *полной* на множестве изображений $\{\omega\}$, если для всякого изображения ω из этого множества найдется АК, принимающая на нем значение «истинно». Следует отметить, однако, что полнота системы АК не исключает того, что для некоторого изображения могут найтись две АК, принимающие на нем значение «истинно».

В некоторых случаях требуется, чтобы ни одно исследуемое изображение не было отнесено одновременно к нескольким классам (например, если априори известно, что распознаваемые классы изображений не пересекаются). Это требование должно быть отражено в АК. Систему АК будем называть *непротиворечивой,* если существует только одна АК, истинная на любом данном изображении. Из приведенного определения следует, что непротиворечивая система АК исключает возможность пересечения классов, описываемых этой системой аксиом. Очевидно, что непротиворечивость системы АК всегда можно эффективно проверить.

Вариативность изображений, порождаемая пространственными преобразованиями воспринимаемых предметов, а также действием разного рода помех и искажений, требует, чтобы система обладала определенной инвариантностью и помехозащищенностью. Например,

всевозможные изображения стола, отличающиеся от эталонного (по которому строится соответствующая АК) сдвигом, поворотом, масштабом, а также некоторыми незначительными искажениями или помехами (лишние линии, незначительные изменения пропорций и т. п.), должны описываться одной и той же АК «стол», т. е. должны классифицироваться как эквивалентные. Систему АК будем называть *инвариантной* по отношению к заданной группе преобразований, если каждая входящая в нее аксиома принимает одно и то же значение на изображениях, отличающихся преобразованиями из этой группы. Таким образом, инвариантность системы АК позволяет «снять» охарактеризованную выше вариантность изображений и тем самым облегчить распознавание сцен по их изображениям.

Задачи распознавания и анализа изображений могут быть переформулированы как задачи логического вывода. Эти задачи решаются в режиме распознавания. Роботу предъявляются сцена, изображение $\tilde{\omega}$ которой может содержать одно или несколько изображений предметов. Эти изображения отличаются от эталонов некоторыми преобразованиями и даже могут частично перекрываться,

Описание изображения сцены $S(\tilde{\omega})$ представляет собой конъюнкцию всех первичных предикатов, истинных на данном изображении $\tilde{\omega}$. В этих условиях задача отыскания на изображении $\tilde{\omega}$ изображения из определенного класса Ω_k, т. е. задача распознавания, сводится к нахождению доказательства теоремы $S(\tilde{\omega}) \rightarrow \sigma(k)$. Саму эту теорему можно трактовать как вопрос: имеется ли на данном изображении $\tilde{\omega}$ предмет из k -го класса? Ясно, что в процессе ответа на этот вопрос описание $S(\tilde{\omega})$ может быть использовано не полностью. Поэтому измерение тех или иных первичных признаков и предикатов должно производиться по мере необходимости в процессе логического вывода.

Задача анализа изображения сцены заключается в распознавании на ней всех изображений предметов из различных классов. Формально эта задача сводится к последовательному доказательству теорем $S(\tilde{\omega}) \rightarrow \exists\, \sigma(k)$, $i = 1, \ldots, N - 1$, где $S_1(\tilde{\omega}) = S(\tilde{\omega})$, а $S_{i+1}(\tilde{\omega})$ получается из $S_i(\tilde{\omega})$ вычеркиванием всех предикатов, участвовавших в выводе i-й теоремы. Содержательно это означает, что, как только выделяется очередное изображение предмета из некоторого класса, оно при дальнейшем анализе не рассматривается. Полный анализ изображения сцены заканчивается распознаванием (и тем самым выделением) всех видимых изображений предметов, составляющих сцену.

В режиме обучения и распознавания на различных изображениях может встречаться один и тот же набор первичных признаков,

характеризующих, например, фрагмент изображения. В таких случаях естественно ввести вторичные признаки, каждый из которых представляет собой некоторую совокупность из первичных признаков, а также вторичные предикаты, определяющие соответствующие фрагменты изображения как новые понятия.

Аксиомами обучения (АО) будем называть ППФ вида

$$\bigwedge_{j=1}^{r} \xi_j (x_{j1}, \ldots, x_{jm}) \rightarrow \alpha_i, \qquad (2)$$

где $\xi_j(x_{j1}, ..., x_{jm})$, $j = 1, ..., r$ — первичные предикаты, дающие полное описание вторичного предиката α_i, т. е. определяющие некоторый фрагмент изображения как новое понятие. Использование АО позволяет не только более экономно представить АК, но и повысить эффективность системы логического вывода в процессе распознавания и анализа.

Как мы уже отмечали, универсальным средством логического вывода в исчислении предикатов является метод резолюций. Поэтому любой конкретный алгоритм распознавания или анализа определяется стратегией метода резолюций. Важно отметить, что для распознавания на изображении сцены нужного предмета не обязательно строить доказательство теоремы $S(\tilde{\omega}) \rightarrow \sigma(k)$ полностью, т. е. перебирать все элементы искомого простого изображения на изображении сцены $\tilde{\omega}$. Вместо этого достаточно найти фрагмент искомого изображения, который содержится лишь в изображениях *k*-го класса и не содержится ни в одном изображении предметов из других классов. Именно это обстоятельство позволяет сильно ограничить число шагов логического вывода, а также распознавать частично закрытые изображения предметов.

Качество работы алгоритмов распознавания и анализа естественно характеризовать числом обращений к информационно-измерительной (сенсорной) системе с целью определения нужных признаков. Заметим, что в нашей формализации это в точности совпадает с числом шагов логического вывода, т. е. с числом резольвент, формируемых в процессе распознавания и анализа. Стратегию логического вывода будем называть *оптимальной,* если число шагов доказательства (число резольвент) минимально.

Важно, отметить, что система логического вывода способна совершенствовать алгоритмы распознавания и анализа за счет использования элементов обучения, а именно: АО и адаптивной подстройки стратегии. Адаптивная подстройка стратегии осуществляется путем ее перестройки (например, путем перестройки оптимального распознающего графа) по мере распознавания новых

изображений и формирования аксиом обучения (АО). Использование АО сокращает логический вывод на длину ее описания, а процесс построения оптимального **распознающего графа** — на значение экспоненциальной функции от этой длины.

Проиллюстрируем описанный метод на примере решения задачи описания, распознавания и анализа обстановки в цехе. Предположим, что в результате предварительной фильтрации исходные изображения предметов и сцен превращаются в контурные изображения, составленные из отрезков. Введем необходимые для дальнейшего первичные признаки и предикаты, представленные в табл. 3. Вторичные признаки и предикаты (аксиомы обучения) представлены в табл. 4.

Таблица 3

Первичный признак	Предикат	Интерпретация
Вертикальный отрезок	$ВТ\ (x,\ y)$	Истинен, если точки x и y соединены вертикальным отрезком.
Горизонтальный отрезок	$ГР\ (x,\ y)$	Истинен, если точки x и y соединены горизонтальным отрезком.
Параллельность двух отрезков	$ПРЛ\ (x, y, u, v)$	Истинен, если отрезки $(x,\ y)$ и $(u,\ v)$ параллельны.

Таблица 4

Вторичный признак-понятие	Символ понятия	Логическое описание понятия
Параллелограмм	$ПР\ (x_1,\ x_2,\ x_3,\ x_4)$	$ПРЛ\ (x_1,\ x_2,\ x_3,\ x_4) \wedge$ $ПРЛ\ (x_1,\ x_4,\ x_2,\ x_3)$
Горизонтальный параллелограмм	$ГПР\ (x_1,\ x_2,\ x_3,\ x_4)$	$ПР\ (x_1,\ x_2,\ x_3,\ x_4) \wedge$ $ГР\ (x_1,\ x_2) \wedge ГР\ (x_2,\ x_3)$
Вертикальный прямоугольник	$ВПР\ (x_1,\ x_2,\ x_3,\ x_4)$	$ПР\ (x_1,\ x_2,\ x_3,\ x_4) \wedge$ $ВТ\ (x_1,\ x_2) \wedge ГР\ (x_2,\ x_3)$

В режиме обучения роботу предъявляются эталонные контурные изображения токарного и сверлильного станков, представленные на рис. 10.

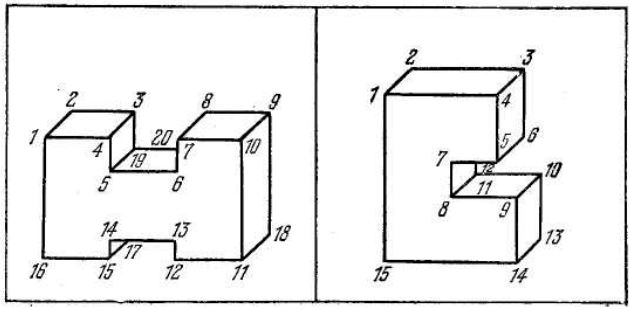


Рис. 10. Эталонные изображения (режим обучения).

По этим данным он строит АК, общий вид которых приведен в табл. 5.

Таблица 5

Класс-понятие	Символ класса	Логическое описание класса
Токарный станок	σ (*ТС*)	$ГПР\ (x_1, x_2, x_3, x_4) \wedge ВПР\ (x_5, x_4, x_3, x_{19}) \wedge ГР\ (x_{19}, x_{20}) \wedge ГР\ (x_5, x_6) \wedge ВТ\ (x_6, x_7) \wedge ГПР\ (x_7, x_8, x_9, x_{10}) \wedge ВПР\ (x_{11}, x_{10}, x_9, x_{18}) \wedge ГР\ (x_{12}, x_{11}) \wedge ВТ\ (x_{12}, x_{13}) \wedge ГР\ (x_{14}, x_{13}) \wedge ВТ\ (x_{15}, x_{14}) \wedge ГР\ (x_{15}, x_{17}) \wedge ГР\ (x_{16}, x_{15}) \wedge ВТ\ (x_{16}, x_1)$
Сверлильный станок	σ (*СС*)	$ГПР\ (x_1, x_2, x_3, x_4) \wedge ВПР\ (x_5, x_4, x_3, x_6) \wedge ГР\ (x_7, x_5) \wedge ВТ\ (x_8, x_7) \wedge ГПР\ (x_8, x_{11}, x_{10}, x_9) \wedge ВТ\ (x_{11}, x_{12}) \wedge ВПР\ (x_{14}, x_9, x_{10}, x_{13}) \wedge ГР\ (x_{15}, x_{14}) \wedge ВТ\ (x_{15}, x_1)$

Так в режиме обучения автоматически формируется описание классов

В режиме распознавания роботу предъявляется изображение произвольной сцены — обстановки в цехе, попавшей в «поле зрения» робота. Для определенности пусть это будет изображение сцены, представленное на рис. 11.

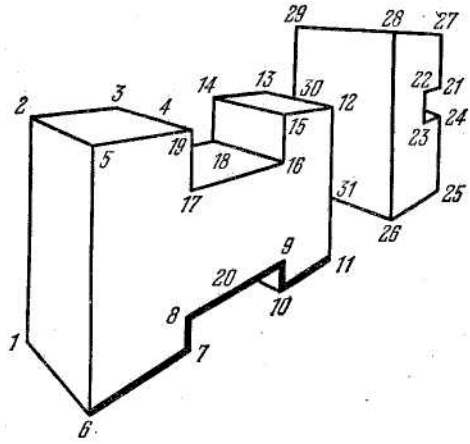


Рис. 11. Изображение сцены (режим распознавания).

Описание этой сцены (на языке первичных и вторичных предикатов) имеет вид
1. *ВПР (6, 5, 2, 1);* 2. *ГПР (5, 2, 3, 4);* 3. *ВТ (17, 4);* 4. *ГР (17, 16);* 5. *ВПР (16, 15, 14, 18);* 6. *ГР (19, 18); 7. ГПР (15, 14, 13, 12);* 8. *ВТ (11, 12);* 9. *ГР (10, 11);* 10. *ВТ (10, 9);* 11. *ГР (10, 20);* 12. *ГР (8, 9,);* 13. *ВТ (7, 8);* 14. ГР *(6, 7);* 15. *ВТ (30, 28);* 16. *ГР (29, 28);* 17. *ГР (29, 27);* 18. *ВТ (21, 27);* 19. *ГР (22, 21);* 20. *ВТ (23, 22);* 21. *ГР (23, 24);* 22. *ВТ (25, 24);* 23. *ГР (26, 25);* 24. *ВТ (26, 29);* 25. *ГР (26, 31).*

Предположим, что мы спрашиваем робота (или он сам задается этим вопросом): «есть ли на изображении сцены токарный станок?». Для ответа на вопрос робот пытается доказать теорему

$$\bigwedge_{j=1}^{25} \xi_j (c_{j1}, \ldots, c_{jk}) \rightarrow \sigma (\mathrm{TC}),$$

где $\xi_j (c_{j1}, \ldots, c_{jk})$, $j = 1, \ldots, 25$, — предикаты, входящие в приведенное выше описание изображения сцены.

Доказательство этой теоремы с помощью стратегии лозы потребовало в зависимости от порядка записи АК от 8 до 54 резольвент. Однако, как уже отмечалось, для распознавания токарного станка на изображении сцены полное доказательство соответствующей теоремы не обязательно, — возможно распознавание по фрагменту. Использование оптимальной стратегии в этом примере позволило сократить число резольвент до 5. При этом был автоматически выделен и существенно использовался в процессе логического вывода фрагмент изображения, обведенный на рис.11 жирной линией. Интересно отметить, что решение той же задачи без использования АО приводит

к существенному снижению эффективностисистемы логического вывода. В этом случае описание изображения сцены требует уже не 25, а 47 первичных предикатов. В процессе распознавания токарного станка с помощью стратегии лозы строится по меньшей мере 73 резольвенты.

Резюмируя содержание, отметим, что *язык исчисления предикатов и адаптивная система логического вывода* позволяют эффективно решать не только задачу автоматизированного планирования поведения робота в условиях априорной неопределенности, но и задачи описания классов, распознавания и анализа ситуаций в окружающей робота среде.

6. Основы логики высказываний

Пусть x_1 и x_2 — некоторые *высказывания,* которые могут быть истинными (1) или ложными (0), например: «Я пойду в театр» *(x_1)* и «Я встречу друга» *(x_2).* Дизъюнкцией $x_1 \vee x_2$ является сложное высказывание «Я пойду в театр *или* встречу друга», а конъюнкцией $x_1 \wedge x_2$ — высказывание «Я пойду в театр *и* встречу друга».

Ясно, что если высказывание истинно, то его отрицание ложно. Сложное высказывание, образованное дизъюнкцией двух высказываний, истинно при условии, что истинно хотя бы одно из них. Сложное высказывание, образованное конъюнкцией двух истинных высказываний истинно, если истинны оба эти высказывания одновременно.

Итак, высказывания можно рассматривать как двоичные переменные, а связки «не», «или», «и», с помощью которых образуются сложные высказывания,— как операции над этими переменными. В алгебре высказываний используются еще две операции: *импликация* $x_1 \rightarrow x_2$, соответствующая связке «если, то» и *эквиваленция* $x_1 \sim x_2$, соответствующая связке «если и только если». Они задаются следующими таблицами:

$x_1 \to x_2$

x_1	x_2	
	0	1
0	1	1
1	0	1

$x_1 \sim x_2$

x_1	x_2	
	0	1
0	1	0
1	0	1

В нашем примере импликацией будет высказывание: *«Если* я пойду в театр, *то* встречу друга», а эквиваленцией— «Я пойду в театр, *если и только если* встречу друга». Как видно из таблиц, импликация высказываний ложна только в случае, когда первое из простых высказываний истинно, а второе ложно. Эквиваленция является истинным высказыванием, если оба простые высказывания истинны или ложны одновременно.

Обозначив буквами простые высказывания, можно представить сложное высказывание формулой с помощью соответствующих связок. Например, высказыванию *«Если* давление масла на шарик клапана больше усилия его пружины (x_1), *то* масло открывает клапан (x_2) *и* частично перетекает из нагнетательной полости во впускную *(x_3)»* соответствует формула $x_1 \to x_2 x_3$.

6.1. Закон исключения третьего

Рассматривая высказывания как двоичные переменные, обычно считают, что они удовлетворяют *закону исключения третьего:* каждое высказывание может быть истинным или ложным (третьего не дано). При этом высказывание не может быть одновременно и истинным и ложным *(закон противоречия).* Значения «истина» и «ложь», соответствующие 1 и 0 в двузначной логике, в логике высказываний обозначаются через «И» и «Л».

Истинность данного высказывания в повседневной жизни устанавливается на основе анализа его смысла. Например, высказывание «Киев — столица Украины» — истинно, а «100 < 10» — ложно. Однако даже в таких категоричных случаях их истинность относительна. Первое предложение перестает быть истинным, если речь идет о периоде, когда столицей УССР был Харьков. Второе

предложение становится истинным, если считать, что число 100 записано в двоичной системе счисления, а 10 — в десятичной («4 < 10»).

Таким образом, высказывание может быть либо истинным, либо ложным в зависимости от обстоятельств, которыми руководствуются при его истолковании. Обычно эти обстоятельства не фигурируют явно в простом высказывании. Например, истинность таких высказываний, как «Хорошая погода», «Сегодня — 16 января», «Результат измерений диаметра цилиндра равен 52 мм» зависит соответственно от вкусов или критерия оценки погоды, сегодняшней даты, требуемой точности измерения. Логика высказываний отвлекается от конкретного смысла предложений, и ответственность за их истолкование возлагает на лиц, компетентных в соответствующей области. **Она дает лишь общие методы анализа сложных высказываний и принципы логических рассуждений и доказательств.**

Принятие закона исключения третьего позволяет полностью использовать в логике высказываний аппарат двузначной логики. Дальнейшее развитие логики высказываний основано на допущении нескольких значений истинности (например, кроме значений «истина» и «ложь» допускается третье значение — «неопределенность»). В подобных случаях используется аппарат многозначной логики. Если истинность предложений определяется с некоторой вероятностью, то логика высказываний превращается в вероятностную логику. Мы рассмотрим только двузначную логику высказываний, причем для обозначения значения «истина» будем применять 1, а значения «ложь» — 0.

6.2. Сентенциональные связки

Сентенциональными связками называют слова «не», «и», «или», «если..., то» и «если и только если», с помощью которых в обычном языке из простых предложений образуются сложные предложения. Как указывалось выше каждой из этих связок соответствует своя логическая операция: отрицание, крнъюнкция, дизъюнкция, импликация и эквиваленция. Обычно высказывания обозначают прописными буквами, а для операций используются те же символы, что и в алгебре логики. Таблицы соответствия в логике высказываний называют *истинностными таблицами.* Для указанных пяти связок они имеют вид:

P	0	1
$\overline{P}$	1	0

P	0	0	1	1
Q	0	1	0	1
PQ	0	0	0	1
$P \vee Q$	0	1	1	1
$P \rightarrow Q$	1	1	0	1
$P \sim Q$	1	0	0	1

Сентенциональные связки в разговорном языке допускают различные варианты. Поэтому при записи сложного предложения в виде формулы алгебры логики важно выяснить характер логической связи между предложениями, не вдаваясь в смысл самих предложений.

Истолкование *отрицания* $\overline{P}$, *конъюнкции* PQ и *дизъюнкции* $P \vee Q$ обычно не вызывает трудностей.

Импликации $P \rightarrow Q$ в обычной речи соответствует условное предложение «если *P,* то *Q*», причем *P* называется *посылкой (антецедентом),* а *Q* — *следствием (консеквентом).* Могут встретиться и другие выражения, имеющие тот же тип логической связи, например: «*P* влечет *Q*», «*P* только тогда, когда *Q*», «*P* есть достаточное условие для *Q*», «*Q* при условии, что *P*», «*Q* есть необходимое условие для *P*» и т. п.

Эквиваленция $P \sim Q$ определяет логическую связь в так называемых *биусловных предложениях* типа «*P*, если и только если *Q*» или в других грамматических формах: «*P* тогда и только тогда, когда *Q*», «если *P*, то *Q* и обратно, если *Q*, то *P*», «*Q* есть необходимое и достаточное условие для *P*».

6.3. Формулы и подстановки

Всякое сложное предложение, которое состоит из простых предложений, связанных сентенциональными связками, можно представить в символической форме. В результате получаем *высказывательную формулу.* На каждом наборе значений истинности букв (переменных) формула принимает некоторое значение. Следовательно, всякую формулу логики высказываний можно рассматривать как *истинностную функцию.*

Рассмотрим, например, сложное высказывание: *«Если* применить стальные конструкции *(P), то* масса снижается *(Q) и* стоимость увеличивается *(R).* Стальные конструкции *не* применяются *($\overline{P}$), а* масса снижается *(Q)».* Соответствующая формула $(P \rightarrow QR)\,\overline{P}\,Q$ представляется следующей таблицей истинности:

P	0	0	0	0	1	1	1	1
Q	0	0	1	1	0	0	1	1
R	0	1	0	1	0	1	0	1
QR	0	0	0	1	0	0	0	1
$P \to QR$	1	1	1	1	0	0	0	1
$\bar{P}Q$	0	0	1	1	0	0	0	0
$(P \to QR)\,\bar{P}Q$	0	0	1	1	0	0	0	0

Отсюда видно, что сложное предложение истинно на двух наборах значений аргументов *P, Q,R,* а именно: (0, 1,0) и (0, 1. l), а на остальных наборах оно ложно.

В логике высказываний дается следующее определение формулы: 1) переменные высказывания суть формулы; 2) если *A* и *B* — формулы, то *(AB), (A* $\vee$ *B*), *(A*→ *B), (A* ~ *B)* и $\overline{A}$ также формулы. Это определение имеет *рекурсивный характер* в том смысле, что первая его часть определяет элементарные формулы, а вторая позволяет из любых формул образовать новые формулы. При записи формул используются обычные упрощения, указанные ранее. Пусть, например, требуется получить формулу $(A \to \overline{AB}) \to ((C \vee D) \to AB)$. Выбираем необходимое множество элементарных формул *A, B, C, D.* Затем последовательно получаем формулы

$$\overline{AB},\ A \to \overline{AB},\ (C \vee D) \to AB,\ (A \to \overline{AB})((C \vee D) \to AB).$$

Как видно, процесс образования формулы происходит путем расширения их множества до тех пор, пока это множество не будет содержать требуемую формулу. Все формулы, построенные в указанном процессе, называются *частями результирующей формулы.*

Если имеется некоторая высказывательная формула, то можно построить соответствующее сложное предложение, заменяя буквы простыми предложениями (одинаковые вхождения букв замещаются одним и тем же предложением). Полученное таким путем предложение называется *подстановкой в данную формулу.* Так, полагая *P* — «идет снег», *Q* — «2 × 2 = 4» и *R* — «слоны зеленые», по формуле $P \to QR$ получаем подстановку: «Если идет снег, то 2 × 2 = 4 и слоны зеленые».

Истинность этого высказывания определяется только приведенной выше таблицей и никоим образом не связана с конкретным содержанием как простых предложений, так и полученного в результате их объединения сложного предложения.

Как видно из таблицы, истинностная функция истинна на всех наборах значений аргументов, кроме наборов (1, 0, 0), (1, 0, 1) и (1, 1, 0). Например, при P = 0, *Q*= 0 и *R* = 1, получим истинное высказывание: «Если не идет снег, то 2 × 2 = 4 и слоны зеленые».

6.4. Сложные высказывания и «здравый смысл»

При первом знакомстве с логикой высказываний трудно без чувства юмора принять подобные предложения. Наш опыт подсказывает, что подвергать сомнению истину «2×2 = 4» так же нелепо, как и утверждать, что «слоны зеленые». Кроме того, между посылкой «идет снег» и ее следствием нет причинной связи. Поэтому с точки зрения «здравого смысла» такие высказывания кажутся несуразными и возможность их появления в логике высказываний следовало бы исключить.

Однако необходимо преодолеть психологический барьер и понять, что ограничения, основанные на «здравом смысле» и причинной связи в логике высказываний не только невозможны, но и нежелательны. В п.6.1 уже указывалось на относительность истинности или ложности того или иного высказывания. Если бы множество допустимых высказываний было подвергнуто испытанию «здравым смыслом», то возникли бы непреодолимые трудности **из-за отсутствия строгого определения, что следует под этим понимать**. Человеку, который никогда не видел снега и не слышал о нем, фраза «идет снег» покажется бессмысленной, а высказывание «слоны зеленые» может иметь вполне определенный смысл, если речь идет, например, о выборе цвета для игрушечных слонов. Аналогичные соображения можно привести и в пользу допущения логической связи между любыми предложениями без учета причинной зависимости между ними.

Поэтому **логика высказываний, отвлекаясь от конкретного содержания высказываний, по существу занимается лишь анализом и синтезом высказывательных формул и изучением отношений между высказываниями.** Что же касается **«здравого смысла», то он должен проявляться при использовании законов логики высказываний в ее конкретных приложениях.**

6.5. Тавтологии

Тождественно истинная формула, т. е. такая формула, которая принимает значения 1 при любых значениях ее компонентов, называется *тавтологией.* Тождественно ложная формула на всех наборах ее компонентов принимает значение 0 и называется *противоречием.* Если в технических приложениях логические функции, выражаемые тавтологиями или противоречиями, практически не

представляют интереса, то в логике высказываний они играют первостепенную роль.

Примером тавтологии может служить высказывание: «Если внедрить новую технологию *(P),* то качество продукции улучшится *(Q).* При улучшении качества продукции (*Q*), ее сбыт увеличивается *(R).* Новая технология внедрена *(P).* Следовательно, сбыт продукции увеличился *(R)».* Оно выражается формулой $(P \to Q)(Q \to R)P \to R$.

Чтобы выяснить, является ли данная формула тавтологией, можно составить для нее истинную таблицу. Так, для приведенной выше формулы имеем:

P	0	0	0	0	1	1	1	1
Q	0	0	1	1	0	0	1	1
R	0	1	0	1	0	1	0	1
$P \to Q$	1	1	1	1	0	0	1	1
$Q \to R$	1	1	0	1	1	1	0	1
$(P \to Q)(Q \to R)P$	0	0	0	0	0	0	0	1
$(P \to Q)(Q \to R)P \to R$	1	1	1	1	1	1	1	1

Можно также воспользоваться зависимостями

$$x_1 \to x_2 = \tilde{x}_1 \vee x_2,\ x_1 \sim x_2 = x_1 x_2 \vee \tilde{x}_1 \tilde{x}_2 = (x_1 \vee \tilde{x}_2)(\tilde{x}_1 \vee x_2)$$

и преобразовать высказывательную формулу к нормальной форме. Если хотя бы один член дизъюнктивной нормальной формы окажется равным 1, то соответствующая ей формула является тавтологией. Если хотя бы один член конъюнктивной нормальной формы окажется равным 0, то соответствующая ей формула является противоречием. Так, для нашего примера имеем:

$$(P \to Q)(Q \to R)P \to R = (\bar{P} \vee Q)(\bar{Q} \vee R)P \to R =$$

$$= (\bar{P}\bar{Q} \vee \bar{P}R \vee QR)P \to R = PQR \to R = \overline{PQR} \vee R = \bar{P} \vee \bar{Q} \vee \bar{R} \vee R =$$

$$= \bar{P} \vee \bar{Q} \vee (\bar{R} \vee R) = \bar{P} \vee \bar{Q} \vee 1 = 1.$$

Очевидно, формула не является тавтологией, если она принимает значение 0 хотя бы на одном наборе значений переменных. Этим обстоятельством можно воспользоваться для распознавания тавтологий сокращенным методом «обратного рассуждения», заключающемся в поиске таких переменных, при которых формула оказываекя ложной. Так, приведенная выше формула можем принять значение 0, если и только если *R* ложно, а $(P \to Q)(Q \to R)P$ истинно. При этом должны бьпь истинны $P \to Q$, $Q \to R$ и *P*. При истинном *P* формула $P \to Q$ истинна только при истинном *Q*. В свою очередь, при

истинном Q формула $Q \rightarrow R$ истинна только при истинном R. Таким образом, анализируемая формула может быть ложной, если и только если R одновременно и истинно и ложно, что невозможно в силу закона противоречия. Следовательно, она является тавтологией.

Для указания на то, что данная формула является тавтологией, используется знак $\models$, который помещается перед формулой, например:

$$\models (P \rightarrow Q)(Q \rightarrow R)P \rightarrow R.$$

6.6. Законы логики высказываний

Различные подстановки в тавтологию, независимо от их конкретного содержания, всегда являются истинными предложениями в силу одной только своей логической структуры. Иначе говоря, **тавтологии можно рассматривать как некоторые *логически истинные схемы* рассуждений или утверждений**. Поэтому они играют роль *законов (теорем) логики высказываний,* претендующих на установление методов построения правильных умозаключений.

Существует бесконечное множество тавтологий, а значит, и законов логики высказываний. Наиболее часто используемые из них следующие: $P \rightarrow P$ *(закон тождества),* $P \vee \overline{P}$ *(закон исключения третьего),* $\overline{P\overline{P}}$ *(закон противоречия),* $\overline{\overline{P}} \sim P$ *(закон двойного отрицания),* $P \rightarrow (Q \rightarrow P)$ *(добавление антецедента* или *verum ex quodlibet* — истина из чего угодно), $\overline{P} \rightarrow (P \rightarrow Q)$ *(ex falso quodlibet* — из ложного что угодно), $(P \rightarrow Q)P \rightarrow Q$ *(закон отделения* или *modus ponens),* $(P \rightarrow Q)\overline{Q} \rightarrow \overline{P}$ *(modus tollens),* $(P \rightarrow Q)(Q \rightarrow R) \rightarrow (P \rightarrow R)$ *(закон силлогизма),* $(P \rightarrow Q) \rightarrow (\overline{Q} \rightarrow \overline{P})$ *(закон контрапозиции).*

Каждый из законов логики высказываний отображает в символической форме некоторую схему доказательства. Например, в соответствии с законом отделения, если истинно, что некоторое высказывание P имплицирует высказывание Q и, кроме того, P истинно, то истинно и Q. Modus tollens применяется при доказательстве от противного: желая доказать утверждение P, предполагается, что P ложно, и показывается, что P имплицирует некоторое высказывание Q, о котором известно, что оно ложно ($\overline{Q}$ истинно). Отсюда заключается, что P истинно.

6.7. Равносильность

Две формулы называются *равносильными,* если на всех наборах значений входящих в них переменных эти формулы принимают одинаковые значения. Для обозначения этого отношения часто употребляют символ $\Leftrightarrow$, так что равносильность формул A и B символически записывается как $A \Leftrightarrow B$. Легко видеть, что равносильность — это отношение эквивалентности: оно рефлексивно $(A \Leftrightarrow A)$, симметрично (если $A \Leftrightarrow B$, то $B \Leftrightarrow A$) и транзитивно (из $A \Leftrightarrow B$ и $B \Leftrightarrow C$ следует, что $A \Leftrightarrow C$). Поэтому равносильность называют также *логической эквивалентностью.*

Равносильность формул логики высказываний вытекает из тождественности соответствующих формул алгебр и логики. Так, в соответствии с тождественными преобразованиями получаем следующие равносильности:

$\bar{\bar{A}} \leftrightarrow A;\ A \vee A \leftrightarrow A;\ AA \leftrightarrow A;\ A \vee B \leftrightarrow B \vee A;\ AB \leftrightarrow BA;\ A \vee (B \vee C) \leftrightarrow (A \vee B) \vee C;\ A(BC) \leftrightarrow (AB)C;\ A(B \vee C) \leftrightarrow AB \vee AC;\ A \vee BC \leftrightarrow (A \vee B)(A \vee C);\ \overline{A \vee B} \leftrightarrow \bar{A}\bar{B};\ \overline{AB} \leftrightarrow \bar{A} \vee \bar{B};\ A \vee AB \leftrightarrow A;\ A(A \vee B) \leftrightarrow A;\ A \vee \bar{A}B \leftrightarrow A \vee B$ и т. д. Кроме того, с помощью отношения равносильности выражаются различные связки между формулами:

$A \to B \leftrightarrow \bar{A} \vee B;\ A \sim B \leftrightarrow AB \vee \bar{A}\bar{B} \leftrightarrow (A \vee \bar{B})(\bar{A} \vee B);\ A \vee B \leftrightarrow \bar{A} \to B;$
$AB \leftrightarrow \overline{A \to \bar{B}};\ A \sim B \leftrightarrow (A \to B)(B \to A).$

Эти и подобные им равносильные соотношения можно использовать для преобразования и упрощения структуры сложного высказывания. Так, для примера из (п.6.3) имеем:

$$(P \to QR)\ \bar{P}\,Q \Leftrightarrow (\bar{P} \vee QR)\ \bar{P}\,Q \Leftrightarrow \bar{P}\,Q \vee \bar{P}\,QR \Leftrightarrow \bar{P}\,Q.$$

Между отношением равносильности и эквиваленцией формул существует следующая связь: если A и B — равносильны, то $A \sim B$ — тавтология, и обратно, если $A \sim B$ — тавтология, то A и B— равносильны. Это сокращенно записывается так: $\models A \sim B$, если и только если $A \Leftrightarrow B$. Справедливость этого утверждения следует непосредственно из определения равносильности и таблицы истинности для эквиваленции. Действительно, если $A \Leftrightarrow B$, то A может принимать только то значение, что и B и, следовательно, их эквиваленция $A \sim B$ всегда истинна и является тавтологией. Если $A \sim B$ — тавтология, то A и B могут иметь только одинаковые значения (0 или 1) и, следовательно, $A \Leftrightarrow B$.

Из изложенного ясно, что тавтологии можно получить из равносильности заменой знака $\Leftrightarrow$ на $\sim$. Так, из равносильности $A \vee AB \Leftrightarrow A$ получаем тавтологию $\models (A \vee AB) \sim A$. Доказательств тавтологии,

например $\models (A \to B)\ (A \to C) \sim (A \to BC)$ можно выполнить с помощью преобразований:

$$(A \to B)(A \to C) \Leftrightarrow (\overline{A} \vee B)(\overline{A} \vee C) \Leftrightarrow \overline{A} \vee \overline{A}B \vee \overline{A}C \vee BC \Leftrightarrow$$
$$\Leftrightarrow \overline{A} \vee BC \Leftrightarrow A \to BC.$$

6.8. Логическое следствие

Говорят, что формула B является *логическим следствием* формулы A и пишут $A \Rightarrow B$, если B истинно на всех наборах значений переменных, для которых A истинно. Легко убедиться, что $A \Rightarrow B$, если и только если $\models A \to B$. Действительно, в соответствии с определением импликации $A \to B$ ложно только при истинном A и ложном B и, следовательно, если $A \to B$ — тавтология, то из истинности A всегда следует истинность B, т. е. $A \Rightarrow B$. Обратно, если $A \Rightarrow B$, то исключается случай, когда A истинно и B ложно, а значит $A \to B$ истинно на всех наборах значений переменных, т. е. $\models A \to B$.

Логическое следствие $A \Rightarrow B$ означаем что из истинности A следует истинность B, но если A ложно, то относительно B ничего утверждать нельзя. Это отношение обобщается на совокупность высказываний: B есть логическое следствие высказываний $A_1, A_2, ..., A_m$, если из истинности всех A_i $(i = 1, 2, ..., m)$ следует истинность B. Из определения конъюнкции можно заключить, что это сводится к соотношению $A_1 A_2 ... A_m \Rightarrow B$, необходимым и достаточным условием которого является тавтология $\models A_1 \times A_2 ... A_m \to B$.

Пусть, например, даны высказывания $(A \to B)(C \to D)$, $BD \to E$, $\overline{E}$ и необходимо установить, является ли высказывание $\overline{A} \vee \overline{C}$ логическим следствием. Это сводится к доказательству тавтологии $\models ((A \to B)\ (C \to D))\ (BD \to E)\ \overline{E} \to (\overline{A} \vee \overline{C})$. Воспользовавшись методом «обратного рассуждения», положим, что следствие $\overline{A} \vee \overline{C}$ ложно (A и C истинны) при истинном значении всех посылок. Тогда, как следует из первой посылки, B и D должны быть истинны, а из истинности BD и второй посылки следует истинность E. Но это противоречит третьей посылке $\overline{E}$, что и доказывает данную тавтологию.

Между логическим следствием и логической эквивалентностью имеется связь, которая вытекает из соотношения $A \sim B \leftrightarrow (A \to B) \wedge (B \to A)$, приведенного в (3.1.7). Оно означает: $A \sim B$, если и только если $A \to B$ и $B \to A$. Пусть $A \sim B$ — тавтология, тогда $A \to B$ и $B \to A$ — также тавтологии, т. е. $\models A \sim B$, если и только если $\models A \to B$ и

$\models B \to A$. А это равносильно утверждению: $A \Leftrightarrow B$, если и только если $A \Rightarrow B$ и $B \Rightarrow A$.

Логическое следствие есть отношение порядка; так, оно рефлексивно ($A \Rightarrow A$), транзитивно (если $A \Rightarrow B$ и $B \Rightarrow C$, то $A \Rightarrow C$) и антисимметрично (из $A \Rightarrow B$ и $B \Rightarrow A$ следует $A \Leftrightarrow B$).

6.9. Правила вывода

Формальная теория вывода ставит своей главной задачей образование из некоторой совокупности исходных тавтологий новых формул, которые также являются тавтологиями. Эта задача решается с помощью *правил вывода:*

1) если A — тавтология, то, заменяя в ней букву X всюду, где она входит, произвольной формулой B, получаем также тавтологию *(правило подстановки);*

2) если A и $A \to B$ суть тавтологии, то B — также тавтология *(правило заключения).*

Первое из этих правил почти очевидно, а второе непосредственно следует из закона *modus ponens* (6).

Формула называется *выводимой в исчислении высказываний,* если она может быть получена из конечной совокупности исходных формул путем конечного числа шагов применения правил вывода. Вообще говоря, не все тождественно истинные формулы могут быть выведены из произвольного множества тавтологий. В то же время строго доказано, что можно выбрать такую конечную совокупность исходных тавтологий *(аксиом исчисления высказываний)*, из которой выводимы все тождественно истинные формулы. Это важное положение решает проблему *полноты исчисления высказываний.*

Предложено много различных систем аксиом исчисления высказываний. Одна из них включает следующие тавтологии:

1) $A \to (B \to A)$;

2) $((A \to B) \to A) \to A$; 3) $(A \to B) \to ((B \to C) \to (A \to C))$; 4) $AB \to A$;
5) $AB \to B$; 6) $(A \to B) \to ((A \to C) \to (A \to BC))$; 7) $A \to (A \vee B)$;
8) $B \to (A \vee B)$; 9) $(A \to C) \to ((B \to C) \to ((A \vee B) \to C)))$; 10) $(A \sim B) \to$
$\to (A \to B)$; 11) $(A \sim B) \to (B \to A)$; 12) $(A \to B) \to ((B \to A) \to (A \sim B))$;
13) $(A \to B) \to (\bar{B} \to \bar{A})$; 14) $A \to \bar{\bar{A}}$; 15) $\bar{\bar{A}} \to A$.

Выведем, например, тавтологию $AB \to BA$. Подстановка в аксиому (6) вместо A формулы AB даст $\models (AB \to B) \to ((AB \to C) \to (AB \to BC))$, что после подстановки A вместо C приводится к

$\models (AB \to B) \to ((AB \to A) \to (AB \to BA))$. Посылка в этой формуле есть аксиома (5), поэтому на основе правила заключения

$\models (AB \to A) \to (AB \to B\,A)$. Так как посылка в полученной тавтологии является аксиомой (4), то, применяя еще раз правило заключения, получаем $\models AB \to BA$, что и требовалось доказать.

Формализация процесса вывода имеет большое теоретическое значение и позволяет построить схему доказательства, которая может быть реализована на вычислительных машинах. Однако сложность аксиоматического подхода к выводу тавтологий заставляет искать и применять специальные правила, которые сокращают многократное применение основных правил вывода.

6.10. Дедуктивный метод

Более краткий и простой способ вывода основан па *теореме дедукции:* если формула B является логическим следствием формул $A_1, A_2, \ldots, A_m$, т. е. $A_1, A_2, \ldots, A_m \Rightarrow B$, то $\models A_1 \to (A_2 \to (\ldots \to (A_m \to B) \ldots))$. При этом говорят, что формула B *выводима из формул* $A_1, A_2, \ldots, A_m$.

Дадим алгебраическое доказательство теоремы дедукции, рассматривая в соответствии с (8) логическое следствие $A_1, A_2, \ldots, A_m \Rightarrow B$ как $A_1\,A_2 \ldots A_m \Rightarrow B$. Преобразуем по формулам из (7) тавтологию

$$A_1A_2 \ldots A_m \to B \leftrightarrow \overline{A_1A_2 \ldots A_m} \vee B \leftrightarrow \overline{A}_1 \vee \overline{A}_2 \vee \cdots \vee \overline{A}_m \vee B \leftrightarrow \overline{A}_1 \vee$$

$$\vee\, \overline{A}_2 \vee \ldots (A_m \to B) \Leftrightarrow (A_1 \to (A_2 \to (\ldots \to (A_m + B) \ldots))).$$

Так как исходная формула —тавтология, то полученная логически эквивалентная ей формула также является тавтологией, что и требовалось доказать.

Значение теоремы дедукции состоит в том, что логическое следствие B из совокупности посылок $A_1, A_2, \ldots, A_m$ представимо в виде тавтологий типа $\models A_1A_2 \ldots A_p \to (A_{p+1} \to \ldots (A_m + B) \ldots)$. Справедливо и обратное утверждение: если имеется тавтология, содержащая цепочку импликаций типа $(A_1 \to (A_2 \to \ldots (A_p \to (A_{p+1} \to \ldots (A_m \to B) \ldots))))$, то она может быть представлена эквивалентной формулой $\models A_1A_2 \ldots A_p \to (A_{p+1} \to \ldots (A_m \to B) \ldots)$, которой соответствует соотношение $A_1A_2 \ldots A_p \Rightarrow (A_{p+1} \to \ldots (A_m \to B) \ldots)$. Из теоремы дедукции и определения логического следствия вытекают следующие положения:

1) $A_1, A_2, \ldots, A_m \Rightarrow A_i$ $(i = 1, 2, \ldots m)$, т. е. любая из совокупности посылок является логическим следствием этой совокупности:

2) если $A_1A_2 \ldots A_m \Rightarrow B_j$ $(j = 1, 2, \ldots n)$ и $B_1, B_2, \ldots, B_n \Rightarrow B$, то $A_1, A_2, \ldots, A_m \Rightarrow B$.

С помощью этих правил можно представить доказательство того, что формула *B* есть логическое следствие формул A_1, A_2, ..., A_m в виде *цепочки формул,* последней из которых является *B.* Промежуточные формулы B_1, B_2, ..., B_n получаются на основании известных логических законов, аксиом и эквивалентностей. На основе теоремы дедукции используемые тавтологии и результирующее соотношение преобразуются к требуемой форме.

В качестве примера докажем, что $(A \vee B) \rightarrow C$, $C \rightarrow (D \vee E)$, $E \rightarrow F$, $\overline{D}\overline{F} \Rightarrow \overline{A}$. Из первой пары посылок на основе закона силлогизма получаем $(A \vee B) \rightarrow (D \vee E)$. Из последней посылки следует $\overline{D}$ и $\overline{F}$. Из посылки $E \rightarrow F$ и $\overline{F}$ выводим (modus tollens) $\overline{E}$. Из $\overline{D}$ и $\overline{E}$ получаем $\overline{D}\,\overline{E} \Leftrightarrow \overline{D \vee E}$, что совместно с $(A \vee B) \rightarrow (D \vee E)$ в соответствии с modus tollens дает $\overline{A \vee B} \Leftrightarrow \overline{A}\,\overline{B}$, откуда выводим $\overline{A}$. Наглядно этот процесс вывода изображается диаграммой, показанной на рис. 1.

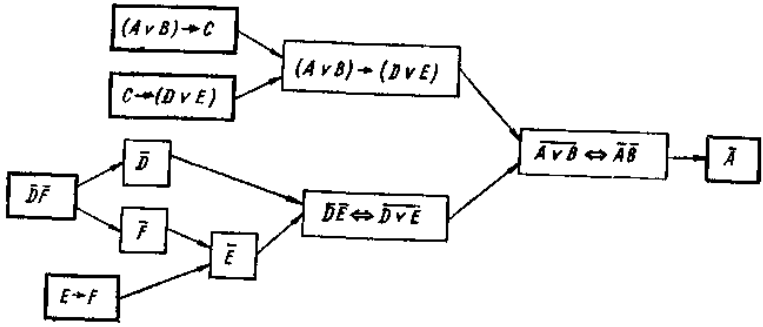


Рис. 1. Диаграмма вывода $\overline{A}$ из посылок $(A \vee B) \rightarrow C$, $C \rightarrow (D \vee E)$, $E \rightarrow F$, $\overline{D}\overline{F}$.

Если в качестве логического следствия выводится конъюнкция некоторого высказывания и его отрицания $A \wedge \overline{A}$, то это свидетельствует о *противоречивости* посылок (из нее выводится произвольное высказывание, как истинное, так и ложное).

6.11. ЛОГИКА ПРЕДИКАТОВ

Обычно высказывания выражают свойства одного или нескольких объектов. Содержательная часть высказывания играет роль

определяющего свойства совокупности объектов, для которых это высказывание истинно, и называется *предикатом.* Например, высказывание «Иванов — отличник» истинно или ложно в зависимости от оценок, которые имеет данный студент. В то же время предикат *«x* — отличник» определяет подмножество отличников на некотором множестве студентов (группа, курс, факультет). Подставив вместо *x* фамилии студентов, получим множество высказываний. Совокупность истинных высказываний и будет соответствовать подмножеству отличников.

Предикат представляет собой логическую функцию *P(x),* принимающую, как и булевы функции, значение 0 или 1, но в отличие от них, значения аргумента *x* выбираются из некоторого множества *M* объектов *(x ∈ M).* В общем случае такая функция может зависеть от многих аргументов $x_1, x_2, ..., x_n$, принимающих значения из одного и того же или различных множеств. Ее записывают $P(x_1, x_2, ..., x_n)$ и называют *n-местным предикатом.* Например: «*x*— четное число», «*x* — компонента коммуникационной цепи» —одноместные предикаты *P(x); «x* брат *y», «x* меньше *y»* — двуместные предикаты *P(x, y);* *«x* и *y* — родители z», «*x* — сумма *y* и z» — трехместные предикаты *P(x, y,* z) и т. д. Если аргументы $x_1, x_2, ..., x_n$ замещены конкретными значениями (объектами), то предикат переходит в высказывание, которое рассматривают как 0-*местный предикат.*

Так как предикаты способны принимать только значения 0 и 1, то их, как и булевы переменные, можно связывать логическими операциями. В результате получаем формулы, определяющие более сложные предикаты. Так, если *P(x)* означает *«x* — инженер», а *Q(x)* — «*x* — сотрудник нашего отдела», то $P(x) \wedge Q(x) = R(x)$ есть одноместный предикат *«x* — инженер и сотрудник нашего отдела» или проще *«x* — инженер нашего отдела». Очевидно, если *P* — множество инженеров, а *Q* — множество сотрудников данного отдела, то этот предикат соответствует пересечению $P \bigcap Q$. Таким образом, имеет место тесная связь между логикой предикатов и операциями над множествами.

6.12. Высказывания и предикаты

В то время как логика высказываний проявляет интерес только к логической связи между предложениями, логика предикатов проникает и в структуру самих предложений в смысле связи того, о ком или о чем идет речь *(субъект)* с тем, что говорится о данном предмете *(предикат).* Поэтому язык логики предикатов лучше приспособлен для

выражения логических связей между различными понятиями, утверждениями, коммуникационными предписаниями и директивами.
Как указывалось выше, *n-местный предикат* $P(x_1, x_2, ..., x_n)$ является неоднородной двузначной логической функцией. Аргументы
$x_1, x_2, ..., x_n$ представляют собой объекты из множеств их определения $X_1, X_2, ..., X_n$, т. е. $x_1 \in X_2, x_2 \in X_2, ..., x_n \in X_n$ и называются *предметными переменными.* Конкретные значения аргументов называют *предметными постоянными.* Предметные переменные и предметные постоянные образуют класс логических понятий, называемых *термами.*
При замещении аргумента x_k (предметной переменной) некоторым его значением *a* (предметной постоянной) *n*-местный предикат
$P(x_1, x_2, ..., x_n)$ превращается в *(n* — 1)-местный предикат $P(x_1 .., x_{k-1}, a, x_{k+1}, ... , x_n)$ и от переменной x_k он уже не зависит. Приписав значения всем переменным $x_1, x_2, ..., x_n$ из соответствующих областей определения, мы получим высказывание, которое можно рассматривать как 0-*местный предикат.*
Например, трехместный предикат $P(x_1, x_2, x_3)$ = «x_1 есть сумма x_2 и x_3» при подстановке $x_1 = 5$ переходит в двуместный предикат $P(5, x_2, x_3)$ = «5 есть сумма x_2 и x_3», а при дальнейшей подстановке $x_2 = 2$ — в одноместный предикат P $(5, 2, x_3)$ = «5 есть сумма 2 и x_3». Очевидно, при $x_3 = 3$ он становится истинным высказыванием, а при всех $x_3 \neq 3$ ложным высказыванием.

6.13. Кванторы

В логике предикатов большое значение имеют две операции, называемые *кванторами,* с помощью которых выражают отношения общности и существования. Пусть *P(x)* — предикат, определенный на множестве *M.* Утверждение, что все $x \in M$ обладают свойством *P(x),* записывают с помощью *квантора общности* $\forall x$ в виде $\forall xP(x)$, что читается «для всех *x, P* от *x*». Утверждение, что существует хотя бы один объект *x* из *M,* обладающий свойством *P(x),* записывают с помощью *квантора существования* $\exists x$ в виде $\exists xP(x)$, что читается «существует такое *x,* что *P* от *x».*
Хотя в выражениях $\forall xP(x)$ и $\exists xP(x)$ и встречается буква *x,* но они не зависят от значений этой переменной. Кванторы $\forall x$ и $\exists x$ *связывают переменную x,* превращая одноместный предикат в высказывание. Очевидно, $\forall xP(x)$ истинно только при условии, что *P(x)* тождественно истинный предикат, а во всех остальных случаях

это высказывание ложно. Высказывание $\exists x P(x)$ всегда истинно, кроме единственного случая, когда $P(x)$ — тождественно ложный предикат.
Рассмотрим, например, предикат $P(x)$=«x — простое число», определенный на множестве натуральных чисел. Подставляя вместо x числа натурального ряда, получаем счетное множеаво высказываний. Некоторые из них, например P (1), P (2), P (3), P (5) и т. д., являются истинными. Высказывание $\forall x P(x)$ — «все натуральные числа простые» — ложно, а $\exists x P(x)$ — «некоторые из натуральных чисел — простые» — истинно.
Между кванторами $\forall x$ и $\exists x$ имеют место соотношения, обобщающие законы де Моргана: $\forall x \overline{P(x)} = \exists x P(x)$; $\overline{\exists} x P(x) = \forall x \overline{P(x)}$.

6.14. Связанные и свободные переменные

Применение квантора к n-местному предикату превращает его в $(n$—1)-местный предикат. Кванторы можно также применять к нескольким различным переменным (по одному квантору какого-либо типа к каждой переменной). Если к n-местному предикату применяется k кванторов, ю он превращается в $(n - k)$-местный предикат, а при $n=k$ — в высказывание. Переменные, к которым применяются кванторы, называются *связанными,* а остальные переменные — *свободными.* Например, из двухместного предиката $P(x, y)$ с помощью кванторов получаем одноместные предикаты $\forall x P(x, y)$; $\exists x P(x, y)$; $\forall P(x, y)$ и $\exists y P(x, y)$, а также высказывания $\forall x \forall y P(x, y)$; $\forall x \exists y P(x, y)$; $\exists x \exists y P(x, y)$ и т. п.
Порядок следования одноименных кванторов не имеет значения, но разноименные кванторы переставлять нельзя. Так, $\forall x \forall y P(x, y)$ эквивалентно $\forall y \forall x P(x, y)$;, но высказывания $\forall x \exists y P(x, y)$; и $\exists y \forall x P(x, y)$, вообще говоря, различны. В этом можно убедиться на примере предиката $P(x, y)$ = «x делит y», который в первом случае превращается в высказывание «для всякого x существует такое y, что x делит y» (истинно), а во втором — «существует такое y, что любое x делит y» (ложно).
Квантор связывает переменную в области своего действия. Эта область обычно заключается в скобки, если она содержит не один предикат, а совокупность предикатов, связанных символами логических операций. Выражения, которые можно образовать применением к предикатам сентенциональных связок и кванторов, представляют собой *формулы логики предикатов.* Переменная *свободна в формуле,* если хотя бы на

одно ее вхождение не распространяется действие квантора. Переменная *связана в формуле,* если она связана по меньшей мере одним квантором. Например, в формуле $\exists y \forall x P(x, y) \rightarrow \forall z Q(z))$ вхождение каждой из переменных связано, а в формуле $\forall x(P(x, y) \vee \exists y\ Q(y)) \vee R(x)$ переменная *x* одновременно и свободная и связанная.

6.15. Категорические высказывания

Перевод предложений с русского или какого-либо другого языка на символический язык логики предикатов вызывает определенные трудности из-за отсутствия механических правил. Он основан не столько на форме обычных предложений, сколько на выявлении их смысловой связи.

В традиционной логике большое внимание уделяется четырем типам *категорических высказываний,* которые обычно обозначаются заглавными латинскими буквами *A, E, I, O:*

A — общеутвердительное высказывание «Всякое *S* суть *P»:*

$\forall\ x(S(x) \rightarrow P(x))$, что означает: «Для всех *x,* если *x* обладает свойством S, то *x* обладает и свойством *P»;*

E — общеотрицательное высказывание «Никакое *S* не есть *P»:*

$\forall\ x(S(x) \rightarrow P(x))$, что означает «Для всех *x,* если *x* обладает свойством *S*, то он не обладает свойством *P»;*

I — частноутвердительное высказывание «Некоторые S суть *P»:*

$\exists x(S(x) \wedge P(x))$, что означает «Существует такой объект *x,* обладающий свойством *S*, который также обладает свойством *P»;*

O — частноотрицательное высказывание «Некоторые S не суть *P»:*

$\exists x\ (S(x) \wedge \overline{P(x)})$, что означает «Существует такой объект *x,* который обладает свойством *S* и не обладает свойством *P*».

Пусть, например, *S(x)* = «*x* — селедка» (свойство «быть селедкой») и *P(x)* = «*x* — рыба» (свойство «быть рыбой»). Тогда четырем типам категорических высказываний соответствуют следующие утверждения: *A* = «Всякая селедка — рыба»; *E* = «Никакая селедка не является рыбой»; *I* = «Некоторые селедки — рыбы»; *O* = «Некоторые селедки не являются рыбами».

На основе правил преобразования высказываний (п.3.1.7) и зависимостей между кванторами (2) можно записать: $\forall\ x(S(x) \rightarrow P(x)) \Leftrightarrow \neg\overline{\exists} x\ (\overline{S(x)} \vee P(x)) \Leftrightarrow \overline{\exists} x\ (S(x) \wedge \overline{P(x)})$. Аналогично преобразуются и другие типы высказываний, в результате чего получаем зависимости:

$$\forall x\,(S(x) \to P(x)) \leftrightarrow \overline{\exists} x\,(S(x) \wedge \overline{P(x)});$$
$$\forall x\,(S(x) \to \overline{P(x)}) \leftrightarrow \overline{\exists} x\,(S(x) \wedge P(x));$$
$$\overline{\forall} x\,(S(x) \to \overline{P(x)}) \leftrightarrow \exists x\,(S(x) \wedge P(x));$$
$$\overline{\forall} x\,(S(x) \to P(x)) \leftrightarrow \exists x\,(S(x) \wedge \overline{P(x)}).$$

Как видно из приведенных равносильностей, высказывания *A* и *O,* а также *E* и *I* являются отрицаниями друг от друга (если одно из них истинно, то другое ложно и обратно) и называются *противоположными.* Из коммутативности операции конъюнкции следует, что суждения *E* и *I* допускают перестановку предикатов *S(x)* и *P(x),* т. е.

$$\overline{\exists} x\,(S(x) \wedge P(x)) \leftrightarrow \overline{\exists} x\,(P(x) \wedge S(x));$$
$$\exists x\,(S(x) \wedge P(x)) \leftrightarrow \exists x\,(P(x) \wedge S(x)).$$

6.16. Непосредственные заключения

Приняв одно из категорических высказываний в качестве посылки, а другое — в качестве следствия, можно построить так называемые *непосредственные заключения.* Истинность или ложность заключения зависит только от его формы или, как говорят, от его *модуса.*

Обычно категорические высказывания сокращенно обозначают совокупностью трех букв *SaP, SeP, SiP, SoP,* где *a, c, i, o* указывают на тип высказывания *(A, E, I, O); S* и *P — понятия,* означающие свойства (в таком порядке, в каком они входят в высказывание). Например, непосредственное заключение уж $\forall\ x(S(x) \to P(x)) \to \exists x\ (P(x) \wedge S(x))$ в принятых обозначениях запишется как $SaP \to SiP$.

Простой анализ показывает, что *SiP* является логическим следствием *SaP,* а *SoP* — логическим следствием *SeP.* Высказывания *SaP* и *SeP* могут одновременно быть ложными, но не истинными и поэтому называются *противоречивыми.* Высказывания *SiP* и *SoP* могут быть одновременно истинными, но не ложными и поэтому называются *антипротиворечивыми.*

Традиционная схема отношений между категорическими высказываниями, называемая *логическим квадратом,* показана на рис. 255.

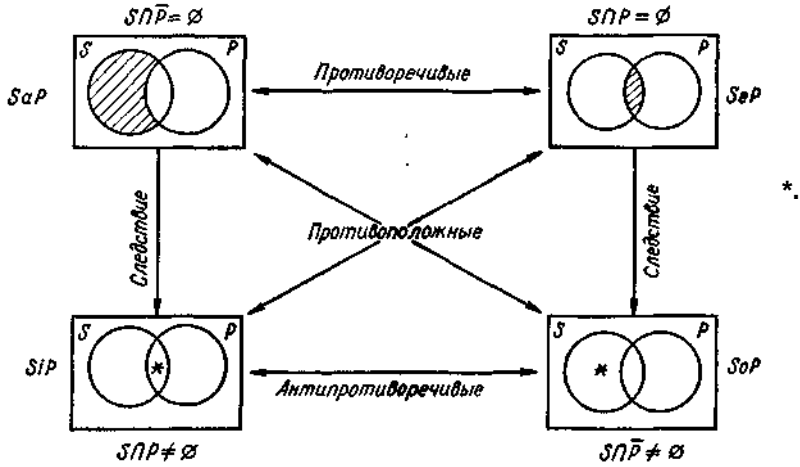


Рис. 2. Логический квадрат.

Там же приведены диаграммы Венна для каждого из четырех типов высказываний. Они непосредственно вытекают из правых частей выражений в (4) и теоретико-множественной интерпретации логических операций над предикатами, причем заштрихованные области соответствуют пустым множествам, а отмеченные звездочкой (*) — непустым множествам. Так как $S \cap \overline{P} = \varnothing$, если и только если $S \subset P$, то высказывание *SaP* соответствует отношению включения множеств $S \subset P$. В случае высказывания *SeP* множества S и *P* являются непересекающимися, а в случае высказывания *SiP* множества S и *P* должны иметь непустую общую часть. Наконец, высказывание *SoP* в силу тождества $S \cap \overline{P} = S \backslash P$ соответствует дополнению S до *P*.

Поскольку каждый из четырех типов высказываний может быть как посылкой, так и следствием, то всего можно образовать 4 • 4 = 16 модусов непосредственных заключений с одинаковым положением понятий *S* и *P* в посылках и следствиях. Изменив порядок следования понятий в следствиях *(SP* на *PS),* получим еще 16 модусов. Итак, имеется всего 32 существенно различных модусов непосредственных заключений. Анализ (например, с помощью диаграмм Венна) показывает, что только 10 из них являются тавтологиями, т. е. *правильными модусами.* Кроме четырех модусов, в которых посылки и следствия являются одинаковыми высказываниями, и двух модусов, допускающих перестановку понятий *(SeP* $\Rightarrow$ *PeS, SiP* $\Rightarrow$ *PiS),* правильными являются модусы: *SaP* $\Rightarrow$ *SiP; SeP* $\Rightarrow$ *SoP; SaP* $\Rightarrow$ *PiS; SeP* $\Rightarrow$ *PoS.*

К таким выводам приходим, если, следуя традиционной формальной логике, считать, что понятия *S* и *P* всегда соответствуют непустым множествам, т. е. предикаты *S(x)* и *P(x)* не могут быть тождественно ложными. Если же, например, *S(x)*—тождественно ложно (*S*= $\varnothing$), то высказывания *SaP* и *SeP* всегда истинны, а *SiP* и *SoP* — ложные (это хорошо видно из рис. 2). Тем самым нарушается правильность ряда модусов традиционной логики.

Пусть, например, *S(x)* = «*x* — летающие черепахи», а *P(x)* означает «жить в зоопарке». Тогда категорические высказывания четырех типов суть следующие:

SaP = «Все летающие черепахи живут в зоопарке»,

SeP = «Никакие летающие черепахи не живут в зоопарке»,

SiP = «Некоторые летающие черепахи живут в зоопарке»,

SoP = «Некоторые летающие черепахи не живут в зоопарке».

Первые два высказывания истинны, что ясно из их эквивалентного представления: *SaP* = «Не существует такого объекта *x,* который был бы летающей черепахой и не жил в зоопарке» и *SeP* = «Не существует такого объекта *x,* который был бы летающей черепахой и жил в зоопарке». Истинность этих высказываний следует уже из того, что действительно «не существует такого объекта, который был бы летающей черепахой», т. е. в силу тождественной ложности предиката *S(x).* По этой же причине ложными являются два других высказывания *SiP* и *SoP.*

Ясно, что при тождественно ложном *S(x)* высказывание *I* не является следствием *A* и высказывание *O* не является следствием *E,* т. е. модусы $SaP \Rightarrow SiP;\ SeP \Rightarrow SoP;\ SaP \Rightarrow PiS;\ SeP \Rightarrow PoS$ перестают быть правильными. Теряют смысл и некоторые отношения между высказываниями, изображенные на логическом квадрате. Традиционная логика находит выход из этого положения, не допуская тождественно ложных предикатов, а значит, и пустых множеств. Но современная логика предикатов не может встать на такую точку зрения, которая сильно сузила бы область ее применения. О целесообразности допущения пустых множеств уже говорилось ранее. Рассматривая пустое множество как подмножество любого множества, мы не нарушаем теоретико-множественных соотношений для различных типов категорических высказываний (рис. 2).

6.17. Категорические силлогизмы

Так называтсуждения типа $XY \rightarrow Z$, где *X, Y* и Z — категорические высказывания. Из истинности конъюнкции *XY* (она истинна только при

истинных *X* и *Y*) на основании modus ponens можно выводить истинность высказывания Z. Если $\models XY \rightarrow Z$, то $XY \Rightarrow Z$ — *правильный силлогизм.*

Во всяком силлогизме *X* — *большая посылка,* содержащая понятия *M* и *P*; *Y* — *малая посылка,* содержащая понятия *M* и S, и Z — *заключение,* в котором S играет роль *подлежащего* и *P* — *сказуемого.* Таким образом, в силлогизме участвуют три понятия, называемые: S — *малый термин, M — средний термин* и *P — большой термин,* причем некоторое суждение от *S* и *P* выводится из двух высказываний — посылок, в которых участвует средний термин *M,* отсутствующий в заключении. Например, *MaP • SaM → SaP* означает: «Если все *M* суть *P* и все S суть *M,* то все S суть *P*», что принято записывать в виде:

Все *M* суть *P*
Все *S* суть *M*
Все *S* суть *P*

В зависимости от порядка следования понятий в посылках совокупность силлогизмов распадается на четыре группы, называемые *фигурами* силлогизмов:

$$\frac{MP}{\underline{SM}}; \quad \frac{PM}{\underline{SM}}; \quad \frac{MP}{\underline{MS}}; \quad \frac{PM}{\underline{MS}}.$$
$$\overline{SP} \qquad \overline{SP} \qquad \overline{SP} \qquad \overline{SP}$$

В данной фигуре каждое из высказываний может относиться к одному из четырех типов *A, E, I, O,* поэтому из нее можно образовать $4^3 = 64$ модуса, а общее количество модусов для всех четырех фигур равно $64 • 4 = 256$. Основная задача теории силлогизмов состоит в выделении множества правильных модусов, т. е. таких, которые при любых конкретных понятиях позволяют из истинных посылок делать истинные заключения. Можно доказать, что из 256 модусов правильными являются только 15. Для наименования правильных модусов применяются слова, содержащие три из четырех букв *a, e, i, o,* которые указывают последовательно на типы высказываний посылок и заключения. Они выглядят (по фигурам) следующим образом:

	Barbara	Celarent	Darii	Ferio
1)	MaP SaM $\overline{SaP}$	MeP SaM $\overline{SeP}$	MaP SiM $\overline{SiP}$	MeP SiM $\overline{SoP}$
	Cezare	Camestres	Festino	Baroco
2)	PeM SaM $\overline{SeP}$	PaM SeM $\overline{SeP}$	PeM SiM $\overline{SoP}$	PaM SoM $\overline{SoP}$
	Datisi	Feriso	Disamis	Bocardo
3)	MaP MiS $\overline{SiP}$	MeP MiS $\overline{SoP}$	MiP MaS $\overline{SiP}$	MoP MaS $\overline{SoP}$
	Calemes	Fresison	Dimatis	
4)	PaM MeS $\overline{SiP}$	PeM MiS $\overline{SoP}$	PiM MaS $\overline{SiP}$	

Традиционная логика признавала правильными еще девять модусов, которые имеют место при условии, что понятиям соответствуют непустые множества объектов. Правильность модусов доказывается на основе законов логики высказываний. Так, для модуса Celarent имеем:

$\forall x\,(M(x) \to \overline{P(x)})) \;\; \forall x\,(S(x) \to M(x)) \Leftrightarrow \forall x((S(x) \to M(x)) \wedge (M(x) \to$

$\to \overline{P(x)}))$. Согласно закону силлогизма $((A \to B)\,(B \to C)) \Rightarrow (A \to C)$, если для всякого x выражение $(S(x) \to M(x))\,(M(x) \to \overline{P(x)})$ истинно, то истинно и выражение $S(x) \to \overline{P(x)}$. Таким образом, в сокращенной записи имеем $MeP \bullet SaM \Rightarrow SeP$, что и представляет собой силлогизм Celarent. Аналогично доказываются и другие правильные силлогизмы. Придавая понятиям *S, M, P* конкретное содержание, из истинных посылок всегда будем получать истинные заключения. Например, в соответствии с модусом Festino имеем:

Никакие черепахи не летают
Некоторые животные летают
Некоторые животные — не черепахи

В то время как правильность модуса требует строгого доказательства, для установления неправильности какого-либо модуса достаточно привести опровергающий его контрпример. Так, модус
$MaP \bullet MiS \to SaP$ опровергается ложным суждением:

Всякое четное число делится на 2
Некоторые четные числа — простые
Всякое простое число делится на 2

Правильный вывод из приведенных посылок «Некоторые простые числа делятся на 2» (множество таких простых чисел содержит единственный элемент 2) следует в соответствии с модусом Datisi.

6.18. Символизация языка

Логика предикатов располагает более общими и универсальными методами обоснования правильных выводов, чем традиционная формальная логика. Первым этапом построения какого-либо доказательства или теории является символизация исходных положений, подвергающихся логическому анализу или принимаемых в качестве аксиом данной теории. Этот процесс обычно сводится к переводу некоторых высказываний на символический язык логики предикатов. Приведем некоторые примеры.

Рассмотрим сложное высказывание, выраженное на обычном языке: «Некоторые студенты выполнили все задания. Ни один студент не выполнял графиков. Следовательно, ни одно задание не являлось графиком». В первом предложении участвуют одноместные предикаты — свойства *P(x)* = «*x* — студент», *Q(y)* = «*y* — задание» и двуместный предикат *R(x, y)* = «*x* — выполнил *y*». Так как в нем говорится о «некоторых студентах», то соответствующая форма будет $\exists x(P(x) \wedge A(x))$, где *A(x)* — сложное высказывание, характеризующее предикат *P(x),* а именно: «выполнили все задания». Поскольку речь идет о «всех заданиях», то переменная *y* связывается квантором общности и высказывание *A(x)* представляется формулой $\forall y(Q(y) \rightarrow R(x, y))$, которая дословно переводится «для всякого *y,* если *y* — задание, то *x* выполнили *y*», смысл которого соответствует фразе «выполнили все задания». Итак, символическая запись первого предложения имеет вид: $\exists x(P(x) \wedge \forall y(Q(y) \rightarrow R(x, y)))$. Аналогично записывается и второе предложение $\forall x(P(x) \rightarrow \forall y\{S(y) \rightarrow \overline{R(x, y)})\}$, где *S(y)* = «*y* — график». Заключение «Ни одно задание не являлось графиком» представляет собой категорическое высказывание типа *QeS.* Таким образом, получаем окончательно:

$\exists x(P(x) \wedge \forall y(Q(y) \rightarrow R(x, y))) \wedge \forall x(P(x) \rightarrow \forall y(S(y) \rightarrow R(x, y))) \rightarrow$
$\rightarrow \forall x\,(Q(x) \rightarrow \overline{S(x)})$.

Рассмотрим примеры символической записи свойств и определений. Пусть *P(x, y)* — бинарное отношение, определенное на некотором

множестве *M*. Рассматривая его как двуместный предикат, записываем основные свойства отношений: $\forall x P(x,x)$ — рефлексивность, $\forall x \forall y (P(x,y) \rightarrow (P(y, x))$ — симметричность, $\forall x \forall y \forall z (P(x,y) \wedge \wedge P(y, z) \rightarrow P(x, z))$ — транзитивность, $\forall x \forall y (P(x,y) \wedge P(y,x) \rightarrow (x = y))$ — антисимметричность и т. д. С помощью этих и подобных им выражений определяются любые типы бинарных отношений, обладающих некоторой совокупностью свойств. Так, отношение эквивалентности определяется как двуместный предикат, удовлетворяющий формуле: $\forall x P(x, x) \wedge \forall x \forall y (P(x, y) \rightarrow \rightarrow P(y, x)) \wedge \forall x \forall y \forall z (P(x, y) \wedge P(y, z) \rightarrow P(x, z))$.

Язык логики предикатов широко используется в теории коммуникаций. Поэтому необходимо научиться уверенно расшифровывать формулы, записанные на этом языке. Пусть, например, $\forall x (P(x) \rightarrow \exists y (Q(y) \wedge R(x, y)))$, где $P(x)$ = «*x* — простое число», $Q(x)$ = «*x* — четное число», $R(x, y)$ — «*y* делится на *x*». Это общеутвердительное высказывание, в котором $P(x)$ играет роль подлежащего, а $\exists y (Q(y) \wedge R(x, y))$ — сказуемого. В свою очередь, сказуемое является частноутвердительным высказыванием относительно переменной *y* (*x* — свободная переменная) и означает: «Существует такое четное число *y*, которое делится на *x*». Тогда исходная формула расшифровывается следующим образом: «Для всех *x*, если *x* — простое число, существует такое четное число *y*, которое делится на *x*» или проще: «Для всякого простого числа можно подыскать такое четное число, которое делится на это простое число».

6.19. Оценочная процедура

Истинное значение формулы в логике предикатов можно установить с помощью *оценочной процедуры*. Она сводится к определению значений входящих в данную формулу предикатов при замещении свободных переменных элементами из множества их определения. При этом последовательно используются общие свойства сентенциональных связок и кванторов. Исходными данными являются неоднородные функции, представляющие предикаты, и конкретные значения высказываний и свободных переменных, для которых требуется найти значение формулы.

Проиллюстрируем рассматриваемую процедуру на примере формулы $\forall x (P(x, y, z) \rightarrow \exists y Q(x, y)) \vee Q(x, y) S$, где предикаты заданы на двухэлементном множестве $\{a, b\}$ таблицами соответствия:

$P(x, y, z)$

	y	a	a	b	b
	z	a	b	a	b
x	a	0	1	1	0
	b	0	1	0	1

$Q(x, y)$

	y	a	b
x	a	0	0
	b	0	1

Пусть S = 0; $x = b$; $y = a$; $z = a$. Подставляя эти значения в формулу, получаем $\forall x(P(x, a, a) \rightarrow \exists y Q(x, y)) \vee Q(b, a) \bullet 1$. Так как $Q(b, a) = 0$, то формула упрощается к виду $\forall x(P(x, a, a) \rightarrow \exists y(Q(x, y))$. Это выражение представляет собой высказывание, для установления значения которого необходимо выяснить, является ли одноместный предикат в скобках истинным для всех значений x. Соответствующая таблица имеет вид:

x	$P(x, a, a) \rightarrow \exists y Q(x, y)$	
a	0	0
b	0	1

Здесь значения $P(x, a, a)$ взяты из первого столбца таблицы для $P(x, y, z)$. Значения $\exists y Q(x, y)$ получены на основе таблицы для $Q(x, y)$. Так как первая ее строка содержит только нули, то $\exists y Q(x, y)$ при $x = a$ получает значение 0. Во второй строке имеется единица, откуда заключаем, что $\exists y Q(x, y)$ при $x = b$ имеет значение 1. Истинностные значения выражения $P(x, a, a) \rightarrow \exists y Q(x, y)$ помещены в таблице под знаком импликации (так часто поступают для сокращения места).

Как видим, это выражение тождественно истинно относительно переменной x, следовательно, $\forall x(P(x, a, a) \rightarrow \exists y Q(x, y))$ также истинно, т. е. исследуемая формула имеет значение 1. Аналогично можно определить истинностные значения формулы и при других значениях переменных x, y, z и высказывания S. Рассмотренная процедура трудоемка даже для сравнительно простых формул, особенно, если требуется найти истинностные значения на всевозможных наборах (при этом необходимо выполнить эту процедуру для всех функций $P(x, y, z)$ и $Q(x, y)$).

6.20. Общезначимость

Особый интерес представляют общезначимые формулы, которые истинны (принимают значения 1) при каждом приписывании значений входящих в них свободных переменных и предикатов. Если A — общезначимая формула, то она, как и тавтологии, обозначается $\models A$.

Для доказательства общезначимости формул используется аппарат логики высказываний, дополненный теоремами для выражений, содержащих кванторы. Приведем некоторые из них.

1) Пусть $Q(x)$ — формула, свободная для y; тогда:

а) $\models \forall x Q(x) \to Q(y)$; *б)* $\models Q(y) \to \exists x Q(x)$;

2) Пусть R — формула, не содержащая свободных вхождений переменной x, и $Q(x)$ — какая-либо формула; тогда: а) если $\models R \to Q(x)$, то $\models R \to \forall x Q(x)$; б) если $\models Q(x) \to R$, то $\exists x Q(x) \to R$.

3) $\models Q(x)$, если и только если $\models \forall x Q(x)$ (следствие из теорем 1 и 2).

На основе этих теорем строятся правила вывода, которые, наряду с правилами исчисления высказываний (правила подстановки и заключения, теорема дедукции и др.), используются для доказательства логических следований.

Правило универсальной конкретизации (УК): из $\forall x Q(x)$, которая свободна для y, выводится $Q(y)$ подстановкой в $Q(x)$ вместо x переменной y (теорема 1 а).

Правило универсального обобщения (УО): если $Q(x)$— следствие посылок, ни одна из которых не имеет свободных вхождений x, то из нее выводится $\forall x Q(x)$ (теорема 2 а).

Кроме того, можно использовать еще два правила, представляющие собой аналоги приведенных выше правил для квантора существования.

Правило экзистенциальной конкретизации (ЭК) позволяет перейти от $\exists x P(x)$ к $P(\alpha)$, где α — неизвестный, но вполне определенный элемент такой, что, если $\exists x P(x)$ истинно, то $P(\alpha)$ также истинно.

Правило экзистенциального обобщения (ЭО) позволяет перейти от $P(\alpha)$ к $\exists x P(x)$, т. е., если существует такое α, что $P(\alpha)$ истинно, то истинно и $\exists x P(x)$.

В логику предикатов полностью переносятся все тавтологии, в частности соотношения: а) $\models A \sim B$, если и только если $A \Leftrightarrow B$;

б) $\models A \to B$, если и только если $A \Rightarrow B$.

6.21. Доказательство логического следствия

Исходя из понятия общезначимости, можно дать следующее определение *логического следствия в логике предикатов:* формула B есть логическое следствие формул $A_1, A_2, ..., A_m$, т. е. $A_1, A_2, ..., A_m \Rightarrow B$, если для каждого множества определения и для каждого приписывания формулам A_i $(i = 1, 2, ..., m)$ в этом множестве формула B истинна при условии, что все A_i истинны. При этом для всех свободных вхождений некоторой переменной x в какие-нибудь A_i выбирается одно и то же

значение x из множества определения, т. е. такое x по существу рассматривают как постоянную.

Следуя общей схеме рассуждений, изложенной в (3.1.10), а также дополнительным правилам вывода (9), рассмотрим пример из (7), где $\exists x(P(x) \wedge \overline{R(x,y)})\, y(Q(y) \rightarrow R(x,y)))$ и $\forall x\ (P(x) \rightarrow \forall y(S(y) \rightarrow \overline{R(x,y)}))$ — посылки и $\forall x(Q(x) \rightarrow \overline{S(x)})$ — заключение. Процесс доказательства представляется диаграммой, показанной на рис. 3.

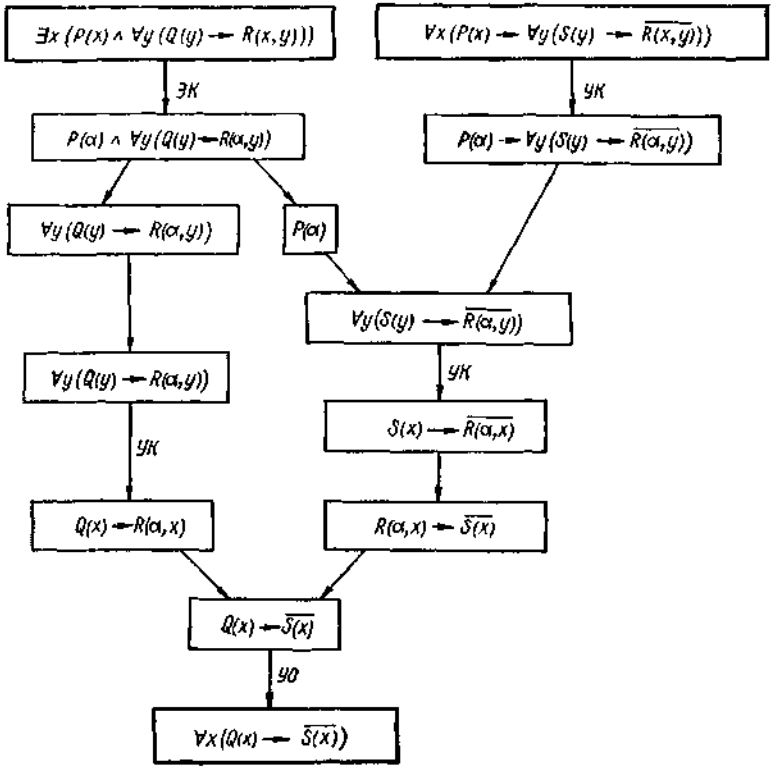


Рис.3. Диаграмма вывода $\forall x(Q(x) \rightarrow \overline{S(x)})$ из посылок

$\exists x(P(x) \wedge \overline{R(x,y)})\, y(Q(y) \rightarrow R(x,y)))$ и $\forall x(P(x) \rightarrow \forall y(S(y) \rightarrow \overline{R(x,y)}))$.

Применение правил вывода, специфических для логики предикатов, указано здесь их сокращенными обозначениями. Остальные правила заимствованы из логики высказываний.

6.22. Моделирование внешней среды

Любая система, вырабатывающая целесообразное поведение, — будь то человек, животное или робот, — всегда использует знания о своих собственных функциональных возможностях и о свойствах внешней среды. О таких системах принято говорить, что они обладают «внутренней моделью внешнего мира». Наличие модели среды позволяет быстро и глубоко проникать в механизмы внешних явлений, намного облегчает процессы обучения на опыте и адаптации.

Каковы же особенности и способы представления знаний о внешнем мире в памяти интеллектуального объекта ? Прежде всего заметим, что способность к моделированию внешней среды (включая свое собственное «я») присуща лишь интеллектуальным объектам.

Окружающий нас мир настолько сложен, что не только интеллектуальнй объект, но и человек бывает доволен, если ему удается уловить и понять хотя бы некоторые самые простые из присущих этому миру закономерностей. Для этого человек строит упрощенные и идеализированные модели, освобожденные от маловажных подробностей и отражающие, как он надеется, наиболее существенные свойства рассматриваемых реальных объектов. По такой же схеме должен действовать и интеллектуальный объект, желающий создать в себе адекватную модель внешнего мира.

Восприятие внешнего мира осуществляется с помощью искусственных органов чувств интеллектуального объекта. Это значит, что реальные ситуации описываются в памяти интеллектуального объекта с помощью набора показаний сенсорных датчиков. Именно в терминах этих показаний — элементарных высказываний о свойствах среды и самого интеллектуального объекта — и формируется «внутренняя модель внешнего мира».

Однако совокупность показаний сенсорных датчиков — это «сырая» информация, которую мы будем называть первичным описанием реальной ситуации. Анализ и обработка этой информации, как мы видели в предыдущем разделе, позволяют строить обобщенные описания ситуаций, называемые понятиями. В эти информационные представления о мире могут входить не только показания органов чувств (например, «красное», «горячее», «твердое» и т. п.), но и утверждения о соотношениях между этими показаниями. Одним из

наиболее удобных средств для формирования и обработки этих представлений является описанное выше исчисление предикатов; вместе с адаптивной системой логического вывода. На языке исчисления предикатов свойства внешнего мира задаются с помощью элементарных предикатов и функций от показаний сенсорных датчиков, а также с помощью логических средств формирования из этих элементов сложных утверждений (описаний) в зависимости от имеющегося опыта и поступающей информации. В двух предыдущих пунктах мы видели, каким образом осуществляется уточнение (коррекция) и расширение представлений робота о внешнем мире в процессе обучения на опыте и адаптации к существующим (заранее неизвестным) условиям.

Моделирование внешнего мира в памяти интеллектуального объекта не является самоцелью. Оно служит главным образом для «мысленного» планирования поведения и принятия решений о тех или иных целенаправленных действиях интеллектуального объекта. При этом весьма важно, чтобы в модели внешнего мира был отражен сам интеллектуальный объект, его структурные и функциональные свойства, особенности взаимодействия с окружающей средой. Благодаря этому интеллектуальный объект получает возможность анализировать не только окружающую среду, в которой он функционирует, но и свое поведение в этой среде. Мы можем смело сказать, что интеллектуальный объекта обладает «сознанием» и «самосознанием». Уточним эти интуитивно понятные термины.

Следуя терминологии, предложенной Д. А. Поспеловым, «сознанием» интеллектуального объекта мы будем называть его способность отображать (моделировать) внешнюю среду в своей памяти и анализировать закономерности этой среды, а также результаты своих воздействий на среду. Под «самосознанием» будем понимать свойство интеллектуального объекта отображать себя в модели среды и анализировать закономерности воздействия среды на свою структуру и функционирование. Именно «сознание» и «самосознание» принципиально отличают интеллектуальных объектов.

6.23. Алгоритмы построения программных движений

Задача построения программных движений (ПД) робота решается на пятом уровне иерархии его системы управления. Особенность этой задачи заключается в том, что движение исполнительных механизмов робота задается законом изменения обобщенных координат, а происходит в реальной внешней среде. При этом допустимость тех или

иных конфигураций исполнительных механизмов непосредственно наблюдается в реальной же среде и никаким простым способом не выражается «на языке» пространства обобщенных координат. Другой особенностью является наличие двигательной избыточности, присущей роботам с большим числом степеней свободы. Эта избыточность, однако, полезна. Она позволяет среди множества возможных ПД отобрать наиболее «экономные», оптимальные программные движения.

Рассмотрим кратко некоторые идеи и алгоритмы построения ПД. Эти алгоритмы служат для того, чтобы робот мог, еще не совершая каких-либо реальных движений, «мысленно» рассчитать, а если нужно — скорректировать программу своих целенаправленных движений. В качестве примера рассмотрим задачу управления многозвенным манипулятором. Цель управления в этом случае заключается в отслеживании схватом некоторой траектории, рассчитанной на более высоком уровне планирования поведения. Задача осложняется наличием разного рода препятствий и конструктивных ограничений.

Программным движением называется такой закон изменения обобщенных координат, при котором достигается цель управлений, удовлетворяются конструктивные ограничения и обеспечивается обход препятствий.

Идея «глобального» подхода к построению ПД заключается в следующем. Вначале на основе информации о траектории схвата с учетом конструктивных ограничений и препятствий формируется *план движения — последовательность промежуточных конфигураций, ведущая к цели.* Затем по «кускам» (например, в классе кусочно-полиномиальных функций) строится само ПД в соответствии с этим планом.

Большой интерес представляет также «локальный» подход, основанный на моделировании так называемого тропизма.

Тропизм — это свойство, присущее простейшим живым организмам; оно заключается в направленных движениях организмов в результате действия односторонних раздражителей. Применительно к уровню построения ПД под таким *«раздражителем» будем понимать информацию о взаимном расположении манипулятора и препятствий в реальном трехмерном пространстве, а под тропизмом — движения манипулятора, направленные на удаление от препятствий*. Не приводя (ввиду громоздкости) явные формулы алгоритма построения ПД, имитирующего свойства тропизма, отметим только, что синтезируемые им целенаправленные движения реализуются за счет двигательной избыточности манипулятора, в то время как схват движется по заданной траектории.

Мы бегло охарактеризовали различные способы построения ПД манипулятора. А как строить ПД колесного или гусеничного шасси для робота, функционирующего на местности с препятствиями? Эта задача естественным образом распадается на две*: прокладка безопасного маршрута и построение закона изменения обобщенных координат, удовлетворяющего конструктивным ограничениям и обеспечивающего движение по этому маршруту.*

Рассмотрим один из подходов к построению *оптимального маршрута* на местности с препятствиями, основанный на методе динамического программирования. Для простоты предположим, что местность представляет собой плоскость, а препятствия задаются ломаными линиями. Пусть заданы координаты *исходной точки* (где находится робот) и *целевой точки* (куда он должен передвинуться). ***Задача заключается в построении маршрута (ломаной линии) из исходной точки в целевую, который не пересекает ломаных препятствий и имеет наименьшую длину***. Такой маршрут будем называть оптимальным.

Прежде всего заметим, что если сформулированная задача не имеет тривиального решения (когда маршрутом является просто отрезок, соединяющий исходную и целевую точки), вершинами ломаной наименьшей длины должны быть вершины ломаных — препятствий. Поэтому в дальнейшем будем рассматривать только эти вершины. Введем функцию f_i определяющую минимальную длину пути из некоторой точки i в целевую точку. Сразу же возникает вопрос — как найти эту функцию? Один из классических способов нахождения функции состоит в том, чтобы задать уравнения, определяющие эту функцию.

Приступая к поиску такого уравнения, зададимся вопросом: существует ли такое внутреннее свойство процесса поиска оптимального пути, которое можно использовать для получения уравнения? Ответ положительный. В самом деле, анализируя задачу, мы видим, что отправляясь из точки i, робот на первом шаге попадет в точку j, пока неясно, какую именно. Далее, мы замечаем, что, если робот ищет кратчайший путь от точки i до целевой точки, то в какую бы точку он ни попал, путь из точки j в целевую должен иметь наименьшую длину. Это почти очевидное свойство докажем от противного. Пусть путь из точки i в целевую точку является кратчайшим, тогда часть пути из точки j в целевую точку также должна иметь минимальную длину, так как, если бы эта часть пути не была кратчайшей, то ее можно было бы заменить более кратким путем и тем самым сократить общую длину пути, а это противоречит тому, что f_i по определению есть минимальная длина пути.

Таким образом, мы установили существенное внутреннее свойство процесса поиска оптимального маршрута: «хвост» (конец) процесса — это всегда кратчайший маршрут. Что все это дает для определения функции f_i? Обозначим через s_{ij} — длину пути между точками i и j (хотя все еще неизвестно, что это за точка j), а через f_j — наименьшую длину пути между точкой j и целевой точкой. Выбирая в качестве точки j такую точку, которая минимизирует сумму $s_{ij} + f_j$, получаем уравнение

$$f_i = \min_{j \neq i} [s_{ij} + f_j]. \tag{1}$$

Это — основное уравнение динамического программирования. Оно обладает тем свойством, что задает две функции f_i и $j(i)$. В самом деле, если мы нашли f_i, то значение j, на котором достигается минимум (1), как раз и указывает ту точку, в которую роботу нужно двигаться дальше. Важно отметить, что функция $j(i)$ представляет собой по существу стратегию поиска оптимального маршрута, т. е. правило, которое позволяет роботу, находящемуся в произвольной точке i, определить, куда ему следует двигаться дальше.

Трудность решения уравнения (1) заключается в том, что неизвестная функция входит в обе части равенства. В такой ситуации приходится прибегать к классическому методу последовательных приближений в функциональном пространстве, используя рекуррентную формулу

$$f_i^{(k+1)} = \min_{j \neq i} [s_{ij} + f_j^{(k)}], \tag{2}$$

где $f_i^{(k)}$ — k-е приближение искомой функции. Можно доказать сходимость алгоритма (2) для произвольной неотрицательной начальной функции f_i^0 с единственным ограничением, что значение функции f в целевой точке равно нулю.

Решение уравнения (1) можно искать и в пространстве стратегий. Этот путь представляется более естественным: «мыслить» в терминах пространства стратегий роботу более удобно, чем в терминах искусственно выбранных функций f_i. К тому же понятие стратегии сохраняет смысл и в тех случаях, когда функция не может быть определена. Какую же приближенную стратегию можно выбрать в задаче прокладки оптимального маршрута? Одна из возможных стратегий такова: робот решает непосредственно двигаться из точки i в целевую точку. Тогда получается приближение $f_i^{(0)} = s_{i\text{ц}}$, где $s_{i\text{ц}}$ — длина пути между точкой i и целевой точкой. Следующее приближение получится, если робот будет искать решение в классе двухзвенных

ломаных. Дальнейшие приближения ищутся в классе трехзвенных, четырехзвенных и т. д. ломаных.

Одно из преимуществ выбора приближений в пространстве стратегий состоит в следующем: выбрав из общих соображений стратегию, мы тут же получаем соответствующую функцию $f_i^{(k)}$. Понятно, что $f_i^{(k)} \le f_i$. В заключение, не входя в детали, отметим только, что описанный метод последовательных приближений в пространстве стратегий действительно сходится к решению уравнения (1) и допускает чрезвычайно простую программную реализацию.

После того, как оптимальный безопасный маршрут построен, можно тем или иным методом (например, с помощью кусочно-полиноминальной аппроксимации) построить программное движение колесного или гусеничного шасси робота. Алгоритмическое решение этой последней задачи обычно не вызывает никаких принципиальных затруднений.

6.24. Алгоритмы адаптивного управления движением

После того как программное движение (ПД) робота построено, возникает следующая задача: *найти закон управления исполнительными приводами, реализующий ПД*. Решение этой задачи существенно осложняется *неопределенностью* условий функционирования робота. Природа этой неопределенности многообразна: изменение характеристик исполнительных двигателей и механизмов (например, в результате старения и износа), технологические допуски, возможные неисправности, а также изменение условий, внешних по отношению к роботу, но оказывающих на него полностью или частично неконтролируемые воздействия. Если степень неопределенности велика, то для построения законов управления ПД робота классические методы теории автоматического управления могут оказаться недостаточными. В подобных условиях рекомендации по широко используемым следящим приводам, параметры которых выбираются из априорных соображений о «средних» условиях функционирования, зачастую не обеспечивают реализацию ПД с требуемой точностью. Поэтому возникает необходимость в адаптивном управлении, восполняющем недостающую информацию в процессе функционирования робота. Синтез *адаптивного управления ПД* осуществляется в два этапа. Вначале в предположении, что уравнения движения робота полностью известны, строится закон управления, обеспечивающий близость (с любой наперед заданной точностью) реального и программного движений. Эту «неадаптивнаю» задачу можно рекомендовать решаеть

методами классической теории управления. Однако воспользоваться таким «идеальным» законом управления практически нельзя, так как он зависит от ряда *варьируемых параметров* уравнения движения, значения которых неизвестны. Варьируемыми параметрами могут быть, например, масса и моменты инерции объекта манипулирования, распределение нагрузки на шасси робота, коэффициенты сцепления с грунтом и т. п. Поэтому на втором этапе на основе «идеального» закона строится адаптивное управление, обеспечивающее близость реального и ПД при любых возможных изменениях варьируемых параметров.

Идея адаптивного управления проста. Она заключается в замене неизвестных параметров «идеального» закона управления их оценками, которые должны целенаправленно «настраиваться» в процессе функционирования робота. Алгоритмы, по которым осуществляется «настройка» параметров управления, принято называть *алгоритмами адаптации.* Многие известные в настоящее время алгоритмы адаптации с математической точки зрения представляют собой *разностные или дифференциальные уравнения*, определяющие закон целенаправленного изменения параметров управления на основе сенсорной информации.

Примером эффективных алгоритмов адаптации, допускающих простую реализацию, могут служить *конечно-сходящиеся* алгоритмы решения целевых неравенств. Смысл целевых неравенств заключается в том, что если они выполнены, то закон управления, в котором вместо неизвестных параметров используется их оценка, обеспечивает требуемую близость реального и ПД. В этом случае коррекция («настройка») параметров не производится. Если же целевые неравенства в некоторый момент времени нарушаются, то осуществляется адаптивная коррекция параметров по некоторому алгоритму. Этот алгоритм может быть выбран так, что число коррекций будет конечным (и даже минимально возможным).

Описанный конечно-сходящийся процесс адаптации хорошо согласуется с содержательным представлением об адаптации. В самом деле, с интуитивной точки зрения адаптация должна проявляться в том, что в неизменяющихся («стационарных») условиях функционирования робота алгоритм адаптации «работает» не все время, а с течением времени «отключается», осуществляя переход на автоматическое управление без «настройки» параметров. Лишь значительное изменение условий функционирования робота вызывает необходимость «включения» алгоритма адаптации для коррекции параметров управления.

6.25. Об организации целесообразного поведения интеллектуального объекта

Одно из важнейших качеств интеллектуального объекта — это целесообразность, или разумность, его поведения. Естественно возникает вопрос: каким образом сложное целесообразное поведение интеллектуального объекта складывается из взаимодействия простых «мыслительных» блоков —-описанных выше элементов его интеллекта? Не претендуя на полный ответ на этот трудный вопрос, попытаемся кратко в самых общих чертах описать схему организации целесообразного поведения интеллектуального объекта.
Целесообразное поведение предполагает согласованную работу элементов интеллекта в процессе выполнения интеллектуальным объектом стоящих перед ним задач. Для выполнения такой согласованной работы служит специальный блок — *координатор,* изображенный на рис. 1.

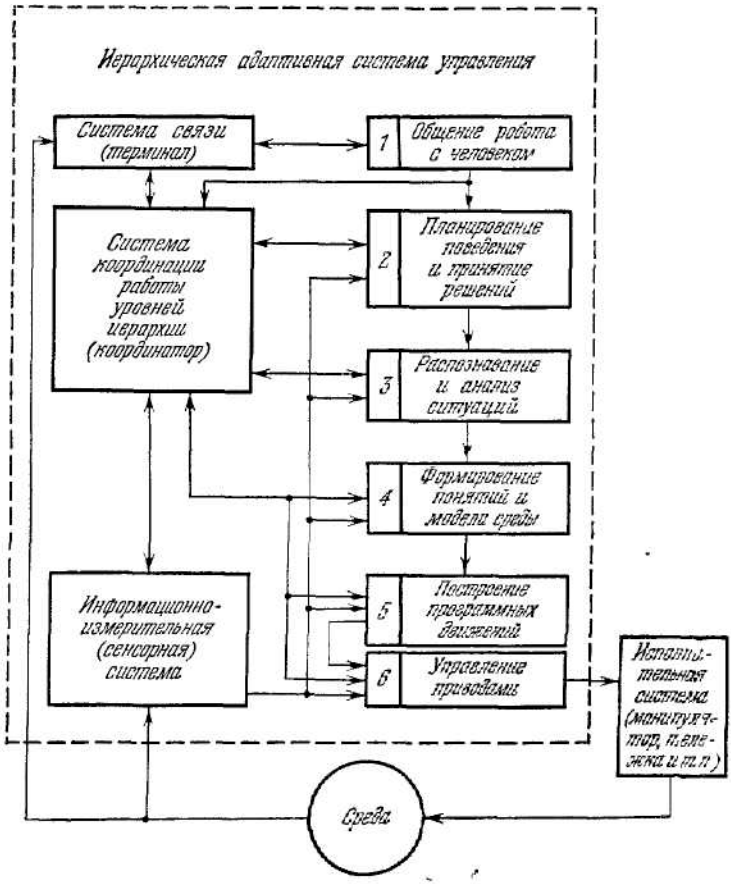


Рис. 1. Схема иерархической адаптивной системы управления.

Этот блок подает сигналы о начале работы систем различных уровней иерархии и получает от них сигналы обратной связи, свидетельствующие об успешном окончании их работы или о невозможности закончить работу с указанием причин. Таким образом, координатор играет роль своеобразного диспетчера для распределения информации по отдельным уровням иерархии (включая и их предохранение от излишней информации), а также для обеспечения относительно независимой (автономной) работы этих уровней.

Мы уже отмечали, что идея иерархического управления основана на разложении исходной задачи на отдельные подзадачи, для решения которых могут быть приме иены различные специальные методы

(элементы интеллекта). Как это реально осуществляется? Сформулировав на том или ином языке задание, человек вводит его с помощью терминала в память робота. Затем на уровне планирования поведения осуществляется разбиение задания на подзадачи и определяется порядок их выполнения. Эта информация поступает в координатор, который инициирует последовательное, или, если это возможно, параллельное решение отдельных подзадач на разных уровнях иерархии. Основу координатора составляет специальное рабочее поле оперативной памяти — *поле внимания.* Сосредоточивая внимание на той или иной подзадаче, координатор заполняет поле внимания соответствующей информацией. Это может быть информация, воспринимаемая сенсорными датчиками, априорная информация, заложенная в память разработчиком интеллектуального объекта, или новые понятия и данные, сформулированные интеллектуальным объектом в процессе обучения на опыте и адаптации. Например, при решении подзадачи анализа и распознавания ситуаций в поле внимания поступают показания сенсорных датчиков и соответствующие программы их обработки. Зная, к какому классу относится текущая ситуация, интеллектуальный объект выбирает адекватное ей действие. Затем координатор последовательно обращает внимание на подзадачи построения программных движений и адаптивного управления приводами, после чего интеллектуальный объект совершает требуемые целенаправленные действия. Такова грубая схема формирования целесообразного поведения интеллектуального объекта.

7. Решение задач методами эвристического поиска

7.1. Вводные замечания

Основой всех стратегий решения задач в области ИИ является метод поиска. Для задач практической степени сложности поиск должен направляться с помощью некоторого механизма управления, причем мы отвергаем возможность использования исчерпывающего поиска или поиска, управляемого случайным механизмом. Решающим аргументом в пользу такого решения является тот факт, что в принципе пространство поиска может быть бесконечным.

Эвристически эффективная стратегия представляет собой совокупность двух механизмов: *механизма генерации элементов* пространства поиска — кандидатов в решающую последовательность и *механизма управления генерацией,* работа которого основывается на информации о пространстве поиска, а также свойствах цели и уже построенных элементов.

Степень эвристической эффективности стратегий определяется прежде всего тем, как извлекаются указанная информация и свойства и как они используются механизмом управления в поиске решения. Таким образом, мы можем выделить в каждой стратегии ***синтаксическую,* или структурную компоненту (генерация), и *семантическую,* или смысловую компоненту (управление поиском)**. Важно отметить, что семантическая компонента вносится в процесс решения задачи *стратегией* во всех предписаниях, кроме семантических.

При построении стратегий мы стоим перед необходимостью разрешить противоречие между *качеством решения* и ею *эффективностью,* где под *качествам* мы понимаем свойства решения (длина решающего пути, количество вершин в решающем графе и т. д.), а под *эффективностью* — количество ресурсов, затраченное при поиске решения (общее число генерированных вершин и т. д).

Поскольку нашей задачей является построение эвристически эффективных стратегий, мы не рассматриваем стратегии типа поиска в ширину, которые, хотя и позволяют найти все решения и, следовательно, наиболее качественные, представляют собой модель исчерпывающего поиска, а потому эвристически неэффективны. План настоящего раздела состоит в том, чтобы для каждого из предписаний в виде задачи эвристического поиска построить стратегию, позволяющую найти решение наилучшего качества, если оно имеется, а затем исследовать пути повышения эвристической эффективности, возможно, с риском потерять решение наилучшего качества.

7.2. Стратегии, основанные на поиске в графе вывода

7.2.1. Алгоритм поиска решающего графа

Ранее мы рассмотрели одно обобщенное декларативное предписание в виде графа вывода, из которого в качестве частных случаев вытекают предписания в виде пропозиционального графа, графа доказательства предписаний и графа пространства состояний. В обобщенном декларативном предписании решение задачи сводится к поиску

беспосылочного редукционного вывода (решающего графа) в графе вывода. Мы опишем алгоритм поиска решающего графа в графе вывода как механизм генерации элементов пространства поиска, а затем рассмотрим некоторые общие характеристики этого алгоритма, в том числе касающиеся механизма управления.

Этот алгоритм носит скорее иллюстративный характер: во-первых, он демонстрирует стиль построения алгоритмов в частных теоретико-графических представлениях, во-вторых, на примере этого алгоритма мы покажем общий сценарий, по которому в последующих параграфах будут рассматриваться стратегии в этом классе представлений. Мы не утверждаем, что этот алгоритм окажется наиболее эффективным (в отношении быстродействия и требуемой памяти) для всех частных случаев.

Введем два направления поиска — *прямое,* т. е. от посылок к заключению *(F),* и *обратное (B).* Будем называть *вывод генерированным,* если генерированы все его высказывания и акты вывода, даже если это произошло в процессе генерации другого вывода. На каждом шаге мы будем выбирать одно из направлений поиска, причем, если не окажется ни одного вывода — кандидата на генерацию, то алгоритм терпит неудачу. В противном случае он выбирает наилучшего кандидата в текущем направлении и генерирует все ранее негенерированные высказывания применением одного ранее негенерированного акта вывода, принадлежащего этому направлению. Если при этом не получено решение, то алгоритм вновь выбирает направление, выбирает кандидата на вывод, и т. д. до тех пор, пока не будет получено решение. Соответственно направлениям поиска мы выделяем два множества *F* и *B. F* содержит все уже генерированные высказывания и акты вывода, принадлежащие беспосылочным выводам (БВ). *B* содержит все уже генерированные высказывания и акты вывода, принадлежащие редукционным выводам (РВ). Заметим, что пересечение *F* и *B* не обязательно должно быть пустым до получения решения.

Определим формально кандидаты на генерацию в направлениях *F* и *B*. Пусть существует акт вывода, не принадлежащий *F,* но все посылки которого принадлежат *F*. Тогда любой БВ, состоящий из этого акта вывода, его заключения и, для любой посылки акта вывода, точно одного БВ, содержащегося в *F* и имеющего эту посылку как заключение, называется *кандидатом на генерацию в направлении F*. Пусть D_0 — РВ, содержащийся в *B*, и каждая посылка D_0 есть заключение некоторого акта вывода, не принадлежащего *B*. Тогда любой РВ, состоящий из точно одного такого акта вывода, его посылок и D_0, называется *кандидатом на генерацию в направлении B*.

Предположим, что на выводах определена мера качества, так что для любой совокупности выводов можно выбрать вывод наилучшего качества (ни один из выводов, принадлежащий этой совокупности, не имеет лучшего качества, чем выбранный).

Тогда алгоритм поиска беспосылочного редукционного вывода может быть представлен следующими шагами:

1. *F* и *B* в начальном состоянии — пустые множества.
2. Выбрать направление *X* генерации *(F* или *B).*
3. Если нет кандидатов на генерацию в направлении *X,* неудача. В противном случае продолжать.
4. Выбрать и генерировать вывод-кандидат *D* для *X,* имеющий наилучшее качество в направлении *X.* Генерация производится путем добавления к множеству *X* одного акта вывода и всех определений в *D,* еще не принадлежащих *X.*
5. Добавить к *F* и *B* все акты вывода и высказывания, принадлежащие уже сгенерированному БВ высказывания в *B.*
6. Успешное решение, если а) решающий вывод содержится в *F* или *B,* б) нет кандидата на генерацию в направлении *F* или *B,* имеющего лучшее качество, чем *D.* В противном случае перейти к шагу 2.

На рис. 1 представлен пример состояния пространава поиска в начале цикла работы алгоритма.

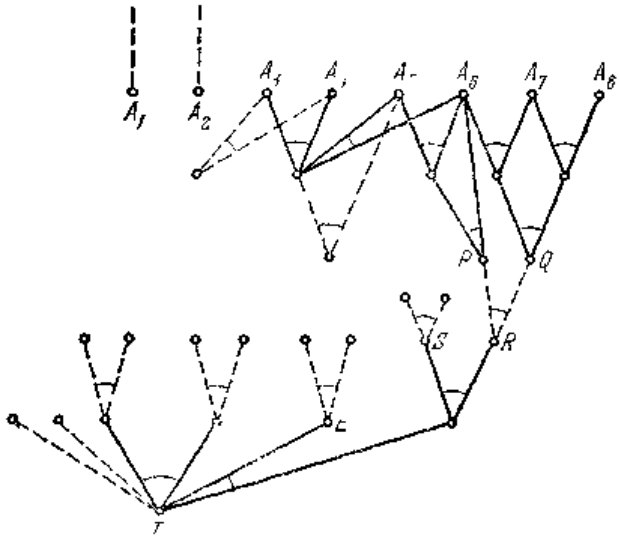


Рис. 1. Состояние пространства поиска в начале цикла работы алгоритма.

Сплошные линии — дуги, принадлежащие уже сгенерированным выводам. Пунктирные линии — дуги, принадлежащие выводам, которые являются кандидатами на генерацию. Остальные выводы на рисунке не показаны. У аксиом A_1 и A_2 пунктиром показаны фиктивные акты вывода (по определению, A_1 и A_2 являются их заключениями). У A_3—A_8 фиктивные акты вывода не показаны. Акт вывода с посылками P и Q и заключением R является частью кандидата на генерацию как в направлении F, так и в направлении B. Множества F и B пока не имеют общих элементов.

Пусть на шаге 2 алгоритма выбрано направление B и выбранным кандидатом на генерацию является РВ с посылками E, S, P, Q. На шаге 4 к B добавляются высказывания P, Q и акт вывода, содержащий их как посылки. Этот же акт вывода добавляется к F вместе с заключением R. Одновременно к B добавляются все высказывания и акты вывода в выводах, имеющих P и Q как заключения.

7.2.2. Свойства алгоритма

Полнота. *Стратегия поиска* называется *полной,* если она найдет решение всегда, когда оно существует. *Поисковая стратегия* будет *исчерпывающей* для направления X, если она генерирует все высказывания и акты вывода, которые могут быть генерированы в направлении X. Очевидно, что любая исчерпывающая поисковая стратегия полна. Однако и в случае бесконечного количества альтернатив вывода (бесконечного дизъюнктивного ветвления) поисковая стратегия может быть полна.

Упорядочение выводов по качеству называется *квазиконечным* для направления X, если для любого вывода D существует только конечное число выводов, имеющих качество, лучшее или равное качеству D.

Теорема 1. *Любая двунаправленная поисковая стратегия, использующая квазиконечное упорядочение выводов по качеству для направления X, полна при условии, что она не становится однонаправленной в противоположном X направлении.*

Доказательство. Пусть D^* — решение, не порожденное поисковой стратегией. Тогда существует некоторый $D \subset D^*$, который на каком-то шаге стал кандидатом на генерацию в направлении X, но не был генерирован. Тогда, если поисковая стратегия не становится однонаправленной в направлении X и не нашла другого решения, она должна генерировать в направлении X кандидаты на вывод D_1, D_2, ..., D_n, D_{n+1}, . . ., не оканчивая работы. Но по определению квазиконечности один из них, например D_n, должен иметь качество, худшее, чем D, что противоречит нашему предположению.

Допустимость. Поисковая стратегия называется *допустимой,* если она находит наилучшее решение всегда, когда оно имеется. Определим понятие наилучшего решения. Пусть f — действительная неотрицательная функция, определенная на множестве выводов. Назовем f *оценочной функцией,* если D_1 имеет *лучшее качество,* чем D_2, когда

$$f(D_1) < f(D_2). \quad (1)$$

D_1 и D_2 *равнокачественны,* когда

$$f(D_1) = f(D_2). \quad (2)$$

Оценочная функция *монотонна,* если из $D' \subseteq D$ следует

$$f(D') \leq f(D). \quad (3)$$

Монотонная оценочная функция характеризует меру сложности решающего вывода. Существует несколько вариантов меры сложности:

— *размер* — число высказываний (директив) в выводе;

— *уровень* — наибольшее число высказываний (директив), измеряемое вдоль какой-либо ветви вывода;

— *суммарная сложность (стоимость)* — сумма всех сложностей (стоимостей), каждая из которых связана с одним из высказываний (директив) в выводе;

— *максимальная сложность (стоимость)* — наибольшая сумма сложностей (стоимостей), измеренная для всех высказываний (директив) вдоль какой-либо ветви вывода.

Решение называется *наилучшим,* если оно обладает наименьшей мерой сложности среди всех решающих выводов на графе вывода.

Диагональной поисковой стратегией (ДПС) называется стратегия, использующая такую оценочную функцию, что

$$h(D) = f(D) - g(D) \geq 0 \quad (4)$$

для всех выводов D, $g(D)$ — мера сложности вывода D такая, что $g(D') \leq g(D)$, если $D' \subseteq D$. Функция h называется *эвристической функцией* и оценивает сложность части наилучшею решения D^*, дополнительную к D. Тогда оценочная функция может рассматриваться как оценка сложности наилучшею решения D^*, содержащего D.

Таким образом, ДПС выбирает и генерирует всегда тот из кандидатов на вывод, который имеет наименьшую оценочную функцию вида

$$f(D) = g(D) + h(D). \quad (5)$$

Геометрическая интерпретация ДПС иллюстрируется на рис. 2.

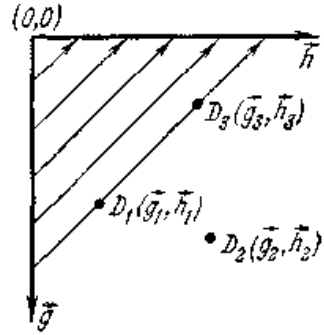


Рис. 2. Геометрическая интерпретация диагональной поисковой стратегии

Здесь точки в системе координат $(\vec{g},\vec{h})$ представляют собой акты вывода, порожденные поисковой стратегией в процессе выбора кандидата на вывод D с $g(D)=\vec{g}$ и $h(D)=\vec{h}$. В силу монотонности функции f и свойств ДПС поиск решения идет от *более коротких диагоналей к более длинным,* так что D_2 будет всегда порождаться после D_1. ДПС называется *ограниченной,* если для всех кандидатов на вывод D

$$f(D)\leq g(D^*), \qquad (6)$$

где $D\subseteq D^*$, D^* - решающий вывод.

Из приведенного определения очевидно, что оценочная функция для ограниченной ДПС монотонна и $h(D^*)=0$, т. е. в интерпретации рис. 2 все решающие выводы лежат на оси $\vec{g}$.

Теорема 2. *Ограниченная ДПС допустима.*

Доказательство. Предположим, что ограниченная ДПС нашла решение D^*, но существует решение D_0, лучшее, чем D^*, а окончание работы алгоритма произошло потому, что ни один кандидат на вывод в направлении X не имеет качества, лучшего, чем D^*. Тогда в момент окончания работы некоторый $D'_0\subset D_0$ является кандидатом на генерацию в направлении X. Однако в силу свойств ограниченной ДПС

$$f(D'_0)\leq g(D_0)<g(D^*)=f(D^*), \qquad (7)$$

что противоречит условию окончания работы алгоритма.

Следствие. *Двунаправленная ограниченная ДПС, использующая квазиконечное упорядочение выводов по качеству для направления X, допустима при условии, что она не становится однонаправленной в противоположном направлении.*

Оптимальность. *Допустимая стратегия* с эвристической функцией h_1 называется *оптимальной,* если она никогда не порождает большего количества высказываний (директив), чем другая допустимая стратегия с такой эвристической функцией h_2, что $h_1(D) \geq h_2(D)$ для всех выводов и $h_1(D) > h_2(D)$ для некоторых из них.

Оптимальность алгоритма поиска решающего вывода в графе вывода может быть доказана лишь для некоторых его частных случаев. Поэтому рассмотрение вопроса об оптимальности мы выносим в параграфы, посвященные описанию алгоритмов поиска решения в частных представлениях.

Итак, в описанных стратегиях в качестве механизма генерации используется абстрактное преобразование, определенное на множестве высказываний в графе; в качестве механизма управления поиском используется оценочная функция, содержательно интерпретируемая количеством усилий, которые надо затратить для получения решения.

7.3. Поиск в пространстве состояний

7.3.1. Алгоритм и его свойства

Решение задачи в продукционной системе сводится к поиску *решающего пути* в графе пространства состояний. Таким образом, механизм генерации в этом представлении определяется преобразованием $\Gamma(x)$, порождающим все дочерние вершины x. Выбор альтернативных путей поиска, или вершин-кандидатов на генерацию, связывается с классом оценочных функций $f(x)$, где x — вершины графа. Мы будем выбирать для генерации ту из вершин-кандидатов, для которой функция f принимает *минимальное значение.* Мы рассматриваем вариант алгоритма для одной начальной вершины $x_0 \in X_o$.

Пусть S — множество уже выбранных вершин, $\overline{S}$ — множество вершин-кандидатов, $S \cap \overline{S} = \varnothing$, x_0 — начальная вершина, X_t — множество конечных вершин, $x_t \in X_t$, x — текущая вершина, $x \in X$, x_i — ее дочерние вершины. Тогда алгоритм поиска в графе пространства состояний представляется следующими шагами:

1) Поместить x_0 в $\overline{S}$ и вычислить для нее $f(x_0)$.

2) Если $\overline{S} = \varnothing$, неудача, иначе продолжать.

3) Выбрать такую $x \in \overline{S}$, что $f(x) = \min_{y \in \overline{S}} f(y)$, и поместить ее в S, изъяв из $\overline{S}$.

4) Если $x \in X_t$, путь найден, окончание, иначе продолжать.

5) Найти *все* $x_i = \Gamma(x)$. Если $\Gamma(x) = \varnothing$, к шагу 2. Иначе вычислить все $f(x_i)$.

6) Для каждого x_i:

а) Если $x_i \notin S \cup \overline{S}$, то поместить x_i в $\overline{S}$.

б) Если $x_i \in \Gamma(x) \cap S$, то сопоставить x_i наименьшую из старой и вновь полученной оценку $f(x_i)$.

в) Если $x_i \in \Gamma(x) \cap \overline{S}$, то сопоставить x_i наименьшую из старой и вновь полученной оценку $f(x_i)$, поместить x_i в $\overline{S}$ (изъяв ее из S).

г) В остальных случаях не изменять S и $\overline{S}$.

7) Перейти к шагу 2.

Пункты 6б и 6в отражают действия алгоритма в случае, когда оператор Г порождает уже рассмотренные ранее вершины, которые к этому моменту находятся либо в $\overline{S}$, либо в S. Естественно, что мы хотим приписать этим вершинам наименьшие из возможных оценочные функции. Этой проблемы не возникало бы при поиске в дереве, поскольку в этом случае имеется точно один путь из начальной вершины в данную.

Рассмотрим свойства этого алгоритма для класса ДПС, а затем укажем на некоторые возможные обобщения.

Поскольку представление в виде графа в пространстве состояний является частным случаем графа вывода, получаемым при исключении конъюнктивных вершин и ограничении числа дочерних вершин конечным числом, мы заключаем, что представленный выше алгоритм, управляемый ДПС, обладает полнотой.

Более того, в случае ограниченной ДПС алгоритм является допустимым, т. е. всегда находит наилучший в смысле минимума f решающий путь, если он существует.

Из определения оптимальности (п.7.2) ясно, что нам необходимо показать, что допустимая стратегия с эвристической функцией h_2 будет порождать любую вершину, которую будет порождать допустимая стратегия с функцией h_1.

Теорема 3. *Ограниченная ДПС оптимальна.*

Доказательство. Пусть нам даны ограниченная ДПС $\sum_1$ с эвристической функцией h_1 и другая допустимая стратегия $\sum_2$ с эвристической функцией h_2. Предположим, что существует такая

вершина x, которую генерирует $\sum_1$, но не генерирует $\sum_2$. Тогда для $\sum_1$ оценка в вершине x

$$f(x)=g(x)+h_1(x). \quad (8)$$

В силу свойства ограниченности стратегии

$$f(x)\leq f(x_t), \quad (9)$$

где $x_t \in X_t$, откуда

$$h_1(x)\leq f(x_t)-g(x) \quad (10)$$

Стратегия $\sum_2$ в силу допустимости может не генерировать вершину x только по одной причине: существует некий менее сложный путь к x_t, чем путь через вершину x. Поэтому для $\sum_2$ действительная стоимость пути к x_t через x

$$f^*(x)\geq f(x_t). \quad (11)$$

Подставляя (10) в (11), получаем

$$h_1(x)\leq(g^*(x)-g(x)+ h_2(x) \quad (12)$$

Однако из анализа шага 6 алгоритма ясно, что оценка $g(x)$ в результате повторного возврата к вершине x может, в крайнем случае, уменьшиться, т. е. для всех $x \in X$ и на любом шаге работы алгоритма $g^*(x)\leq g(x)$. Отсюда

$$h_1(x)-h_2(x) \leq 0, \quad (13)$$

что противоречит определению оптимальности. Отсюда заключаем, что $\sum_2$ генерирует любую вершину, которую генерирует $\sum_1$, что доказывает теорему.

Замечание. Условие (3) монотонности ДПС позволяет вывести свойство меры сложности $g(x)$. Если $x \in S$, то это означает по построению алгоритма, что в этот момент $f(x)\leq f(y)$ для всех $y \in \overline{S}$. Поэтому и в силу монотонности, если мы вновь придем к вершине x из любой другой вершины, то новая оценка $f'(x)\geq f(x)$. Это означает, что при первой же генерации вершины x мы находим ее истинную меру сложности или, другими словами, при первом же включении x в S она уже находится на оптимальном пути из начальной вершины в x. Отсюда

$$g(x)=g^*(x) \quad (14)$$

и, следовательно, для алгоритма поиска, управляемого монотонной оценочной функцией, т. е. для допустимого алгоритма, шаги 6б и 6в могут быть без ущерба изъяты.

Необходимо указать на ряд важных особенностей алгоритма поиска, управляемого ограниченной ДПС. Допустимость стратегии гарантирует нахождение оптимального пути, однако при этом никак не оговаривается число генерируемых при этом вершин, т. е. фактически вычислительные ресурсы, необходимые для получения решения. Оптимальный путь будет найден с минимальными затратами, но лишь среди других допустимых стратегий.

Вполне вероятно, что путем отказа от допустимости мы могли бы, теряя гарантию найти оптимальный решающий путь, сократить число генерируемых вершин, получив, таким образом, некоторый компромисс между качеством получаемого решения и ресурсами, необходимыми для его получения, т. е. эффективностью алгоритма.

Далее, существенным дефектом ДПС является исследование всех альтернативных путей, имеющих одинаковое качество. Выбор вершины среди множества вершин с одинаковой оценочной функцией в рассмотренном выше алгоритме произволен, однако, в силу монотонности оценочной функции, алгоритм все время будет возвращаться к исследованию вершин в множестве $\overline{S}$, обладающих равным качеством с уже занесенной в *S* вершиной. В случае значительного количества равнокачественных текущих решений управление поиском с помощью ДПС окажется чрезвычайно слабым и алгоритм по своим вычислительным свойствам будет приближаться к полному перебору вершин из $\overline{S}$, т. е. будет скорее поиском в ширину, чем в глубину. Между тем в практически важных случаях пространство поиска весьма часто имеет вблизи решения мезаструктуру, т. е. форму плато с мало отличающимися в точках пространства значениями оценочной функции. Если же это плато находится вблизи локального оптимума (пространство поиска не унимодально), то это дополнительно ухудшает качество работы алгоритма.

Для улучшения положения можно наложить еще одно ограничение на ДПС. Если два кандидата на генерацию обладают равной оценочной функцией, но неодинаковой мерой сложности, т. е. для $x, y \in \overline{S}$

$$\left.\begin{array}{l} f(x) = f(y), \\ g(x) > g(y) \end{array}\right\} \qquad (15)$$

то выбирается для включения в *S* та вершина *x,* которая обладает большей мерой сложности, или большей стоимостью уже пройденного пути. Основанием для введения условий (15) является предположение, что путь от этой вершины к цели окажется короче *(h (x)<h (y)).* ДПС, удовлетворяющая условиям (15), называется *восходящей ДПС.* Геометрическая интерпретация этой стратегии очевидна из рис.2. Дополнительно к направлению генерации, определяющему ДПС, добавляется направление генерации вдоль диагонали снизу вверх, предшествующее переходу с диагонали на диагональ, так что последовательность рассмотрения точек в координатах *(g, h)* будет D_1, D_3, D_2. На качество работы алгоритма существенное влияние оказывает метод получения эвристической функции *h*.

Теорема 4. *Для любой ограниченной ДПС и для всех x ∈ X, лежащих на некотором пути к целевой вершине,*
а) *h(x)≤h* (x), где h* (x)— истинная мера сложности оптимального пути от вершины x до целевой вершины;*
б) *стратегия с функцией h* (x) генерирует вершины только на оптимальном пути.*
Таким образом, можно предположить, что с приближением *h* к *h ** алгоритм, управляемый ограниченной ДПС, будет генерировать все меньшее число вершин. Однако эвристическая функция определяется на основании специфической информации о решаемой задаче, т. е. на основании знания глобальных свойств пространства поиска. Слабость знаний решателя задач об этих свойствах является отличительной чертой всех систем предписаний, поэтому мы не можем рассчитывать на сколько-нибудь точное знание эвристической функции (почему она и называется эвристической). Попытка более точно определить эвристическую функцию, в лучшем случае, приведет к более сложным вычислениям на каждой вершине, что отрицательно скажется на качестве работы алгоритма.
Заканчивая анализ ДПС как механизма управления поиском в пространстве состояний, еще раз подчеркнем, что
1) Оптимальная стратегия гарантирует эффективность алгоритма только в классе допустимых стратегий.
2) Наличие равнокачественных решений ухудшает направленность поиска.
3) Выбор эвристической функции может серьезно повлиять на качество работы алгоритма.

7.3.2. Методы повышения эффективности поиска

Рассмотрим некоторые методы отказа от допустимости, которые могут привести к повышению эффективности алгоритма поиска. Опишем следующие возможности:
1) Более общее задание оценочной функции.
2) Динамическое изменение оценочной функции в ходе поиска решения.
3) Динамическое переупорядочение множества операторов Г.
4) Динамическое отсечение вершин.
Общее задание оценочной функции. Можно задать оценочную функцию в виде

$$f(x) = (1—\omega)g(x) + \omega h(x), \quad 0 \le \omega \le 1, \qquad (16)$$

где *f(x)* —монотонная оценочная функция.
Отметим ряд частных случаев этой функции:

1) ω=0 — алгоритм поиска в ширину, или *стратегия насыщения уровня*.

2) $\omega = \frac{1}{2}$ — только что рассмотренный класс ДПС.

3) ω=1 — алгоритм, управляемый эвристической функцией.

Стратегия с оценочной функцией вида (16) обладает следующими свойствами*:*

1) *Стратегия полна при* ω <1.

2) *Стратегия с истинной эвристической функцией h* оптимальна при* $^1/_2 \leq \omega \leq 1$.

Чтобы проиллюстрировать повышение эффективности поиска для $^1/_2 \leq \omega \leq 1$ по сравнению с ДПС, приведем таблицу 1 результатов, экспериментально полученных для одной из задач (действительная минимальная длина пути равна 32).

Таблица 1

ω	Количество генерированных вершин	Длина пути
0,5	1000	Отказ от продолжения поиска
0,75	317	38
0,89	431	62
0,94	279	62
1,00	514	80

Динамическое изменение оценочной функции. Рассмотрим два способа динамического изменения оценочной функции: *аналитический* и *эвристический.* В первом случае задаем оценочную функцию в виде

$$f(x)=g(x)+\omega(x)h(x). \qquad (17)$$

Весовой коэффициент является функцией вершины. Желательно, чтобы он был убывающей функцией расстояния вершины от начальной вершины. В этом случае в близких к начальной вершине уровнях осуществлялся бы поиск в глубину, избегающий рассмотрения равнокачественных вершин вдали от целевой вершины. По мере приближения к цели поведение алгоритма могло бы изменяться в сторону стратегий с насыщением уровня, т. е. к более подробному исследованию окрестности текущей вершины $x \in \overline{S}$. Положим

$$\omega(x) = 1 + e - \frac{e \cdot l(\mu(x_0, x))}{n}, \quad (18)$$

где $0<e<1$, $l(\mu(x_0, x))$ — длина пути от начальной вершины к вершине x. Оценочная функция (17) с весовым коэффициентом (18) будет обладать описанным выше поведением, если $g(x)$ меняется без больших скачков. Стратегия, управляемая функцией (17), обладает одним важным свойством.

Теорема 5. *Стратегия с оценочной функцией* (17), *взвешивающим коэффициентом* (18) *и эвристической функцией* $h(x) \le h^*(x)$ *для всех* $x \in X$ *находит решающий путь, стоимость которого не превышает стоимость оптимального пути более чем в* $(1+e)$ *раз.*

Доказательство. Пусть m — стоимость пути, найденного стратегией при данном e, а L — стоимость оптимального пути. Пусть $x \in \overline{S}$ — вершина на оптимальном пути. Тогда из (17) и (18)

$$f(x) < g(x) + (1+e)h(x). \quad (19)$$

Так как $h(x) \le h^*(x)$, то

$$f(x) < g(x) + h^*(x) + eh^*(x). \quad (20)$$

Так как x лежит на оптимальном пути, то

$$f(x) < L + eh^*(x), \quad (21)$$

откуда

$$f(x) < L(1+e). \quad (22)$$

Но по оценке нашей стратегии $m \le f(x)$, так что

$$m < L(1+e), \quad (23)$$

что доказывает теорему.

В таблице 2 приведены экспериментальные результаты поиска в графе, сравнивающие работу стратегии с $\omega(x)h(x)$ и ДПС (известная оптимальная длина решающего пути равна 246).

Таблица 2

Стратегия	Длина решающего пути	Количество генерированных вершин
Стратегия со взвешиванием, $e=0,6$	260	353
Стратегия со взвешиванием, $e=0,4$	253	474
Диагональная поисковая стратегия	Отказ от продолжения поиска	500

Идея *эвристического* метода динамического изменения оценочной функции заключается в том, чтобы обновлять ее в процессе решения задачи на основе анализа уже построенных частичных поисковых деревьев и с помощью соответствующей техники усреднения и оптимизации. **Мы называем этот метод эвристическим, потому что нет никакой гарантии, что поисковое пространство достаточно однородно, чтобы частичные деревья являлись представительной выборкой.**

Кроме того, успех этого метода в значительной степени зависит от того, насколько удачно выбрана оценочная функция (с точки зрения ее близости к действительной стоимости пути).

Динамическое отсечение вершин. Несмотря на те или иные меры, принятые для улучшения качества работы алгоритма, возникает вопрос, что делать, когда вычислительные ресурсы (память или время) исчерпаны, а решение еще не найдено. В случае переполнения памяти выход заключается в том, чтобы отметить частичный путь из вершины $x \in \overline{S}$ с наименьшим значением $f(x)$ к начальной вершине и затем стереть ту или иную часть построенного к этому моменту дерева. Например, можно запомнить часть этого пути, начиная с начальной вершины, и стереть все дерево поиска, кроме той его части, которая строится из последней вершины на запомненном пути. Этот вариант отсечения вершин является, в известной мере, аварийным. Более эффективной представляется процедура оценки вершин в зависимости от оценок дочерних вершин путем их частичной генерации. Идея заключается в том, чтобы подвергнуть относительно большое число вершин в $\overline{S}$ частичной генерации, используя лишь небольшое подмножество множества применимых к ним операторов. При этом часть вершин на основании такого просмотра вперед может быть сразу оценена как бесперспективная и отсечена. Эта оценка может быть приведена, например, на основе смешанной оценочной функции вида

$$f'(x) = \frac{f(x) + w \sum f(\Gamma'_i(x))}{1 + w(j-1)}, \qquad (24)$$

где Γ'_i, i=l, 2, . . ., j—1—выбранные для просмотра операторы, w— вес.

Здесь мы сталкиваемся с необходимостью оценить относительные затраты на поиск без отсечения и дополнительные вычисления оценочных функций во всех вершинах, сопоставив их в каждом конкретном случае для принятия решения об использовании этого метода. Кроме того, при динамическом отсечении нет никакой гарантии, что мы не отсекаем именно те вершины, которые окажутся на оптимальном пути (отказ от допустимости).

Динамическое переупорядочение операторов. Идея динамического переупорядочения операторов заключается в том, чтобы по мере накопления опыта в процессе решения задачи или класса задач разработчик мог упорядочивать множество операторов (директив) по их полезности, испытывать затем при решении этой задачи или сходных с ней задач операторы (директивы) в установленном порядке, сокращать постепенно это множество до наиболее полезных операторов (директив) и в идеале сопоставлять определенные группы операторов (директив) определенным категориям состояний.

Желательно, чтобы решатель задач по мере накопления опыта синтезировал классификационную таблицу множества операторов (директив) и классов состояний, к которым эти операторы (директивы) целесообразно применять. Применительно к представлению в пространстве состояний задача может быть представлена следующим образом. Вместо оператора (директивы) Γ мы вводим упорядоченное множество операторов (директив) $\Gamma' = \{ \Gamma'_i / i=1, 2, \ldots, m\}$, так что Γ'_i — i-й элемент этого множества, $\Gamma'_i = X \rightarrow X$. Γ' следующим образом связан с Γ: пусть $X_j = \Gamma(x_j)$ и

$$X'_j = \bigcup_{i=1}^{m} \Gamma'_i(x_j);$$

тогда, если $x_k \in X$ то $x_k \in X'_j$. Другими словами для всех j $X_j \subseteq X'_j$, т. е, мы допускаем синтез некоторых операторов (директив) и добавление их к множеству Γ.

В наиболее очевидном варианте динамическое переупорядочение операторов (директив) может быть реализовано следующим образом. Первоначально порядок операторов (директив) в Γ' произволен (например, задается случайным механизмом генерации индексов i, i=1, 2, . . ., m). Задается функция $\varphi: X \rightarrow \Gamma'$, отображающая $x_j \in X$, такие, что $f(x_j) = \min_{x_k \in \overline{S}} f(x_k)$ в порядковый номер оператора (директивы), т. е. для каждой вершины, выбранной для перенесения в S, операторы (директивы) испытываются в порядке, установленном к данному моменту. Далее, в процессе анализа частичных поисковых деревьев те операторы (директивы), которые были применены и соответствуют дугам на оптимальном пути в этих деревьях, «поощряются» уменьшением их индекса в множестве Γ', а примененные операторы (директивы), соответствующие дугам вне оптимального пути, «наказываются» увеличением индекса. Эта схема может быть усилена, так что операторы (директивы) из некоторого подмножества $\Gamma'_r \subset \Gamma'$, $r=s, s+1, \ldots, m$, в случае получения ими очередного «наказания» отбрасываются. В то же время к вершинам $x_j \in \overline{S}$, выбранным для

генерации, применяется лишь ограниченное множество операторов (директив) $\Gamma'_p \subset \Gamma'$, p=1, 2, ..., s—1. В этом случае обновление Γ'_p происходит за счет перехода «наказанных» операторов (директив) из «высшей лиги» в «низшую» и замены их операторами (директивами) из низшей лиги.

Кардинальное решение проблемы переупорядочения операторов (директив) следует искать на пути разбиения поисковых пространств на обладающие различными свойствами области в духе рассмотренных нами в задаче о миссионерах и людоедах или образов Сэндуолла. Естественно предположить, что каждой такой области в результате накопления опыта или непосредственно из ее анализа можно было бы соотнести свое небольшое специализированное множество операторов (директив). Такие множества могли бы составить основу для эффективного поиска решений, однако решение этой задачи является производным от более общей проблемы глобального исследования пространства поиска и преобразования представлений, которая, как отмечалось выше, еще не нашла своего решения

Доказательство допустимости и оптимальности ДПС имело своей целью построение такой стратегии, которая находила бы наилучшие в смысле минимума оценочной функции решения. Из обсуждения, касающегося работы ДПС в случае большого числа равнокачественных альтернатив, ясно, что ДПС даже в большей степени подходит для поиска всех наилучших решений, чем для поиска только одного такого решения. Это вытекает из того, что если алгоритм ДПС нашел наилучшее решение, то к этому моменту он либо раскрыл остальные лучшие решения, либо находится весьма близко к их раскрытию. Однако часто, в особенности для больших поисковых пространств, ставится задача поиска любого решения В этом случае применение ДПС является весьма неэффективным опять-таки в силу параллельного исследования равнокачественных альтернатив.

Таким образом, при необходимости найти любое решение целесообразен сознательный отказ от допустимости алгоритма и использование других поисковых стратегий. В частности, хорошо для этой цели могут подойти стратегии с большим ω в формуле (17).

7.4. Двунаправленный поиск решения в пространстве состояний

В случае, когда целевое состояние задано явным образом, возможно осуществлять поиск решения в графе как от начальной вершины к конечной, так и от конечной вершины к начальной. Эти два

направления поиска могут быть объединены в единый процесс двунаправленного поиска. Основанием для конструирования такого алгоритма является факт экспоненциального роста поисковых деревьев и предположение, что при этом два более коротких диаметра поиска будут генерировать меньше вершин, чем один длинный (здесь под *диаметром поиска* мы понимаем глубину проникновения механизма генерации в поисковом пространстве, т. е длину максимально удаленного от начала поиска пути).

Вновь введем некоторую оценочную функцию для управления поиском. Однако, поскольку поиск производится в двух направлениях, оценочная функция должна быть определена для обоих направлений — прямого *(F)* и обратного *(B)*. Пусть она равна f_F и f_B соответственно. В следующем определении алгоритма сохраняются все обозначения из предыдущего параграфа, $a_{\min}$ равна текущему минимуму стоимости уже найденного пути, $\overline{T}$ и T — множества вершин-кандидатов и генерированных вершин в направлении B, $f_F(x)=g_F(x)+h_F(x)$, $f_B(x)=g_B(x)+h_B(x)$, где $h_F(x)$ и $h_B(x)$ — оценки минимальных стоимостей путей $\mu(x, \ldots, x_t)$ и $\mu(x_0, \ldots, x)$ соответственно.

1) Поместить x_0 в $\overline{S}$ и вычислить $f_F(x_0)$, поместить x_t в $\overline{T}$ и вычислить $f_B(x_t)$. Установить $a_{\min}=\infty$.

2) Выбрать направление F (продолжать) или B (к шагу 4).

3) Выбрать такую $x \in \overline{S}$, что $f_F(x) = \min_{y \in \overline{S}} f_F(y)$. Поместить x в $\overline{S}$ (изъяв ее из $\overline{S}$) и проверить все $x_i \in \Gamma(x)$ на следующие возможности:

а) Если $x_i \notin S \bigcup \overline{S}$, то поместить ее в $\overline{S}$.

б) Если $x_i \in \Gamma(x) \cap \overline{S}$, то сопоставить x_i наименьшую из старой и вновь полученной оценку $f_F(x_i)$.

в) Если $x_i \in \Gamma(x) \cap S$, то сопоставить x_i наименьшую из старой и вновь полученной оценку $f_F(x_i)$, поместить x_i в $\overline{S}$ (изъяв ее из S).

г)В остальных случаях не делать никаких изменений в S и $\overline{S}$.К шагу 5.

4) Выбрать такую $x \in \overline{T}$, что $f_B(x) = \min_{y \in \overline{T}} f_B(y)$. Поместить x в T (изъяв ее из $\overline{T}$) и проделать те же проверки, что и в шаге 3, относительно $x_i \in \Gamma^{-1}(x)$, используя T и $\overline{T}$. Продолжать.

5) Если $x \in S \cap T$, то $a_{\min} := \min(a_{\min}, g_F(x)+g_B(x))$. Если $a_{\min} \leqslant \max\left[\min_{x \in \bar{S}} f_F(x), \min_{x \in \bar{T}} f_B(x)\right]$, то выход алгоритма, $a_{\min}$ дает длину наилучшего пути. Иначе к шагу 2.

Трудности реализации двунаправленного поиска связаны с двумя вопросами: выбором метода определения направления генерации и оптимизацией «точки встречи» прямого и обратного поиска.

Одним из простейших правил выбора между направлениями *F* и *B* явилось бы правило эквидистантности от начальной и конечной вершины. Однако это привело бы нас фактически к малоэффективному поиску в ширину как в прямом, так и в обратном направлениях. Д. Поль включил в критерии выбора направления генерации размеры множеств $\bar{S}$ и $\bar{T}$, отражающие плотность дочерних вершин и сужающие поиск в случае предпочтения направления с меньшей мощностью множества. Правило выбора *«если* $|\bar{S}| \leq |\bar{T}|$, *то продолжать, иначе к шагу* 4», названное Полем *принципом сравнения мощности,* может быть подставлено в шаг 2 алгоритма двунаправленного поиска. Однако для случая бесконечного дизъюнктивного ветвления этот критерий не годится. В духе определения квазиконечного упорядочения качества было бы переформулировать правило выбора следующим образом: *«Если* $\bar{S}$ *содержит меньше вершин с наилучшим качеством, продолжать, иначе к шагу* 4».

Более сложным является вопрос об управлении точкой встречи прямого и обратного поиска. Мы должны разрешить противоречие, органически присущее эвристическому поиску: эвристическая функция вводится для того, чтобы сделать поиск более направленным, и с этой точки зрения целесообразно, чтобы стратегия поиска обеспечивала исследование относительно узких областей в окрестности оптимального пути. Однако при этом возникает риск, что пути, полученные в прямом и обратном направлениях, разойдутся. Поскольку ДПС допустима, то оптимальный путь будет получен, но ожидаемые преимущества двунаправленного поиска не будут реализованы. Более того, если встреча произойдет в районе начальной (конечной) вершины, то эффективность двунаправленного поиска будет ниже, чем у поиска в одном направлении. Таким образом, встает задача привести поисковые деревья к такой точке встречи, чтобы длина путей в направлениях *F* и *B* $\mu_F = \mu_B$. Один из способов решения этой задачи — определение промежуточной вершины x_i такой, что

$$h_F(x_i) \cong h_B(x_i) \cong {}^1/_2 h_F(x_0). \qquad (25)$$

Другими словами, мы пытаемся направить поиск к промежуточной вершине как в направлении *F*, так и в направлении *B*, что не означает, что встреча произойдет именно в этой точке. Это решение в духе редукционного подхода, напоминающее идею ключевых состояний и операторов (директив).

Другой возможностью является непрерывное обновление эвристической функции с целью направить поиск в одном из направлений к фронту поиска в противоположном направлении. Это может быть сделано изменением целевой вершины при каждой перемене направления поиска: в качестве целевой вершины выбирается последняя вершина, исследованная в противоположном направлении (т. е. последняя вершина, помещенная в список S или T).

Мы рассмотрим одну из возможностей управления эвристической функцией h_X с помощью оценочной функции f_Y, где *X* и *Y*— противоположные направления. Пусть произошел переход от генерации вершин в направлении *X* к генерации в направлении *Y (X* и *Y* могут быть как *F*, так и *B)*. Мы оцениваем ситуацию в поисковом пространстве с позиций оценочной функции $f_X(x)$, где *x*—последняя генерированная вершина в направлении *X*, т. е. $x \in S$, если $X=F$, и $x \in T$, если $X=B$. Предположим, что в направлении *Y* имеется множество вершин-кандидатов $y \in \overline{S}$, если $Y=F$, и $y \in \overline{T}$, если $Y=B$. Поиску в направлении *X* соответствует оценочная функция

$$f_X(x)=g_X(x)+h_X(x), \qquad (26)$$

в направлении *Y* —

$$f_Y(y)=g_Y(y)+h_Y(y) \qquad (27)$$

(аргументы в дальнейшем опускаем).

Рассмотрим два варианта:

1. С позиций f_X суммарный путь не дешевле g^*, где g^* — действительная стоимость, т. е. $f_X \le g_X+g_Y$. Учитывая (26), получаем для этого варианта $h_X \le g_Y$. В этом случае мы заключаем, что поиск со стороны *Y* зашел слишком далеко (заметим, что g_X — истинная стоимость оптимального пути в направлении *X)*. Следует переопределить оценочную функцию для $y \in Y$ таким образом, чтобы она выбирала для генерации те вершины, которые находятся ближе к фронту глубины g_X. Иначе говоря, следует, чтобы f_Y более реалистично оценивала суммарную длину g_X+g_Y. Для этого достаточно, чтобы $f_Y \ge f_X$, откуда, так как $h_X \le g_Y$, $h_Y \ge g_X$ для всех $y \in Y$. Если же для некоторых $y \in Y$ $h_Y < g_X$, то устанавливается $h_Y=g_X$.

2. С позиций f_X суммарный путь дешевле, чем g^*, т. е. $f_X > g_X + g_Y$ или $h_X > g_Y$. В этом случае, в силу ограниченности ДПС и того, что $x \in X$, $g_X+g_Y<f_X \le g^*$, так что f_X представляет собой нижнюю оценку наилучшего решения. Поэтому можно положить $f_Y \ge f_X$, откуда получаем

$h_Y \geq f_X - g_Y$. В случае невыполнения последнего неравенства устанавливается $h_Y = f_X - g_Y$. Мы объединим эти два варианта в один, записав

$$h_Y \geq g_X + (h_x \dot{-} g_Y), \qquad (28)$$

где

$$a \dot{-} b = \begin{cases} 0, & a \leqslant b, \\ a - b, & a > b. \end{cases}$$

Формула (28) дает нам возможность обновлять эвристическую функцию для поиска в направлении *Y* после каждого перехода к этому поиску от поиска в направлении *X*.

7.5. Поиск решения в пропозициональных графах

7.5.1. Алгоритм поиска минимального решающего графа

Мы ставим проблему решения задачи путем сведения ее к подзадачам как *поиск минимального решающего графа в неявно заданном пропозициональном графе.*

Определим понятие минимального решающего графа. С этой целью связываем с каждой дугой пропозиционального графа меру сложности, или стоимость дуги. Определим *стоимость графа* как сумму стоимостей всех его дуг. Тогда решающий граф, имеющий минимальную стоимость, называется *минимальным решающим графом.* Соответственно *минимальным путевым графом из* $s_1, s_2, .., s_k$ в $t_1, t_2, . . ., t_m$ называется путевой граф из $\{s_i\}$ в $\{t_l\}$, имеющий минимальную стоимость.

Введем ряд дополнительных определений. Мы по-прежнему называем применение оператора (директивы) сведения задачи к подзадачам, порождающее из некоторой вершины *s* ее дочерние вершины $s_1, s_2, . . ., s_k$, *генерацией, или раскрытием вершины s.* Очевидно, что конечная вершина никогда не раскрывается. Вершина, не являющаяся конечной и которая еще не генерирована, называется *кандидатом на генерацию.* Пусть $S = S_1 S_2 . . S_k$— конъюнкция. Конъюнкция называется *раскрытой* тогда и только тогда, когда раскрыты все не являющиеся конечными вершины $s_1, s_2, . . ., s_k$. Пусть $s=\{s_i/i=1, 2, . . ., q\}$ — множество начальных вершин и для s задана некоторая импликанта $P = N_1 N_2 . . .N_r$. Мы связываем с каждой такой импликантой оценочную функцию $f(P)=g(P)+h(P)$, где $g(P)$ — оценка стоимости минимального

путевого графа из вершин $s_1, s_2, \ldots, s_q$ в вершины $n_1, n_2, \ldots, n_r$, $h(P)$— оценка стоимости минимального решающего графа, начинающегося в вершинах $n_1, n_2, \ldots, n_r$. Соответствующие истинные значения обозначим через $g^*(P)$ и $h^*(P)$. Заметим, что в минимальном решающем графе

$$h^*(P) \leqslant \sum_{i=1}^{r} h^*(N_i) \tag{29}$$

(для пропозиционального дерева верно равенство).

Основываясь на оценочной функции $f(P)$, мы можем построить алгоритм поиска решения в пропозициональном графе. Стратегия поиска напоминает ДПС, но существенно отличается от нее способом образования кандидатов на генерацию. Однако благодаря этому мы сможем доказать допустимость и оптимальность этого алгоритма. **План дальнейшего изложения состоит в том, чтобы формально определить алгоритм, доказать его допустимость и оптимальность, а затем рассмотреть связь выбранной стратегии с ДПС и некоторые другие стратегии.**

В приведенном ниже алгоритме $W=\{W_j\}$ — множество конъюнкций, соответствующих вершинам-кандидатам на генерацию, $V(P)$ — множество импликант S, получаемых из предписаний P замещением каждой N_i, не являющейся конечной, одной из ее непосредственных импликант ($P = N_1N_2 \ldots N_r$), S— предписание, связанное с начальными вершинами, R — множество конъюнкций, соответствующих уже генерированным вершинам.

1) Положить $W=\{S\}$, $R = \varnothing$.

2) Вычислить $f(Q)$ для каждого $Q \in W$. Выбрать такое $P \in W$, что $f(P) = \min\limits_{Q \in W} f(Q)$. В случае равенства выбор произволен с предпочтением $Q \in T$, T — множество предписаний, соответствующих конечным вершинам.

3) Пусть $P = N_1N_2 \ldots N_r$, N_i—предписания, связанные с вершинами n_i, $i = 1,2, \ldots, r$. Если все n_i — конечные вершины, выход с решающим графом. Иначе раскрыть все нераскрытые вершины n_i, не являющиеся конечными.

4) Положить $R=R \cup \{P\}$, $W=(W \cup V) \backslash R$. Если $W= \varnothing$, выход, решающего графа нет. Иначе к шагу 2.

Пример. Рассмотрим граф на рис. 3, *а*.

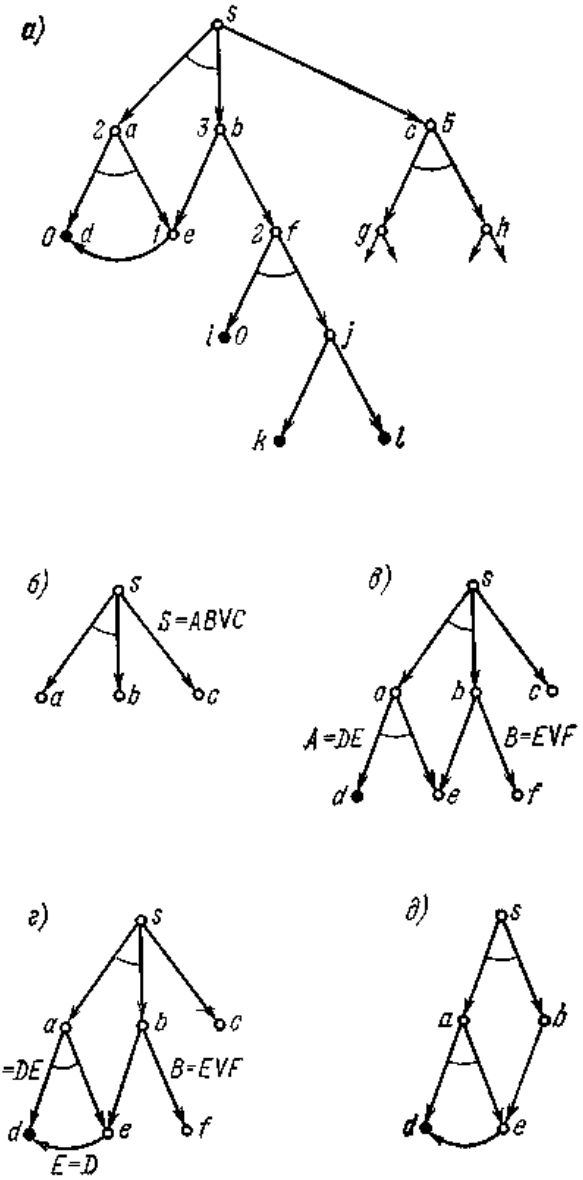


Рис. 3. Пример работы алгоритма: *а)* исходный граф, *б), в), г)* шаги алгоритма, *д)* решающий граф.

Положим $h(P) = r$ — 1+ min $\{h(N_1),h(N_2), ...,h(N_r)\}$. Числа рядом с вершинами n — оценки $h(N)$. Стоимости дуг положены равными единице. Последовательные шаги алгоритма представлены на рис. 3, *б, в, г* соответственно.

1) Раскрытие *s*. $W=\{AB, C\}$, $R = \{S\}$, $V=\{AB, C\}$.
$h(AB)= (2—1)+ \min (h(A), h(B)) = 1+\min(2, 3)=3$.
$f(AB)=2+3=5$, $f(C) = 1+5=6$.
Выбираем *AB* для дальнейшего раскрытия.

2) Раскрытие *a* и *b*. Поскольку $AB=DE(E+F)=DE+DEF$, то
$V=\{DE, DEF\}$, $R = \{S, AB\}$, $W= \{DE, DEF, C\}$.
$h(DE)=(2—1)+\min(h(D), h(E)) = 1+\min(0, 1) = 1$.
$h(DEF)=(3-1)+ \min (h(D), h(E), h(F))=2+\min(0,1, 2) =2$.
$f(DE)=5+1=G$, $f(DEF)=5+2=7$, $f(DE)=f(C)$.
Предположим, что для раскрытия выбрано *DE*.

3) Раскрытие *e*. Поскольку $DE=D$, то $V=\{D\}$, $R = \{S, AB, DE\}$, $W=\{D, DEF, C\}$, $h(D)=0$, $f(D)=6+0=6$, $f(D)=f(C)$.
D — конечная вершина, поэтому выбираем *D*. Алгоритм заканчивает работу с решающим графом, представленным на рис. 3, *д*.
Заметим, что в приведенном алгоритме не указаны некоторые детали, например, расстановка указателей для извлечения решения и т. д.

7.5.2. Свойства алгоритма поиска минимального решающего графа

Допустимость.

Лемма 1. *Пусть* $h(Q)\le h^*(Q)$ *для всех Q, являющихся импликантами S. Если имеется минимальный решающий граф, то алгоритм выбирает такую импликанту P, что* $f(P)\le f^*(S)$.

Доказательство. Пусть G_m— минимальный решающий граф, G_Q — частично раскрытый граф, $G_Q \subseteq G_m$, порожденный к моменту выбора *P*. Пусть $q_1, q_2, \ldots, q_r$ — конечные и нераскрытые вершины G_Q. Тогда $Q=Q_1Q_2 \ldots Q_r$— импликанта *S*, $Q \in W$. В силу минимальности G_m и $G_Q \subseteq G_m$ $f^*(S)=f^*(Q)$ и $g(Q)=g^*(Q)$. Так как $h(Q)\le h^*(Q)$, то

$$f(Q) \le g^*(Q)+h^*(Q)=f^*(Q)=f^*(S),$$

Но так как $f(P) = \min\limits_{Q\in W} f(Q)$, то

$$f(P)\le f(Q\le f^*(S). \qquad (30)$$

Назовем пропозициональный граф *G* δ-*графом,* если стоимость всех дуг *G* не менее, чем δ.

Теорема 6. *Пусть $h(Q) \le h^*(Q)$ для всех Q, являющихся импликантами S. Для всех δ-графов алгоритм поиска минимального решающего графа допустим.*

Доказательство. Предположим, что δ-граф G содержит минимальный решающий граф G_m. Докажем последовательно следующие три утверждения:

1) Алгоритм закончит работу.
2) Алгоритм закончит работу с решающим графом.
3) Алгоритм закончит работу с минимальным решающим графом.

1) Рассмотрим $Q \in W$, $Q=N_1N_2 \ldots N_n$ такую, что вершины $n_1, n_2, \ldots, n_r$ находятся на расстоянии d от ближайшей из начальных вершин. Тогда $f(Q) \ge d\delta$. Следовательно, если имеемся G_m с $f^*(S)$, для всех импликант Q с вершинами $n_1, n_2, \ldots, n_r$ удаленных более чем на $f^*(S)/\delta$ от ближайшей из начальных вершин, $f(Q) > f^*(S)$. Но по лемме 5.1 алгоритм выберет такую P, что $f(P) \le f^*(S)$. Следовательно, Q не будет выбрана алгоритмом, и поэтому он закончит работу.

2) Алгоритм не может закончить на шаге 4, так как имеется минимальный решающий граф, и $W \neq \varnothing$. Тогда остановка может произойти только на шаге 3, т. е. с решением.

3) Пусть T — импликанта, выбранная алгоритмом непосредственно перед окончанием его работы. По лемме 5.1 $f(T) \le f^*(S)$. Тогда $f^*(T) \le f(T) \le f^*(S)$. Поскольку $f^*(T)$ не превосходит минимальной стоимости, то $f^*(T)$ минимальна.

Оптимальность. Поскольку для доказательства допустимости алгоритма мы наложили ограничение $h(Q) \le h^*(Q)$, то очевидно, что в пределах этого ограничения, в зависимости от конкретных значений $h(Q)$, существует множество допустимых алгоритмов. Мы хотели бы выбрать из них такой алгоритм, который обладал бы наибольшей эффективностью в том смысле, что он раскрывал бы при нахождении минимального решения минимальное количество вершин. Таким образом, накладывая некоторые дополнительные ограничения на эвристическую функцию h, мы докажем оптимальность алгоритма в указанном выше смысле. Следует подчеркнуть, что вновь, как и в задачах поиска пути минимальной стоимости, оптимальность доказывается лишь в классе допустимых алгоритмов. Вводятся следующие дополнительные ограничения на эвристическую функцию:

1) Для всех P, являющихся импликантой S и содержащих высказывания, связанные только с конечными вершинами,

$$h(P) \equiv 0. \qquad (31)$$

2) Для любых Q, являющихся импликантой S, и P, являющихся импликантой Q,

$$h(Q) - h(P) \le k(Q, P), \qquad (32)$$

где $k(Q, P)$— стоимость минимального путевого графа из$\{q_i\}$ в$\{p_j\}$.

Лемма 2. *Если алгоритм, управляемый оценочной функцией, удовлетворяющей* (32), *выбирает импликанту P, то* $g(P)=g^*(P)$.

Доказательство. Предположим, что $g(P)>g^*(P)$. Пусть $P=P_1P_2 \ldots P_r$. Существует минимальный путевой граф G_0 из начальных вершин в $p_1, p_2, \ldots, p_r$, частично раскрытый в силу $g(P)>g^*(P)$. Пусть $q_1, q_2, \ldots q_m$— все конечные и нераскрытые вершины в G_0. Очевидно, что $Q \in W$, $Q=Q_1Q_2 \ldots Q_m$, а P является импликантой Q. Поскольку G_0—минимальный путевой граф, $g(Q)=g^*(Q)$. По предположению леммы и из определения функции $g^*(P)$

$$g(P)>g(Q)+k(Q, P). \qquad (33)$$

Добавляя $h(P)$ к общим частям неравенства (33), получаем $g(P)+h(P)>g(Q)+h(P)+k(Q, P)$ и в силу (32) $f(P)>f(Q)$. Но это означало бы, что алгоритм выберет импликанту Q, что противоречит условию леммы. Поэтому

$$g(P)=g^*(P). \qquad (34)$$

Теорема 7. *Пусть* $\sum_1$ *и* $\sum_2$ *— две допустимые стратегии, управляемые эвристическими функциями* h_1 *и* h_2 *соответственно. Если*

1) *для всех P, являющихся импликантами S и содержащих по крайней мере одно предписание, связанное с неконечной вершиной,* $h_1(P)>h_2(P)$;

2) *удовлетворяются ограничения* (31) *и* (32), *то для любого* δ*-графа, имеющего минимальный решающий граф, импликанта, выбираемая* $\sum_1$ *будет также выбираться* $\sum_2$.

Доказательство. Пусть $P_1, P_2, \ldots$— последовательность импликант, выбранных $\sum_1(P_1=S)$. Предположим, что теорема неверна. Тогда существует P_k такая, что она выбирается $\sum_1$, но не выбирается $\sum_2$. Тогда для $\sum_2$ $f_2(P_k) \geq f^*(S)$ или $h_2(P_k)>f^*(S)-g_2(P_k)$. Так как P_k— первая импликанта, выбранная $\sum_1$ и не выбранная $\sum_2$, то $\sum_2$, так же как и $\sum_1$ успела породить последовательность импликант $P_1P_2, \ldots, P_k$. Но по лемме 5.2 для $\sum_1$ $g_1(P_k) = g^*(P_k)$. Тогда по построению алгоритма для $\sum_2$ $g_2(P_k) = g^*(P_k)=g_1(P_k)$. Следовательно, $h_2(P_k) \geq f^*(S)—g^*(P_k)$. Для $\sum_1$ получаем (по лемме 5.1) $g_1(P_k)+h_1(P_k) \leq f^*(S)$, откуда, учитывая лемму 5.2, $h_1(P_k) \leq f^*(S)—g^*(P_k)$ и, наконец, $h_1(P_k) \leq h_2(P_k)$, что противоречит условию 1 теоремы.

Следствие. *При условиях теоремы* 7 *каждая вершина, раскрываемая* $\sum_1$ *раскрывается* $\sum_2$.

Мы уже упоминали выше, что рассмотренная стратегия поиска решающего графа напоминает ограниченную ДПС, определенную (6). Однако имеется одно существенное различие: алгоритм поиска решающего вывода в графе вывода на каждом шаге генерирует *только один* кандидат на вывод, т. е. по определению кандидата на вывод в направлении *B, точно один акт вывода и его посылки* присоединяются

к исходному РВ. В алгоритме раскрываются одновременно *все* еще не раскрытые вершины. В понятиях обобщенного предписания это означает, что кандидатом на вывод в направлении B является вывод D, если он содержит уже генерированный РВ $D_0 \subseteq D$, уже генерированные акты вывода, принадлежащие D_0, и для каждой посылки D_0— один акт вывода, который еще не генерирован и который имеет в качестве заключения эту посылку.

Смысл параллельного анализа всех нераскрытых вершин заключается в том, что если вывод (или, что то же самое, частично раскрытый путевой граф) зачислен в кандидаты на генерацию с определенной оценкой, то он будет иметь эту оценку все время, пока не будет раскрыт (или пока алгоритм не закончит работу). Это условие всегда удовлетворяется для поиска в пространстве состояний. Однако структура пространства поиска в пропозициональных графах существенно сложнее, и в общем случае вывод, который на данном шаге является кандидатом на генерацию, затем может перестать быть им и не будет генерирован на более поздних стадиях, поскольку данный вывод может стать частью вывода, который имеет худшее качество, чем другие кандидаты.

Оптимальность алгоритма покупается за счет параллельного исследования всех вершин, дочерних по отношению к конъюнктивной, и поэтому он весьма близок к классу алгоритмов поиска в ширину, т. е. алгоритмов, весьма слабо управляемых эвристической функцией. С этой точки зрения он является относительно неэффективным, а оптимальность его не играет большой роли, так как доказана в классе допустимых алгоритмов, также слабо управляемых функцией h.

Мы имеем две возможности сделать поиск решающего графа более направленным. Первая из них заключается в том, чтобы модифицировать алгоритм, допустив раскрытие только одной вершины. Легко сделать это, внеся изменения в шаге 3 и в определение множества V исходного алгоритма.

3) Пусть $P=N_1N_2 \ldots N_r$, N_i — предписания, связанные с вершинами n_i, $i=1,2, \ldots, r$. Если все n_i— конечные вершины, выход с решающим графом. Иначе раскрыть нераскрытую вершину n_k, $1 \le k \le r$, такую, что $f(N_k) = \min\limits_i f_k(N_i)$.

Определение множества *V*. $V(P)$ — множество импликант S, получаемых из предписаний P замещением каждой раскрытой N_i, не являющейся конечной, одной из ее непосредственных импликант ($P=N_1N_2 \ldots N_r$).

Легко видеть, что с этими модификациями алгоритм становится частным случаем алгоритма поиска решающего вывода в графе

вывода. Как показано в п.7.2, этот алгоритм, управляемый оценочной функцией $f(P)=g(P)+h(P)$, допустим. Однако оптимальность этого алгоритма не может быть установлена по следующей причине. При доказательстве оптимальности необходимо показать, что если акт вывода генерируется стратегией $\sum_1$ с эвристической функцией h_1, то он также генерируется и $\sum_2$, управляемой такой эвристической функцией h_2, что $h_2<h_1$. Далее, для $\sum_2$ надо показать, что этот акт принадлежит кандидату на вывод лучшего качества, чем минимальное решение, и поэтому он генерируется перед окончанием алгоритма. Этот прием работает для тех случаев, когда выбранный в качестве кандидата вывод остается кандидатом неизменного качества, пока он не будет выбран для генерации (поиск пути минимальной стоимости, алгоритм). В нашем же случае качество вывода может стать хуже и тогда акт вывода, принадлежащий ранее кандидату на вывод, имевшему стоимость, меньшую стоимости минимального решения, будет теперь принадлежать только кандидатам с худшим, чем у решения, качеством. Вторая возможность повышения эффективности поиска—управление с помощью оценочной функции вида $h(P)$, т. е. без учета стоимости уже построенного частично раскрытого путевого графа.

7.5.3. Поиск решающего графа в аддитивном пропозициональном графе

Рассмотрим другое возможное обобщение алгоритма поиска минимального решающего дерева. Это обобщение, называемое *поиском решающего графа в аддитивном пропозициональном графе,* представляет интерес по нескольким причинам. Во-первых, на этом алгоритме мы продемонстрируем отличный от импликативного метод определения разрешимых вершин и решающих графов, хорошо реализуемый в списочных структурах, — так называемый **метод указателей**. Во-вторых, на примере работы этого алгоритма мы проиллюстрируем изложенный выше тезис об ухудшении качества вывода при раскрытии одной вершины, дочерней для конъюнктивной и имеющей минимальную стоимость. В-третьих, рассмотрение этого алгоритма даст нам повод к обсуждению проблемы уменьшения избыточности при представлении редукционных систем графами, а не деревьями.

Предположим, что нам дано пропозициональное дерево. Естественно, что одинаковые подзадачи, возникающие в ходе разбиения задач более высокого уровня, представляются в дереве различными вершинами. Аддитивный пропозициональный граф может быть получен из такого дерева, если

1) Одинаковые подзадачи в дереве идентифицируются как корни одних и тех же поддеревьев и представляются одной вершиной в графе без циклов.
2) Стоимость решающего подграфа приравнивается стоимости соответствующего решающего поддерева, т. е. мера сложности решения идентифицированных подзадач вычисляется один раз, но прибавляется к стоимости решающего подграфа столько раз, сколько она встречается.
На рис. 4, *а* представлен пример аддитивного пропозиционального графа, а на рис. 4, *б, в* даны его решающие графы.

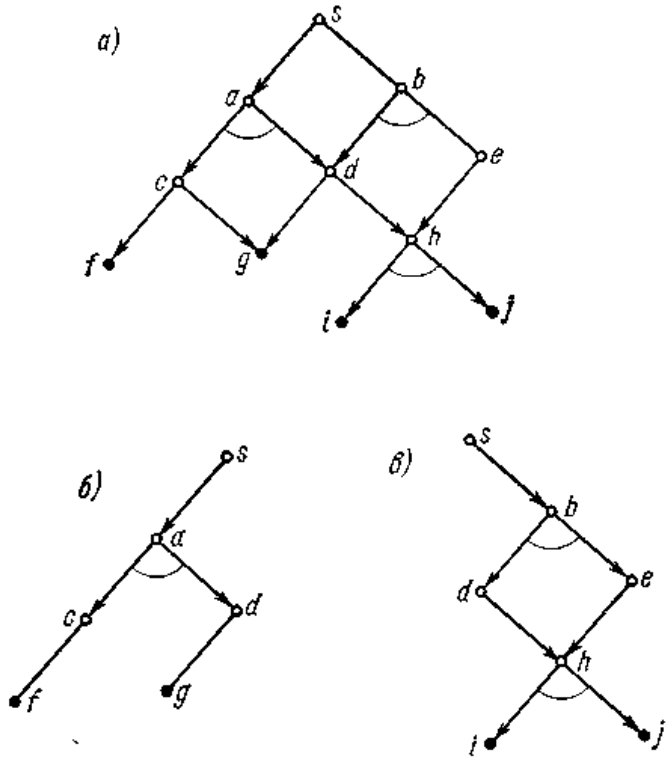


Рис. 4. Аддитивный пропозициональный граф *(а)* и его решающие графы *(б, в)*.

В таблице 3 показаны стоимости этих решающих графов, подсчитанные для общего вида пропозиционального графа и

аддитивного пропозиционального графа, в предположении единичных стоимостей дуг.

Таблица 3

Вид пропозиционального графа	Стоимость решающего графа	
	рис. 5.4, *б*	рис. 5.4, *в*
Общий вид (сумма стоимостей всех дуг)	5	7
Аддитивный (сумма стоимостей решающих поддеревьев)	5	9

Заметим, что неравенство (29) в последнем случае превратилось бы в равенство в силу указанного в пункте 2.

В общем, *стоимость решающего графа* определяется рекурсивно следующим образом:

1) С каждой конечной вершиной связывается стоимость $C = 0$.

2) Стоимость, связанная с любой другой вершиной x,

$$C(x) = \sum_{i=1}^{k} (C_i + c(x, x_i)) \qquad (35)$$

как для дизъюнктивных, так и для конъюнктивных вершин. Здесь x_i, i=l, 2, . . ., k,— дочерние вершины вершины x, $c(x, x_i)$— стоимость дуги, связывающей x с x_i.

3) Стоимость, связанная с начальной вершиной, есть стоимость решающего графа.

Будем считать также, что все вершины аддитивного пропозиционального графа либо конъюнктивные, либо дизъюнктивные (ранее мы допускали, чтобы одна и та же вершина могла быть по отношению к одним дочерним вершинам конъюнктивной, а по отношению к другим — дизъюнктивной). Остальные понятия являются стандартными для пропозициональных графов.

Решающий граф пропозиционального графа обладает следующими свойствами:

1) Если дизъюнктивная вершина включена в решающий граф, то *одна и только одна* ее дочерняя вершина также включена в решающий граф.
2) Если конъюнктивная вершина включена в решающий граф, то *все* ее дочерние вершины также включены в решающий граф.

Мы уже упоминали выше, что для указания частично построенного графа минимальной стоимости используется метод указателей. Это означает, что на каждом шаге цепочка указателей, начиная от начальной вершины, определяет текущего кандидата в решающий граф и следующую генерируемую вершину. Эта же цепочка указателей используется для извлечения ответа в конце работы алгоритма. Образование цепочки указателей управляется обновлением оценок вершин, предшествующих раскрываемым во время работы алгоритма. Заметим, что, в отличие от алгоритма, где раскрывались все вершины, данный алгоритм существенно зависит от работы механизма образования указателей.

За оценку стоимости вершины мы будем принимать стоимость минимального решающего графа, начинающегося в этой вершине. Эта оценка является эвристической функцией $h(x)$.

Алгоритм поиска решающего графа в аддитивном пропозициональном графе представляется следующими шагами:

1) Выбрать начальную вершину x_0.
2) Если x_0 разрешима, то выход с решающим графом, получаемым, начиная с x_0, прослеживанием направлений указателей вплоть до конечных вершин. Иначе продолжать.
3) Проследить указатели от x_0 к вершине x и раскрыть ее. Пусть вершины, дочерние по отношению к x, — x_i , i =l,2, ...,k.
4) Обновить оценки и направления указателей вершины x и всех вершин, предшествующих x, по следующим правилам:

а) если x_i уже раскрывалась ранее, она уже имеет оценку $h(x_i)$;

б) если x_i раскрыта впервые, то если x_i— конечная вершина, $h(x_i)=0$, то x_i отмечается как разрешимая; если x— дизъюнктивная вершина, $h(x) = \min\limits_i (h(x_i) + c(x, x_i))$, то направить указатель от x к x_i и в случае разрешимости x_i отметить вершину x как разрешимую; если x— конъюнктивная вершина, $h(x) = \sum\limits_{i=1}^{k} (h(x_i)+c(x, x_i))$, то направить указатель от x к той из не отмеченных как разрешимая вершин x_i, которая имеет наибольшую оценку, и в случае разрешимости всех x_i отметить x как разрешимую;

в) повторить процедуру обновления оценок и направления указателей для всех вершин, предшествующих *x,* вплоть до начальной вершины. К шагу 2.

Сделаем ряд замечаний к этому алгоритму.

1) Мы выбираем направления указателя от конъюнктивной вершины к той из ее дочерних вершин, которая имеет наибольшую оценку. Эвристическое основание для такого выбора состоит в том, что мы хотим как можно раньше опровергнуть гипотезу о том, что построенный к настоящему моменту частично раскрытый граф является верхней частью решающего графа. Действительно, если конъюнктивная вершина *x* на самом деле входит в решающий граф, то порядок раскрытия ее дочерних вершин безразличен, поскольку придемся раскрыть все дочерние вершины. Но если это не так, то желательно отбросить *x* и граф, начинающийся с *x,* как можно быстрее. Мы предполагаем, что выбор вершины *x* с максимальной оценкой в наибольшей степени повысит суммарную стоимость построенного графа.

2) Поскольку в аддитивном графе между двумя вершинами может быть больше одного пути, обновление оценки в данной вершине может происходить несколько раз. Тем не менее эта процедура закончится, поскольку граф не имеет циклов.

На рис. 5 показан пример поиска минимального решающего графа в графе (рис. 5, *а*). Стоимости дуг приравнены единице. Числа, стоящие рядом с вершинами, являются оценками стоимости минимального решающего графа, начинающегося с этой вершины. Разрешимые вершины отмечены жирными точками. На рис. 5,*б*—5,*ж* показаны шаги алгоритма с обновленными оценками, указателями и последовательной отметкой разрешимых вершин. На рис. 5, *з* представлен решающий граф.

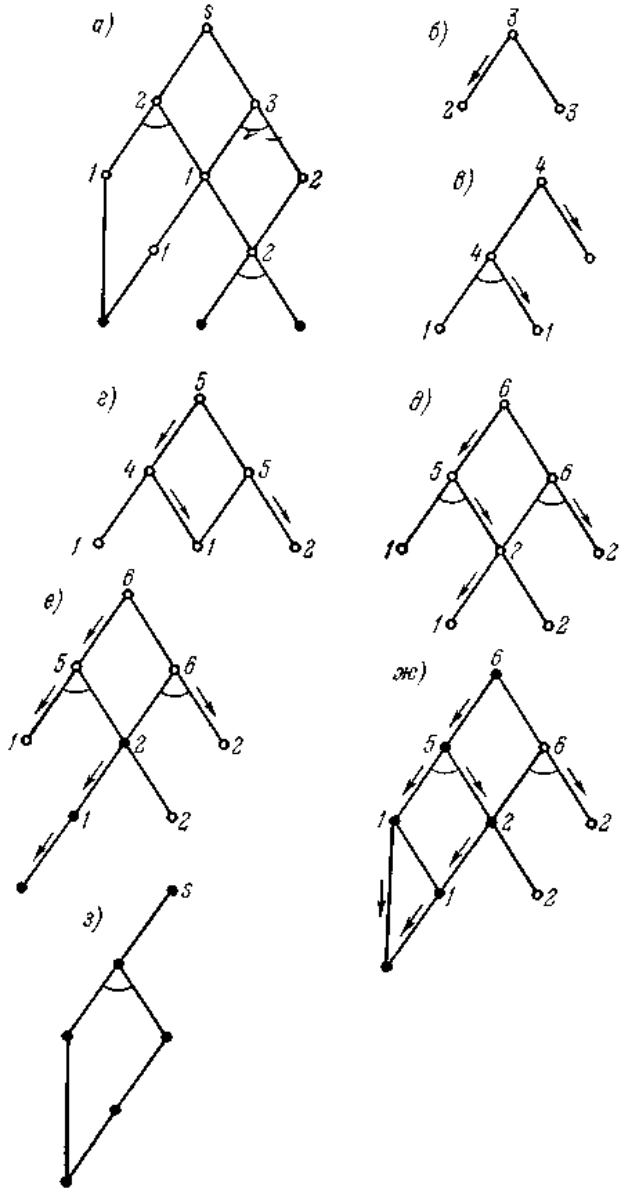


Рис. 5 Пример работы алгоритма а) исходный граф, *б), в), г), д), е), ж)* шаги алгоритма, *з*) решающий граф.

Использование в рассмотренных алгоритмах пропозициональных графов вместо деревьев устраняет избыточность, возникающую, когда порождаются идентичные подзадачи, а дальнейший поиск идет так, как будто бы они являются различными. Это приводит в дальнейшем к независимому, выращиванию эквивалентных поддеревьев, соответствующих сведению идентичных задач к их подзадачам. Таким образом, предписание в виде пропозициональных деревьев может в значительной степени снизить эффективность алгоритмов. В то же время использование графов требует распознавания и идентификации одинаковых подзадач, что может представить некоторые трудности чисто вычислительного характера
Именно такой вид избыточности устраняется в аддитивных пропозициональных графах. Существует и другой вид избыточности. Он возникает, когда имеются различные сведения первоначальной задачи к одним и тем же подзадачам или, в терминологии обобщенного представления, различные выводы *D* имеют одинаковые посылки. Задачу можно поставить в более общем виде устранение избыточности в случае, когда множество подзадач, полученное с помощью одного вывода, является подмножеством множества подзадач, полученного с помощью другого вывода По-видимому, эта задача не может быть решена в связи с тем, что одни и те же акты вывода могут одновременно принадлежать различным выводам. Устраняя избыточный вывод путем ликвидации соответствующего акта вывода, мы можем одновременно потерять другие выводы, которые, возможно, затем окажутся кандидатами наилучшего качества.

8. Решения задач методами доказательства теорем

8.1. Структура процедур доказательства теорем

Ранее мы отмечали, что для решения многих задач может потребоваться логический анализ. Для этих целей мы ввели формальный язык — исчисление предикатов, на котором можно формулировать исходные утверждения и делать из них логические выводы. Благодаря общности и логической силе язык исчисления предикатов может претендовать на использование его в машинном построении умозаключений для широкого класса задач.

В этом разделе мы сначала опишем теоретическую модель, являющуюся основой для построения многих практических методов, а затем приведем наиболее эффективные методы доказательства теорем . Будем рассматривать процедуру доказательства теорем как пару (T, ∑), где T— аксиомы и правила вывода, а ∑ — стратегия поиска для T.

Задача стратегии поиска состоит в том, чтобы выбрать из всего множества формул те, к которым на очередном шаге процедуры доказательства надо применять правило вывода. **Задача правила вывода — получить непосредственное следствие из выбранного множества формул.**

Увеличение эффективности программ, доказывающих теоремы, осуществляется в основном за счет использования в процессе доказательства семантической информации, встраивания специфических свойств конкретных теорий в правила вывода и алгоритмы унификации и предоставления пользователю возможностей вмешательства в процесс доказательства.

В следующем параграфе мы рассмотрим теоретические основы, используемые при построении большинства программ доказательства теорем. Изложение материала будет основываться на принципе резолюции, позволяющем на одной теоретической базе рассмотреть основные проблемы, возникающие при доказательстве теорем.

8.2. Теоретические основы построения программ доказательства теорем

Ранее задача доказательства теорем определена нами как в понятиях выполнимости, так и в понятиях выводимости. На практике будем использовать оба подхода. В данном разделе изложение строится на основе понятия выполнимости. При этом подходе задача доказательства теорем состоит в определении невыполнимости множества дизъюнктов, полученных из исходного множества формул исчисления предикатов.

Невыполнимость множества S дизъюнктов обозначает, что *S* ложно (т. е. конъюнкция дизъюнктов множества *S* принимает значение *F)* при всех интерпретациях на любых областях. Данное определение не является конструктивным, так как невозможно рассмотреть все интерпретации на любых областях. Однако существует одна специальная область *H* такая, что *S* является невыполнимым тогда и только тогда, когда S является ложным при всех интерпретациях на этой области. Эта область называется *универсумом Эрбрана* для множества *S* и определяется следующим образом.

Пусть H_0—множество констант, появляющихся в множестве S дизъюнктов. Если в S нет констант, то в H_0 включается некоторая константа, например, $H_0=\{a\}$. Для i=0,1,2, . . .H_{i+1} есть объединение H_i и множества всех понятийных термов в форме $f^n(t_1, . . ., t_n)$ для всех n-местных функций f^n, встречающихся в S, где t_j, j=1, . . ., n, есть члены множества H_i. Тогда будем $H=H_\infty$, называть *эрбоановским универсумом* S, а H_i — его уровнем i.

Пример 1. Пусть

$S = \{P (x) \vee Q (a) \vee \sim P (f (x)), \sim Q (b) \vee P (g(x))\}$.

Тогда

$H_0 = \{a, b\}$,

$H_1\{a, b, f(a), f(b), g(a), g(b)\}$,

$H_2 = \{a, b, f(a), f(b), g(a), g(b), f(f(a)), f(f(b)), f(g(a)), f(g(b)), g(f(a)), g(f(b)), g(g(a)), g(g(b))\}$, ...

Пусть S — множество дизъюнктов, C — дизъюнкт множества S. *Фундаментальным примером дизъюнкта* C называется дизъюнкт, полученный замещением переменных (понятий) в C членами эрбрановского универсума S таким образом, что все вхождения одной и той же переменной (понятием) в C заменяются на один и тот же терм. *Эрбрановской базой* множества S дизъюнктов называется множество всех фундаментальных примеров атомов в S.

Пример 2. Пусть $S = \{R(x), P (g(y)) \vee Q(y)\}$. Определим для данного множества S эрбрановский универсум, фундаментальный пример дизъюнкта $R (x)$ и эрбрановскую базу.

Эрбрановский универсум для S равен

$$H=\{a, g(a), g(g(a)), ...\}.$$

Фундаментальным примером дизъюнкта $R(x)$ является, например, дизъюнкт

$$R(g(a)).$$

Эрбрановской базой (A) для S является множество:

$$A = \{R(a), P(a), Q(a), R(g(a)), P(g(a)), Q(g(a)), ...\}.$$

Определим над эрбрановским универсумом для S специальный вид интерпретации, называемый H-интерпретацией для S.

Пусть S —множество дизъюнктов, H—эрбрановский универсум для S и I — интерпретация S над H. Интерпретация I называется *H-интерпретацией* S тогда и только тогда, когда она удовлетворяет следующим условиям:

1. I отображает все константы множества S сами в себя.

2. Пусть f есть n-местная функциональная буква и $h_1, . . ., h_n$ суть элементы H. В интерпретации I функциональной букве (понятию) f ставится в соответствие функция, которая отображает $(h_1, . . ., h_n)$ в $f(h_1, . . ., h_n)$ (т. е. отображает H^n в H).

H-интерпретация не накладывает ограничений на соответствия, устанавливаемые для *n*-местных предикатных букв в *S*. Пусть $A = \{A_1, A_2, .. ., A_n, . . .\}$ является эрбрановской базой для *S*. Тогда *H* -интерпретация *I* может быть представлена множеством:

$$I=\{m_1, m_2, . . ., m_n, . . .\},$$

в котором m_j есть A_j или $\sim A_j$ для $j = 1,2, . . .$ При этом, если m_j есть A_j, то A_j назначается значение *T,* иначе A_j назначается значение *F*,

Пример 3. Приведем для множества S из примера 2 некоторые *H*-интерпретации:

$I_1=\{R(a), P(a), Q(a), R(g(a)), P(g(a)), Q(g(a)), ...\}$,

$I_2= \{\sim R(a), \sim P(a), \sim Q(a), \sim R(g(a)), \sim P(g(a)), \sim Q(g(a)), ...\}$,

$I_3 = \{ R(a), \sim P(a), Q(a), R(g(a)), \sim P(g(a)), Q(g(a)),... \}$.

Для компактности будем записывать I_1 в виде $I_1= \{R(x), P(x), Q(x)\}$ или $I_1=\{R, P, Q\}$. Аналогично $I_2= \{\sim R,\sim P, \sim Q\}$, а $I_3= \{ R, \sim P, Q\}$.

Справедлива следующая теорема.

Теорема 1. *Множество S дизъюнктов невыполнимо тогда и только тогда, когда S ложно при всех H-интерпретациях S.*

Таким образом, для установления невыполнимости множества дизъюнктов достаточно рассматривать не все его интерпретации, а только *H*-интерпретации.

Приведем без доказательства теорему Эрбрана.

Теорема 2. *Множество S дизъюнктов невыполнимо тогда и только тогда, когда существует конечное невыполнимое множество фундаментальных примеров дизъюнктов в S.*

Приведенная формулировка теоремы Эрбрана является основой для процедуры опровержения (установления невыполнимости множества дизъюнктов). Действительно, будем для множества *S* дизъюнктов генерировать множества $S_1, . . ., S_n, . . .$, где S_i есть множество всех фундаментальных примеров (предписаний) $S(H_i)$, полученных замещением переменных (понятий) в S константами из уровня *i* множества *H*, для S. Будем последовательно проверять на ложность множества S_i, i=l,2, ... Теорема Эрбрана гарантирует, что если множество невыполнимо, то данная процедура обнаружит такое *n,* что S_n является ложным.

Основным комбинаторным препятствием для достижения эффективности подобных процедур является огромная скорость роста конечных множеств S, с ростом *i.* Для каждого конечного множества *S* дизъюнктов, которое невыполнимо и для которого фактически можно построить опровержение, существует по крайней мере одно конечное подмножество *P* эрбрановского универсума для *S,* имеющее приемлемые размеры, такое, что S*(P)* невыполнимо, причем *P*

минимально (в том смысле, что S(Q) выполнимо для любого собственного подмножества Q множества *P).*
Ниже мы опишем основную идею принципа резолюции, являющегося теоретической базой для построения большинства методов доказательства теорем.
Введем необходимые определения.

8.2.1. Алфавитный порядок символов

Будем говорить, что множество всех символов *упорядочено в алфавитном порядке,* если
1) символы расположены в таком порядке: понятия, директивы, переменные, функции, предикаты, связка отрицания;
2) понятия расположены в алфавитном порядке;
3) директивы расположены в алфавитном порядке;
4) символ функции (предиката) меньшей размерности предшествует символам функций (предикатов) большей размерности, а при равных размерностях функции и предикаты упорядочены в алфавитном порядке предикатных и функциональных букв;
5) переменные расположены в алфавитном порядке.

8.2.2. Лексикографический порядок выражений

Выражением будем называть понятийные термы и функциональные литеры. Множество всех выражений *упорядочено в лексикографическом порядке* посредством следующего правила: короткие выражения предшествуют длинным, а при равенстве длин выражений A и B раньше ставится выражение, у которого первый (первые) символ имеет ранний алфавитный порядок.

8.2.3. Подстановочные компоненты

Подстановочная компонента — это выражение t/x, где x — переменная, t - терм, отличный от x, причем x называют переменной компоненты t/x, а t — термом компоненты.

8.2.4. Подстановки

Множество подстановочных компонент попарно различными переменными называется *подстановкой* и записывается в виде $\alpha=\{t_1/x_1, \ldots, t_n/x_n\}$. Применение подстановки к некоторой формуле A

обозначает замену всех вхождений в A переменной x_i, $1 \le i \le n$, на вхождение терма t_i. Получившуюся формулу будем обозначать $A\alpha$ и называть α-примером A.
Например, применив к формуле $R(x, f(y))$ подстановку $\alpha=\{\varphi(a)/x, f(z)/y\}$, получим α-пример $R(\varphi(a), f(f(z)))$.

8.2.5. Композиция подстановок

Композицией $\alpha\beta$ *двух подстановок* α и β называется результат применения β к термам подстановки α с добавлением из β всех подстановочных компонент, содержащих переменные, отсутствующие в α. Например, если $\alpha=\{f(x, y)/z\}$, $\beta=\{a/x, b/y, c/w, d/z\}$, то

$$\alpha\beta = \{f(a, b)/z, a/x, b/y, c/w\}.$$

Можно показать, что применение к литере P последовательности подстановок α и β дает тот же результат, что и применение к P подстановки αβ. Можно также показать, что композиция подстановок ассоциативна: $(\alpha\beta)\gamma=\alpha(\beta\gamma)$.

8.2.6. Унификация

Множество литер $\{L_i\}$ называется *унифицируемым,* если существует такая подстановка α, что в результате применения ее к каждому элементу $\{L_i\}$ получаем, что $L_1\alpha=L_2\alpha=\ldots$ В этом случае подстановку α называют *унификатором* для $\{L_i\}$ и обозначают $\{L_i\}_\alpha$. Например, подстановка $\alpha=\{f(a)/x, b/y\}$ унифицирует множество литер $\{R(f(y), x), R(f(b), f(a))\}$ и дает в качестве ответа $\{R(f(b), f(a))\}$. *Наиболее общим* (простейшим) *унификатором* (НОУ) для $\{L_i\}$ называется такой унификатор α, что если β — какой-нибудь унификатор для $\{L_i\}$, дающий $\{L_i\}_\beta$, то найдется подстановка δ, для которой $\{L_i\}_{\alpha\beta} = \{L_i\}_\beta$. С точностью до алфавитных вариантов существует *единственный* НОУ.

8.2.7. Алгоритм унификации

Алгоритм, который находит НОУ для унифицируемого множества литер $\{L_i\}$ и сообщает о неудаче, если множество неунифицируемо, будем называть *алгоритмом унификации.* Приведем описание работы алгоритма.

1. Положить $k=0$ и $\sigma_k=\varepsilon$ (пустая подстановка). Перейти к пункту 2.
2. Если $\{L_i\}_{\sigma_k}$ не является одноэлементным множеством, то перейти к пункту 3. В противном случае положить $\sigma = \sigma_k$ и закончить работу.

3. Каждая из литер в $\{L_i\}_{\sigma_k}$ рассматривается как цепочка символов, и выделяются первые подпредписания литер, не являющихся одинаковыми у всех элементов $\{L_i\}_{\sigma_k}$. Указанные подпредписания, расположенные в лексикографическом порядке, образуют множество рассогласования B_k для $\{L_i\}_{\sigma_k}$. Пусть V_k—самый первый (левый), а U_k — следующий за ним элемент B_h. Тогда, если V_k является переменной, не входящей в U_k, то $\sigma_{k+1}=a_k\{U_k/V_k\}$, $k=k+1$ и перейти к пункту 2; в противном случае окончить работу.

Можно показать, что описанный процесс всегда завершается.

Поясним работу алгоритма на примере. Пусть $\{L_i\} = \{P(x, z, v), P(x, k(y), y), P(x, z, b)\}$. На первом шаге работы алгоритма будет получено множество рассогласования $B_0=\{z, k(y)\}$ и подстановка $\sigma_1= \{k(y)/z\}$. На втором шаге $\{L_i\}\sigma_1$ не является одноэлементным множеством. $\{L_i\}\sigma_1 = \{P(x, k(y), v), P(x, k(y), y), P(x, k(y), b)\}$, множество $B_1=\{ v, y, b\}$, $\sigma_2=\{k(y)/z, y/v\}$. На третьем шаге $\{L_i\}\sigma_2 =\{P(x, k(y), y), P(x, k(y), y), P(x, k(y), b)\}=\{P(x, k(y), y), P(x, k(y), b)\}$, $B_2= \{y, b\}$, $\sigma_3=\{k(y)/z, y/v, b/y\}$. На четвертом шаге $\{L_i\}\sigma_3=\{P(x, k(b), b)\}$ является одноэлементным множеством, а наиболее общим унификатором является подстановка

$$\{k(y)/z, y/v, b/y\}.$$

Дизъюнкт можно рассматривать как множество литер $\{L_i\}$. Если подмножество литер некоторого дизъюнкта $\{L_i\}$ унифицируемо с помощью НОУ α, то $\{L_i\}_\alpha$ называется *фактором* $\{L_i\}$. У одного дизъюнкта может быть несколько факторов, но число их конечно. Например, факторами дизъюнкта

$R(g(x)) \vee R(x) \vee Q(a, g(u)) \vee Q(x, g(b)) \vee Q(z, w)$

являются дизъюнкты

$R(g(z)) \vee R(z) \vee Q(a, g(u)) \vee Q(z, g(b))$, $R(g(a)) \vee R(a) \vee Q(a, g(b))$.

Первый фактор получен унификацией двух последних вхождений литеры Q в исходный дизъюнкт, а второй — трех. В приведенном предложении два вхождения литеры R нельзя унифицировать, так как в множестве рассогласования $\{x, g(x)\}$ переменная x входит в терм $g(x)$, что противоречит требованию, содержащемуся в пункте 3 алгоритма унификации.

8.2.8. Резольвента.

Пусть $\{L_i\}$ и $\{M_i\}$ — два дизъюнкта, не имеющие общих переменных (это можно всегда получить переименованием переменных). Пусть $\{l_i\}$ и $\{m_i\}$—такие два подмножества $\{L_i\}$ и $\{M_i\}$, что

1) $\{l_i\} \subseteq \{L_i\}$ и $\{m_i\} \subseteq \{M_i\}$;

2) для $\{l_i\}\ U\{\sim m_i\}$ существует наиболее общий унификатор α, т. е. $\{m_i\}_\alpha$ и $\{l_i\}_\alpha$ являются дополнительными.

Тогда говорят, что исходные дизъюнкты *разрешаются,* и из них выводим новый дизъюнкт, называемый *резольвентой:*

$$[\{L_i\}\text{-}\{l_i\}]_\alpha \bigcup [\{M_i\}\text{-}\{m_i\}]_\alpha.$$

Два дизъюнкта могут иметь более одной резольвенты, так как способ выбора $\{l_i\}$ и $\{m_i\}$ может оказаться не единственным. Если условия 1 и 2 не соблюдаются, то дизъюнкты не имеют резольвент.

Поясним процесс образования резольвент на примере. Пусть даны два дизъюнкта:

$$\{L_i\} = R\{y, g(a)) \vee R(y, g(x)) \vee P(x),$$
$$\{M_i\} = \sim R\ (z, g(a)) \vee \sim P(z).$$

Выбирая в качестве $\{l_i\}$ и $\{m_i\}$ соответственно $\{R\ (y,g\ (a))\}$ и $\{\sim R(z, g(a))\}$, мы получаем резольвенту $R(z,g(x)) \vee P(x) \vee \sim P(z)$. Если в качестве $\{l_i\}$ и $\{m_i\}$ выбрать соответственно $\{R(y, g(a)), R(y, g\{x))\}$ и $\{\sim R(z, g(a))\}$, то резольвентой будет $P(a) \vee \sim P(z)$. Всего для этих двух дизъюнктов можно образовать четыре резольвенты.

8.2.9. Резолюция

Пусть S — множество дизъюнктов. Будем называть *резолюцией* правило вывода, генерирующее резольвенты из множества S дизъюнктов. Объединение S с множеством всех резольвент, которые могут быть образованы из дизъюнктов, входящих в S, будем обозначать $R(S)$. Обозначим $R_0(S)=S$ и определим $R_{n+1}(S)=R(R_n(S))$ для $n\geq 0$. Справедлива следующая теорема, устанавливающая полноту логической системы.

Теорема 3. *Если S — произвольное конечное множество дизъюнктов, то S невыполнимо тогда и только тогда, когда* $R_n(S)$ *содержит для некоторого* $n\geq 0$ *пустой дизъюнкт.*

Итак, мы определили формальную систему, использующую одно правило вывода (правило резолюции) и не требующую логических аксиом.

На основе приведенной выше теоремы можно построить процедуру опровержения. Эта процедура состоит в вычислении для конечного множества S дизъюнктов последовательности множеств S, $R_1(S)$, $R_2(S)$, ... и может закончиться одним из трех исходов.

1. $R_n(S)$ содержит пустой дизъюнкт и, следовательно, множество S невыполнимо.
2. $R_n(S)$ совпадает с $R_{n+1}(S)$ и, следовательно, S выполнимо.

3. Процедура для определенных множеств S продолжает работу бесконечно в связи с неразрешимостью исчисления предикатов.

В некоторых случаях бесконечный процесс можно распознавать и прерывать работу.

Процесс опровержения, использующий принцип резолюции, удобно представлять в виде графа. Вершинам графа соответствуют дизъюнкты. Дизъюнкты исходного множества изображаются в виде *концевых вершин,* т. е. вершин, не имеющих предшественников. Если два дизъюнкта образуют резольвенту, то из этих дизъюнктов проводятся ребра к вершине—дизъюнкту, соответствующему выведенной резольвенте. Вершина графа, у которой нет следующих за ней вершин, называется *корневой вершиной.* Она соответствует дизъюнкту, выведенному этим графом. Пустой дизъюнкт будем обозначать символом *NIL.* Принято корень графа изображать внизу, а исходные дизъюнкты вверху. На рис. 1 приведен пример представления процесса опровержения в виде графа для невыполнимого множества дизъюнкюв $\{R(x) \vee \sim Q(x) \vee \sim P(x), R(x) \vee P(x), R(x) \vee Q(x), \sim R(x)\}$.

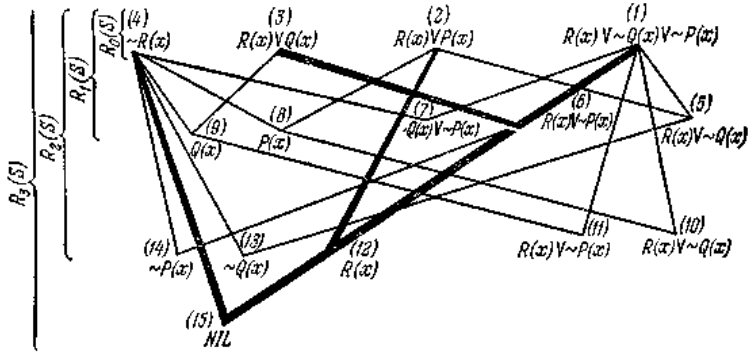


Рис. 1. Поиск опровержения с использованием бинарной резолюции.

На графе изображены дизъюнкты, получаемые в процессе применения бинарной резолюции к исходному множеству.

Процедура опровержения, построенная непосредственно на основе теоремы о резолюции, является не вполне эффективной, так как множества $R_1(S)$, $R_2(S)$,... очень быстро разрастаются. В следующих параграфах мы опишем практические процедуры доказательства, сокращающие перебор как за счет введения правил, ограничивающих принцип резолюции (8.3 и 8.4), так и за счет применения более эффективных стратегий поиска (8.5).

8.3. Системы вывода в исчислении предикатов без равенства

Как было указано в 8.1, процедура доказательства может быть представлена парой (Т, ∑), где Т — аксиомы и правила вывода, а ∑ — стратегия поиска. В данном параграфе и в 8.4 будут рассмотрены правила вывода, ограничивающие принцип резолюции (директивы). Для лучшего понимания правил вывода мы будем приводить примеры их использования в контексте процедур доказательства. В этих примерах будет использоваться простейшая стратегия поиска — стратегия поиска в ширину (см.7.3.2).

8.3.1. Семантическая резолюция

На рис. 8 мы привели пример процесса опровержения, полученного бинарной резолюцией. Из всех генерированных дизъюнктов для вывода *NIL* достаточны (6) и (12). Остальные дизъюнкты являются неуместными или избыточными. Рассмотрим на данном примере, как можно было бы избежать генерации лишних дизъюнктов. Один из способов состоит в разделении множества S исходных дизъюнктов на два подмножества, при этом для образования резольвенты используют дизъюнкты из разных подмножеств. На практике для разбиения множества на два класса используют **интерпретацию.**

Пусть *I*— некоторая интерпретация множества S дизъюнктов. Тогда в одно подмножество *(S_1)* включаются дизъюнкты, которым выбранная интерпретация не удовлетворяет, а в другое (S_2) включаются дизъюнкты, которым интерпретация удовлетворяет. Так как рассматриваемое множество дизъюнктов невыполнимо, то любая интерпретация разделит его на два непустых подмножества.

В рассматриваемом нами примере интерпретация *I= {~R, ~Q, ~P}* разделит исходное множество дизъюнктов на S_1={(2), (3)} и S_2={(1), (4)}. Это позволит не генерировать дизъюнкт (7), так как он образуется из элементов одного подмножества.

Для ограничения количества генерируемых дизъюнктов используется понятие *упорядочения предикатных букв*. Упорядочим в исходном множестве (рис. 1) предикатные буквы следующим образом: *P>Q>R.* В двух дизъюнктах (один из S_1, а другой из S_2) будем резольвировать только такую литеру дизъюнкта из S_1, которая является *наибольшей (старшей) предикатной буквой* в данном дизъюнкте. Это ограничение позволит нам не резольвировать дизъюнкты (8) и (9), так как у них резольвируемая предикатная буква не является старшей *(R<P, R<Q).*

Для примера на рис. 1 при $I=\{\sim R, \sim Q, \sim P\}$, $P>Q>R$ и введенных нами ограничениях на уровне $R_1(S)$ возможно сгенерировать только дизъюнкты (5) и (6). Им обоим удовлетворяет интерпретация I, следовательно, мы их включаем и подмножеаво S_2. После этого можно образовать два одинаковых дизъюнкта: R (из (3) и (5)) и R (из (2) и (6)). Дизъюнкту R выбранная интерпретация не удовлетворяет, и он включается в подмножество S_1. Но в S_2 есть дизъюнкт $\sim R$. Резольвируя R и $\sim R$, получим пустой дизъюнкт. Итак, существует два способа, которыми может быть получен дизъюнкт R:(((1) (2)) (3)) и (((1) (3)) (2)). Способы отличаются только порядком использования дизъюнктов. Так как для доказательства достаточно одного способа получения дизъюнкта, то необходимо избежать второго способа получения дизъюнкта R. Для этого вводится понятие *клаша*. Идея клаша состоит в том, чтобы генерировать R непосредственно из (1), (2) и (3) без образования «промежуточных» дизъюнктов (5) и (6).

Семантическая резолюция для ограничения резолюции использует объединение указанных выше понятий: **интерпретации, упорядочения предикатных букв и клаша**.

Приведем теперь формальное определение *семантической резолюции*.

Пусть I есть интерпретация. Пусть P — упорядочение предикатных букв. Конечное множество дизъюнктов $\{E_1, \ldots, E_q, N\}$, $q\geq 1$, называется *семантическим клашем* по отношению к P и I (*PI*-клашем) тогда и только тогда, когда $E_1, \ldots, E_q$ (называемые *сателлитами)* и N (называемое *ядром)* удовлетворяют следующим условиям:

1. Интерпретация I не удовлетворяет $E_1, \ldots, E_q$ (т. е. $E_1, \ldots, E_q$ ложны в I).
2. Пусть $R_1=N$. Для каждого $i=1, \ldots, q$ существует резольвента R_{i+1}, образованная из R_i и E_i.
3. Резольвируемая литера в E_i является наибольшей предикатной буквой в E_i, $i=1, \ldots, q$.
4. Интерпретация I не удовлетворяет R_{q+1} (т. е. R_{q+1} ложно в I). R_{q+1} называется *PI*-резольвентой (из *PI*-клаша $\{E_1, \ldots, E_q, N\}$).

Будем называть *семантической резолюцией (PI-резолюцией)* правило вывода, генерирующее *PI*-резольвенты из множества дизъюнктов.

Пример 4. Пусть

$$E_1 = T(x) \vee Q(x), \quad E_2 = Q(x) \vee R(x),$$
$$N = \sim T(x) \vee \sim Q(x) \vee R(x).$$

Пусть $I=\{\sim T, \sim Q, \sim R\}$ и P есть упорядочение, в котором $T>Q>R$. Тогда $\{E_1, E_2, N\}$ есть *PI*-клаш. *PI*-резольвентой этого клаша является $R(x)$.

В этом примере ни $\{E_1, N\}$ ни $\{E_2, N\}$ не являюгся *PI*-клашем, так как интерпретация I удовлетворяет образованным резольвентам (нарушение условия 4).

Отметим, что при изменении упорядочения P на такое, что $R>Q>T$, множество дизъюнктов $\{E_1, E_2, N\}$ не является *PI*-клашем (нарушено условие 3).

В *PI*-клаше $\{ E_1, E_2, .. ., E_q, N\}$ E_1 рассматривается как первый сателлит, E_2— второй, а E_q — последний. Фактически порядок сателлитов не является существенным. Мы можем пометить любой сателлит как первый, среди оставшихся любой как второй и т. д. Независимо от выбранного порядка мы получим одну и ту же *PI*-резольвенту. Следовательно, мы можем избежать генерации одной резольвенты многими способами.

Пусть I — интерпретация для множества S дизъюнктов и P — упорядочение предикатных букв, появляющихся в S. Вывод из S называется *PI-выводом* тогда и только тогда, когда каждый дизъюнкт в выводе является или дизъюнктом из S, или *PI*-резольвентой.

Пример 5. Пусть

$S = \{Q\{a) \vee R(x), \sim Q(x) \vee R(x), \sim R(a) \vee \sim T(a), T(x)\}$

Пусть $I=\{\sim Q, R, \sim T)$ и P— упорядочение предикатных букв $R>Q>T$.

На рис. 2, *а* показан *PI*-вывод пустого дизъюнкта из S.

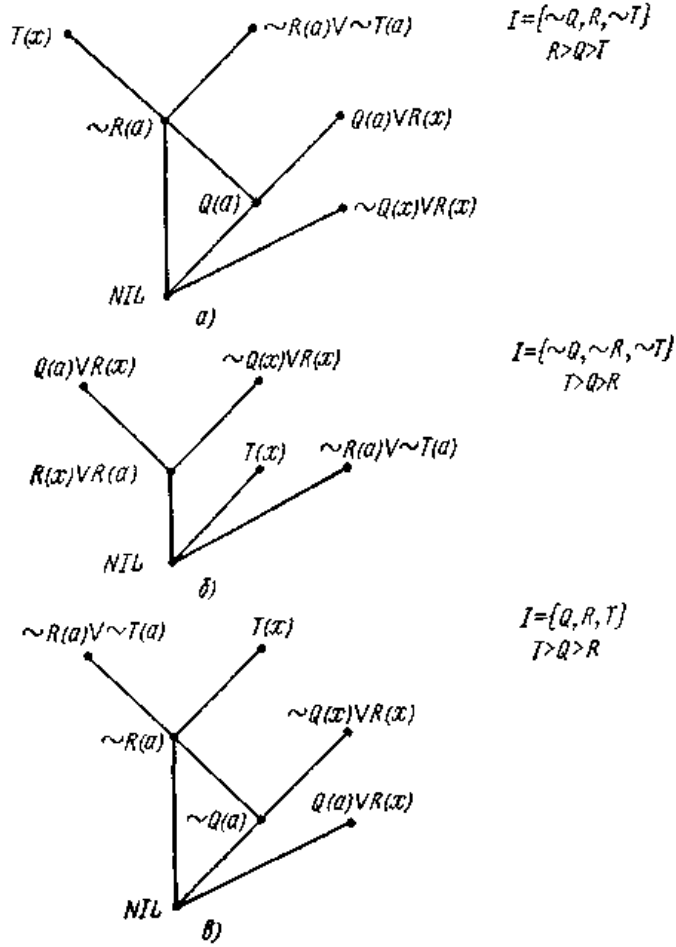


Рис. 2. Граф опровержения, полученный *PI*–резолюцией, положительной гиперрезолюцией и отрицательной гиперрезолюцией.

Особенностью семантической резолюции является тот факт, что можно использовать любую интерпретацию и любое упорядочение предикатных букв. Например, если в примере 5 выбрать $I=\{\sim Q, \sim R, \sim T\}$ и $T>Q>R$, то мы получим *PI*-вывод пустого дизъюнкта, изображенный на рис. 2, б.

Можно показать, что *PI* -резолюция полна, т. е. *если P есть упорядочение предикатных букв в конечном невыполнимом множестве*

S дизъюнктов и I есть интерпретация множества S, то существует PI-вывод пустого дизъюнкта из S.

8.3.2. Специализация семантической резолюции

В этом разделе мы опишем два правила вывода, которые могут быть получены из семантической резолюции выбором специальных интерпретаций. Одна интерпретация ведет к *гиперрезолюции*, а другая — к *стратегии множества поддержки*.

Будем называть дизъюнкт *положительным,* если он не содержит знаков отрицания. Дизъюнкт называется *отрицательным,* если каждая его литера содержит знак отрицания. Дизъюнкт называется *смешанным,* если он не является ни положительным, ни отрицательным.

Выберем интерпретацию *I*, в которой каждая литера является отрицанием атома. В этой частной интерпретации *PI*-резолюция называется *положительной гиперрезолюцией.* В положительной гиперрезолюции все сателлиты и *PI*-резольвенты являются положительными дизъюнктами.

Отрицательной гиперрезолюцией называегся специальный случай *PI*-резолюции, в которой интерпретация *I* не содержит литер со знаками отрицания. В отрицательной гиперрезолюции все сателлиты и *PI*-резольвенты являются отрицательными дизъюнктами.

Из полноты *PI*-резолюции следует, что положительные и отрицательные гиперрезолюции полны.

Для примера 5 и упорядочения $T>Q>R$ вид положительной и отрицательной гиперрезолюции (гипердирективы) изображен на рис. 2, *б* и рис. 2, *в* соответственно.

Рассмотрим другую специализацию семантической резолюции.

Пусть *S* — конечное невыполнимое множество дизъюнктов и *T* есть подмножество S такое, что *S—T* выполнимо. Так как *S— T* выполнимо, то существует интерпретация *I*, которая удовлетворяет дизъюнктам множества *S — T*. Выберем данную интерпретацию. Пусть *P* — любое упорядочение предикатных букв в S. На основании полноты *PI*-резолюции можно утверждать, что при выбранных *P* и *I* существует вывод *D* пустого дизъюнкта из S. Рассмотрим в *D* каждый *PI* -клаш $\{E_1, E_2, \ldots, E_q, N\}$. По определению *PI* -клаша сателлиты ложны в *I*, поэтому ни один дизъюнкт из S — *T* не является сателлитом.

PI-резольвента этого клаша может быть получена как результат последовательности бинарных резолюций E_1 и *N*, затем E_2 и резольвенты, полученной на предыдущем шаге (от E_1 и *N*), и т. д.

Таким образом, в каждой бинарной резолюции одновременно оба резольвируемых дизъюнкта не принадлежат множеству *S—T.*
Резолюция, удовлетворяющая указанному требованию, называется *резолюцией множества поддержки (опорного множества),* а множество *T — множеством поддержки.*
Разложив *PI*-клаши в *D* в последовательность бинарных резолюций, мы получим *вывод множества поддержки,* т. е. вывод, в котором каждая резолюция есть резолюция множества поддержки.
Из приведенного рассуждения и полноты *PI*-резолюции следует, что стратегия множества поддержки полна.

8.3.3. Семантическая резолюция, использующая упорядоченные дизъюнкты

Упорядочение предикатных букв, используемое в *PI*- выводе, в общем случае не дает возможности выбрать в сателлите единственную литеру, которая должна резольвироваться.
Пример 6. Пусть

$$S = \{R(a) \vee R(b) \vee R(c) \vee R(d), \sim R(x)\},$$
$$I = \{\sim R(x)\}$$

и *P* - любое упорядочение. Тогда в *PI*-клаше
$\{R(a) \vee R(b) \vee R(c) \vee R(d), \sim R(x)\}$ каждая из четырех литер в сателлите имеет одинаковое старшинство и, следовательно, является кандидатом на резольвирование с ядром.
Стремление иметь единственного кандидата на резольвирование в каждом сателлите приводит к идее упорядочения дизъюнктов.
Упорядоченным дизъюнктом называется определенная последовательность литер. Упорядоченный дизъюнкт, так же как и дизъюнкт, является дизъюнкцией входящих в него литер. Различие состоит в том, что дизъюнкт рассматривается как множество литер, а упорядоченный дизъюнкт как некоторая последовательность литер.
Будем говорить, что *литера* L_2 *старше литеры* L_1 в упорядоченном дизъюнкте (или L_1 *младше,* чем L_2) тогда и только тогда, когда L_2 следует за L_1 в последовательности, определенной упорядоченным дизъюнктом. Отметим, что *старшая (наибольшая) литера* дизъюнкта является последней литерой дизъюнкта, а *младшая (наименьшая) литера* — первой.
Рассмотрим $R(a) \vee R$ (Ь) $\vee R(c)$ как упорядоченный дизъюнкт. В нем $R(c)$—старшая (наибольшая) литера, а $R(a)$— младшая литера.
Мы будем определять семантическую резолюцию, использующую упорядоченные дизъюнкты. Предварительно введем необходимые понятия.

Если две или больше литер (с одинаковыми знаками) упорядоченного дизъюнкта *C* имеют наиболее общий унификатор σ, то упорядоченный дизъюнкт, полученный из последовательности *C*σ вычеркиванием любой литеры, идентичной младшей литере, называется *упорядоченным фактором дизъюнкта C.*

Пример 7. Для дизъюнкта $C=R(x) \vee Q(x) \vee R(a)$ первая и третья литеры имеют НОУ $\sigma = \{a/x\}$. Следовательно, $C\sigma = R(a) \vee Q(a) \vee R(a)$. В последовательности *C*σ существуют две идентичные литеры (первая и третья литеры). В соответствии с определением младшей литерой считается литера, расположенная левее. Для получения упорядоченного фактора надо из *C*σ удалить литеру, идентичную младшей литере. В рассматриваемом дизъюнкте это последняя литера. Таким образом, *упорядоченным фактором является последовательность* $R(a) \vee Q(b)$.

Пусть C_1 и C_2 — упорядоченные дизъюнкты. *Упорядоченная бинарная резольвента* дизъюнкта C_1 и дизъюнкта C_2 (не имеющих общих переменных) определяется следующим образом. Пусть L_1 и $\sim L_2$ — две литеры в C_1 и C_2 соответственно. Если L_1 и $\sim L_2$ имеют НОУ σ и *C* есть упорядоченный дизъюнкт, полученный из конкатенации последовательностей $C_1\sigma$ и $C_2\sigma$ путем устранения $L_1\sigma$ и $L_2\sigma$ и вычеркивания из оставшейся последовательности любой литеры, которая идентична младшей литере последовательности, то *C* называется *упорядоченной бинарной резольвентой.*

Заметим, что упорядоченная бинарная резольвента C_1 и C_2 не тоже самое, что упорядоченная бинарная резольвента C_2 и C_1.

Пример 8. Рассмотрим упорядоченные дизъюнкты $C_1=R(x) \vee Q(x) \vee T(x)$ и $C_2=\sim R(a) \vee Q(a)$. Вычислим упорядоченную бинарную резольвенту C_1 и C_2. $L_1=R(x)$, $L_2=\sim R(a)$, $\sigma=\{a/x\}$. Конкатенация $C_1\sigma$ и $C_2\sigma$ дает последовательность $R(a) \vee Q(a) \vee T(a) \vee \sim R(a) \vee Q(a)$. Устранив $L_1\sigma$ и $L_2\sigma$, получим последовательность $Q(a) \vee T(a) \vee Q(a)$. В оставшейся последовательности первая литера является младшей, а третья литера идентична ей. Устранив третью литеру, получим последовательность $Q(a) \vee T(a)$, являющуюся упорядоченной бинарной резольвентой C_1 и C_2.

Упорядоченной резольвентой упорядоченных дизъюнктов C_1 и C_2 называется одна из следующих упорядоченных бинарных резольвент:

1) C_1 и C_2;
2) C_1 и упорядоченного фактора C_2;
3) упорядоченного фактора C_1 и C_2;
4) упорядоченного фактора C_1 и упорядоченного фактора C_2.

Упорядоченной резолюцией называется правило вывода, которое генерирует упорядоченные резольвенты из множества упорядоченных дизъюнктов. Можно показать, что упорядоченная резолюция полна.

Рассмотрим теперь *семантическую резолюцию для упорядоченных дизъюнктов.*

Пусть I — интерпретация. Конечная последовательность упорядоченных дизъюнктов $\{E_1, E_2, \ldots, E_q, N\}$ называется *упорядоченным семантическим клашем* по отношению к I (для краткости *OI*-клашем) тогда и только тогда, когда $E_1, E_2, \ldots, E_q$ (называемые *упорядоченными сателлитами)* и N (называемое *упорядоченным ядром)* удовлетворяют перечисленным ниже условиям.

1. Интерпретация I не удовлетворяет $E_1, E_2, \ldots, E_q$.
2. Пусть $R_q=N$. Для каждого $i=q, q$—1, ..., 1 существует упорядоченная резольвента R_{i-1} сателлита E_i и R_i.
3. Резольвируемая литера в E_i является последней литерой в E_i, $i = 1, \ldots, q$, резольвируемая литера в R_i является наибольшей литерой среди литер, которым удовлетворяет I.
4. Интерпретация I не удовлетворяет R_o, R_o называется *OI*-резольвентой *OI* -клаша $\{E_1, E_2, \ldots, E_q, N\}$.

Пример 9. Рассмотрим упорядоченные дизъюнкты:

(1) $Q(a) \vee R(x)$,

(2) $S(x)$,

(3) $\sim R(x) \vee \sim S(a) \vee T(b)$.

Пусть I есть интерпретация, в которую все литеры входят со знаком отрицания. Интерпретация I не удовлетворяет дизъюнктам (1) и (2), поэтому они могут быть использованы как сателлиты. Так как (3) имеет литеры, которым удовлетворяет I, то (3) может быть использован как упорядоченное ядро. В (3) есть две литеры, которым удовлетворяет $I(\sim R(x)$ и $\sim S(a))$, следовательно, необходимы два сателлита. Пусть $R_2=N=\sim R(x) \vee \sim S(a) \vee T(b)$. В R_2 наибольшей из литер, которым удовлетворяет I, является $\sim S(a)$. Так как $\sim S(a)$ и последняя литера (2) могут резольвироваться, то $E_2=S(x)$. Упорядоченная резольвента R_1 сателлита E_2 и R_2 есть $R_1=\sim R(x) \vee T(b)$. В R_1 наибольшей литерой, которой удовлетворяет I, является $\sim R(x)$. Так как $\sim R(x)$ и последняя литера (1) могут резольвироваться, то $E_1=Q(a) \vee R(x)$. Упорядоченная резольвента R_0 сателлита E_1 и R_1 есть $R_0=Q(a) \vee T(b)$ Так как интерпретация I не удовлетворяет R_0, мы заключаем, что (E_1, E_2, N) или ((1), (2), (3)) есть *OI*-клаш и $Q(a) \vee T(b)$ есть *OI*-резольвента этого клаша.

Из приведенного примера видно, что порядок упорядоченных сателлитов определяется порядком литер в упорядоченном ядре. Поэтому для рассмотренного примера ((2), (1), (3)) не является *OI* -клашем.

Если рассматривать дизъюнкты из примера 6 как упорядоченные дизъюнкты и *I={~R},* то для них существует только одна *OI*-резольвента *(R* (*a*) ∨ *R* (*b*) ∨ *R* (*c*)), в отличие от четырех *PI*-резольвент.

Пусть *S* — множество упорядоченных дизъюнктов и *I*— интерпретация для S. Вывод из S называется *OI-выводом* тогда и только тогда, когда каждый упорядоченный дизъюнкт в выводе есть или упорядоченный дизъюнкт из S, или *OI* -резольвента.

Для множества S дизъюнктов из примера 6 и *I={~R} OI*-вывод пустого дизъюнкта из *S* (рис. 3) требует всего четыре *OI* -резольвенты.

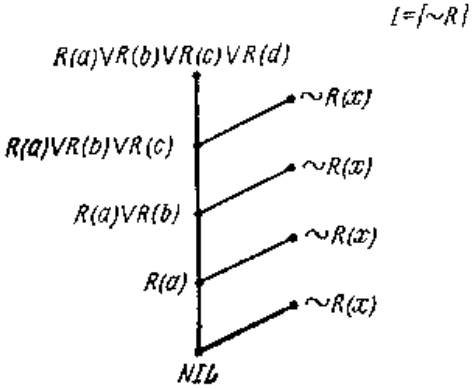


Рис. 3. Пример *OI*-вывода пустого дизъюнкта.

При использовании *PI*-резолюции (директивы) генерируется 40 *PI*-резольвент до того момента, пока метод насыщения уровня (поиск в ширину) не найдет пустой дизъюнкт.

8.3.4. Выполнение семантической резолюции

Хотя *OI*-резолюция неполна, можно использовать понятие упорядоченных дизъюнктов для выполнения *PI*–резолюции.

Упорядоченным положительным дизъюнктом называется упорядоченный дизъюнкт, не содержащий литер со знаками отрицания. *Упорядоченным отрицательным дизъюнктом* называется упорядоченный дизъюнкт, в котором каждая литера содержит знак

отрицания. *Упорядоченным неположительным (неотрицательным) дизъюнктом* называется упорядоченный дизъюнкт, не являющийся положительным. Ниже мы рассмотрим положительную гиперрезолюцию. Отрицательная гиперрезолюция может быть представлена подобным образом. Будем рассматривать только положительные упорядоченные дизъюнкты как кандидаты в сателлиты и неположительные как ядра. Примем соглашение, что в любом упорядоченном неположительном дизъюнкте отрицательные литеры располагаются после положительных.

Пусть S—множество упорядоченных дизъюнктов и P — упорядочение предикатных букв в S. Приводимый ниже алгоритм вычисляет положительные гиперрезольвенты.

1. Пусть M и N есть множество всех положительных и неположительных упорядоченных дизъюнктов в S.
2. j=0.
3. $A_0 = \varnothing$, $B_0 = N$.
4. i=0.
5. Если A_i содержит *NIL*, то конец (опровержение найдено), иначе переход к следующему шагу.
6. Если B_i пусто, то переход к шагу 9, иначе переход к следующему шагу.
7. Вычислить множество W_{i+1},

W_{i+1}= {упорядоченные резольвенты C_1 и C_2, где C_1— упорядоченный дизъюнкт или упорядоченный фактор упорядоченного дизъюнкта в M, а C_2 — упорядоченный дизъюнкт в B_i. Резольвированная литера C_1 содержит наибольшую предикатную букву в C_1, а резольвированная литера в C_2 является «последней» литерой C_2}.

Пусть A_{i+1} и B_{i+1} будут множествами всех положительных и отрицательных упорядоченных дизъюнктов в W_{i+1} соответственно.

8. $i=i+1$, переход к шагу 5.
9. $T=A_0 \bigcup \ldots \bigcup A_i$ и $M = T \bigcup M$.
10. $j=j+1$, i=1.
11. Вычислить множество R,

R = {упорядоченные резольвенты C_1 и C_2, где C_1 есть упорядоченный дизъюнкт или упорядоченный фактор упорядоченного дизъюнкта в T, а C_2 есть упорядоченный дизъюнкт в N. Резольвированная литера в C_1 содержит наибольшую предикатную букву в C_1}. (Отметим, что резольвированная литера в C_2 может быть любой литерой в C_2.) Пусть A_1 и B_1 будут множествами всех положительных и неположительных упорядоченных дизъюнктов в R соответственно.

12. Переход к шагу 5.

В приведенном алгоритме j является номером уровня. Множество гиперрезольвент, сгенерированных на уровне j, размещается в множестве A_i,. При этом i указывает на количество сателлитов, участвующих в образовании данной гиперрезольвенты. В каждом уровне j B_i будет убывать до пустого множества, так как максимальное число отрицательных литер в любом упорядоченном дизъюнкте в B_i уменьшается на единицу при увеличении i на единицу.

Шаг 11 введен для того, чтобы на уровне (j+1) не генерировать резольвенты, полученные на уровне j. Можно показать, что если S невыполнимо, то пустой дизъюнкт генерируется приведенным алгоритмом.

С рассмотренным алгоритмом без потери свойства полноты может быть связана стратегия вычеркивания. То есть для T и M, полученных на 9-м шаге алгоритма, любой дизъюнкт в T или M, поглощаемый другими дизъюнктами в T или M, может быть вычеркнут. (Говорят, что **дизъюнкт C_1 поглощает дизъюнкт C_2, если существует такая подстановка, что $C_1\sigma \subseteq C_2$.)**

Пример 10. Пусть S есть множество упорядоченных дизъюнктов, $S=\{W(x) \vee S(x), \vee V(x) \vee S(x), S(x) \vee P(x), \sim P(x) \vee \sim Q(x), R(x) \vee \sim W(x), \sim S(x), Q(x) \vee \sim V(x) \vee \sim R(x)\}$.

Пусть предикатные буквы упорядочены следующим образом: $W>V>S>R>Q>P$.

Приведенный ранее алгоритм, примененный к множеству S породит следующие шесть положительных гиперрезольвент: $S(x) \vee R\{x), P(x), R(x), S(x) \vee Q(x), Q(x), NIL$.

Для получения гипервывода пустого дизъюнкта из S надо проследить последовательность порождения алгоритмом резольвент. Для рассматриваемого примера последовательность представлена на рис. 4, *а*.

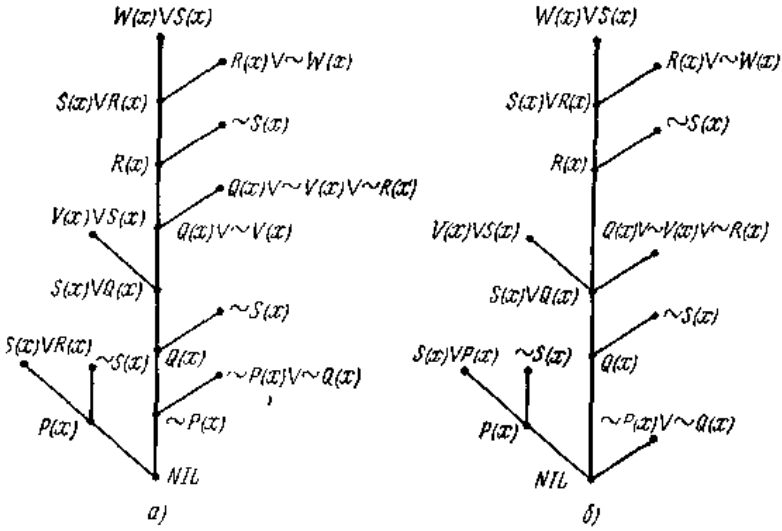


Рис. 4. Гипервывод пустого дизъюнкта.

Указанная последовательность естественным образом трансформируется в гипервывод *NIL* из 5 (рис. 4, *б)*.

8.3.5. Линейная резолюция, использующая упорядоченные литеры и информацию о резольвированных литерах

Предварительно определим простейшее и эффективное правило вывода, называемое входной резолюцией. Пусть *S* — исходное множество дизъюнктов. Дизъюнкты множества S будем называть *входными дизъюнктами. Входной резолюцией* называется бинарная резолюция, у которой хотя бы один из дизъюнктов является входным. Эта стратегия хорошо ограничивает перебор, но не является полной.

Небольшое усложнение входной резолюции приводит к линейному выводу.

Линейным выводом D из множества *S* дизъюнктов называется последовательность дизъюнктов (C_1, . . .,C_n), в которой $C_1 \in S$, а каждый член C_{i+1}, i=1, . . ., n—1, является резольвентой дизъюнкта C_i (называемого *центральным дизъюнктом)* и дизъюнкта *B* (называемого

боковым дизъюнктом), который удовлетворяет одному из двух условий:

1) $B \in S$ (в этом случае B называют *входным дизъюнктом* дизъюнкта C_{i+1});

2) B является некоторым дизъюнктом C_j, предшествующим в выводе дизъюнкту C_i, т. е. $j<i$ (в этом случае B называется *предшествующим дизъюнктом* дизъюнкта C_{i+1}).

Если C_{i+1} получен на основании первого из условий, то говорят, что имела место *входная резолюция,* в противном случае — *резолюция предшествования.*

Пример 11. Рассмотрим невыполнимое множество W дизъюнктов, $W=\{\sim P\{x), P(x) \vee Q(x), \sim Q(x) \vee R(x), \sim R(x) \vee S(x), \sim R(x) \vee \sim S(x) \vee T(x), P(x) \vee \sim T(x)\}$.

На рис. 5, *а* приведен вид графа опровержения, удовлетворяющего условиям линейного вывода, для невыполнимого множества W дизъюнктов.

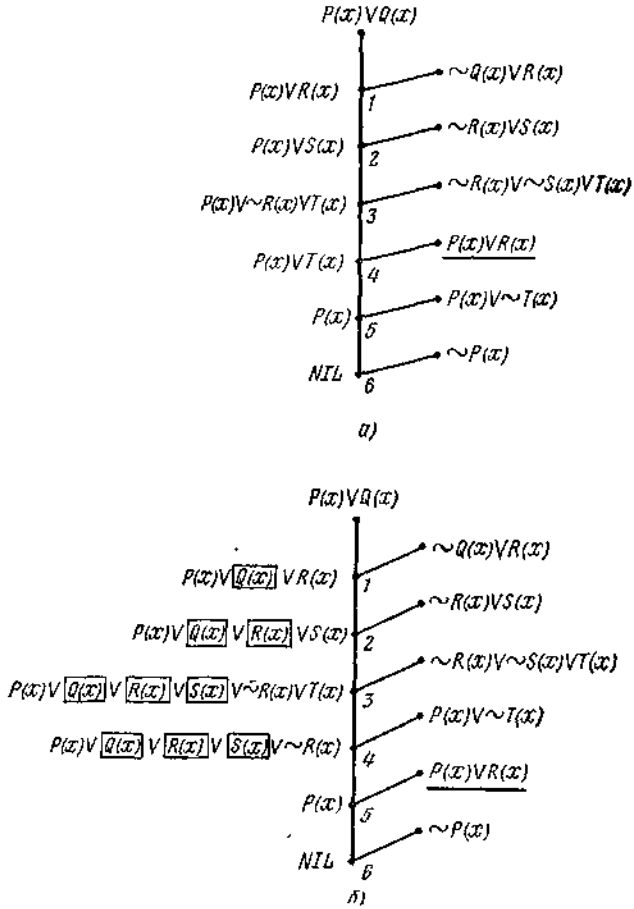


Рис. 5. Пример линейного опровержения.

Здесь слева изображены центральные дизъюнкты, а справа — боковые. Резольвенты 1, 2, 3, 5 и 6 получены входной резолюцией, а резольвента 4 получена резолюцией предшествования.

Концевую вершину *A* в графе, удовлетворяющем условиям линейного вывода, будем называть *начальной,* если любая другая вершина графа либо является концевой, либо следует за *A*.

Сформулируем без доказательства теорему, утверждающую, что для любого невыполнимого множества дизъюнктов всегда существует граф опровержения, удовлетворяющий условию линейного вывода.

Теорема 4. *Пусть G — граф опровержения для невыполнимого множества S дизъюнктов и A — некоторый дизъюнкт из S, появляющийся в G. Тогда для S существует граф опровержения G', удовлетворяющий условиям линейного вывода, для которого A служит начальной вершиной.*

Приведенная теорема устанавливает полноту линейного вывода.

Отметим, что при выборе начальной вершины в графе опровержения, удовлетворяющем условиям линейного вывода, допускается некоторый произвол. Начальная вершина должна появляться в каком-нибудь графе опровержения *G*, т. е. она выбирается из некоторого подмножества $K \subseteq S$, содержащего дизъюнкты, появляющиеся в каком-нибудь опровержении. Например, в качестве *K* могут выбираться дизъюнкты, возникающие при отрицании доказываемого определения.

Линейная резолюция является довольно примитивным правилом вывода. Однако это правило можно усилить введением упорядоченных дизъюнктов и использованием информации о резольвированных литерах. Связывание понятия упорядоченных дизъюнктов (см. п. 8.3.3) с линейной резолюцией не нарушает ее полноты, но существенно увеличивает эффективность метода.

Обычно при выполнении резолюции образование резольвенты происходит путем устранения резольвированных литер. Однако оказывается, что эти литеры несут очень полезную информацию, которая может быть использована для усиления линейной резолюции. До введения механизма, использующего информацию о резольвированных литерах, рассмотрим простой пример. Пусть дано множество *S* дизъюнктов, линейное опровержение для которого изображено на рис. 5, *а*. Заметим, что в этом графе все резольвенты, кроме одной $(P(x) \vee R(x))$, удовлетворяют входной резолюции. Было бы полезно найти необходимое и достаточное условие, когда надо применять резолюцию предшествования и с каким дизъюнктом. Это позволило бы в процессе линейного вывода, до тех пор пока не выполнено указанное условие, применять более эффективную входную резолюцию. Покажем, что это условие может быть определено, если используется понятие упорядоченных дизъюнктов и соответствующим образом записывается информация о резольвированных литерах. Вывод, использующий оба эти понятия, называется *линейным упорядоченным выводом (OL-выводом)*. До формального определения этого вывода рассмотрим его применение к множеству упорядоченных дизъюнктов из примера 11 (рис. 5, б).

Начальной вершиной выберем упорядоченный дизъюнкт $P(x) \vee Q(x)$, последняя литера которого резольвирует с дизъюнктом $\sim Q(x) \vee R(x)$. Будем образовывать резольвенту по правилам, подобным образованию

резольвенты для упорядоченных дизъюнктов. Отличие будет состоять в том, что вместо устранения обеих резольвированных литер будем оставлять в резольвенте первую из них, но помечать ее особым образом. Будем записывать резольвированные литеры в рамке и называть их *A*-литерами. Остальные литеры будем называть *B*-литерами. Если за *A*-литерой не следуют *B*-литеры, то *A*-литера (литеры) вычеркиваются.

В приведенном примере в качестве первой резольвенты получим дизъюнкт $P(x) \vee \boxed{Q(x)} \vee R(x)$. Аналогичным образом получаются вторая, третья и четвертая резольвенты (рис. 5, б).

Все четыре первые резольвенты образованы входной резолюцией. Четвертая резольвента имеет вид $P(x) \vee \boxed{Q(x)} \vee$

$\vee \boxed{R(x)} \vee \boxed{S(x)} \vee \sim R(x) \vee \boxed{T(x)}$. Так как за *A*-литерой $\boxed{T(x)}$ не следуют *B*-литеры, то эта *A*-литера вычеркивается. Отличительной особенностью полученной резольвенты

$$\left(P(x) \vee \boxed{Q(x)} \vee \boxed{R(x)} \vee \boxed{S(x)} \vee \sim R(x)\right)$$

является тот факт, что последняя *B*-литера *(~R(x))* является дополнительной к *A*-литере $\left(\boxed{R(x)}\right)$. Данное обстоятельство и является указанием на необходимость использования при образовании очередной резольвенты не входной резолюции, а резолюции предшествования.

Так как за *A*-литерами в пятой резольвенте $\left(P(x) \vee \boxed{Q(x)} \vee \boxed{R(x)} \vee \boxed{S(x)} \vee \boxed{\sim R(x)}\right)$ не следуют *B*-литеры, то *A*-литеры удаляются и резольвента принимает вид *P(x)*. Шестая резольвента равна пустому дизъюнкту.

Введем теперь формальное определение *OL*-вывода.

Упорядоченный дизъюнкт *C* называется *уменьшающимся упорядоченным дизъюнктом* тогда и только тогда, когда последняя литера *C* унифицируется с отрицанием некоторой *A*-литеры в *C*.

При получении уменьшающегося упорядоченного дизъюнкта нет необходимости искать, с каким из полученных ранее дизъюнктов он

образует резолюцию предшествования. Вместо этого можно просто вычеркивать последнюю литеру в этом упорядоченном дизъюнкте. Мы будем называть это вычеркивание *операцией уменьшения.* Операция уменьшения позволяет не запоминать в *OL*-выводе промежуточных дизъюнктов. Этот аспект *OL*-вывода делает его очень удобным для выполнения на вычислительной машине.

Операцию устранения *A*-литер, за которыми не следуют *B*-литеры, будем называть *операцией сокращения.*

Пусть *C* — уменьшающийся дизъюнкт и *L* — его последняя литера, унифицируемая с некоторой *A*-литерой наиболее общим унификатором σ. Тогда упорядоченный дизъюнкт, полученный из *C*σ применением операций уменьшения и сокращения, будем называть *уменьшенным упорядоченным дизъюнктом* дизъюнкта *C.*

Итак, эффективность *OL*-вывода вызвана двумя факторами:

1. Обнаружением уменьшающегося дизъюнкта.
2. Введением операции уменьшения.

Упорядоченный фактор дизъюнкта C определим так же, как в п. 8.3.3. Условимся при образовании упорядоченного фактора применять операцию сокращения.

Упорядоченная бинарная резольвента дизъюнкта C_1 и дизъюнкта C_2 (не имеющих общих переменных) определяется аналогично определению, данному в п.8.3.3. Пусть L_1 и L_2 — две литеры в упорядоченных дизъюнктах C_1 и C_2 соответственно. Если L_1 и $\sim L_2$ имеют НОУ σ и *C* есть упорядоченный дизъюнкт, полученный из конкатенации последовательностей $C_1\sigma$ и $C_2\sigma$ путем

1) заключения в рамку $L_1\sigma$;
2) устранения $L_2\sigma$;
3) вычеркивания из оставшейся последовательности любой *B*-литеры, которая идентична младшей *B*-литере последовательности;
4) применения операции сокращения,

то дизъюнкт *C* называется *упорядоченной бинарной резольвентой* C_1 и C_2.

Упорядоченную резольвенту определим так же, как в разделе 8.3.3.

Пусть дано множество S упорядоченных дизъюнктов и упорядоченный дизъюнкт C_1 из *S*. Линейный вывод дизъюнкта C_n из *S* с начальным дизъюнктом C_1 называется *OL-выводом,* если выполнены следующие условия:

1. Для i=l, . . ., n—1, C_{i+1} является упорядоченной резольвентой дизъюнкта C_i и B_i (называемых соответственно *центральным* и *боковым),* при этом резольвированная литера C_i (или упорядоченного фактора C_i) является последней литерой C_i.

2. B_i является или некоторым дизъюнктом C_j, $j<i$ (если C_i есть уменьшающийся дизъюнкт), или дизъюнктом из S (во всех остальных случаях). Если B_i есть некоторый дизъюнкт C_j, $j<i$, то C_{i+1} является уменьшенным дизъюнктом.
3. В выводе нет тавтологий.

Определение упорядоченного дизъюнкта может быть использовано для доказательства следующего утверждения.

В *OL*-выводе, если C_i есть уменьшающийся упорядоченный дизъюнкт, то существует центральный упорядоченный дизъюнкт C_j, $j<i$, такой, что уменьшенный дизъюнкт C_{i+1}, полученный из C_i, является упорядоченной резольвентой C_i и C_j. На рис. 5,*б* этот дизъюнкт подчеркнут.

Следующая теорема устанавливает полноту *OL*-резолюции.

Теорема 5. *Если C есть упорядоченный дизъюнкт в невыполнимом множестве S упорядоченных дизъюнктов и если S — {C} выполнимо, то существует OL-опровержение из S с упорядоченным дизъюнктом C как начальным дизъюнктом.*

Легко показать, что *OL*-резолюция включает в себя резолюцию множества поддержки, т. е. полнота *OL*-резолюции гарантирует полноту резолюции множества поддержки.

8.3.6. Линейный вывод

Рассмотрим вопросы эффективного выполнения линейного вывода. Предположим, что для невыполнимого множества *S* дизъюнктов в качестве начального выбран дизъюнкт C_1. После резольвирования C_1 со всеми возможными боковыми дизъюнктами мы получим резольвенты $R_1, \ldots, R_m$. Каждый R_i, $1 \le i \le m$, является возможным центральным дизъюнктом. Если некоторый R_i является пустым дизъюнктом, то доказательство завершено. Иначе, для каждого *i* мы находим все возможные боковые дизъюнкты, которые могут резольвировать с R_i. Указанный процесс продолжается до получения пустого дизъюнкта. Для удобства представления указанного процесса будем изображать центральные дизъюнкты в виде вершин графа, а боковые в виде помеченных дуг, связывающих соответствующий центральный дизъюнкт и образующуюся резольвенту.

Проиллюстрируем сказанное на примере.

Пример 12. Пусть множество $S=\{R(x) \vee Q(x), \sim R(x) \vee Q(x), R(x) \vee \sim Q(x), \sim R(x) \vee \sim Q(x)\}$. В качестве начального дизъюнкта выберем дизъюнкт $R(x) \vee Q(x)$. На рис. 6 изображено дерево поиска пустого дизъюнкта с *OL*-резолюцией. (Для иллюстративных целей

тавтологии в примере не устраняются.) Номер у вершины указывает порядок получения резольвенты.

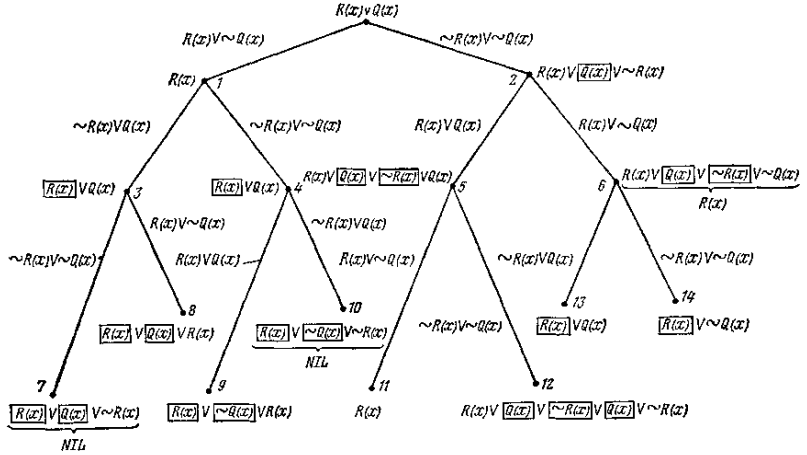


Рис. 6. Представление *OL*-опровержения в виде дерева поиска.

Нетрудно видеть, что задача нахождения *OL*-опровержения в указанной постановке подобна задаче эвристического поиска. В понятиях эвристического поиска задача нахождения опровержения, представленная на рис. 6, решена методом *поиска в ширину*. Здесь состояния представлены центральными дизъюнктами, а операторами являются боковые дизъюнкты. Начальным состоянием является дизъюнкт $d=\{R(x) \vee Q(x)\}$, а конечным *NIL*.

Следует отметить, что *OL*-вывод эффективно конкурирует практически со всеми методами за счет простоты организации поиска. Эта простота объясняется как тем, что в *OL*-выводе один из резольвируемых дизъюнктов определен, так и тем, что здесь не требуется запоминать промежуточные дизъюнкты.

8.4. Правила вывода в исчислении предикатов с равенством

Равенство является важным и широко используемым отношением. Многие определения могут быть легко записаны с использованием равенства. Если отношение равенства используется для выражения некоторого определения и доказательство ведется с использованием единственного правила резолюции, то, кроме аксиом, необходимо добавить набор аксиом, описывающих свойства равенства.

Аксиомы равенства для множества S дизъюнктов имеют вид:

1. $\{x=x\}$ (рефлексивность).
2. $\{x\neq y \vee y=x\}$ (симметричность).
3. $\{x\neq y \vee y\neq z \vee x=z\}$ (транзитивность).
4. $\{ x_j\neq x_0 \vee \sim P(x_1, \ldots, x_j, \ldots, x_n) \vee P\{x_1, \ldots, x_0, \ldots, x_n)\}$, где $j = 1, \ldots, n$, для каждой n-местной предикатной буквы P, встречающейся в S.
5. $\{x\neq x_o \vee f(x_1, \ldots, x_j, \ldots, x_n)=f(x_1, \ldots, x_0, \ldots, x_n)\}$, где $j = 1, \ldots, n$, для каждой n-местной функциональной буквы f, встречающейся в S.

Аксиомы 4 и 5 определяют свойство подстановочности равенства.

В приведенных аксиомах для обозначения предикатного символа равенства мы используем = и пишем $a= b$ и $a\neq b$ вместо $=(a, b)$ и $\sim(=(a, b))$ соответственно.

Введение равенства с помощью аксиом и универсального правила вывода (резолюции) имеет определенные недостатки. Покажем это на примере.

Пусть дано: 1) $Q(a)$; 2) $a=b$. Надо доказать 3) $Q(b)$.

Если пользоваться аксиомами равенства и принципом резолюции, то сначала получим 4) $a\neq x_2 \vee Q(x_2)$, резольвируя выражение 1) и аксиому равенства $x_1\neq x_2 \vee \sim Q(x_1) \vee Q(x_2)$. Затем, резольвируя 4) и 2), получим $Q(b)$. Естественно было бы не получать вспомогательное выражение 4), т. е. с помощью специальных правил вывода получить доказательство, имеющее вывод меньшей длины. На практике более существенным оказывается даже не увеличение длины вывода, а тот факт, что появившиеся бесполезные дизъюнкты (выражение 4)) приводят в свою очередь к появлению новых бесполезных дизъюнктов, «загрязняют» пространство поиска, а это приводит к снижению эффективности метода.

Следует указать, что сам факт присутствия аксиом равенства, увеличивающих общее количество формул, также является недостатком.

Существуют другие способы введения отношения равенства. Например, Дарлингтон использовал аксиомы второго порядка, Робинсон ввел обобщенный принцип резолюции, который обеспечил встраивание равенства, Моррис предложил *E*-резолюцию. Мы будем в этом параграфе рассматривать правило парамодуляции, являющееся наиболее естественным из предложенных правил.

Введем необходимые определения. Мы определили ранее, что множество S дизъюнктов является невыполнимым тогда и только тогда, когда S ложно на всех интерпретациях. Однако существует много множеств дизъюнктов, которые истинны в одних интерпретациях, но ложны в других. Пусть W есть множество всех интерпретаций S. Пусть Q является непустым подмножеством W.

Будем называть множество S дизъюнктов *Q-невыполнимым* тогда и только тогда, когда S ложно для каждого элемента *Q*.

Определим теперь *R*-невыполнимые множества, где *R* обозначает равенство.

R-интерпретация I множества *S* дизъюнктов есть интерпретация, удовлетворяющая следующим условиям:

а) $(\alpha=\alpha) \in I$;

б) если $(\alpha=\beta) \in I$, то $(\beta=\alpha) \in I$;

в) если $(\alpha=\beta) \in I$ и $(\beta=\gamma) \in I$, то $(\alpha=\gamma) \in I$;

г) если L' есть результат замещения некоторого одного появления α в L на β и $(\alpha=\beta) \in I$, то $L' \in I$.

Здесь α, β и γ— любые термы в эрбрановском универсуме для *S*, а *L* — любая литера в *I*.

Фактически *R*-интерпретация удовлетворяет условиям рефлексивности, симметричности, транзитивности и подстановочности, т. е. является моделью теории равенства. Многие авторы рассматривали специальные классы моделей различных теорий (частичного упорядочения, теории множеств и т. п.). Довольно общий подход к решению этой задачи предложен Слэйглом и продемонстрирован им на примерах теории равенства, частичного упорядочения и теории множеств, а позднее на теориях с коммутативностью и ассоциативностью. Цель этого подхода заключается в том, чтобы заменить некоторые аксиомы рассматриваемой теории полными (по опровержению) и эффективными (по времени) правилами вывода. Новые правила позволяют исключить огромное число лишних следствий, возникающих при использовании правила резолюции и аксиом теории. На основании этих идей частичное и общее упорядочение были встроены в правила вывода и получены экспериментальные результаты.

В этой работе будут рассматриваться только модели теории равенства, так как подход к теории равенства и другим теориям является подобным.

Множество *S* дизъюнктов называется *R-удовлетворимым,* если существует *R*-интерпретация, удовлетворяющая всем дизъюнктам в S. Иначе S называется *R-неддовлетворимым* (*R*-невыполнимым).

Пусть *S* — множество дизъюнктов. Определим *множество F функционально рефлексивных аксиом* для *S* как $F=\{f(x_1, \ldots,x_n)=f(x_1,\ldots x_n)$ для всех *n*-местных функциональных букв *f* встречающихся в *S*$\}$. *S* будем называть *функционально-рефлексивной системой,* если S содержит *F* и *{x=x},* и просто *рефлексивной системой,* если S содержит *{x=x},* но не содержит *F*.

Теорема 6. *Пусть S — множество дизъюнктов и P — множество аксиом равенства для S. S является R-невыполнимым тогда и только тогда, когда S* $\bigcup$ *P невыполнимо.*

Для *R*-невыполнимых множеств справедливо расширение теоремы Эрбрана.

Теорема 7. *Конечное множество S дизъюнктов R-невыполнимо тогда и только тогда, когда существует конечное множество S' фундаментальных примеров дизъюнктов в S такое, что S' является R-невыполнимым.*

8.4.1. Парамодуляция

Определим правило вывода для равенства, называемое *парамодуляцией*. Используя резолюцию и парамодуляцию, мы всегда можем вывести пустой дизъюнкт из *R*-невыполнимого множества дизъюнктов.

По правилу парамодуляции из дизъюнкта C_1, равного $\{L(t) \vee C'_1\}$, и дизъюнкта C_2, равного $\{r=s \vee C'_2\}$, не имеющих общих переменных (где C'_1 и C'_2 — дизъюнкты, r и s — термы и $L(t)$ — литера, содержащая терм t) и таких, что t и r имеют НОУ σ, выводится дизъюнкт, называемый *бинарным парамодулянтом* C_1 и C_2:

$$L\sigma\,[s\sigma] \bigcup C'_1\sigma \bigcup C'_2\sigma,$$

где $L\sigma[s\sigma]$ обозначает результат замены одного появления $t\sigma$ в $L\sigma$ на $s\sigma$. Литеры L и $r=s$ называют *парамодулированными литерами.*

Пример 13. Рассмотрим дизъюнкты:

C_1: $Q(g(f(x))) \vee R(x)$;

C_2: $f(g(b))=a \vee T(g(c))$.

Образуем бинарный парамодулянт C_1 и C_2. Здесь L есть $Q(g(f(x)))$, C'_1 есть $R(x)$, r есть $f(g(b))$, s есть a и C'_2 есть $T(g(c))$. L содержит терм t, равный $f(x)$, который унифицируется с r. НОУ t и r есть $\sigma=\{g(b)/x\}$. Следовательно, $L\sigma\,[t\sigma]$ равно $Q(g(f(g(b))))$, а $L\sigma[s\sigma]$ есть $Q(g(a))$. Так как $C'_1\sigma$ равно $R(g(b))$ и C'_2 равно $T(g(c))$ то бинарный парамодулянт C_1 и C_2 равен

$$Q(g(a)) \vee R(g(b)) \vee T(g(c)).$$

Парамодулированными литерами являются $Q(g(f(x)))$ и $f(g(b))=a$.

Парамодулянт дизъюнктов C_1 и C_2 есть один из следующих бинарных парамодуляптов:

1) бинарный парамодулянт C_1 и C_2;
2) бинарный парамодулянт C_1 и фактора C_2;
3) бинарный парамодулянт фактора C_1 и C_2;
4) бинарный парамодулянт фактора C_1 и фактора C_2.

Приведем пример совместного использования правила парамодуляции и резолюции для получения опровержения.

Пример 14. Если $x^2=e$ для всех x в группе, то группа коммутативна.

1. $f(e, x) = x$.
2. $f(x, e)=x$.
3. $f(x, f(y, z))=f(f(x, y), z)$.
4. $f(x_t\ x)=e$.
5. $f(a, b)=c$.
6. $c\neq f(b,a)$.
7. $f(x, e)=f(f(x, y), y)$, парамодулянт 3 и 4, $t=f(y,z)$.
8. $x=f(f(x, y), y)$, парамодулянт 7 и 2, $t=f(x, e)$
9. $a=f(c, b)$, парамодулянт 8 и 5, $t=f(x, y)$.
10. $f(y, f(y, z))=f(e, z)$, парамодулянт 3 и 4, $t=f(x,y)$.
11. $f(y, f(y, z))=z$, парамодулянт 10 и 1, $t=f(e, z)$.
12. $f(c, a)=b$, парамодулянт 11 и 9, $t=f(y, z)$.
13. $c=f(b, a)$, парамодулянт 8 и 12, $t=f(x, y)$.
14. *NIL*, резольвента 13 и 6.

Приведем алгоритм получения опровержения, используя правила парамодуляции и резолюции. Пусть S_0 — множество всех факторов дизъюнктов из S. Для нечетного $i>0$ пусть S_i формируется из S_{i-1} добавлением всех дизъюнктов, которые могут быть получены парамодуляцией двух дизъюнктов в S_{i-1}. Для четного $i>0$ пусть S_i формируется из S_{i-1} добавлением всех факторов дизъюнктов, которые могут быть получены резольвированием двух дизъюнктов в S_{i-1}. Если на некотором шаге получен пустой дизъюнкт, то это обозначает *R*-невыполнимость множества S.

Робинсон и Вос показали, что если S является конечным функционально-рефлексивным множеством дизъюнктов, то приведенный выше алгоритм является полуразрешающей процедурой для *R*-невыполнимости.

Непосредственное использование резолюции и парамодуляции неэффективно, поэтому используются различные стратегии. Ограничивающие стратегии для парамодуляции более слабы, чем для резолюции. Неполны аналоги *PI*-резолюции и *OL*-резолюции в парамодуляции. Гиперрезолюция и линейная резолюция могут быть распространены на парамодуляцию (гипермодуляция и линейная парамодуляция).

8.4.2. Гиперпарамодуляция

Пусть P — упорядочение предикатных букв, которое включает буквы в дизъюнктах C_1 и C_2. Парамодулянт C_1 и C_2 называется

P-гиперпарамодулянтом тогда и только тогда, когда выполняются следующие условия:

1. C_1 и C_2—положительные дизъюнкты.
2. Парамодулированные литеры в C_1 и C_2 содержат наибольшую предикатную букву в C_1 и C_2 соответственно.

Напомним, что *P*-гиперрезольвента есть *PI*-резольвента, когда каждая литера в интерпретации *I* содержит знак отрицания. Заметим, что в этом случае сателлиты и *P*-гиперрезольвенты являются положительными дизъюнктами.

Пусть *P* — упорядочение предикатных букв в множестве *S* дизъюнктов. Тогда *P-гипервывод с резолюцией и парамодуляцией* есть вывод, в котором каждый дизъюнкт есть дизъюнкт из S или *P*-гиперрезольвента, или *P*-гиперпарамодулянт. *P-гиперопровержение с резолюцией и парамодуляцией* есть *P*-гипервывод с резолюцией и парамодуляцией пустого дизъюнкта.

Можно показать, что *P*-гиперпарамодуляция полна, т. е. справедлива следующая теорема.

Теорема 8. *Если P — упорядочение предикатных букв в конечном R-невыполнимом множестве S дизъюнктов, то существует P-гиперопровержение с резолюцией и парамодуляцией из* $(S \cup \{x=x\} \cup F)$, *где F—множество функционально-рефлексивных аксиом для S.*

Полнота гиперпарамодуляции говорит, что можно рассматривать для резолюции только клаши с положительными сателлитами, а для парамодуляции — только положительные дизъюнкты. Эта процедура вывода особенно хорошо работает при малом количестве положительных дизъюнктов. Отметим, что упорядочение *P* предикатных букв может играть важную роль в дедуктивной процедуре. Действительно, введя для предикатной буквы равенства наименьший (наибольший) порядок, мы будем получать больше (меньше) резолюций, чем парамодуляции. Если предикатной букве равенства присвоить наименьший порядок, то мы получим следующее следствие теоремы 8.

Следствие 1. *Если P есть упорядочение предикатных букв в конечном R-невыполнимом множестве S дизъюнктов таких, что букве равенства R (*или *=) присвоен наименьший порядок в* P, *и если F есть множество функционально-рефлексивных аксиом для S, то пустой дизъюнкт выводится из* $(S \cup \{x=x\} \cup F)$ *резолюцией и парамодуляцией, в которых*

а) *все резольвенты суть P-гиперрезольвенты;*

б) *парамодуляция выполняется только между дизъюнктами, состоящими из положительных литер с единственной предикатной буквой (буквой равенства), и положительными дизъюнктами.*

Заметим, что если мы используем *P*-гиперпарамодуляцию и *P*-гиперрезолюцию, то множество *F* функционально-рефлексивных аксиом требуется для получения доказательства. Например, рассмотрим $S=\{a=b,\ f(a)\neq f(b)\}$. *S R*-невыполнимо, но не существует *P*-гиперопровержения с резолюцией и парамодуляцией из $S \bigcup \{x=x\}$. Однако сущестует *P*-гиперопровержение с резолюцией и парамодуляцией из $\{S \bigcup \{x=x\} \bigcup F\}$.

Пример 15. Произведем на примере поиска опровержения невыполнимого множества *S* дизъюнктов сравнение парамодуляции и гиперпарамодуляции. Для обоих методов будем использовать поиск в ширину.

Для данного множества *S* дизъюнктов мы будем вырабатывать последовательности дизъюнктов $S_0, S_1, \ldots, S_n, \ldots$ до тех пор, пока не получим пустой дизъюнкт. Пусть $S_0=S$.

$$S_n = \begin{cases} \text{резольвенты из } S_0 \cup \ldots \cup S_{n-1} \text{ для нечетных } n, \\ \text{парамодулянты из } S_0 \cup \ldots \cup S_{n-1} \text{ для четных } n. \end{cases}$$

Кроме того, после генерации нового дизъюнкта будем пользоваться стратегией вычеркивания поглощаемых дизъюнктов. Вычеркнутые дизъюнкты будем помечать звездочкой.

Пусть $S=\{\sim Q(a) \vee \sim S(a) \vee \sim T(a) \vee a=b,\ Q(a),\ S(a),\ T(a),\ f(a)\neq(b),\ f(x)=f(x)\}$.

Пусть *P* будет следующее упорядочение предикатных букв: $Q>S>T>=$.

А) Рассмотрим *P*-гиперпарамодуляцию с ограничениями следствия 1.

S_0: *1. $\sim Q(a) \vee \sim S(a) \vee \sim T(a) \vee a=b$.
2. $Q(a)$.
3. $S(a)$.
4. $T(a)$.
5. $f(a)\neq f(b)$.
6. $f(x)=f(x)$

S_1: 7. $a=b$, P-гиперрезольвента 1, 2, 3 и 4.

S_2: 8. $Q(b)$, P-гиперпарамодулянт 2 и 7.
9. $S(b)$, P-гиперпарамодулянт 3 и 7.
10. $T(b)$, P-гиперпарамодулянт 4 и 7.
11. $f(a)=f(b)$,
12. $f(b)=f(a)$, } P-гиперпарамодулянты 6 и 7.
13. *NIL*, P-гиперрезольвента 5 и 11.

Б) Рассмотрим теперь парамодуляцию без ограничений.

S_0: *1. $\sim Q(a) \vee \sim S(a) \vee \sim T(a) \vee a=b$.
2. $Q(a)$.
3. $S(a)$.
4. $T(a)$.
5. $f(a) \neq f(b)$.
6. $f(x)=f(x)$.

S_1. *7. $\sim S(a) \vee \sim T(a) \vee a=b$, резольвента 1 и 2.

S_2: *8. $\sim S(a) \vee \sim T(a) \vee Q(b)$, парамодулянт 2 и 7.
*9. $\sim S(a) \vee \sim T(a) \vee S(b)$, парамодулянт 3 и 7.
*10. $\sim S(a) \vee \sim T(a) \vee T(b)$, парамодулянт 4 и 7.
*11. $\sim S(a) \vee \sim T(a) \vee f(b) \neq f(b)$, парамодулянт 5 и 7.
*12. $\sim S(a) \vee \sim T(a) \vee f(a) = f(b)$,
*13. $\sim S(a) \vee \sim T(a) \vee f(b) = f(a)$, } парамодулянты 6 и 7.

S_3: *14. $\sim T(a) \vee a=b$, резольвента 3 и 7.
*15. $\sim T(a) \vee Q(b)$, резольвента 3 и 8.
*16. $\sim T(a) \vee S(b)$, резольвента 3 и 9.
*17. $\sim T(a) \vee T(b)$, резольвента 3 и 10.
*18. $\sim T(a) \vee f(b) \neq f(b)$, резольвента 3 и 11.
*19. $\sim T(a) \vee f(a)=f(b)$, резольвента 3 и 12.
*20. $\sim T(a) \vee f(b)=f(a)$, резольвента 3 и 13.

S_4 *21. $\sim T(a) \vee \sim f(f(a) = f(f(b))$,
*22. $\sim T(a) \vee \sim f(f(b) = f(f(a))$, } парамодулянты 6 и 19

S_5: 23. $a = b$, резольвента 4 и 14.
24. $Q(b)$, резольвента 4 и 15.
25. $S(b)$, резольвента 4 и 16.
26. $T(b)$, резольвента 4 и 17.
27. $f(b) \neq f(b)$, резольвента 4 и 18.
28. $f(a)=f(b)$, резольвента 4 и 19.
29. $f(b)=t(a)$, резольвента 4 и 20.
30. $f(f(a))=f(f(b))$, резольвента 4 и 21.
31. $f(f(b))=f(f(a))$, резольвента 4 и 22.

S_6: 32. $f(f(f(a))) = f(f(f(b)))$,
33. $f(f(f(b))) = f(f(f(a)))$, } парамодулянты 6

S_7: 34. *NIL*, резольвента 5 и 28.

8.4.3. Линейная парамодуляция

Для данного множества S дизъюнктов и дизъюнкта C_1 из S *линейным выводом* C_n *с резолюцией и парамодуляцией* с начальным дизъюнктом C_1 называется последовательность дизъюнктов $C_1, . ., C_n$, в которой
1) для i=l, .. ., n—1 C_{i+1} является резольвентой или парамодулянтом дизъюнктов C_i, и B_i;
2) каждый B_i является или некоторым дизъюнктом из S, или некоторым C_j, $j<i$.
Линейным опровержением с резолюцией и парамодуляцией называется линейный вывод с резолюцией и парамодуляцией пустого дизъюнкта.
Пример 16. Пусть S есть $\{\sim R(c) \vee c=d, \sim R(c) \vee g(c)\neq g(d), R(c) \vee a=b, R(c) \vee g(a)\neq g(b), g(x)=g(x)\}$. Тогда линейное опровержение из S с начальным дизъюнктом $\sim R(c) \vee c=d$ представлено на рис. 7.

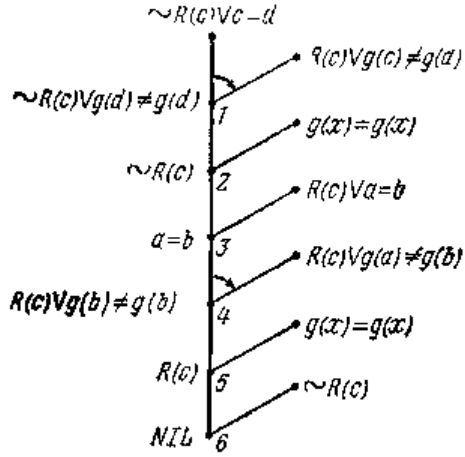


Рис. 7. Линейное опровержение с резолюцией и парамодуляцией.

Здесь шесть боковых дизъюнктов. Пять из них из множества S, а один ($\sim R$ (c)) является предшествующим.
Можно показать, что линейная парамодуляция полна, т. е. справедлива следующая теорема.
Теорема 9. *Если С есть дизъюнкт в R-невыполнимом множестве S дизъюнктов, включающем x=x и функционально-рефлексивные аксиомы, и если S —*{C} *является R-выполнимым, то S имеет линейное опровержение с резолюцией и парамодуляцией с начальным дизъюнктом С.*

8.5. Стратегии поиска

В 8.1 указано на то, что **процедура доказательства теоремы может быть разделена на правило вывода и стратегию поиска.**

В предыдущих параграфах описаны правила вывода, используемые для поиска опровержения. В данном параграфе будут рассмотрены стратегии поиска. Задача стратегии поиска — выбрать те дизъюнкты-кандидаты, к которым на очередном шаге процедуры доказательства теорем следует применять правило вывода.

Ранее было показано, что пространство поиска при доказательстве теоремы может быть представлено в виде абстрактного графа доказательства теорем (G, s), в котором различным выводам одной и той же формулы соответствуют различные вершины. Если граф доказательства теорем, изображенный на рис.8, интерпретировать как граф, получаемый применением принципа резолюции, используя метящую функцию $c: G \rightarrow B^*$, то два дизъюнкта $c(n_7)$ и $c(n_8)$ должны представлять собой все резольвенты пары $c(n_3)$ и $c(n_4)$.

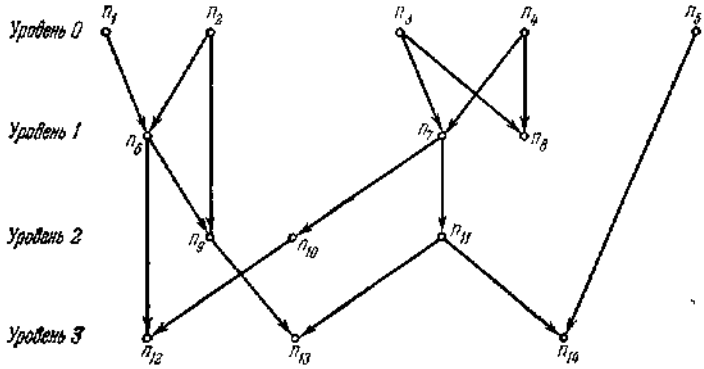


Рис.8. Абстрактный граф доказательства теорем.

Дизъюнкт $c(n_8)$ не должен резольвировать ни с одним из дизъюнктов $c(n_i)$, где $1 \le i \le 14$. Дизъюнкты $c(n_{10})$ и $c(n_{11})$ являются или факторами дизъюнкта $c(n_7)$, или получены из $c(n_7)$ резольвированием $c(n_7)$ с самим собой. Если, например, $C=c(n_6)=c(n_7)=c(n_{14})$, то C имеет три вывода: два первого уровня и один третьего уровня. Отметим, что выводы нет необходимости представлять в виде дерева. В приведенном примере дизъюнкт $c(n_2)$ используется дважды в выводе $c(n_{13})$, но представлен в графе в виде одной вершины.

Как было указано ранее, абстрактная задача доказательства теоремы может быть представлена в виде четверки $P=(G, s, F, g)$, где $F \subseteq G$

является множеством терминальных вершин для P (или решающих вершин) и $g: G \rightarrow R$ — оценочная функция, выражающая меру сложности вывода, (R — множество вещественных чисел).

Решение задачи P состоит в разработке стратегии поиска $\sum$, которая генерирует из G_o вершину $n \in F$.

Стратегия поиска $\sum$ для P есть функция $\sum: 2^G \rightarrow 2^G$, которая вырабатывает подмножество G из другого подмножества G. Задача стратегии поиска в данной постановке состоит в выборе наиболее перспективных дизъюнктов для применения к ним правил вывода. Другими словами, **задача состоит в упорядочении дизъюнктов на основании некоторой меры, называемой обычно оценочной функцией.** При этом дизъюнкты, имеющие наименьшее значение оценочной функции, считаются предпочтительными и именно к ним и первую очередь применяются правила вывода.

Наибольшее распространение на практике получил специальный вид оценочной функции $f(n)=g(n)+h(n)$ (для всех $n \in G$), выраженной в терминах меры сложности K и дополнительной эвристической функции h (см.8.2). Эвристическая функция h для P есть функция $h:G \rightarrow R$ такая, что $h(n) \geq 0$, для всех $n \in G$. Интерпретация эвристической функции h состоит в том, что $f(n)=g(n)+h(n)$ оценивает стоимость g (n^*), где n^*—терминальная вершина $n^* \in F$ такая, что $n \leq n^*$, т. е. $h(n)$ оценивает $g(n^*)—g(n)$.

В терминах этой оценочной функции можно описать ранние поисковые стратегии. Обычно в них $g(n)$ характеризовало уровень в графе доказательства, а $h(n)$ — длину дизъюнктов. Выбор в качестве $h(n)$ длины дизъюнкта обосновывается тем фактом, что резольвирование дизъюнктов, содержащих одну литеру, сразу приводит к опровержению. Стратегии *насыщения уровня* могут быть охарактеризованы как стратегии, использующие упорядочение на основании оценочной функции $f(n)=g(n)$, т. е. в первую очередь получаются все резольвенты исходного множества дизъюнктов (нулевого уровня), затем первого уровня, второго и т. д. Эта стратегия эквивалентна стратегии поиска в ширину.

Стратегия *предпочтения единичным дизъюнктам* (одночленам) может быть представлена оценочной функцией $f(n)=h(n)$ при условии, что $g(n)<g_{max}$. Данная стратегия широко используется на практике. Однако эта стратегия является неэффективной при наличии в исходном множестве аксиом большого количества единичных дизъюнктов (что имеет место в вопросно-ответных системах) .

Стратегия *предпочтения наименьшим литерам,* предложенная Слэйглом, также выражается функцией $f(n)=h(n)$. Отличие ее от стратегии предпочтения единичным дизъюнктам состоит только в

способе вычисления *h(n).* Здесь *h(n)* вычисляется не для дизъюнкта-кандидата, а для пары дизъюнктов-кандидатов. Предпочтительной считается такая пара дизъюнктов (*A, B),* которая образует резольвенту *(R)* наименьшей длины. Длина резольвенты может быть вычислена до образования резольвенты по следующей формуле:

$$h(R) \le h(A)+h(B) - 2.$$

На практике, так как важно не абсолютное значение длины, а относительное, *h(R)* считают по формуле *h(R)=h(A)+h(B).*

Мы привели выше простейшие стратегии поиска, используемые в программах доказательства теорем. Ковальский обратил внимание на необходимость разработки более сложных стратегий и определил класс диагональных поисковых стратегий (ДПС) (см.7.2) и восходящих ДПС (см.7.3).

Рассмотрим предложенный Ковальским частный алгоритм *восходящей диагональной стратегии поиска,* названной им ∑*-алгоритмом.

Пусть *m* будет вершина в пространстве поиска, а *c(n*)— соответствующий ей дизъюнкт. В ∑*-алгоритме оценочная функция определяется парой *(i, j),* где *i* — длина дизъюнкта *c(n*), а *j* — уровень дизъюнкта *c(n*). Если даны дизъюнкты $c(n_1)$ с оценкой (i_1, j_1) и дизъюнкт $c(n_2)$ с (i_2, j_2), то $c(n_1) < c(n_2)$ (т. е. $c(n_1)$ *лучше* $c(n_2)$), если

а) $i_1 + j_1 < i_2 + j_2$ или б) $i_1 + j_1 = i_2 + j_2$ и $i_1 < i_2$.

Если $i_1 + j_1 = i_2 + j_2$ и $i_1 = i_2$, то $n_1 =_{d^u} n_2$ (т. е. n_1 и n_2 равны в смысле введенного порядка).

Таким образом, при равенстве суммы длины и уровня двух дизъюнктов предпочтение отдается более короткому дизъюнкту, исходя из того, что цель стратегии поиска— получить дизъюнкт нулевой длины.

На основе введенного упорядочения Ковальский разделил пространство поиска на множества *A (i, j).* Каждое множество *A (i, j)* состоит из дизъюнктов, имеющих одинаковую меру упорядочения $<_{d^u}$. Алгоритм ∑* пытается генерировать дизъюнкты в соответствии с упорядочением восходящей ДПС. Сначала ∑* пытается найти среди исходных дизъюнктов дизъюнкт для множества *A* (0, 0), что возможно, только если в исходном множестве S есть пустой дизъюнкт. Затем делается попытка последовательно генерировать (путем выбора из исходного множества или применением правил вывода) дизъюнкты в множества *A* (0, 1), *A* (1,0), *A* (0, 2), *A* (1, 1), *A* (2,0), *A* (0,3), . . ., *A (i, j),* где *i* - длина дизъюнкта, а *j* — уровень. ∑* генерирует дизъюнкты для множества *A (i, j)* одним из следующих способов:

1. При *j* = 0 дизъюнкт исходного множества длиной *i* заносятся в *A* (*i, j*).
2. При *j* >0 фактор (факторы) дизъюнктов из *A*(*i*+1, *j*—1) заносятся в

A (i, j).

3. При $j>0$ в множество A (i, j) заносится резольвента дизъюнкта $c(n_1)$ из A (i_1, j_1) и дизъюнкта $c(n_2)$ из $A(i_2, j_2)$, где $i=i_1+i_2-2$ и $j=\max(j_1, j_2)+1$.

В начале доказательства все множества A пусты. Они заполняются программой НАПОЛНИТЬ *(i, j)*, которая генерирует первым или третьим из указанных способов дизъюнкт с оценкой (i, j). Алгоритм $\sum^*$ начинает работу с вызова НАПОЛНИТЬ (0, 0). Дизъюнкты $c(n_1)$ и $c(n_2)$, образующие резольвенту третьим способом, являются дизъюнктами из множества A (i_1, j_1) с лучшей мерой, чем (i, j). Когда программа НАПОЛНИТЬ *(i, j)* образовала все возможные дизъюнкты, то вызывается программа НАПОЛНИТЬ *(i',j')*, где *(i',j')* есть следующая мера упорядочения в смысле $>_{d^u}$. Всякий раз, когда НАПОЛНИТЬ *(i, j)* генерирует новый дизъюнкт *c(n)*, вызывается программа РЕКУРСИЯ (*c (n)*). Ее задача — проверить, взаимодействует ли *c (n)* с ранее генерированными дизъюнктами, и в этом случае выработать способом 2 и 3 все дизъюнкты с оценкой $(i', j')<_{d^u}(i, j)$, которые являются потомками *c(n)*. К полученным дизъюнктам также применяется программа РЕКУРСИЯ. Этот рекурсивный процесс продолжается до тех пор, пока на некотором уровне не удается образовать дизъюнкты с мерой, лучшей, чем *(i, j)*. Тогда управление возвращается предыдущему уровню программы РЕКУРСИЯ. В конце концов осуществляется повторный вход в программу НАПОЛНИТЬ, и указанный процесс продолжается до нахождения пустого дизъюнкта или до исчерпания времени (или памяти), отведенного на доказательство.

Ковальский определил восходящую ДПС для оценочной функции *f(n)=g(n)+h(n).* Можно усложнить данную функцию и рассматривать ее как линейную комбинацию не двух, а большего количества функций. Например,

$$f(n)=W_o g(n)+\omega_1 h_1(n)+\omega_2 h_2(n),$$

где ω_i есть весовой коэффициент $\left(\sum_i \omega_i = 1\right)$, g — уровень дизъюнкта, h_1 — длина дизъюнкта, h_2 — мера функциональной сложности дизъюнкта (например, наибольший уровень вложенности функций, входящих в дизъюнкт).

В изложенных выше стратегиях поиск начинается со всего множества исходных аксиом G_o (формул нулевого уровня). При решении многих задач множество G_o оказывается очень большим, что приводит к невозможности найти опровержение в приемлемое время. Применительно к рассматриваемой нами проблеме ИИ G_0 включает множество всех

исходных знаний интеллектуальным объектом. Будем называть все факты, входящие в G_o, *базовыми данными.* Аксиомы базовых данных, не содержащие переменных, будем называть *константными аксиомами.* Остальные аксиомы будем называть *общими аксиомами.* Необходимо отметить, что для решения любой конкретной задачи требуются не все базовые данные, а только небольшая их часть. Поэтому необходимо при поиске доказательства выбирать из базовых данных только те дизъюнкты, которые уместны для решаемой задачи. Выбор уместных аксиом может быть осуществлен в основном за счет использования семантической информации о дизъюнктах, а не синтаксической, использованной в методах упорядочения (длина, уровень или мера сложности дизъюнктов).

Продемонстрируем методы внесения в процедуру поиска не только синтаксической, но и семантической информации на примере *Q** алгоритма. Данный алгоритм состоит из двух подалгоритмов:

1. *Алгоритма выбора базового дизъюнкта,* осуществляющего выбор из множества всех дизъюнктов тех, которые относятся к решаемой задаче.
2. *Алгоритма дедукции,* определяющего последовательность, в которой к дизъюнктам применяются правила вывода .

Оба алгоритма удобно описать в контексте поиска *в системе редукций.*

Система редукций состоит из трех множеств: начальных задач, множества операторов и конечных (разрешимых) задач.

В контексте доказательства теорем методом резолюции будем дизъюнкт рассматривать как задачу, а литеру как подзадачу. Дизъюнкт при этом соответствует конъюнкции подзадач, так как для решения задачи все литеры должны быть разрешены путем применения операторов (других дизъюнктов).

Состав множества начальных задач зависит от используемой системы вывода. Например, если используется стратегия множества поддержки или стратегия, включающая множество поддержки (например, *OL*-вывод), то множество начальных задач состоит из дизъюнктов, входящих в отрицание доказываемой теоремы.

Множеством операторов является множество дизъюнктов, применимых к задаче посредством правил вывода. Однако множество применимых операторов зависит от используемых правил вывода. Резолюция использует два дизъюнкта, и для многих систем вывода любой из этих дизъюнктов можно рассматривать как задачу, а другой как оператор. Однако для ряда систем вывода кажется естественным делать различие между дизъюнктами. Например, в случае линейной или *OL*-резолюции боковые дизъюнкты естественно рассматривать как операторы. Конечными задачами являются дизъюнкты определенного

вида. Типичным примером является пустой дизъюнкт, сообщающий, что решена вся задача. Более сложным является определение окончания подзадачи.

Подзадача *A*, соответствующая литере в дизъюнкте, может быть решена одним из двух способов:

1) разрешена за один шаг применением опровергающего ее оператора (например, путем резольвирования с одночленным дизъюнктом, дополнительным к *A*);

2) оператор, примененный к *A,* порождает дополнительное множество подзадач; при этом *A* считается разрешимым тогда и только тогда, когда будут разрешены все подзадачи.

Определить на практике, когда данная подзадача разрешена, весьма трудно, так как в зависимости от системы вывода подзадачи рассматриваются в произвольном порядке. Решение указанной задачи в общем случае требует запоминания громоздкой информации. Однако в случае *OL*-резолюции проблема упрощается, так как на данном уровне испытывается только одна подзадача, а информация, требуемая для запоминания, хранится в представлении дизъюнкта. Если решение некоторой подзадачи приводит к другим подзадачам, то в *OL*-резолюции создается *A*-литера, а вновь образованным подзадачам соответствуют *B*-литеры, расположенные справа от данной *A*-литеры. Когда все *B*-литеры будут разрешены (*OL*-резолюция выполняется над самой правой *B*-литерой) и *A*-литера оказывается справа, то *A*-литера устраняется из дизъюнкта операцией сокращения (см. п.8.3.4), и только в этот момент выбирается для решения новая подзадача.

Процесс дедукции, используемый в алгоритме *Q**, подобен процессу поиска, описанному для алгоритма ∑*, и состоит в вычислении оценочной функции. Отличие состоит в том, что с целью ускорения взаимодействия уместных базовых дизъюнктов с вновь полученными дизъюнктами можно вводить базовые дизъюнкты не только в программе НАПОЛНИТЬ, но и в программе РЕКУРСИЯ. В алгоритме широко используются различные *правила вычеркивания,* направленные на устранение избыточных или семантически бессмысленных дизъюнктов. Примерами таких дизъюнктов являются *тавтологии, алфавитные варианты* уже существующих дизъюнктов или *поглощаемые дизъюнкты.*

Алфавитным вариантом некоторого дизъюнкта *C* называется дизъюнкт *C'*, получаемый из *C* подстановкой, заменяющей только названия переменных.

Дизъюнкт C_1 *поглощает* дизъюнкт C_2, если существует такая подстановка, что $C_1\sigma \subseteq C_2$. Например, дизъюнкт *P(x)* поглощает дизъюнкт $P(y) \vee Q(z)$, так как при $\sigma=\{y/x\}$

$$\{P(x)\}_\sigma \subseteq \{P(y) \vee Q(z)\}.$$

Тавтологии, алфавитные варианты и поглощаемые дизъюнкты являются неуместными и излишними, и они должны вычеркиваться, когда для этого предоставляется возможность. Вычеркивание алфавитных вариантов и тавтологий не нарушает полноты правил вывода. Вычеркивание поглощаемых дизъюнктов нарушает полноту некоторых правил вывода. Так, например, для бинарной резолюции (или семантической резолюции) и стратегии насыщения уровня вычеркивание поглощаемых дизъюнктов не нарушает полноты правил вывода, только если оно осуществляется после насыщения уровня.

Для устранения избыточных дизъюнктов применяется способ *означивания предикатов,* осуществляемый путем ссылок на семантическую информацию конкретной проблемной области. Так, например, пусть мы имеем дизъюнкт $C \vee \sim F(Мэри, x)$, где C — дизъюнкт. Если нам известно на основании семантической информации, что для истинности предиката F первый его аргумент должен быть объектом мужского пола, то мы можем определить, что $\sim F(Мэри, x)$ в данной интерпретации принимает значение «истино» и, следовательно, весь дизъюнкт можно отбросить, не нарушая свойств невыполнимости оставшегося множества. Если же некоторая литера в результате означивания приобретает значение «ложно», то она может быть устранена из того дизъюнкта, в который она входит.

Хотя означивание предикатов является полезной стратегией, но она не защищает от генерации неуместных дизъюнктов. Указанная задача может быть с успехом решена путем тщательного выбора уместных базовых дизъюнктов.

Для того чтобы выбрать аксиомы (операторы, директивы), уместные для данной задачи, алгоритм Q^* первоначально генерирует дизъюнкты из отрицания предписания. В зависимости от системы вывода алгоритм рассматривает одну или все литеры итерированных дизъюнктов (дизъюнктов «хозяев») как спецификацию, на основании которой осуществляется выбор аксиом. Каждую из этих литер будем называть *выделенной литерой.* Выделенная литера используется для выбора аксиом, которые *резольвируют* по данной литере с генерированным дизъюнктом. Таким образом, выделенные литеры действуют как образцы. Этот прием подобен идее Грина, на основании которой, только определенные дизъюнкты (активные) принимают участие в процессе поиска доказательства. Количество аксиом, отобранных выделенными литерами, может быть уменьшено отфильтровыванием на основании семантики (например, несоответствие типов аргументов). Оставшиеся после фильтрации операторы заносятся в список СПЕЦ и

упорядочиваются таким образом, чтобы более уместные испытывались в первую очередь.

В результате работы алгоритма выбора базового дизъюнкта дедуктивный алгоритм получит не все базовые дизъюнкты, имеющие данную меру упорядочения. Как следствие этого, выбранные базовые дизъюнкты не обязательно будут генерироваться в соответствии с мерой упорядочения. Следовательно, в отличие от алгоритма $\sum^*$, алгоритм Q^* не обладает допустимостью (см. р. 7). Однако потеря допустимости имеет только теоретическое значение. На практике кажется более важным найти быстро какое-нибудь решение, а не обязательно простейшее решение, требующее больших затрат и влекущее за собой риск не решить задачу в отведенные время и объем памяти.

Опишем теперь более подробно, каким образом происходит **выделение аксиом, их фильтрация и упорядочение**.

Как уже указывалось ранее, литеры генерированных дизъюнктов используются для выделения аксиом, которые становятся кандидатами для генерации. Система вывода, используемая алгоритмом дедукции, будет определять, какие дизъюнкты будут использоваться для этих целей и какие критерии будут применяться для выбора аксиом. Если система вывода использует стратегию опорного множества или если система связана с опорным множеством, как в *OL*-резолюции, то используются только литеры дизъюнктов, обладающих поддержкой. С другой стороны, если система вывода не включает стратегию опорного множества, то все сгенерированные дизъюнкты и общие аксиомы будут использоваться для выделения аксиом. При этом критерий резолюции должен быть ослаблен таким образом, что аксиома будет становиться кандидатом, если она содержит литеры, которые *унифицируют* с литерами сгенерированного дизъюнкта. Ослабление критерия резолюции необходимо для того, чтобы обеспечить опровергающую полноту алгоритма, например, в случае, когда для опровержения теоремы необходимо доказательство леммы, а предписание не используется до тех пор, пока лемма не доказана.

Система вывода указывает также, какие литеры должны использоваться при выделении аксиом. Так, например, такая система вывода, как *OL*-резолюция, указывает одну литеру каждого генерированного дизъюнкта, которая используется. Другие системы вывода используют для этих целей все литеры.

Рассмотрим теперь проблему семантической фильтрации найденных аксиом-кандидатов. Фильтрация может делаться на основе типов переменных и констант, входящих и выделенную литеру. Например, предположим, что литера генерированного дизъюнкта

имеет вид *~ РОДИТЕЛЬ (x, Петр)* и из контекста известно, что переменная *x* имеет тип *МУЖЧИНА*. Предположим, что потенциальными кандидатами, найденными в базовых данных, являются дизъюнкты:

1) *~ОТЕЦ(u, v) ∨ РОДИТЕЛЬ (u, v);*

2) МАТЬ (u, v) ∨ РОДИТЕЛЬ (u, v).

Применяя фильтр, получим, что аксиома 2) является несовместимой с выделенной литерой, так как переменная *u* в литере *РОДИТЕЛЬ* имеет из контекста тип *ЖЕНЩИНА* (так как является первым аргументом предиката *МАТЬ,* требующего типа *ЖЕНЩИНА).* Таким образом, кандидатом становится только дизъюнкт 1).

Если некоторая подзадача может быть полностью решена с использованием константных аксиом, запомненных в явном виде в базовых данных, то нет необходимости генерировать для решения подзадачи общие аксиомы. Например, пусть *МАТЬ (x, Эмиль)* является литерой сгенерированного дизъюнкта. Пусть системе известен факт, что у каждого человека есть точно одна мать. Если аксиома *МАТЬ (Роза, Эмиль)* найдена в базовых данных системы, то она полностью решает подзадачу и мы выбираем только одного кандидата. Никакие другие аксиомы, которые могут резольвировать с предикатом *МАТЬ (x, Эмиль),* не следует вводить в пространство поиска для решения этой подзадачи.

Описанный процесс фильтрации уменьшает число аксиом, которые принимают участие в процессе поиска. Так как меньшее количество аксиом будет участвовать в логическом взаимодействии, то уменьшится число генерируемых дизъюнктов.

Следует отметить, что, хотя фильтрация делает систему неполной в смысле исчерпания всех вариантов решений, она оставляет ее полной в смысле опровержения.

После того как аксиомы-кандидаты выделены и подвергнуты семантической фильтрации, их необходимо упорядочить так, чтобы «более перспективные» обрабатывались в первую очередь.

Упорядочение выполняется в два этапа:

1) упорядочение кандидатов, связанных с выделенной литерой (это соответствует упорядочению операторов, применяемых к подзадаче);

2) упорядочение списков кандидатов, соответствующих различным выделенным литерам (это соответствует упорядочению подзадач).

Упорядочение кандидатов выделенной литеры осуществляется двумя способами. Либо упорядочение совершается на основании рекомендаций пользователя аналогично списку рекомендаций, нпример, в языке PLANNER, либо в соответствии с оценочной

функцией, используемой в дедуктивном алгоритме. В последнем случае константные аксиомы предшествуют общим аксиомам

Для упорядочения подзадач (литер) используется эвристика, состоящая в предпочтении литерам, имеющим одну пли несколько констант. Смысл этой эвристики состоит в том, что

1) константные литеры будут резольвировать (унифицировать) с меньшим числом базовых дизъюнктов, чем общие литеры,

2) если предписпние касается частного утверждения, то предлагаемый механизм будет более уместен, чем слепой перебор на основе общих литер.

На рис. 9 приведен пример доказательства с предпочтением литерам, содержащим константы.

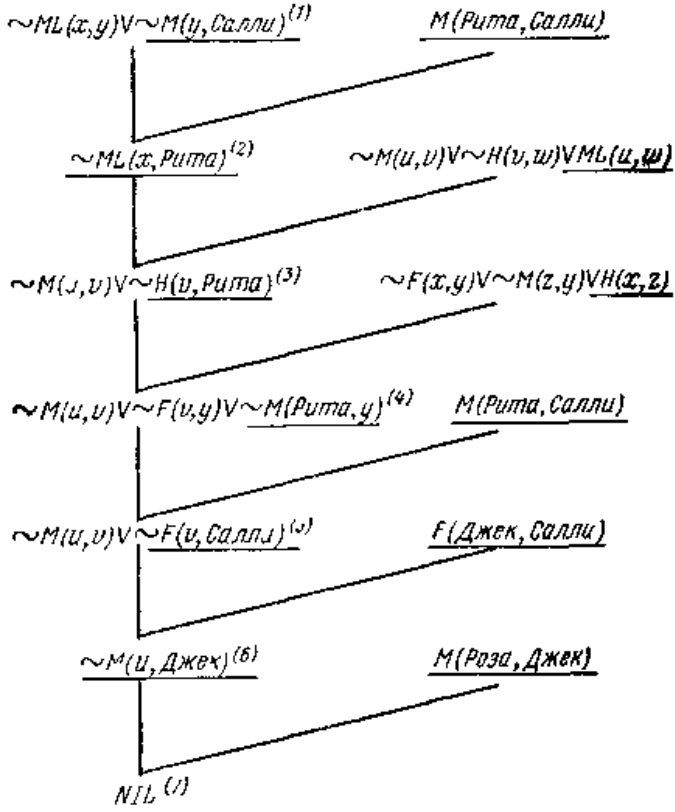


Рис. 9. Использование констант для ведения поиска.

Дизъюнкт 1) имеет две выделенные литеры. Мы отдаем предпочтение литере *~M(y, Салли),* так как она содержит константу. Аксиомы, которые могут взаимодействовать с литерой *~M (y, Салли),* принимают участие в поиске до других аксиом, применяемых к дизъюнкту 1). По аналогичным причинам мы отдаем предпочтение литере *~H (v, Puma)* в дизъюнкте 3).

Если существует несколько подзадач, содержащих константы, то упорядочение осуществляется на основании предсказанного значения оценочной функции резольвенты, полученной из дизъюнкта «хозяина» выделенной литеры и первой аксиомы в списке кандидатов этой литеры.

Следует отметить, что в некоторых базовых данных определенные константы встречаются очень часто. Примером может служить константа, отражающая пол (мужчина, женщина) в картотеке личного состава какого-либо учреждения. Так как такие константы часто встречаются в базе данных и не именуют конкретного индивидуума, то они мало будут помогать в ведении поиска. Поэтому эти типы констант при упорядочении целесообразно рассматривать как переменные.

Определенные события, происходящие в процессе поиска, могут быть использованы для сокращения пространства поиска и устранения определенных кандидатов из списка литер. Мы упоминали при описании фильтрации кандидатов-аксиом, что некоторые подзадачи могут быть полностью разрешены с помощью константных базовых аксиом. В более общей форме можно сказать, что некоторые подзадачи имеют точное число решений, в то время как другие — неопределенное число. Если число решений для данной подзадачи известно, то, когда все решения найдены, часть графа поиска (сформированная для решения данной подзадачи) может быть отброшена. В дополнение к этому аксиомы-кандидаты, выделенные при решении этой подзадачи, могут быть устранены. Результатом такого сокращения является уменьшение числа неуместных дизъюнктов в пространстве поиска и, что даже более важно, устранение их преемников.

Выполнение этого семантического действия весьма сложно для произвольной системы вывода, так как для данного дизъюнкта (задачи) попытки решения всех подзадач выполняются одновременно. Однако для *OL*-резолюции этот подход применим, так как здесь в текущий момент рассматривается только одна подзадача и информация, требуемая для обнаружения решения подзадачи, встроена в представление *OL*-резолюции.

Итак, мы описали приемы по использованию семантики в процессе поиска доказательства теорем. Некоторые из этих приемов являются довольно общими и могут применяться для многих проблемных областей.

Заканчивая описание методов доказательства, отметим, что основными способами устранения причин «экспоненциального взрыва» имеющего место при доказательстве теорем практической степени сложности, является использование семантики и встраивание в правила вывода и алгоритмы унификации специфики конкретной области применения.

8.6. О машинном доказательстве теорем

Современные компьютеры могут выполнять множество сложнейших операций, доступных ранее только человеку. Кроме таких привычных действий, как распознавание образов, перевод текстов с одного языка на другой, они все больше внедряются в самые сложные сферы искусственного интеллекта – научного направления, моделирующего процесс человеческого мышления. Одно из таких направлений – автоматическое доказательство теорем – мы собираемся рассмотреть ниже.

Практически каждый из нас имеет представление о том, что такое аксиомы, определения и как проводятся математические доказательства. Подобную работу могут выполнять компьютеры (вернее, компьютерные программы), что, кажется, должно свидетельствовать о том, что они способны мыслить – в нашем, человеческом, понимании. На самом деле это не так. Они лишь слепо используют определенный набор правил и приемов, которому их нужно предварительно научить. Однако недостаток творческого начала в некоторых случаях вполне может компенсироваться быстродействием, и хорошее тому подтверждение – успехи шахматных программ. А исходной точкой для «машинного интеллекта» всегда будет некоторая формализация наших знаний, каковой является, в частности, математическая логика – дисциплина, посвященная изучению доказательств.

Автоматическое доказательство и логическое программирование

Установление отношений логического следствия, или, что то же самое, доказательство теорем является одной из главных задач логики. Она

представляет не только теоретический, но и практический интерес для многих научных и технических областей. В качестве примера назовем две достаточно широкие научно-прикладные сферы, проблемы из которых могут быть выражены в терминах доказательства:

- анализ программ. Если процесс выполнения программы описывается некоторой логической формулой A, а условие завершения – формулой B, то проверку того, что программа закончит работу, можно сформулировать как доказательство следования B из A;
- преобразование состояний. Если заданы набор состояний и множество операторов над ними, то проблема достижения некоторого заданного состояния из исходного опять же сводится к доказательству логического следствия.

В логике было предложено немало методов установления логического следствия. Большинство из них основаны на доказательстве того, что некоторая логическая формула, связанная с предписание (директивой), истинна или ложна. Однако данные методы требуют большой рутинной работы (переписывания, перебора вариантов) для своей реализации, т. е. слишком трудоемки и громоздки для «ручного» применения. В то же время они легко программируются на ЭВМ, и это делает задачу установления логического следствия гораздо более доступной. Приемы и алгоритмы, используемые для решения таких задач, будем называть автоматическими (машинными) методами доказательства.

Хотя многие алгоритмы были предложены давно и постоянно совершенствуются, их реализация на традиционных языках программирования вызывает существенные трудности. Потребность в адекватных инструментальных средствах привела к созданию целого направления в программировании, называемого логическим, которое включает подходы и методы, приспособленные для решения задач логики, и в рамках которого созданы специализированные языки.

При решении задач на этих логических языках, как правило, описывают предметную область (называемую иногда универсумом рассмотрения), и на ней определяют функции и предикаты, выражающие взаимосвязи между исследуемыми сущностями. Логическое программирование при этом обладает особыми свойствами, отличающими его от других подходов. В функциональном

программировании процедура действует как математическая функция, возвращая значение от аргументов. Логическая же процедура соответствует математическому понятию отношения и обладает большей гибкостью. При определении отношений ничего не говорит о том, какие аргументы исходные и каковы возможные результаты – в разных запросах одни и те же объекты могут играть различные роли.

Само исчисление, реализованное в логическом программировании, состоит из **формализованного языка и механизма вывода.** Последний, в свою очередь, включает некоторое множество исходных предложений, называемых аксиомами, и правил вывода, позволяющих из одних предложений получать другие. Последовательность предложений, каждое из которых либо является аксиомой, либо следствием предыдущих, называется формальным выводом или доказательством, а предложения, получаемые в этом процессе, – предписаниями.

Наличие формальных правил вывода позволяет проверять правильность доказательств предписаний, порождая с их помощью из аксиом все новые и новые следствия – до получения интересующего нас предписания. Не менее часто применяется и обратный метод: берут интересующее предписание и выясняют, из каких посылок оно может быть получено, затем анализируют эти посылки и т. д., пока не придут к аксиомам. К сожалению, в обоих случаях крайне сложно оценить необходимый объем вычислений. Программные пакеты, выполняющие автоматические доказательства или некоторые их элементы, традиционно разделяют на *интерактивные* и *автоматические пруверы* (системы доказательств, interactive/automated theorem provers) и *верификаторы моделей* (model checkers). Данная классификация основана на некоторых параметрах решаемых задач и степени участия пользователя, хотя в целом такое разделение в известной степени условно. Надо отметить, что с развитием машинных методов стал актуальным вопрос о том, что же считать доказательством в математике. **Традиционно под ним понималась обозримая последовательность выводов, которую можно проверить «вручную», условно говоря, с помощью карандаша и листа бумаги. Однако участие в процессе компьютера потребовало пересмотра такого «устаревшего» представления.**

Для иллюстрации сказанного приведем пример фундаментальной математической проблемы, решенной (в основном) машинными

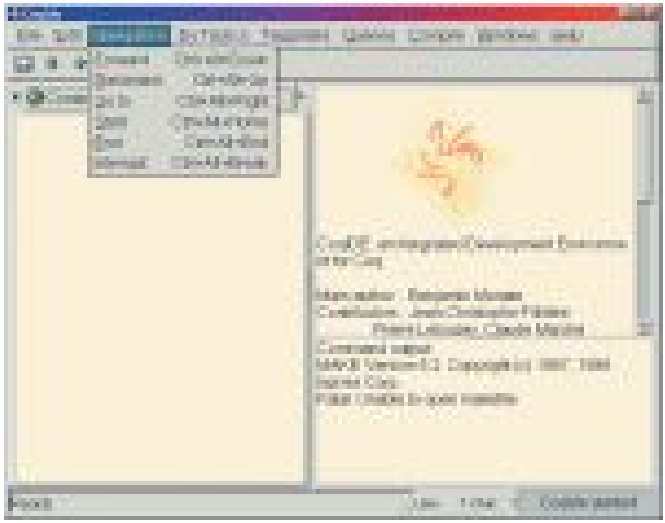
методами. Это так называемая задача о четырех красках, не поддававшаяся аналитическому решению более века. Она была сформулирована в 1852 г., когда один английский географ заметил, что для раскраски карты Британии (при которой соседние графства имели бы разный цвет) достаточно четырех красок. Математики, заинтересовавшиеся этим любопытным фактом, задались вопросом: а достаточно ли четырех красок для раскрашивания произвольной карты? Оказывается, да. Упуская занимательную предысторию, скажем, что первое правильное доказательство этой теоремы было получено только в 1976 г. с использованием машинных методов. Между тем оно до сих пор вызывает сомнение у ряда специалистов. Дело в том, что в нем одновременно с громоздкой аналитической частью доказательства, состоящего из десятков страниц сложнейших выкладок и тысяч (!) дополнительных диаграмм, имеется машинная часть, которую удалось проверить только на компьютере (использовался суперкомпьютер Cray-1A). Авторы доказательства логически свели задачу к исследованию 1482 базовых карт, сформированных специальным образом – их раскраска на Cray потребовала 50 суток компьютерного времени. Это-то и поставило под сомнение надежность данного метода – вероятность ошибки (как программной, так и аппаратной) при таком объеме не является пренебрежимо малой. Впрочем, доказательство 1976 г. было подтверждено позже. В 1990-х годах задача о четырех красках была сведена к раскрашиванию «всего» 633 базовых карт, что может быть реализовано на современном ПК за вполне приемлемое время – всего за несколько часов. Тем не менее поводы для скепсиса не исчезли – ведь задача опять решена машинными методами, и о проверке доказательства «вручную» можно даже не мечтать.

Coq и Gallina

Важной особенностью большинства приложений, реализующих методы автоматического доказательства, является их некоммерческий, экспериментальный характер. С одной стороны, это обеспечивает бесплатный и неограниченный доступ к данному ПО всех желающих, но с другой – подобные системы, как и вообще многие некоммерческие IT-проекты, зачастую не имеют развитой пользовательской поддержки, нередко отличаются нестабильным функционированием и т. д.

Выше мы кратко упомянули об основных классах логических программ. К сожалению, даже тезисный обзор лучших представителей

каждого из них, который смог бы сориентировать читателя, скажем, в выборе подходящего ему пакета, является совершенно необозримой задачей. Поэтому в качестве примера мы детальнее остановимся лишь на одном из продуктов, обладающим, несмотря на общедоступность, весьма развитыми инструментами для решения сложных задач. Система автоматизированного построения доказательств Coq, созданная в исследовательском институте INRIA (Франция), является достаточно сбалансированным приложением, сочетающим поддержку широкого спектра логических процедур, высокую стабильность работы и относительную простоту освоения для широких кругов пользователей. Система и сопровождающая ее документация могут быть свободно загружены с узла продукта. Разработчики определяют Coq как Proof Assistant, что в приведенной классификации соответствует интерактивным пруверам. Кроме того, Coq поддерживает процедуры полностью машинного вывода, характерные для автоматических пруверов, т. е. система обладает качествами сразу двух выделенных нами классов программ. Coq реализован на нескольких платформах (Linux, Mac OS, Windows, Solaris) и имеет набор интерфейсов, облегчающих взаимодействие с пользователем. В ОС Windows основным средством работы с системой является стандартная командная консоль, хотя в поставку входит и графический интерфейс CoqIde.



Интерфейс интегрированной среды разработки системы автоматизированного построения доказательств Coq

Базируется Coq на собственном языке спецификаций Gallina, состоящем из **объявлений, определений и команд**. Объявление ставит в соответствие некоторому имени его спецификацию, что примерно отвечает созданию переменной определенного типа в алгоритмических языках. **Спецификации делятся на логические высказывания, математические коллекции и абстрактные типы.** Например, спецификация nat является аксиоматическим понятием, принадлежащим математической коллекции, и означает натуральное число. Команда Variable n:nat. объявляет объект n как переменную натурального типа.

С помощью определений создаются и именуются новые объекты (в то время как объявления приписывают объектам только типы). Например, в арифметическом модуле Coq определен объект plus, ставящий в соответствие двум натуральным числам их сумму (очевидно, что он является прямым аналогом функционального символа «сумма» из исчисления предикатов).

Как же проводятся доказательства в Coq? Кратко поясним только схему, не вдаваясь в детали. **Определение (в терминах Gallina называемое целью), которое должно быть доказано, записывается с помощью соответствующих операторов**. Заранее, если необходимо, вводятся используемые в нем объекты – переменные, функциональные символы и др. Затем к цели применяются специальные команды, называемые **тактиками**, являющиеся примитивами построения доказательства. Тактика действует на текущую цель, пытаясь построить доказательство исходного определения, возможно, на основе некоторых гипотетических суждений, добавляющихся затем к текущему списку определений. Процесс проведения доказательства с помощью тактик предусматривает активное участие пользователя и осуществляется, как правило, за несколько шагов, число которых может быть весьма значительным. При этом от него требуются довольно глубокие знания языка спецификаций. Таким образом, Coq функционирует как интерактивный прувер. Однако в Gallina также имеются тактики, пытающиеся построить доказательство без участия человека. Для их применения достаточно введения единственной команды, а в случае успеха выдается соответствующее сообщение. В этом случае Coq действует подобно полностью автоматическим системам.

В завершение нашего знакомства с Coq укажем на такое важное качество системы, как модульность и расширяемость. Применяемые в системе понятия и тактики оформлены в виде модулей-библиотек, которые можно подключать по мере необходимости. При обычном старте в систему загружается минимальный набор библиотек, содержащий лишь логические связки и ряд арифметических понятий. Для работы в каких-то специфических предметных областях потребуются соответствующие дополнительные инструменты.

Таким образом, Coq является открытой расширяемой системой, способной выражать самые разные системы понятий и решать в них логические задачи. Так, в 2004 г. была завершена полная формализация в Coq упомянутой теоремы о четырех красках, что продемонстрировало ее высокие возможности. Завершая этот краткий обзор, хотелось бы затронуть вопрос о сравнении интеллектуальных способностей человека и возможностей современных пакетов автоматических доказательств при проведении логических выкладок. Иными словами, способны ли сегодняшние системы доказательств конкурировать, скажем, с математиками, делающими это без применения компьютеров? При всей неоднозначности такой постановки вопроса считается, что машинные системы доказательств пока заметно отстают от профессиональных ученых, тогда как уровень среднего школьника и даже студента ими давно пройден. Для сравнения: компьютерные программы, играющие в шахматы, сегодня практически победили человека; имеется всего лишь несколько шахматистов, способных играть на равных с Fritz или Shredder, особенно с их двухпроцессорными версиями (которые обычно снабжаются приставкой Deep). При этом программы неустанно совершенствуются, так, настоящий фурор произвел шахматный движок Rybka (созданный всего одним человеком, правда, профессиональным шахматистом), возглавляющим сегодня все рейтинги. Вместе с тем имеются области, где пруверы вне конкуренции – например, в Software Engineering для проверки корректности программ или непротиворечивости требований к ПО. Поэтому хотя невозможно спрогнозировать, превзойдут ли машинные системы доказательств человека, в настоящее время они уже нашли свои ниши, и в дальнейшем сфера их применения будет только расширяться.

Формирование и моделирование речевой коммунизации будем осуществлять средствами логики высказываний и логики предикатов.

9. Планирование и выпонение действий интеллектуальных объектах

9.1. Анализ систем решения задач

9.1.1. Миры и планы

Рассмотренные в предыдущих разделах методы представления и решения задач в системах ИИ создают необходимую концептуальную и алгоритмическую основу для перехода к анализу систем решения задач искусственными интеллектуальными объектами (ИИО). Мы предполагаем, что ИИО существует и функционирует в мире, который обновляется как в результате действий самого ИИО, так и в результате некоторых независимых от ИИО действий в этом мире.

Будем предполагать, что ИИО обладает способностью воспринимать информацию о мире, решать задачи и выполнять активные действия в этом мире. С алгоритмической точки зрения это означает, что ИИО обладает описанием мира в виде модели, списком моделей элементарных действий, или операторов, в общем случае изменяемых разработчиком ИИО, либо автоматически в результате обобщения опыта функционирования ИИО, а также общим механизмом создания целенаправленных стратегий в виде последовательности элементарных действий или операторов и воплощения стратегий в рациональные программы действий ИИО в окружающем мире.

Будем различать два больших класса миров — статические и динамические миры.

Статическим миром мы назовем мир, в котором источником активных действий является ИИО или совокупность ИИО, управляемая централизованно, а факты омире, представляемые в его модели, не зависят в явной форме от времени. Соответственно мы можем указать следующие факторы, определяющие *динамичность мира:*

а) наличие в мире независимых от ИИО источников действий, изменяющихмир; к числу таких источников могут относиться другие ИО или «слепые силы природы»;

б) наличие в мире фактов и определение в ней действий, явно зависящих от времени (например, факт OPEN (D1, 17, 19), т. е. «дверь D1 открыта с 17 до 19 часов», или оператор TURN (R1, 20), т. е. «вернуться в комнату R1 к 20 часам».

Следствием статичности мирв является возможность упорядочения множества фактов, описывающих мир, по степени их неизменности в

процессе решения задачи. Такое упорядочение является весьма относительным и может меняться от задачи к задаче. Так, если ИИО перемещает ящики, то факты об их цвете являются более неизменными, чем об их положении. Если же ИИО красит ящики, то ситуация оказывается противоположной Однако, по-видимому, положение ящика является всегда более неизменным фактом, чем положение ИИО.

Возможность ранжирования фактов по степени их неизменности является для нас важной (см. п.9.4.1). Отметим, чго в сложных динамических мирах можно и не найти такое ранжирование.

Целенаправленная стратегия поиска решения задачи называется *планом.*

Определим *пространство моделей M* как множество всех возможных состояний моделей мира, описываемых алгоритмической системой ИИО.

Моделью элементарного действия ИИО в мире, или *оператором,* называется некоторое отношение F_j, определенное в пространстве моделей, т. е. $F_j: M \rightarrow 2^M$. Множество всех операторов обозначим через F. В общем случае отношение, определяющее оператор, является частичным отношением, определенным на собственном подмножестве M_k множества M.

Теперь мы можем дать формальное определение плана.

Планом называется помеченный направленный граф, удовлетворяющий следующим условиям:

I) каждая дуга графа помечена оператором $F_j \in F$, в общем случае оператором-схемой, т. е. семейством операторов, определяемым некоторым параметром;

2) с каждой вершиной графа n_k связана некоторая формула R_k, определяющая в свою очередь подмножество $M_k \subseteq M$,

M — множество моделей;

3) из одной вершины исходят одинаково помеченные дуги;

4) множество M_k, соответствующее n_k, содержится в области определения оператора F_j, помечающего дуги, исходящие из этой вершины.

Это определение является важным по нескольким причинам. Во-первых, оно определяет *обобщенный план,* т. е. множество планов, которое можно получить из исходного конкретной подстановкой значений параметров. Во-вторых, множество дуг, исходящих из одной вершины графа, определяет множество *возможных результатов применения* оператора, т. е. для каждого $m_i \in M_k$ $F_j(m_i)=\{m_1, m_2, \ldots, m_n\}$. Наконец, в-третьих, из определения вытекает применимость оператора F_j к любому состоянию модели $m_i \in M_k$.

План называется *простым,* если для всех вершин n_k оператор является однозначной функцией модели, т. е. $F_j(m_i)=m$, $m_i \in M_k$, $m \in M$ (из каждой вершины исходит ровно одна дуга).
План называется *сложным,* если каждому результату применения оператора m_p, p=1, 2, ..., n, приписывается некоторая оценка C_p правдоподобия его появления или оценка его полезности с точки зрения достижения цели (т. е. оценки приписываются дугам, исходящим из вершины n_k). В случае, если альтернативным результатам применения оператора не приписаны оценки (или, что то же самое, приписаны одинаковые оценки), *план* называется *составным.* План не обязательно может приводить к достижению поставленной цели. Мы вводим в связи с этим определение *полного* и *неполного плана.* План P называется *полным планом из начальной модели* m_0 *к целевой* m, если существует $P' \subset P$ такой, что 1) P' есть план; 2) все вершины P' достижимы из такой вершины n_r, что $m_k \in M_r$; 3) существует по меньшей мере одна такая вершина $n_t \in P'$, что R_t влечет за собой m. *План,* не удовлетворяющий хотя бы одному из этих условий, называется *неполным.*
Алгоритмическая система ИИО, осуществляющая построение планов решения задачи, называется *планирующей системой* (ПС).

9.1.2. Планы и действия

Проблема планирования для ИИО занимает важное место в проблемах решения задач СИИ. Поскольку ИИО должен функционировать в реальном мире, т. е.мире, неполностью соответствующей той модели мира, которая заложена в ИИО, взаимосвязь проблемы планирования и выполнения действий является принципиально важным фактором, определяющим возможность выполнения ИИО поставленных перед ним задач. В отличие от оператора, *элементарное действие* ИИО в мире функционирования есть отношение, определенное на декартовом произведении пространств мира W и моделей M и отображающее его в самого себя,

$$Q_j : W \times M \rightarrow W \times M. \qquad (1)$$

Таким образом, мы можем выделить в действии две компоненты: компоненту нового состояния мира

$$w = Q_{j,w}(w_i, m_i) \qquad (2)$$

и компоненту нового состояния модели этого мира в ИО

$$m = Q_{j,w}(w_i, m_i). \qquad (3)$$

Идесь Q_j представляет из себя *действие-схему,* т. е. множество действий, определяемое некоторым параметром.

В силу неточности и неполноты моделей для достаточно сложных миров, а также погрешностей восприятия окрежающей среды ИИО и его подсистем мы не можем ожидать функционального соответствия между миром его моделью в ИИО, т. е. отношение моделирования $R: W \rightarrow M$ не является функцией n в строгом смысле слова, что не позволяет в общем случае надеяться на полную формализацию процесса построения адекватных моделей мира ИИО.

При создании формальной теории планирования и представления действий в ИИО естественным решением является организация обратной связи, позволяющей корректировать модель мира в соответствии с действительными изменениями, происходящими в нем.

Выполнение планов и корректировка модели мира в соответствии с действительными результатами их выполнения осуществляются специальной алгоритмической системой ИИО, называемой *исполняющей системой* (ИС). Таким образом, ИС выполняет план, обращаясь к определенным действиям, которые, как предполагается, соответствуют операторам в плане, получает планы (в том числе неполные), обращаясь к ПС, и выдает команды ПС о корректировке плана в случае выявления несоответствий действительного состояния мира и его модели в ИИО. В каждом состоянии ИС должна осуществить выбор между планированием и действием.

Определения полных и неполных, простых, составных и сложных планов создают основу для классификации ИС. Эта классификация приведена в таблице 1.

Таблица 1

		Наличие только полных планов	Наличие полных или неполных планов
Отсутствие обратной связи (только простые планы)		A	B
Наличие обратной связи	Простые планы	C	F
	Составные планы	D	G
	Сложные планы	E	H

Примечание. Буквы идентифицируют класс ИС для таблицы 7.2 и дальнейшего изложения.

Здесь под обратной связью понимается проверка модели с помощью ИС после каждого шага выполнения плана. В таблице 2 сведены основные функциональные особенности ИС различного класса.

Таблица 2

Класс ИС	Характер взаимодействия ПС и ИС	Действия ПС при несоответствии мира и текущей модели	Характер работы ИС
A	Полное соответствие последовательности действий плану (даже если он неудачен)	Несоответствие не учитывается, так что управление к ПС в процессе выполнения плана не передается	Простое прохождение списка действий
B	Полное соответствие на уровне неполного плана	При несоответствии неполного плана результатам его выполнения возможен пересмотр плана при хранении дерева планирования с оценками полезности	Простое прохождение списка действий, соответствующего неполному плану
C	Полное соответствие при успешном выполнении. Обмен информацией для проверки соответствия модели миру после каждого шага выполнения	Полное перепланирование, однако при хранении дерева планирования возможно использование промежуточных результатов	Выполнение действия — проверка соответствия — сигнал о выполнении (соответствие) или о перепланировании (несоответствие)
D E	То же, что и C, но производится проверка и модели, и плана	Поиск дочерней вершины, соответствующей новому состоянию модели. Если такой вершины нет, аналогично C	Выполнение действия — проверка соответствия — поиск возможных продолжений, при несоответствии — перепланирование в случае неудачи
F G H	То же, что в C / То же, что в D и E — На уровне неполного плана. Выбор планирования или выполнения в соответствии с оценками	То же, что в C, D и E соответственно для неполного плана. Перепланирование по сигналу ИС	Решение задачи выбора планирования или выполнения на каждом шаге

Наиболее важным является выбор направления деятельности ИС в случае работы с неполными планами. На каждом этапе решения задачи ИС стоит перед дилеммой: выполнять действия в соответствии с намеченным неполным планом или продолжать построение неполного плана. Каждая из альтернатив может привести к нежелательным результатам. Действительно, при выполнении неполного плана ИИО может не достигнуть цели, в то время как дальнейшее планирование могло бы это показать. С другой стороны, продолжение планирования может оказаться нецелесообразным, если выполнение уже построенного плана могло бы показать его бесперспективность. Таким образом, планирование и выполнение для ИС класса F, G и H являются конкурирующими действиями. С каждым из этих действий могут быть связаны некоторые оценки полезности в виде, например, функции стоимости уже построенного плана и оценки стоимости остатка плана, приводящего к цели. Другим возможным подходом являлась бы формулировка задачи планирования как игры с природой. Во всяком случае, именно здесь открываются возможности использования изложенных в предыдущих двух разделах (темах) алгоритмов. В свою очередь ИС по результатам своих действий и оценивая текущее состояние планирования, могла бы определить в каждом конкретном случае, планировать или выполнять.

Постановка задачи о совместной оптимизации процессов планирования и выполнения связана, по-видимому, с некоторым обобщением процессов в единый процесс, аналогичный процессу решения задачи. Два таких возможных обобщения мы рассмотрим в п.9.5.2.

В заключение этого параграфа мы отметим еще одну задачу, встающую в связи с проблемой планирования и выполнения действий,— задачу обобщения моделей и уже найденных планов и использования их как в последующих процессах построения планов, так и при выполнении действий. Один шаг на пути к такому обобщению уже сделан введением операторов-схем. Дальнейшее развитие идей обобщения — это постепенное наращивание множества операторов путем их укрупнения, а также соответствующее обобщение условий их применения, с тем чтобы создать иерархию планов от укрупненных к более детализированным.

Заучивание и обобщение получаемых в процессе решения задач успешных частичных решений является предметом исследований в области теории ИИ. Мы рассмотрим развитие этих идей в преломлении к проблемам планирования и выполнения действий в ИИО.

9.2. Планирующая система «Решатель задач STRIPS»

Рассмотрим вопросы, связанные с планированием и выполнением действий в ИИО, кратким описанием планирующей системы «Решатель задач» (ПС РЗ) по следующим причинам:

1) система представляет весьма широкий класс универсальных решателей задач для ИИО, работающих в достаточно сложном мире;

2) ПС РЗ является, возможно, пока единственной системой, которая работает в реальном мире, взаимодействуя с ИС «Исполнитель планов» (ИП) PLANEX;

3) ПС РЗ является иллюстрацией проблем планирования и выполнения действий и возможных путей их решения;

4) формализм ПС РЗ отражает все преимущества и недостатки декларативных предписаний, создавая в то же время приемлемую основу для ряда обобщений.

Тривиальный ИИО РЗ функцинрует в среде, состоящей из комнат с дверьми и предметами (ящики, призмы), и способна в автоматическом режиме осуществлять с этими предметами относительно простые коммуникации.

Мир ИИО представляется в ПС в виде модели, состоящей из набора правильно построенных формул (ппф) в исчислении предикатов первого порядка, описывающих состояние мира в данный момент.

Действия ИИО моделируются множеством операторов, определяемых *наименованием, списком параметров,* а также *условиями применимости (предусловиями)* и *результатами действия* в виде *схем ппф,* т. е. ппф, зависящих от параметров.

Результаты действия оператора описываются *списком вычеркивания* тех схем ппф, которые перестают быть истинными после применения оператора, и *списком добавлений* схем ппф, которые становятся истинными после применения оператора.

Важно различать параметры, входящие в схему ппф, и обычные связанные переменные, т. е. переменные под знаком кванторов общности или существования. В связи с этим системы доказательства теорем , описанные в р. 8, требуют некоторой модификации.

Пусть имеется некоторая целевая схема ппф $G(p)$, p — множество параметров схемы, которую нужно доказать на множестве M дизъюнктов, т. е. найти опровержение множества дизъюнктов $M \bigcup \{\sim G(p)\}$. Для вычисления частного случая p' множества p, при котором множество $M \bigcup \{\sim G(p)\}$ невыполнимо, можно использовать стандартный алгоритм унификации (п.8.2). С помощью этого алгорит-

ма можно найти наиболее общие частные случаи параметров, при которых обеспечивается унификация. Однако необходимо определить, какие подстановки допустимы в случае параметров. Определим следующие типы термов, которые могут быть подставлены вместо переменной: переменные, константы, параметры и функциональные термы, не содержащие переменных. Вместо параметра могут быть подставлены следующие типы термов: константы, параметры и функциональные термы, не содержащие функций Сколема, переменных или параметров.

Поскольку один и тот же параметр может иметь несколько вхождений в множестве дизъюнктов, он должен замещаться при резолюции термом во всех дизъюнктах, являющихся производными от резольвенты.

В качестве примера приведем оператор «переместить объект *k* из места *m* в место *n*». Наименование:

PUSHTO *(k, m, n) (k, m, n* — параметры).

Предусловия:

At (Отправитель, *m)* ∧ At *(k, m)* (и отправитель, и объект *k* должны быть в месте *m).*

Список вычеркиваний:

At (Отправитель, *m*);

At *(k, m)* (отправитель и объект *k* больше не находятся в месте *m).*

Списокдобавлений: At (Отправитель, *n);*

At(*k, n)* (отправитель и объект *k* находятся в месте *n).*

Конечная модель или цель также описывается в виде ппф.

Трудности использования формальных методов доказательства теорем в качестве стратегий поиска плана, главным образом связанные с проблемой границ, оказались непреодолимыми. В связи с этим в ПС процесс поиска плана полностью отделен от метода доказательства теорем. Последний используется только внутри модели мира для ответа на вопросы, связанные с анализом применимости операторов и проверкой выполнимости условий достижения цели.

Для поиска решения в пространстве моделей ПС РЗ использует описанный ранее механизм редукции GPS. Преимущество такого комбинированного подхода заключается в возможности рассмотрения сложных моделей, используя описательную мощь исчисления предикатов первого порядка, и использования эффективных эвристик разбиения цели на подцели, свойственных механизму редукции GPS.

В процессе поиска механизм редукции порождает иерархию цели, подцелей и моделей, которую можно представить в виде дерева поиска. Каждая вершина дерева имеет вид (модель, (список целей)) и

соответствует задаче достижения по порядку подцелей из списка целей указанной модели среды. Схема работы ПС представлена на рис. 1.

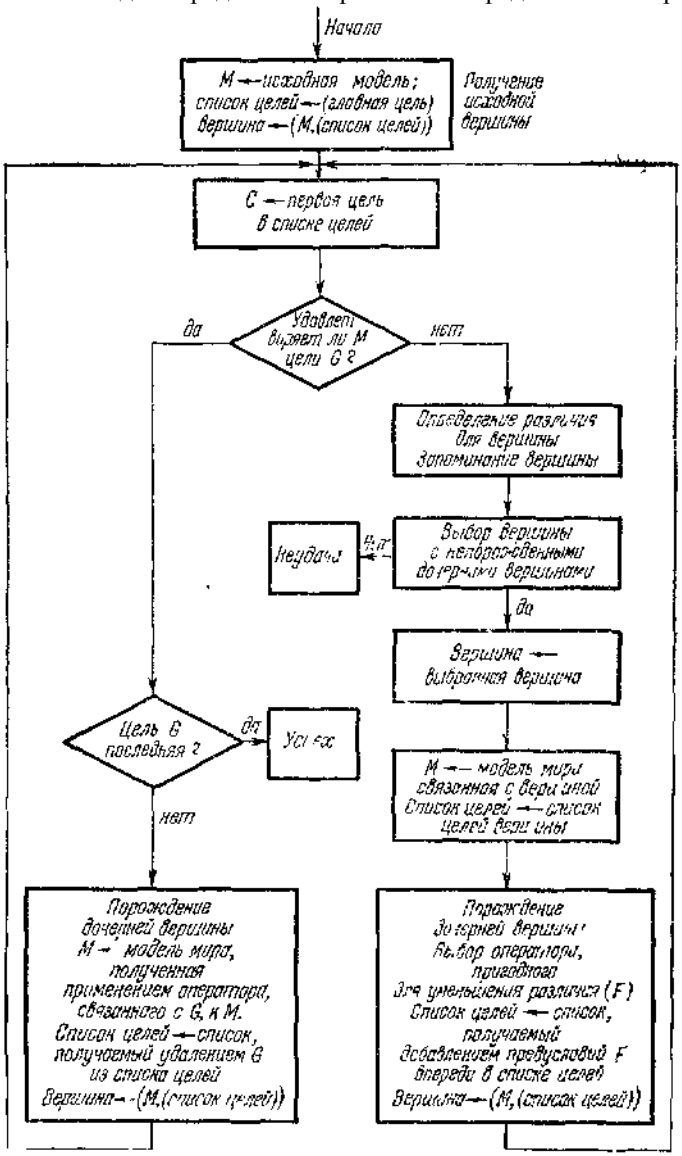


Рис. 1. Схема работы планирующей системы РЗ.

Мы рассмотрим принципы работы системы на примере и дадим для этого примера дерево поиска. Мир отправвителя изображен на рис. 2.

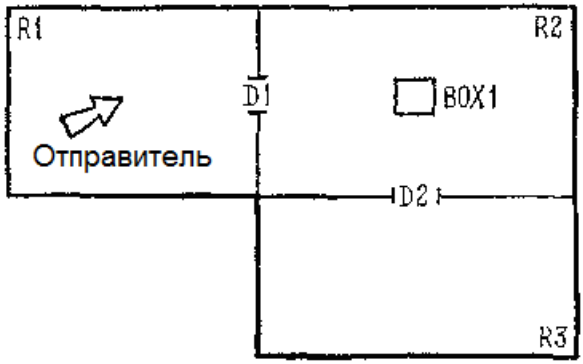


Рис. 2. Мир отправителя для примера задачи, решаемой ПС РЗ.

Пусть перед отправителем ставится задача передвинуть ящик из комнаты R2 в комнату R1.
Начальная модель M_o.
M_o: INROOM (Отправитель, R1) — отправитель находится в комнате R1,
CONNECTS (Dl, Rl, R2) — комнаты R1 и R2 соединены дверью D1.
CONNECTS (D2, R2, R3) — комнаты R2 и R3 соединены дверью D2.
BOX (BOX1) — ящик есть BOX1.
INROOM (BOX1, R2)— ящик BOX1 находится в комнате R2.
($\forall x \forall y \forall z$*)* [CONNECTS (*x, y,* z)→CONNECTS (*x, y,* z)]— порядок комнат в предикате CONNECTS безразличен.
Целевая ппф G_o.
G_o: $(\exists x)$[BOX(*x*) $\wedge$ INROOM (*x*, Rl)].
Нам задан список из двух операторов:
1) GOTHRU(*d*, *r*1, *r*2*)* — отправитель идет через дверь *d* из комнаты *r*1 в комнату *r*2 *(d*, *r*1, *r*2— параметры).
Предусловия:
INROOM (Отправитель, rl) $\wedge$ CONNECTS *(d,* rl, r2).
Список вычеркиваний:
INROOM (Отправитель, $), где $ может принимать любые значения.
Список добавлений.
INROOM (Отправитель, *r2).*

2) PUSHTHRU *(b, d, r*1, *r2)* — отправитель толкает объект *b* через дверь *d* из комнаты *r*1 в комнату *r2 (b, d, r*1, *r2* — параметры).
Предусловия:
INROOM *(b,* rl) ∧ INROOM (Отправитель, *r*1) ∧ CONNECTS (*d, r*l, *r*2).
Список вычеркиваний:
INROOM (Отправитель, $);
INROOM *(b,* $).
Список добавлений
INROOM (Отправитель, r2);
INROOM (*b*, r2).
Поиск решения начинается с попытки доказать, что G_0 следует из M_o Эта попытка терпит неудачу, однако часть целевой ппф BOX *(x)* удовлетворяется при *x*=BOX1. Поэтому определяется различие между G_0 и M_0 в виде INROOM (BOX1, Rl). Система определяет, что частичная подстановка PUSHTHRU (BOX1, *d,* rl, Rl) может обеспечить это высказывание, так как в его списке добавлений имеется INROOM *(b, r2).* Тогда в качестве новой цели G1 устанавливаются предусловия этого оператора с соответствующими подстановками параметров, т. е.
Gl: INROOM(BOX1, *r*1) Л INROOM (Отправитель, *r*1) ∧
∧ CONNECTS(*d, r*l, Rl)
Gl не может быть доказано из M_o и определяется различие между ними в виде INROOM (Отправитель, R2), так как при *r*1= R2 и *d=D*l остальные литералы G1 имеются в M_o. Система устанавливает, что для устранения этого различия уместен оператор GOTHRU при r2=R2 Устанавливается очередная подцель
G2 INROOM (Отправитель, rl) ∧ CONNECTS(*d, r*l, R2), которая может быть выведена из M_o при rl=Rl и *d=D*l Поэтому к M_0 применяется GOTHRU (D1, Rl, R2), преобразуя ее в
Ml: INROOM (Отправитель, R2);
CONNECTS (Dl, Rl, R2);
CONNECTS (D2, R2, R3);
BOX (BOX 1);
INROOM (BOX 1, R2);
($\forall x \forall y \forall z$*)*[CONNECTS *(x, y,* z)→CONNECTS (*x, z,y*)].
Теперь делается попытка доказать Gl из новой модели Ml. Эта попытка приводит к успеху при rl=R2, *d=D*1. Таким образом, к Ml применяется оператор
PUSHTHRU (BOX1, Dl, R2, Rl),
так как остальные подстановки были сделаны ранее. Модель Ml преобразуется в
M2: INROOM (Отправитель, Rl);
CONNECTS (Dl, Rl, R2);

CONNECTS (D2, R2, R3);
BOX (BOX 1);
INROOM (BOX 1, Rl),
($\forall$ *x* $\forall$ *y* $\forall$ *z)* [CONNECTS *(x, y, z)*→CONNECTS *(x, z, y)*].

Теперь делается попытка доказать G_o из М2. Эта попытка оказывается успешной, так что решением является план

GOTHRU (Dl, Rl, R2); PUSHTIIRU(BOX1, Dl, R2, Rl). (4)

Дерево поиска для этой задачи представлено на рис. 3.

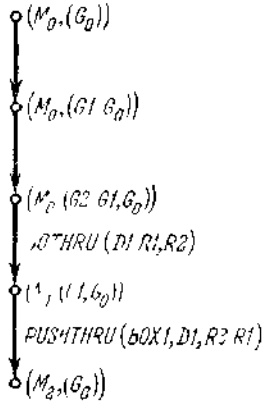


Рис. 3. Дерево поиска для примера задачи

В данном случае наше дерево выродилось в путь, поскольку в примере не возникло альтернативных возможностей (выбор оператора GOTHRU, а не PUSHTHRU, на втором шаге, объясняется точным совпадением различия со списком добавлений GOTHRU). Естественно, что в более сложных задачах и при большем списке операторов деревья поиска могут быть существенно «ветвистее».

Следует отметить два обстоятельства, не затронутые нашим простым примером.

Во-первых, порядок образования дуг дерева поиска, соответствующих применению операторов, вовсе не предопределяет вхождение операюров в этом порядке в окончательный план. Это лишний раз подчеркивает гибкость стратегии поиска механизма редукции GPS, сочетающего в себе свойства как прямого, так и обратного поиска.

Во-вторых, ПС РЗ снабжена эвристическим механизмом выбора узлов дерева поиска. В системе используется оценочная функция, учитывающая такие факторы, как число оставшихся целей в списке целей, число и тип предикатов в оставшихся выражениях цели, а также сложность различий, связанных с данным узлом

Как отмечалось ранее, главной трудностью использования механизма редукции GPS является выбор операторов, пригодных для устранения или уменьшения различий. В рассматриваемой системе этот вопрос решается следующим образом. Предположим, что в дереве поиска образован узел *(M,* (G_i, G_{i-1},…, G_0)), причем система доказательства теорем пытается доказать невыполнимость множества $M \bigcup (\sim G_i\}$. Если доказательство успешно, то к модели *M* применяется оператор с предусловиями G_i. Если же в течение определенного времени опровержение не будет найдено, то незавершенное доказательство или, в случае, если оно велико, его часть, выбираемая из эвристических соображений, и берется в качестве различия между *M* и G_i и связывается с данным узлом

Процесс выбора пригодного оператора происходит в два шага. На первом шаге создается упорядоченный список операторов-кандидатов. Выбор кандидатов основан на простом сравнении предикатов в различии с предикатами списков добавлений операторов. На втором шаге программа доказательства теорем определяет, могут ли высказывания из списка добавлений оператора резольвировать с высказываниями в различиях. Если новые резольвенты являются производными от высказываний в списке добавлений, то соответствующий оператор объявляется пригодным. Заметим, что из одного оператора-схемы может быть образовано несколько пригодных частных случаев.

Ранее мы отмечали трудности решения проблемы границ для декларативных представлений. Представление результатов применения операторов в виде списков добавлений и списков вычеркиваний решает часть этой проблемы в духе метода контекстов и контекстных графов. Однако, как отмечалось, проблема выведенных фактов этим методом не решается, так что в общем случае требуется вновь выводить их в каждом новом состоянии, т. е. в каждой новой модели. Эта проблема разрешается в ПС РЗ путем задания множества простейших предикатов, имеющих наинизший ранг в смысле, указанном в п.9.1.1. Остальные предикаты связываются с этим простейшим множеством. Другими словами, любой производный дизъюнкт связывается с теми из простейших предикатов, от которых зависит его истинность. Таким образом, предикат On(B2, B1) должен быть связан с предикатом At (B1, *A).* Тогда истинность всех производных дизъюнктов может быть определена непосредственно из списков вычеркивания соответствующих операторов.

9.3. Обобщение планов и планирование с помощью макрооператоров

9.3.1. Представление планов

Задача обобщения планов состоит в том, чтобы после построения успешного плана преобразовать этот конкретный план в план, который мог бы быть затем использован для множества подобных задач. Другими словами, мы хотим получить план-схему, т. е. **параметризованное семейство планов**. При этом необходимо, чтобы такой обобщенный план мог быть использован как в последующих процессах планирования в качестве дополнительного к имеющемуся списку операторов, так и при выполнении планов, составленных из таких обобщенных планов-операторов, с помощью ИС.

Прежде всего необходимо определить формализм представления планов, в понятиях которого можно было бы описать и решить поставленную задачу.

Таким формализмом является *треугольная таблица* (ТТ), строкам и столбцам которой соответствуют операторы плана. Пример ТТ представлен на рис. 4.

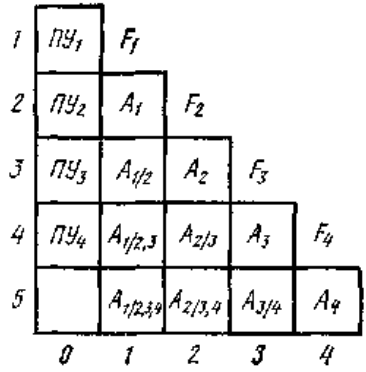


Рис. 4. Треугольная таблица.

Столбцы ТТ, кроме нулевого, помечены операторами F_1, F_2, F_3, F_4 составляющими план. Обозначим через $c_{i,j}$ ячейку ТТ, i— номер столбца, j — номер строки.

В общем случае для каждого столбца i, $i \neq 0$, в ячейку $c_{i,\ i+1}$ помещается *список добавлений* A_i *оператора* F_i. В ячейки $c_{i,\ i+k}$, k=2, 3, ... n+1—i

(n— число операторов в плане), помещаются те высказывания списка добавлений оператора F_i, которые остаются неизменными после применения операторов F_{i+1}, F_{i+2}, ..., F_{i+k-1}. Они обозначаются через $A_{i,\ i+1,i+2,\ldots,\ i+k-1}$. В нашем примере $A_{1/2}$ означает те высказывания A_1, которые остались неизменными после применения F_2, $A_{1/2,3}$ — те высказывания $A_{1/2}$, которые остались неизменными после применения F_3 и т. д.

Рассмотрим теперь строку j ТТ. Эта строка (исключая ячейку $c_{0,\ j}$) содержит список добавлений, получаемый применением последовательно F_1, F_2, . . ., F_{j-1}, или (j—1)*-й «шапкой» плана.* В частности, (n+1)-я строка содержит список добавлений, полученный после выполнения всего плана.

Назовем множество высказываний, используемых для доказательства предусловий оператора, *поддержкой предусловий.* Необходимо, чтобы строка j содержала все ппф в поддержке предусловий F_j.

Часть таких ппф будет добавлена (j—1)-й шапкой плана и поэтому будет включена в строку j. Остающиеся необходимые ппф находятся в начальной модели и не были вычеркнуты ни одним из F_k, k=1, 2, . . ., j—1. Эти высказывания, обозначенные нами $ПУ_j$, и помещаются в столбце i, i=0. Следовательно, нулевой столбец ТТ содержит те высказывания из начальной модели, которые были использованы в доказательствах предусловий для плана. Отметим, что $ПУ_j$, j=1,2,. . ., n, не содержат полное описание начальной модели.

Назовем высказывания, входящие в поддержку предусловий F_j, j=1, 2, . . ., n, *отмеченными высказываниями.* По построению ТТ все высказывания в ячейке $c_{0,\ j}$, j=1, 2,. . . . . ., n, являются отмеченными. Однако не обязательно все высказывания в $c_{0,\ j}$, $i\neq0$, являются отмеченными.

На рис. 5 приведена ТТ для плана решения задачи, рассмотренной в 9.2. Звездочка указывает отмеченные высказывания.

*INROOM(Robot,R1) *CONNECTS(D1,R1,R2)	GOTHRU(D1,R1,R2)	
*INROOM(BOX1,R2) *CONNECTS(D1,R1,R2) *CONNECTS(x,y,z)→CONNECTS(x,z,y)	*INROOM(Отправ,R2)	PUSHTHRU(BOX1,D1,R2,R1)
		INROOM(Отправ,R1) INROOM(BOX1,R1)

Рис. 5. Треугольная таблица для примера (9.2).

Таким образом, отмеченные высказывания в *j*-й строке составляют поддержку предусловий F_j. Представляет интерес исследование условий применимости для последовательности операторов $F_j, F_{j+1}..,F_n$, т. е. *j-го остатка плана.* Очевидно, что *j*-й остаток плана применим к модели, если она содержит ту часть поддержки предусловий F_k, $k=j, j+1, \ldots, n,$ которая не вырабатывается внутри самого остатка.

Назовем *j-м ядром ТТ* такую прямоугольную подтаблицу ТТ, что она содержит ячейку $c_{0,\ n+i}$ и строку *j*. Мы утверждаем, что *j*-й остаток плана применим к модели, если все отмеченные высказывания в *j*-м ядре истинны в этой модели.

Действительно, если все отмеченные высказывания в *j*-м ядре истинны, то F_j применим к модели. В результате применения F_j получится новая модель, в которой будут истинны высказывания A_j. Поскольку по построению таблицы и отмеченные высказывания внутри *j*-го ядра истинны, то все отмеченные высказывания в строке $j+1$ истинны; следовательно, F_{j+1} также применим. Продолжая аналогичные рассуждения относительно строк $j+2, \ldots, n,$ полччаем, что *j*-й остаток плана применим к модели.

Таким образом, доказано достаточное условие применимости *j*-го остатка плана.

Заметим, что первое ядро устанавливает достаточные условия применимости плана в целом, т. е. конъюнкция всех высказываний в нулевом столбце представлюет собой предусловия для всего плана. В свете доказанного достаточного условия выполнение плана может рассматриваться как последовательное преобразование ядер ТТ, т. е. для всех $j, F_j(K_j)=K_{j+1},$ где K_j — *j*-е ядро ТТ.

9.3.2. Обобщение планов

Опишем процедуру обобщения плана, представленного в виде ТТ. Эта процедура осуществляется в три шага.

На первом шаге мы преобразуем исходный конкретный план в наиболее общую форму. Этот шаг осуществляется следующим образом:

1) Каждое появление константы в первом ядре, в том числе одной и той же, замещается отдельным параметром.

2) Остальные столбцы озаглавливаются описаниями операторов-схем с параметрами.

3) В высказываниях в столбцах $i,$ i=l, 2, . . ., $n,$ исходного плана все константы заменяются соответствующими параметрами операторов-схем F_i.

В нашем примере ТТ (рис. 5) преобразуется в ТТ рис. 6).

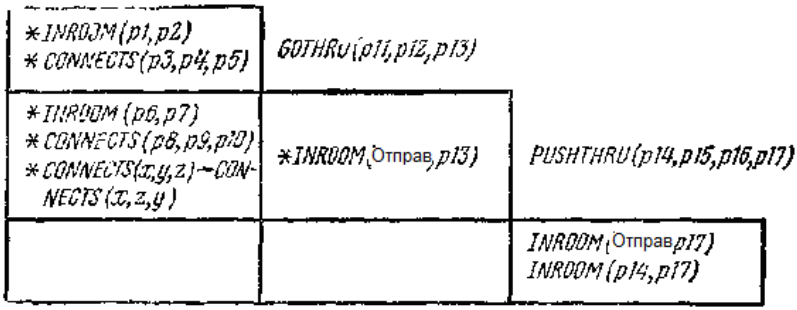


Рис. 6. Треугольная таблица для примера (9.2) после первого шага процедуры обобщения.

На втором шаге мы вводим такие ограничения на полученные параметры, чтобы, с одной стороны, отмеченные высказывания в строке j, j=1, 2, . . ., n, были поддержкой оператора F_j и, с другой стороны, чтобы исходный план оставался частным случаем получающейся ТТ. Этот шаг осуществляется следующим образом:

1) Извлекаются графы опровержения, построенные в процессе доказательства предусловий всех операторов исходного плана.

2) Для каждого такого графа, соответствующего оператору F_j, строится изоморфный образ выполнением на каждом шаге резолюций тех же высказываний и унификаций тех же литер, причем в качестве аксиом используются отмеченные высказывания из строки j, а в качестве доказываемого высказывания — предусловия оператора-схемы F_j из ТТ, полученной на первом шаге.

3) Все получаемые в процессе доказательства подстановки вносятся в полученную на первом шаге ТТ.

На рис 7 и 8 приведены деревья опровержения для предусловий операторов ТТ (рис. 6).

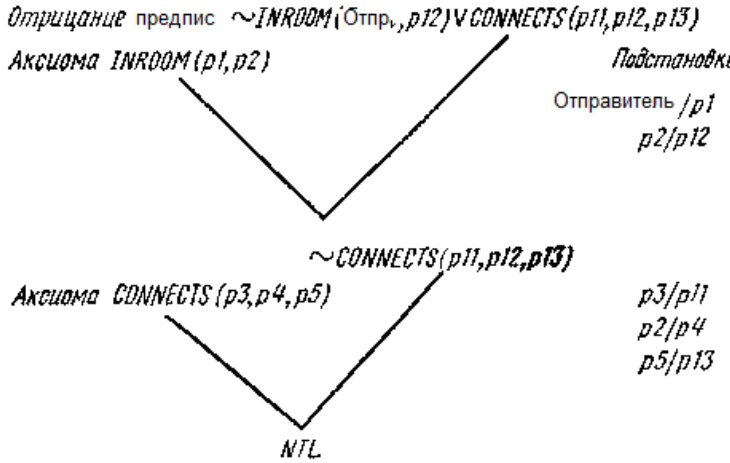


Рис. 7. Дерево опровержения для предусловий оператора GOTHRU (p11, p12, p13).

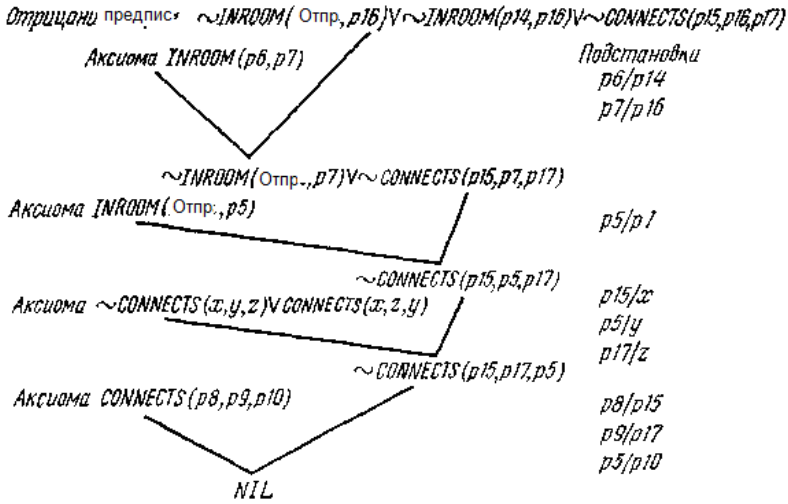


Рис. 8. Дерево опровержения для предусловий оператора PUSHTHRU (p14, p15, p16, p17).

После соответствующих подстановок ТТ (рис. 6) преобразуется в ТТ (рис. 9),

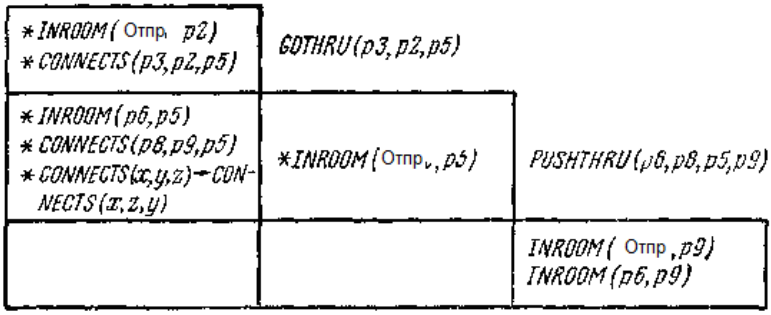


Рис. 9. Обобщенная треугольная таблица для примера (9.2).

На третьем шаге мы проводим два дополнительных преобразования, имеющие своей целью исключить излишние обобщения, а также один вид возможного противоречия.

Источником излишнего обобщения является случай, когда одно и то же высказывание в начальной модели плана входит в поддержку более чем одного оператора. Тогда на первом шаге процедуры обобщения это высказывание преобразуется в два различных высказывания. Во многих случаях это приводит к полезным обобщениям. Так, например, произошло в нашем примере (рис. 9), когда CONNECTS (D1, Rl, R2) преобразовалось в CONNECTS *(p3, p2, p5)* и CONNECTS *(p8, p9, p5),* что обобщило исходный план в том отношении, что теперь отправитель может передвинуть ящик в некоторую третью комнату, а не обязательно в ту, в которой первоначально находился сам.

Однако, если бы различные параметры (в нашем примере *p2* и p9), полученные из одной константы, не использовались бы оба как аргументы операторов в плане, их без ущерба можно было бы связать вместе с последующей подстановкой одного вместо другого в ТТ.

Источником возможных противоречий в обобщенной ТТ является сделанное нами на шаге 1, п. 3, молчаливое предположение о том, что вычеркивания в наиболее общей ТТ могут быть теми же самыми, что и в исходной ТТ.

Рассмотрим случай, когда в исходной ТТ имеются два одинаковых предиката с различными аргументами (например, At (BOX1, Rl) и At(BOX2, R2)). На первом шаге обобщения они превратятся в одинаковые предикаты с различными параметрами (например, *At(p1, p2)* и At(p3, p4)). Предположим, что в результате второго шага обобщения эти параметры продолжают оставаться различными, хотя, возможно, и отличаются от *p1, p2* и *p3, p*4 соответственно. Далее,

вполне возможно, что при использовании обобщенного плана и нахождении его частного случая часть соответствующих параметров в предикатах будет связана с одной и той же константой, в то время как другая часть соответствующих параметров предикатов свяжется с различными константами. Это и приводит к противоречию. В приведенном выше примере эта ситуация возникла бы, если бы *p1* и p3 были связаны с одним объектом, скажем, BOX7, а *p2* и *p4* — с различными местоположениями (R4 и R8 соответственно). В результате мог бы быть доказан план (или его часть), в котором BOX7 одновременно находился бы в R4 и R8. Ясно, что в конкретном плане предикат At(pl, *p2)* был бы вычеркнут, прежде чем была установлена истинность предиката At(p3, *p4).*

В общем случае мы должны провести корректировку обобщенной ТТ путем анализа списков вычеркивания всех операторов, входящих в план. Алгоритм коррекции обобщенной ТТ работает с полученной на втором шаге таблицей и анализирует последовательно списки вычеркивания операторов F_1, F_2, …., F_n. Рассмотрим совместно высказывания в строке j и список вычеркиваний F_j. Очевидно, что применение списка вычеркивания к строке будет менять ее тогда, когда некоторые из высказываний строки будут унифицироваться с некоторыми из литерсписка вычеркиваний и когда для унификации потребуется, чтобы некоторый параметр *p1* замещался другим параметром p2 или константой C. Если *p1=p2* или *p1=C,* то высказывание *R* вычеркивается, иначе оно остается неизменным и переходит в строку j+1. Это условное вычеркивание разрешается путем замещения высказывания в строке j+1 одной из импликаций:

а) $p1 \neq p2 \rightarrow R$,

б) $p1 \neq C \rightarrow R$. (7.5)

Предположим далее, что замещенное высказывание в (j+1)-й строке отмечено, т. е. входит в поддержку предусловий F_{j+1} (если оно не отмечено, алгоритм заканчивает работу). Тогда для того, чтобы поддержка предусловий F_{j+i} сохранилась после введения одной из импликаций (5), мы должны добавить в ячейку $c_{0,\ j+1}$ отмеченное высказывание вида

а) $p1 \neq p2$

или (6)

б) $p1 \neq C$,

так что прежнее в высказывание легко выводится из (5, *а)* и (6, *а)* или (5, *б)* и (6, *б).*

На этом третий шаг обобщения плана завершается, и полученный обобщенный план может быть занесен в список операторов системы. Мы будем называть такой план *макрооператором.*

9.3.3. Особенности планирования с макрооператорами

Рассмотренный в предыдущем параграфе процесс образования макрооператоров представляет собой типичный пример обучения ПС в ходе решения ею задач.

Этот процесс обеспечивает непрерывное обогащение возможностей системы при условии, что она будет способна

1) использовать макрооператор и любую его часть в процессе планирования;

2) свести к минимуму в каждом конкретном использовании избыточность макрооператора, являющуюся естественным следствием его обобщенного высказывания;

3) регулировать процесс накопления макрооператоров в системе.

Предположим, что в процессе планирования для уменьшения различия выбирается некоторый макрооператор, представляющий последовательность операторов $F_1, F_2,, ..., F_n$. Такой макрооператор содержит n различных *списков добавления* $A_{1,j}$, j=2, 3, . . ., n, соответствующих j-й шапке плана. Эти списки добавлений могут быть использованы обычным образом для выбора наиболее пригодного из них с позиции уменьшения различия. В случае выбора равнокачественных списков добавлений естественно выбирать список с наименьшим j, поскольку ему будет соответствовать более короткая последовательность.

Пусть выбран список добавлений $A_{1,j}$. Очевидно, что для целей планирования нас не интересует (j+1)-й остаток полного плана, так что из j-й шапки плана можно без ущерба удалить те операторы, которые предназначены для формирования высказываний, входящих в поддержку предусловий всех F_k, k=j+1, . . ., n. Кроме того, для целей планирования не нужны и те операторы, результатом которых являются высказывания, не используемые для установления пригодности списка добавлений $A_{1,j}$.

Эти соображения приводят нас к следующей формулировке алгоритма упрощения макрооператора:

1) Производится отметка тех высказываний из $A_{1,j}$, которые необходимы для установления пригодности $A_{1,j}$ (т. е. входят в неполный вывод, являющийся различием).

2) Снимаются метки у всех высказываний, являющихся поддержкой предусловий F_{j+1}.

3) Находится первый столбец справа, начиная с j-го, не содержащий отмеченных высказываний. Пусть таким столбцом будет i_k-й, $i_k \leq j$.

Тогда снимаются метки у всех высказываний в i_k-й строке, а оператор F_{i_k} исключается из j-й шапки плана.

4) Процесс, указанный в п. 3), повторяется вплоть до первою столбца. Оставшаяся в результате подпоследовательность операторов выдается как макрооператор, пригодный для уменьшения установленного различия.

Пример. Рассмотрим ТТ (рис. 10).

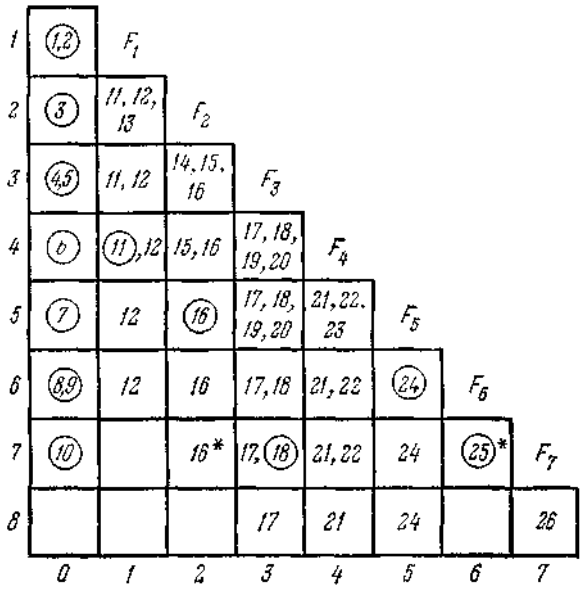


Рис. 10. Пример ТТ для иллюстрации алгоритма упрощения макрооператора.

Числа в ячейке представляют высказывания, отмеченные высказывания обведены кружками, высказывания, пригодные для уменьшения различий, отмечены звездочками. Структура ТТ может быть прослежена с помощью таблицы 3, где I — начальное, а G — конечное состояние плана.

Таблица 3

Оператор	Источник поддержки предусловий	Адресат поддержки предусловий	Оператор	Источник поддержки предусловий	Адресат поддержки предусловий
F_1 F_2 F_3 F_4	I I I I, F_1	F_4 F_5 F_7, G G	F_6 F_8 F_7	I, F_2 I, F_5 I, F_3, F_6	F_8, G F_7 G

В нашем примере j=6. Алгоритм работает следующим образом (высказывания 16 и 25 отмечены как уменьшающие различие).

1) Снимается метка у высказывания 18.
2) Столбец 4 — первый справа, не содержащий отмеченных высказываний. Снимается метка у высказывания 11, оператор F_4 исключается.
3) Столбец 3 — следующий столбец, не содержащий отмеченных высказываний (метка у высказывания 18 уже снята). Оператор F_3 исключается.
4) Столбец 1 — следующий, не содержащий отмеченных высказываний (метка у высказывания 11 уже снята). Исключается оператор F_1.

В результате получается последовательность операторов F_2, F_5, F_6, представляющая собой упрощенный макрооператор. На рис. 11 представлена ТТ для этого макрооператора, а таблица 4 показывает структуру ТТ (рис. 11).

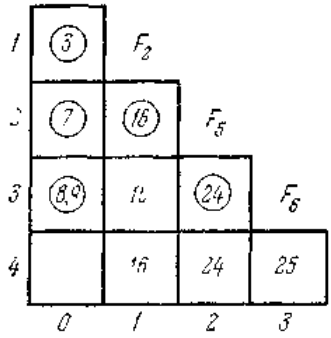


Рис. 11. Результат упрощения ТТ (рис. 10)

Таблица 4

Оператор	Источник поддержки предусловий	Адресат поддержки предусловий
F_2	I	F_5, G
F_5	I, F_2	F_6, G
F_6	I, F_5	G

Отметим, что мы не снимали метки у высказываний в нулевом столбце, так как соответствующие высказывания исключаются автоматически при исключении операторов.

Рассмотрим теперь, как может применяться упрощенный макрооператор для преобразования моделей. Главный вопрос состоит в выборе предусловий для макрооператора.

Если бы мы ограничились применением полного плана, выраженного макрооператором, то очевидно, что в качестве предусловий следовало бы выбрать конъюнкцию высказываний в первом ядре ТТ. Однако в случае, если весь макрооператор неприменим, мы хотим использовать частичную возможность применения макрооператора, поскольку она тоже может привести к полезным преобразованиям модели. Таким образом, мы приходим к необходимости установления предусловий для каждого остатка макрооператора. В п.9.3.1 мы установили достаточное условие применимости *j*-го остатка плана. Таким образом, мы должны установить истинность всех отмеченных высказываний в *j*-м ядре. Это может быть осуществлено с помощью процедуры сканирования ячеек ТТ.

Идея процедуры заключается в следующем. Чем меньше i и чем больше j, тем в большее число ядер одновременно входит ячейка $c_{i,j}$ ТТ. Таким образом, просматривая все ячейки $c_{i,j}$ в порядке возрастания i и убывания j и проверяя истинность всех отмеченных высказываний в этих ячейках доказательством их из текущей модели, мы будем последовательно находить те ядра в порядке убывания их номеров, все отмеченные высказывания в которых истинны (если такие ядра имеются). Как только мы находим некоторую ячейку $c_{i,j}$, в которой хотя бы одно высказывание не может быть доказано из текущей модели, мы исключаем из рассмотрения все ячейки c_k,j, $k \geq i$, так как при этом мы можем найти требуемое ядро с номером не более, чем i—1.

По мере доказательства высказываний в ячейках ТТ из текущей модели мы осуществляем соответствующие подстановки параметров макрооператора. Важно заметить, что сделанные для доказательства высказываний в одной ячейке подстановки должны немедленно распространиться на всю таблицу. Если имеются альтернативные возможности, то следует хранить дерево возможностей и, в случае необходимости, осуществлять возврат к точке ветвления.

С помощью описанной процедуры сканирования ТТ мы находим j-й остаток плана с наивысшим j, применимый к модели. В этом случае последовательность операторов $F_j, F_{j+1}, \ldots, F_n$ может быть выполнена, преобразуя текущую модель в некоторую новую модель. Если же неприменим весь план (т. е. ни один из j-х остатков плана неприменим, j=1,2,..., n), то конъюнкция отмеченных высказываний в первом ядре устанавливается как новая подцель.

Чтобы завершить изложение особенностей планирования с помощью макрооператоров, нам необходимо описать возможные подходы к решению задачи регулирования количества макрооператоров в системе. Хотя, казалось бы, богатый набор макрооператоров повышает возможности системы, наличие множества альтернативных решений задачи всегда будет в определенной степени снижать эффективность поиска плана. Конечно, трудно указать точно границы разумного компромисса между указанными факторами. Во всяком случае, есть два очевидных пути разумного сокращения количества макрооператоров в ПС.

Первый из них — исключение тех макрооператоров, которые являются частью более общих макрооператоров. Это исключение не только разумно, по и необходимо, поскольку ПС, работающая с макрооператорами, все время порождает подобные случаи.

Действительно, пусть решая задачу, ПС включает в план некоторую часть макрооператора или весь макрооператор. Получив полный план решения задачи, ПС обобщает его, порождая новый макрооператор, включающий в себя весь старый макрооиератор или его часть. Если подстановки параметров в старом и новом макрооператорах соответствуют друг другу, то очевидно, что каждый список добавлений старого макрооператора является подмножеством некоторого списка добавлений нового макрооператора. В этом случае старый макрооператор может быть исключен из системы. Аналогичным образом следует поступать и в случае использования в новом макрооператоре частей старого, исключая, однако, при этом те списки добавлений нового макрооператора, которые являются подмножествами списков добавлений уже существующего макрооператора.

В общем процедура исключения избыточных списков добавлений формулируется следующим образом: если любой частный случай *j*-й шапки одной ТТ является также частным случаем некоторой последовательности операторов в этой же или любой другой ТТ, то все списки добавлений *j*-й шапки (т. е. начиная со 2-й и кончая (*j*+1)-й строкой) ТТ исключаются.

П р и м е р. Рассмотрим две последовательности операторов:

A: $F_1(p1)$, $F_2(p1,p2)$, $F_3(p3)$, $F_4(p3,C1)$, $F_1(p3)$, $F_2(p4, p5)$.

B: $F_3(p6)$, $F_4(p6, \mathrm{Cl})$, $F_1(p7)$, $F_5(p6, p7)$.

1) $F_1(p1)$ и $F_2(p1, p2)$ в последовательности *A* могут быть исключены, так как в конце плана есть $F_1(p3)$ и $F_2(p4, p5)$.

2) $F_3(p6)$ и $F_4(p6, \mathrm{Cl})$ в последовательности *B* могут быть исключены, так как в последовательности *A* имеются $F_3(p3)$ и $F_4(p3, C1)$.

3) $F_1(p7)$ в последовательности *B* не может быть исключено последовательностью *A*, так как $F_3(p6)$, $F_4(p6, C1)$, $F_1(p7)$ и $F_3(p3)$, $F_4(p3, C1)$, $F_1(p3)$ могут иметь разные частные случаи.

Следует отметить, что определенная выше процедура исключения не учитывает случая одинаковых операторов, включенных в разные макрооператоры, но имеющих различные поддержки предусловий. Вообще говоря, ни один из таких операторов не должен исключаться.

Второй путь сокращения количества макрооператоров в ПС — введение механизмов «забывания». Подход к решению этой задачи заключается в наборе статистики использования макрооператоров и исключении тех из них, которые используются реже, чем заранее установленный порог частоты использования.

9.4. Обобщение пространств поиска решений и планирование в абстрактных пространствах

9.4.1. Принцип образования иерархии пространств

Обобщение планов путем автоматического создания макрооператоров является важным шагом на пути к созданию универсальных и эвристически эффективных решателей задач, обладающих свойством самоусовершенствования. Однако накопление набора методов решения задач и укрупнение самих методов составляют лишь часть общей проблемы обучения СИИ. Мы отмечали, что решающим шагом на пути к созданию мощных СИИ является глобальное исследование пространства поиска, имеющее своей целью «понимание» решателем задач общих закономерностей этого пространства. Обращаясь к

рассмотренной ранее задаче о миссионерах и людоедах, мы видим, что укрупнение операторов позволило нам лишь исключить некоторые промежуточные состояния. Наиболее существенный выигрыш был получен за счет введения в пространстве поиска решения некоторого комплексного образа, позволившего решить задачу на абстрактном уровне.

Главным недостатком метода обобщения планов (п. 9.3.2) является тот факт, что, укрупняя макрооператоры, мы оставляем другой элемент высказывания — модели — на том же, весьма детальном уровне, что и в случае обычных оператором. В понятиях формальной модели, последовательное обобщение макрооператоров приводит к настолько большим конъюнкциям предусловий, составляющих первое ядро ТТ, и настолько большим спискам добавлений, что эффективность решения достаточно сложных задач может оказаться весьма малой.

Поэтому для планирования решения сложных задач в сложных областях необходимо описывать области в некотором существенно обобщенном пространстве, которое, значительно упрощая описание области, игнорировало бы незначительные детали**. При этом полный процесс решения задачи можно было бы представить как пошаговый процесс в иерархии пространств от наиболее абстрактного к базовому пространству, в котором получалось бы окончательное решение. Существенной характеристикой такого процесса является поиск приблизительного наброска решения задачи в абстрактном пространстве и, при его достаточной перспективности, преобразование этого пространства в пространство более низкого уровня, уточнение наброска решения и т. д. Такой процесс должен значительно увеличивать эвристическую эффективность системы, в то же время допуская достаточно легкое преобразование высказываний от абстрактного к базовому.**

Рассмотрим, каким образом приведенные выше соображения могут быть преломлены в контексте описания моделей области с помощью ппф в исчислении предикатов первого порядка и описания операторов с помощью предусловий, списков вычеркивания и добавления, также выраженных в виде ппф. Прежде всего возникает вопрос, каким образом следует отличать пространства друг от друга. Очевидно, что было бы целесообразно опускать часть описаний моделей и операторов. Однако нам необходим критерий «детальности» выражений, входящих в эти описания. Таким критерием может быть ранжирование фактов об области по степени их неизменности (см. п. 9.1.1.). Эвристическим основанием использования такого ранжирования являются следующие соображения. С точки зрения

построения плана существование предметов и их свойства (наличие комнат и ящиков, расположение дверей) являются более важными фактами, чем положение предметов, которые могут передвигаться отправителем и тем более, чем положение самого отправителя (отправитель не умеет строить двери и ящики!). Поэтому именно эти важные факты должны использоваться в абстрактном пространстве для частичных описаний модели, и после построения наброска плана он может наполняться такими подробностями, как положение предметов и т. д.

Второй вопрос заключается в том, какие из элементов описания моделей и операторов ранжировать по их неизменности. По-видимому, нет смысла разбивать на классы высказывания, относящиеся к описанию модели: второстепенные детали в модели могут просто игнорироваться. Точно так же нет особого смысла ранжировать списки добавлений и вычеркиваний операторов: те из операторов, действие которых сводится к изменению деталей, не будут выбираться как пригодные в пространствах высокого уровня. Кроме того, изменения в списках вычеркивания и добавления приведут к тому, что в различных пространствах результаты действий операторов окажутся весьма различными, поэтому преобразования между пространствами могут оказаться сложными.

В то же время, ранжируя предусловия операторов, мы добиваемся весьма важного результата: **в пространствах высокого уровня мы будем всегда выбирать те операторы, которые производят наиболее существенные преобразования моделей области.**

Таким образом, мы выбираем предусловия оператора как средство различения уровня пространства поиска решения. Могут быть предложены различные методы присвоения весов литерам, входящим в предусловия операторов. Один из них заключается в следующем.

На все предикаты, описывающие мир ИИО, накладывается некоторое частичное упорядочение, которое определяет порядок, в котором будут исследоваться литеры предусловий всех операторов, используемых в этой области. Исследование литер происходит следующим образом:

1) Всем литерам, чье значение истинности не может быть изменено никаким оператором, присваивается максимальный вес.

2) Оставшиеся литеры исследуются по порядку. Если литера может быть достигнута из состояния, где все ранее исследованные литеры истинны, с помощью короткого плана, она считается деталью и ей присваивается вес, равный ее рангу в частичном упорядочении. Если такого плана нет, то ей присваивается вес, больший, чем наивысший ранг в частичном упорядочении.

9.4.2. Планирование в иерархии пространств

Рассмотрим возможную схему процесса планирования в образованной взвешиванием литер предусловий операторов иерархии пространств. Схема системы, приведенная на рис. 12, носит рекурсивный характер.

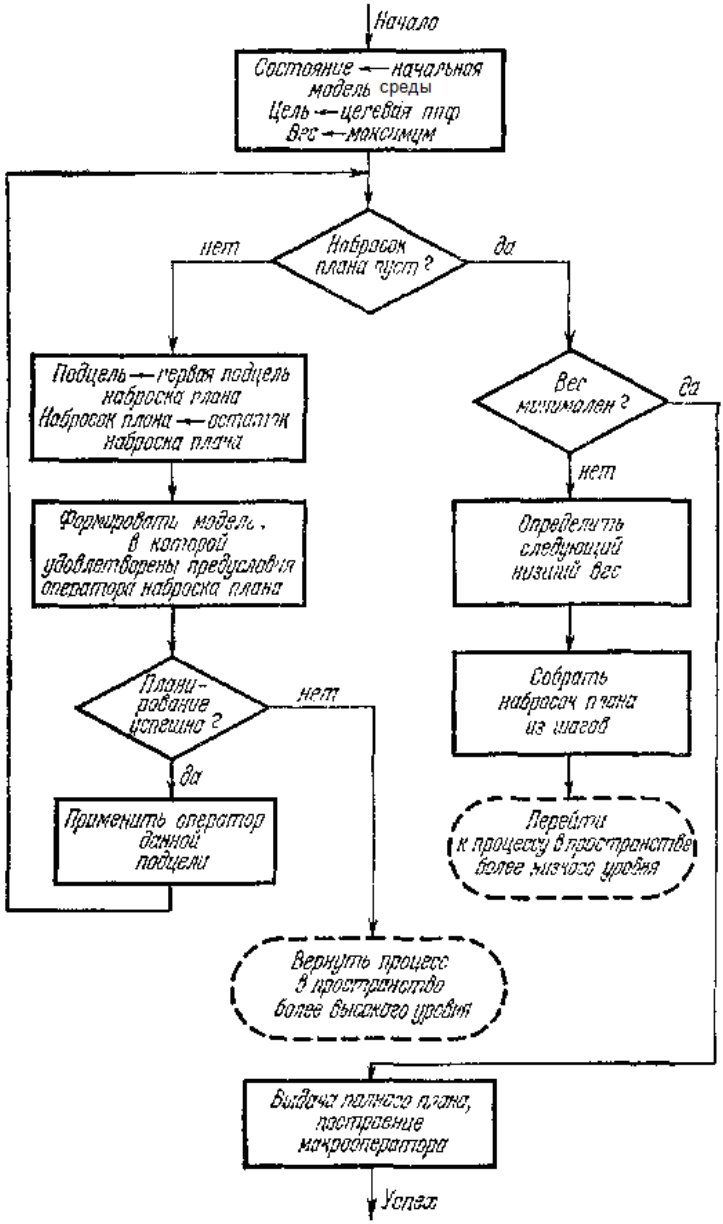


Рис. 12. Схема работы планирующей системы в иерархии пространств.

Сначала схема работает в абстрактном пространстве высшего уровня, строя набросок плана достижения главной цели. Когда этот план найден, система определяет вес следующего более низкого по уровню пространства, в котором требуется планирование, и строит **скелет вершин**, к которым применялись операторы на высшем уровне пространства. Затем система вызывается рекурсивно, решая подзадачи соединения шагов в наброске плана и обеспечения применимости этих шагов в соответствующих точках нового плана. Когда все подзадачи решены, система вновь вызывает себя рекурсивно для планирования в пространстве еще более низкого уровня. Этот процесс продолжается, пока не будет построен полный план в базовом пространстве, т. е. пространстве самого низкого уровня.

Описанный процесс осуществляет планирование в каждом абстрактном пространстве вплоть до главной цели, прежде чем осуществится переход к планированию на более низком уровне. Если какая-либо из подзадач в некотором пространстве не может быть решена, то управление возвращается в пространство более высокого уровня, вершина, послужившая причиной неудачи, исключается из рассмотрения и продолжается поиск успешного плана.

Заметим, что действия системы при неудаче аналогичны описанному ранее механизму возврата к точке ветвления со всеми его преимуществами и недостатками. В связи с этим требования к качеству поиска на высших уровнях иерархии пространств должны быть весьма высоки. Описываемая система должна управляться оценочными функциями, позволяющими строить допустимые стратегии в смысле приведенного ранее определения, возможно, в ущерб эффективности поиска, поскольку каждый лишний шаг в абстрактном пространстве может привести к большому количеству лишних шагов в пространстве более низкого уровня. В частности, на самом высоком уровне планирования используется оценочная функция при $\omega=1/2$ с постепенным увеличением веса эвристической функции по мере перехода в пространства более низкого уровня.

Проблема равнокачесгвенных решений в пространствах высокого уровня также становится весьма серьезной, поскольку произвольный выбор уменьшения различий может привести к неправильным решениям в пространствах более низкого уровня, где при выяснении деталей эти решения могут оказаться не эквивалентными, что приведет к необходимости возврата на более высокий уровень планирования. Поэтому при наличии равнокачественных альтернатив решение откладывается до планирования на низшем уровне путем частичной подстановки значений параметров оператора.

9.4.3. Пример. На рис. 13 изображена область отправителя в начальном и конечном состоянии.

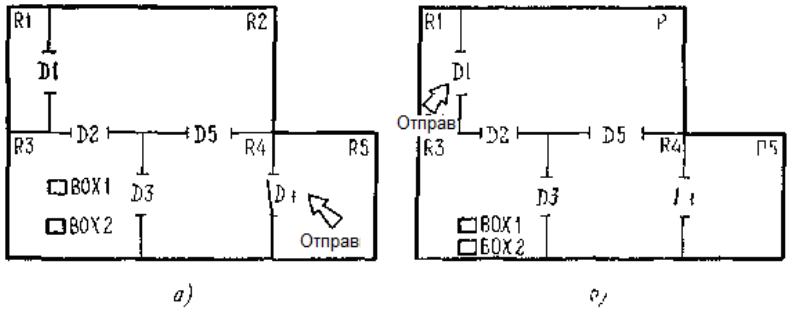


Рис. 13. Начальная (а) и конечная (б) область отправителя в примере (9.4.3).

Эта область достаточно сложна, однако работа системы может быть продемонстрирована только на примере повышенной сложности. В интересах краткости изложения мы не будем давать все формальные шаги построения плана, компенсируя эти пробелы словесным описанием. Мы хотим показать этим примером общий , характер планирования в иерархии пространств. Нам нужны следующие 5 операторов:

F_1: PUSHB *(bx, by)* — передвинуть *bx* к объекту *by.*

Предусловия:

{6} TYPE *(by*, OBJECT) *(by* типа «объект»),

{6} PUSHABLE *(bx) (bx* можно передвигать),

($\exists$ *rx*)[{5} INROOM (*bx*, *rx)* $\wedge$ {5} INROOM *(by*, *rx)* $\wedge$

{5} INROOM (Отправитель, *rx)},*

{1} NEXTTO (Отправитель, *bx)* (отправитель стоит вслед за *bx).*

Вычеркивания:

AT (Отправитель, $), NEXTTO (Отправитель, $), AT (*bx*,$),

NEXTTO *(bx*, $), NEXTTO ($, *bx).*

Добавления:

*NEXTTO(by, *bx),* *NEXTTO(*bx, by),*

NEXTTO (Отправитель, *bx).*

F_2: GOTHRUDR *(dx, rx)* — идти через *dx* в *rx.*

Предусловия:

{6} TYPE(*dx*, DOOR) *(dx* типа «дверь»),

{6} TYPE(*rx*, ROOM) *(rx* типа «комната»),

{2} STATUS (*dx*, OPEN) (состояние двери «открыта»),

($\exists$ *ry*) [{5} INROOM (Отправитель, *ry)* $\wedge$ {6} CONNECTS (*dx*, *ry*, *rx*)].
Вычеркивания:
AT (Отправитель, $), NEXTTO (Отправитель, $),
INROOM (Отправитель, $).
Добавления
* INROOM (Отправитель, *rx).*
F_3: GOTOD *(dx)* — идти к двери *dx.*
Предусловия:
{6} TYPE(*dx*, DOOR),
($\exists$ *rx*)($\exists$ *ry*) [{5} INROOM (Отправитель, *rx*) $\wedge$
{6} CONNECTS (*dx*, *rx*, *ry)}.*
Вычеркивания:
AT (Отправитель, $), NEXTTO (Отправитель, $).
Добавления:
*NEXTTO (Отправитель, *dx).*
F_4: GOTOB *(bx)* — идти к объекту *bx.*
Предусловия:
{6} TYPE(*bx*, OBJECT),
($\exists$ *rx*)[{5} INROOM (bx, *rx)* $\wedge$
{5} INROOM (Отправитель, *rx*)]
Вычеркивания:
AT (Отправитель, $), NEXTTO (Отправитель, $).
Добавления:
* NEXTTO (Отправитель, *bx).*
F_5: OPEN *(dx)* — открыть дверь *dx.*
Предусловия:
{6} TYPE (*dx*, DOOR),
{5} STATUS (*dx*, CLOSED) (состояние двери «закрыта»),
{5} NEXTTO (Отправитель, *dx).*
Вычеркивания:
STATUS (*dx*, CLOSED).
Добавления:
*STATUS(*dx*, OPEN).

Примечания. 1. Числа, стоящие в фигурных скобках, соответствуют весу литерпредусловий.

2. Звездочки, стоящие перед добавлениями, указывают первичные добавления, т. е. те, которые рассматриваются системой при поиске пригодного оператора.

Мы не приводим здесь полного описания начальной модели. Часть этого описания очевидна из рис. 13, *а.* Заметим, что все предикаты типа TYPE, STATUS, CONNECTS, PUSHABLE входят в начальную модель. Кроме того, предполагается наличие двух аксиом:

$(\forall\ x)$ (PUSHABLE (x) → TYPE (x, OBJECT)),
т. е. то, что можно передвинуть, является объектом, и
$(\forall\ x)$ (STATUS (x, CLOSED)↔ ~ STATUS (x, OPEN)),
т. е. если дверь закрыта, то она не открыта, и наоборот.
Мы решаем задачу: «пододвинуть ящик BOX1 к ящику BOX2, отправителю перейти в комнату R1». Целевая ппф имеет вид
NEXTTO (BOX1, BOX2) ∧ INROOM (Отправитель, Rl).
Поиск начинается в пространстве высшего уровня (веса 6). Ищется различие между целью и начальной моделью. Различием является сама целевая ппф. Пригодным для уменьшения первого члена различия является оператор F_1 при bx=BOX1, by=BOX2. Поскольку предусловия веса 6 истинны в начальной модели, оператор применяется, и в новой модели BOX1 придвинут к BOX2. Различие между целью и этой новой моделью—INROOM (Отправитель, Rl). Пригодным для его уменьшения является оператор F_2 при rx=Rl. Его предусловия в пространстве веса 6
TYPE (R1, ROOM) ∧ TYPE *(dx*, DOOR) ∧ *(*∃ *ry)*CONNECTS*(dx, ry*, Rl) удовлетворяются при dx=Dl. Применение F_2: GOTHRUDR (Dl, Rl) порождает модель, в которой удовлетворяется целевая ппф. Таким образом, мы получаем набросок плана в абстрактном пространстве высшего уровня в виде

PUSHB (BOX1, BOX2), GOTHRUDR (Dl, Rl). (7)

Производится переход к планированию в пространстве веса 5. Первая потдель — предисловия веса 5 первого оператора наброска плана, PUSHB (BOX1, BOX2). Различие между начальной моделью и предусловиями будет

INROOM (Отправитель, R3).

Пригодным для уменьшения этого различия является F_2 при rx=R3. Однако F_2 неприменим к начальной модели. Различиями являются INROOM (Отправитель, R2) или INROOM (Отправитель, R4). Для уменьшения первого различия уместен F_2 при rx=R2, однако он неприменим к начальной модели. Для уменьшения второго различия уместен F_2 при rx=R4 и он применим при dx =D4 Таким образом, применяется GOTHRUDR (D4, R4), формируя модель, к которой применим F_2 при rx=R3, dx=D3. Оператор GOTHRUDR (D3, R3) формирует модель, в которой удовлетворяют предусловия PUSHB (BOX1, BOX2) веса 5 Этот оператор применяется.
Тепечь устанавливается новая подцель — предусловия второго оператора (7). Сравнение текущей модели с его предусловиями вырабатывает различие INROOM (Отправитель, R2). Пригодным для уменьшения этого различия является оператор F_2 при rx=R2. Этот оператор применим при dx=D2, так что GOTHRUDR (D2, R2)

формирует новую модель, в которой подцель удовлетворена. Тогда применим GOTHRUDR (Dl, Rl), формируя модель, в которой вновь удовлетворяется целевая ппф. Мы получаем набросок плана в абстрактном пространстве веса 5:

GOTHRUDR (D4, R4); GOTHRUDR (D3; R3);
PUSHB (BOX1, BOX2); GOTHRUDR (D2, R2);
GOTHRUDR (Dl, Rl). (8)

Переходим к планированию в абстрактном пространстве веса 2.

Первая подцель — предусловия веса 2 первого оператора (8). Сравнение их с начальной моделью вырабатывает различие STATUS (D4, OPEN). Пригодным для уменьшения этого различия является оператор F_5 при *dx*=D4, однако он неприменим. Различие между его предусловиями и начальной моделью — NEXTTO (Отправитель, D4). Пригодным для уменьшения этого различия является F_3 при *dx=D4.* Этот оператор применим к начальной модели и преобразует ее в модель, в которой применим и F_5. Поскольку, как легко проверить, все остальные подзадачи непосредственно удовлетворены, мы приходим к модели, в которой удовлетворяется целевая ппф с помощью наброска плана в абстрактном пространстве веса 2:

GOTOD (D4); OPEN (D4); GOTHRUDR (D4, R4);
GOTHRUDR (D3, R3); PUSHB (BOX1, BOX2);
GOTHRUDR (D2, R2); GOTHRUDR (Dl, Rl). (9)

Переходим к планированию в базовом пространстве (веса 1). Первые четыре оператора (9) применяются по очереди, однако пятый оператор неприменим. Различием является NEXTTO (Отправитель, BOX1). Пригодным для уменьшения этого различия является F_4 при
bx =BOXl. Этот оператор применим, вырабатывая модель, в которой применим PUSHB (BOX1, BOX2). Окончательный план выглядит следующим образом:

GOTOD (D4); OPEN (D4); GOTHRUDR (D4, R4);
GOTHRUDR (D3, R3); GOTOB (BOXl);
PUSHB (BOX 1, BOX2);
GOTHRUDR (D2, R2); GOTHRUDR (Dl, Rl). (10)

Если внимательно проследить за ходом построения плана, легко заметить, что система автоматически строит сначала критические части плана, а потом переходит к построению плана для легко преодолеваемых областей базового пространства, действуя точно таким образом, как мы поступали на заключительном этапе анализа задачи о миссионерах и людоедах.

9.5. Стратегии выполнения действий

9.5.1. Исполнительные макрооператоры

Как указывалось в п. 9.1.2, ИС должна осуществлять выполнение планов и корректировку моделей области в соответствии с действительными результатами их выполнения. Очевидно, что стратегией, которая обеспечивала бы на каждом шаге наилучшее соответствие модели области самой области и плана его действительным результатам, была бы стратегия полного перепланирования после каждого шага выполнения. Однако мы отвергаем такую стратегию как крайне неэффективную. Настолько же неэффективной была бы и стратегия полного игнорирования результатов выполнения действия и их несоответствия «задуманному» плану.

Разумный компромисс между этими крайними альтернативами заключается в том, чтобы

1) при получении новой информации, обнаруживающей несоответствие *j*-го остатка плана, распознать ее и исключить из рассмотрения ИС этот остаток;

2) при неудаче *j*-й шапки плана с точки зрения ожидаемых результатов следует распознать неудачу и либо перевыполнить некоторую часть плана, либо осуществить перепланирование.

Такая стратегия обеспечит выполнение простых или составных планов (полных или неполных), т. е. явится основой для создания ИС класса А — Н, в зависимости от эффективности поиска планов с помощью ПС и сложности предъявляемых задач.

Рассмотрим воплощение указанной стратегии в системе, использующей макрооператоры. Мы предполагаем, что, построив план, система обобщает его и передает полученный макрооператор для выполнения.

Таким образом, первоначально мы должны осуществить частичные подстановки в макрооператоре, обеспечивающие доказательство целевой ппф. Важно отметить, что условия, при которых он может быть выполнен, остаются в обобщенном виде.

Алгоритм подготовки макрооператора для выполнения работает следующим образом:

1) В ячейку $c_{o,\ n+1}$ помещаются все высказывания из начальной модели, которые были использованы в процессе построения плана при доказательстве целевой ппф.

2) Высказывания из *(n*+1*)*-й строки используются для доказательства целевой ппф. Подстановки, произведенные в процессе этого доказательства, распространяются на весь макрооператор.
3) Высказывания в (*n*+1)-й строке, представляющие собой поддержку доказательства целевой ппф, отмечаются. Полученный макрооператор готов для управления выполнением действий.
Рассмотрим пример из п.9.2. Обобщенная ТТ для этого примера приведена на рис. 9. Осуществляя шаги алгоритма, мы
1) помещаем в $c_{0,3}$ BOX (BOX1);
2) используем BOX (BOXl), INROOM (Отправитель, *p9)* и INROOM (*p6*, *p9)* для доказательства
G_o: ($\exists$ *x*) [BOX *(x)* $\wedge$ INROOM *(x*, R1)];
3) осуществляем подстановки, произведенные во время доказательства, а именно *p6* =BOX1 и *p9*=*R1* во всей ТТ (рис. 9);
4) отмечаем высказывания BOX (BOX 1) и INROOM (BOX1, Rl).
Подготовленная для выполнения ТТ приведена на рис. 14.

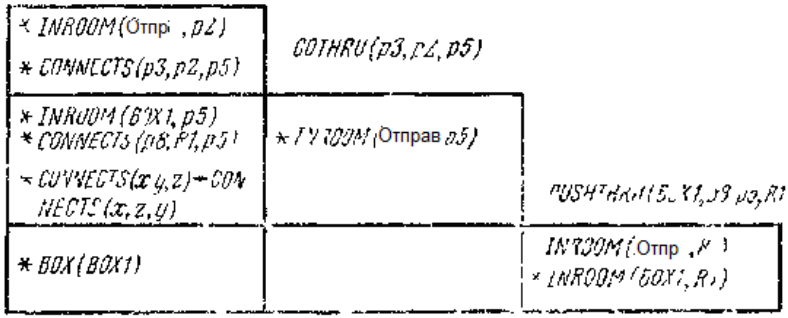


Рис. 14. Подготовленный для выполнения макрооператор (пример из п.9.2).

Полученный *макрооператор* мы будем называть *исполнительным.*
Управление выполнением плана с помощью исполнительного макрооператора основано на рассмотренном в п. 9.3.1 достаточном условии применимости *j*-го остатка плана. Таким образом, для продолжения выполнения действий на каждом шаге необходимо иметь хотя бы одно такое ядро, все высказывания в котором были бы истинны. Заметим, что в начале выполнения плана ИС исходит из начальной модели, так что первое ядро содержит только истинные высказывания. Однако в дальнейшем непредвиденные в плане обстоятельства могут либо неожиданно приблизить нас к цели, либо, наоборот, сделать план невыполнимым.

Для иллюстрации работы с исполнительным макрооператором мы предполагаем, что хотя бы часть плана может быть использована для выполнения (в противном случае должна начаться фаза полного перепланирования). Производится проверка ядер, начиная с конца, т. е. с ядра с наивысшим номером ((*n*+1)-е ядро — это (*n*+1)-я строка ТТ). Если все высказывания в *(n*+ 1)-м ядре истинны, план выполнен. В противном случае мы проверяем на истинность высказываний *n*-е, (*n*—1)-е и т. д. ядра. Пусть первое ядро, все высказывания которого истинны, имеет номер *j*. Тогда выполнены предусловия *j*-го остатка плана, F_j, F_{j+1} . . ., F_n. Мы применяем F_j и вновь ищем ядро с наивысшим номером, все высказывания в котором истинны. В случае полного соответствия ожидаемых результатов планирования действительным, указанная процедура просто выполняет все операторы плана последовательно. Однако она имеет возможность устранить ненужные операторы и ликвидировать неудачи повторным выполнением операторов, например, путем других подстановок параметров в соответствии с изменившейся по каким-либо причинам моделью.

Это последнее обстоятельство представляется особенно важным, поскольку позволяет ИС осуществить фактическое перепланирование без обращений к ПС (при наличии альтернативных возможностей).

Очевидно, что процедура аналогична механизму возврата к точке ветвления, причем точка ветвления находится автоматически в соответствии с текущей моделью области.

Остается заметить, что проверка ядер на истинность всех высказываний, входящих в них, осуществляется с помощью процедуры сканирования, описанной в п. 9.3.3.

9.5.2. Обобщенные процессы планирования и выполнения

Как отмечалось в п. 9.1.2, постановка задачи совместной оптимизации процессов планирования и выполнения действий связана с обобщением этих процессов в единый процесс. Рассмотрим две возможности такого обобщения.

Первая из них связана с рассмотрением ПС как некоторой программы действий, которая оказывает воздействие не на внешнюю область, как это делает ИС, а на абстрактную среду, состояниями которой являются планы. Если оказывается возможным оценить полезность построенного неполного плана, то в каждом состоянии в пространстве планов ИС с помощью некоторого метаоператора может выбрать наиболее выгодное направление планирования. Более того, если оценки для

вновь вырабатываемого плана хуже оценок для уже выработанного плана (полного и неполного), то ИС осуществляет действие, а если лучше, то продолжает планирование. В такой модели описания процессов планирования и выполнения ИС представляет собой по существу метапланирующее устройство, осуществляющее на абстрактном уровне рациональный выбор между планированием и действием и выдающее заказы на планирование ПС. В свою очередь ПС играет роль генератора абстрактных квазидействий, формирующего по запросу из ИС некоторые, возможно, гипотетические планы с оценками их полезности. В этой трактовке ИС изменяет состояния модели и отбрасывает неподходящие для этой модели планы, а ПС, не меняя модели, расширяет существующий для нее план или создает новый план. **Формально действие и планирование могут быть описаны как функциональные отношения в пространстве, состояния которого объединяют состояния модели и плана.**

Другая возможность обобщения процессов планирования и выполнения связана с идеей планирования в иерархии абстрактных пространств. Экстраполируя эту идею от базового пространства, в котором строятся планы, на уровни еще большей детализации, вплоть до непосредственного управления подсистемами ИИО, мы получаем полностью объединенного систему планирования и выполнения действий. Выполнение процесса в такой системе можно представить себе следующим образом

Базовое пространство ПС рассматривается как абстрактное пространство для ИС. Таким образом, уточняя детали, отсутствующие в этом базовом пространстве, общая система постепенно опускается до базового пространства ИС. Можно предполагать, что изменения в заданной области будут всегда в большей степени воздействовать на решения, принятые на низких уровнях иерархии пространств, чем на решения на более высоких уровнях. Тогда на каждом более высоком уровне план можно считать более близким к эффективному. Конечно, в такой трактовке взаимодействия планирования и выполнения действий требования к качеству решений на высоких уровнях иерархии существенно высоки. Перед началом погружения в пространства низкого уровня план должен быть максимально близок к эффективному. Однако важно отметить, что это может быть достигнуто не только за счет введения **эффективных эвристик**, но и за счет загрубления плана (путем исключения деталей). Стратегия выполнения основана на существенно неполном планировании. Выбирается небольшое число шагов плана как набросок плана действий, и он постепенно, по мере перехода к более низким уровням иерархии

пространств ИС, наполняется деталями становясь, возможно, при этом менее эффективным. На некотором уровне детали начинают вводиться только в начальную часть выработанного к этому моменту субплана и т. д. Наконец, вырабатывается короткий субплан в базовом пространстве ИС. Этот субплан выполняется. Расхождения между предполагаемыми на уровне субплана и действительными результатами наиболее вероятно являются деталями для построенного плана в базовом пространстве ПС и тем более для набросков плана на высших уровнях планирования. Тогда можно считать, что эти планы пока близки к эффективному.

Далее подобным образом строится новый субплан в базовом пространстве ИС, отражающий точно все появившиеся расхождения. В процессе построения нового субплана возможен возврат на тот уровень пространства, в котором это расхождение еще не является деталью.

В крайнем случае, когда расхождения оказываются достаточно серьезными, сигнал о неудаче распространяется точно до того уровня в иерархии пространств, в котором это расхождение является деталью.

Перепланирование начинается именно с этого уровня, оставляя наброски плана на высших уровнях неизменными.

Представляется, что использование подобных простых и коротких субпланов особенно эффективно в сложных областях, где степень неопределенности достаточно высока.

10. Планирования при неполном описании мира

10.1. Особенности планирования при неполном описании мира

Для описанных выше ПС в конечном счете нужна исчерпывающая детализация описания внешнего мира. Очевидно, что для большинства реальных задач, с которыми может столкнуться ИИО, детализация описания мира не может быть обеспечена. Кроме того, подход, связанный с точным запоминанием всех деталей окружающей обстановки и других, возможно, выведенных фактов, может оказаться неудовлетворительным. Во-первых, детальная информация может очень часто меняться. Во-вторых, в реальных мирах количество

информации может оказаться существенно большим, чем в состоянии запомнить система.

Таким образом, наряду с дальнейшим развитием ПС, работа которых основывается на представлении всех фактов об окружающем мире, представляется необходимым поиск других подходов, в принципе основанных на неполном знании. Одним из таких подходов является использование альтернативных планов, основанных на использовании составных и сложных (полных или неполных) планов. Рассмотрим возможность построения таких планов в рамках ПС класса описанных в п.9.2. Расширим стандартное описание оператора с целью преобразования его в сложный оператор, т. е. оператор с несколькими выходами, каждому из которых приписана некоторая вероятность. В этом расширении каждый выход оператора будет описываться своими списками добавлений и вычеркиваний, а также указанной выше вероятностью

Типичным примером таких операторов могут служшь операторы сбора информации о состоянии мира, в частности, следующий оператор, проверяющий состояние двери (открыта или закрыта дверь):

CHECKDOOR (DX) — проверить дверь *Dx*.

Предусловия:

TYPE *(Dx,* DOOR); NEXTTO (Robot, DA:).

Выход 1	Выход 2
Список вычеркиваний:	Список вычеркиваний:
ПУСТО.	ПУСТО.
Список добавлений:	Список добавлений:
DOORSTATUS (D*x*, OPEN).	DOORSTATUS (D*x*, CLOSED).
Вероятность: 0,5.	Вероятность: 0,5.

Вообще говоря, в операторах сбора информации предусловия должны содержать совет о том, когда следует применять оператор (например, в виде требования, чтобы собираемая информация не была известна перед применением оператора). Такое требование гарантировало бы, что полученная оператором информация будет добавляться к модели, и, таким образом, позволило бы отличать оператор сбора информации от операторов, производящих сходные действия (в нашем примере— оператор CHECKDOOR от операторов типа F, в п. 9. 4. 3) Заметим, что такого рода требования представляют собой новый класс предусловий, так как требуют получения неудачи как в доказательстве ппф (дверь закрыта), так и в доказательстве отрицания ппф (дверь открыта), выражая, таким образом, неопределенность ситуации. Для таких

предусловий следует определить свой синтаксис, а также добавить возможности получения подобных доказательств.

При использовании многовыходных операторов пространство поиска может быть представлено в виде пропозиционального графа, где вершины соответствуют состояниям мира, а каждая ветвь — выходу операторов, применяемых к вершине, причем выходы из различных операторов образуют дизъюнктивные ветви, а выходы одного оператора — конъюнктивные Тогда план может быть представлен в виде поддерева, состоящего из начальной вершины дерева и, для каждой вершины плана, тех дочерних вершин, которые получены в результате применения выходов одного оператора. Каждая конечная вершина плана соответствует модели мира, удовлетворяющей целевой ппф (если план успешен).

Этот план является условным планом с ветвлением, поскольку в каждой вершине производится проверка, каков действительный выход ИС, и в соответствии с ним выбирается та или иная ветвь плана. При этом вероятность появления каждой вершины равна произведению вероятностей выходов операторов, определяющих путь из начальной вершины дерева к данной вершине.

Возникает вопрос: каковы должны быть условия окончания для алгоритма построения условного ветвящегося плана? Двумя крайними и потому тривиальными вариантами являются достижение абсолютного успеха (все конечные вершины удовлетворяют целевой ппф) и достижение абсолютной неудачи (ни одна из конечных вершин не удовлетворяет целевой ппф). Однако нас интересуют промежуточные случаи. Один из таких случаев будет иметь место, когда дерево полностью построено, и некоторые из его конечных вершин (но не все) удовлетворяют целевой ппф. Следовательно, можно в принципе выбрать такой выход каждого оператора в плане, чтобы план был успешен. Поскольку дерево полностью построено, то произойдет окончание работы алгоритма, и в качестве результата разумно выбрать такой план, для которого сумма вероятностей появления вершин плана, удовлетворяющих целевой ппф, будет максимальной.

Рассмотренные варианты являются полными планами. Для неполного плана необходимо иметь условия окончания, отличные от полного построения графа или достижения абсолютного успеха.

Можно принять за условие окончания нахождение такого плана, вероятность успеха которого превышает некоторый порог. Здесь вероятность успеха неполного плана определяется как сумма вероятностей вершин, построенных к этому моменту. Разумность такого критерия аргументируется исключением лишнего планирования в ситуации, когда план, близкий к успешному, уже найден.

При построении неполного плана, как обычно, возникает задача взаимодействия ПС и ИС. Вновь мы хотели бы иметь стратегию, позволяющую осуществлять продолжение планирования или перепланирование после каждого шага выполнения действий; однако такая стратегия слишком непрактична. Другими возможностями взаимодействия являются:

— вызов ПС, когда конечная вершина плана найдена, а задача не решена;

— вызов ПС, когда найдена вершина, ни одна из дочерних вершин которой не удовлетворяет целевой ппф.

Следует также включить в условия окончания и такие ясные случаи, которые должны останавливать построение плана, вероятность неудачи которого превышает некоторый порог. Это условие может быть выражено двояким образом:

1) когда каждый план в графе имеет вероятность достижения терминальной вершины, не удовлетворяющей целевой ппф, большую, чем некоторый предел;

2) когда каждая нераскрытая вершина в графе имеет вероятность, меньшую, чем некоторый малый предел.

Аргументом в пользу такого условия окончания является предположение, что ни один из планов, вероятно, не приведет к успеху. При планировании целесообразно использовать стратегию выбора наиболее перспективного плана с последующим выбором в нем вершины для раскрытия. При этом в качестве оценок планов могут выступать вероятности успеха и неудачи для вершин плана и эвристические функции в принятом ранее смысле этого понятия. Оценки вершин могут основываться и на эвристических функциях, а также на вероятностях их появления.

Проиллюстрируем сказанное выше на примере (рис.1).

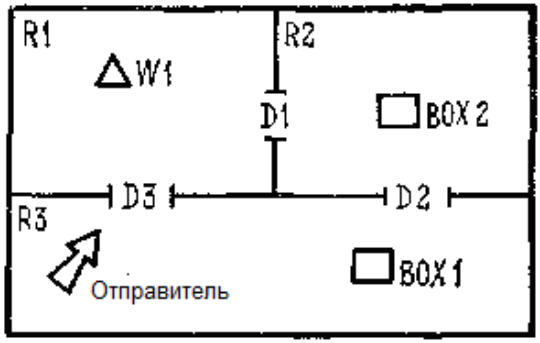


Рис. 1. Пример планирования при неполном описании мира.

Задача отправителя состоит в том, чтобы передвинуть ящик BOX 1 в комнату R1 так, чтобы призма W1 и любой ящик не были в одной комнате. Предположим, что отправитель не знает состояние дверей Dl, D2 и D3 (открыты они или закрыты) и не знает, есть ли призма W1 в комнате R1. План, использующий операторы GOTHRU и PUSHTHRU из примера в п.9.2 в сочетании с введенным выше оператором CHECKDOOR и новым оператором CHECKOBJECT (Wl, R1) (проверить, есть ли W1 в комнате R1), показан на рис.2.

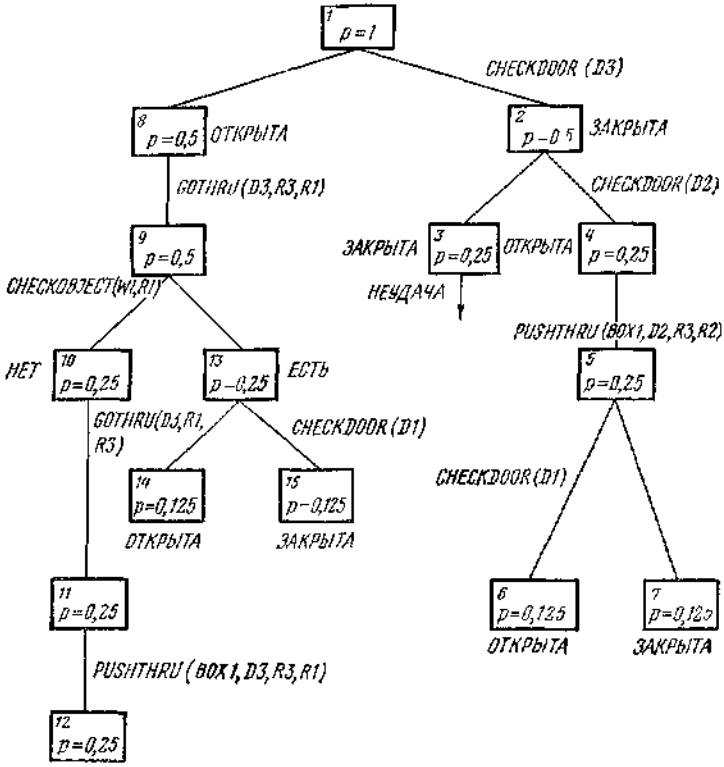


Рис.2. Пример плана со сложными операторами.

Здесь предполагается, что планирование обрывается, когда все нераскрытые вершины имеют вероятность, меньшую 0,2. Номера в квадратах указывают последовательность раскрытия вершин. Мы предоставляем читателю возможность самостоятельно

просмотреть этот план. Отметим только, что вероятность успеха для этого плана равна 0,25, вероятность неудачи — также 0,25, оставшиеся возможные выходы не раскрыты, так как вероятности соответствующих вершин лежат ниже установленного порога.

Дополнительные возможности планирования с многовыходными операторами раскрываются при планировании в иерархии абстрактных пространств. Использование иерархического представления может упростить построение условных планов, потому что неопределенность выходов операторов сохраняется лишь до определенного уровня детализации. В пространстве высшего уровня предусловия и действия операторов могут выражаться с помощью стандартного описания, т. е. без неопределенности. Однако некоторые из действий на этом уровне могут описываться в понятиях параметров, что усложнит преобразование планов в планы в пространствах низших уровней. Тем не менее простота представления сложных операторов делают эту схему перспективной.

Другой подход основан на постановке задачи планирования как игры с миром. Теоретической основой такого подхода может служить, например, формализм дерева лотереи [Сэндуолл]. В такой игре мир «отвечает» на принятые решения выбором одного из своих выходов, который в свою очередь предопределяет следующее решение. Отличие такой игры от обычных игровых алгоритмов заключается в том, что

— мир является безразличным, т. е. не делает специально худших для системы ходов (природа не злонамеренна!);

— безразличие мира предопределяет использование стратегий, основанных на максимизации полезности, а не на минимаксных или α—β-процедурах;

— в такой игре нет полного выигрыша или полного проигрыша: мера полезности растет или падает по ходу игры, т. е. выплаты производятся не в конце игры, а после каждого хода;

— на каждом шаге действия системы нет необходимости принимать новые решения, если старые решения оказываются успешными.

Новые решения необходимы для системы лишь в случаях, когда цель, поставленная предыдущим решением, достигнута или когда получена новая информация, снижающая полезность или исключающая возможность использования предыдущего решения.

Меры полезности в общем случае являются функциями времени в соответствии с общей целью и законами, действующими в мире Стратегия планирования сводится к нахождению лучшего первого хода. Заметим, что при наличии большого числа альтернатив на эту стратегию могут накладываться дополнительные отсекающие эвристики. Другими словами, кроме наилучшего решения среди всех

возможностей, можно выбрать первое решение, мера полезности которого превышает заранее установленный порог, или наилучшее решение среди *k* первых рассмотренных альтернатив и т. д.
Резюмируя сказанное выше, мы выделяем следующие основные элементы стратегии планирования при игре с миром:
1. Генерация множества ветвей на каждом шаге игры (возможно, с эвристическими отсечениями).
2. Правила присвоения значений полезности конечным вершинам ветвей плана.
3 Вычисление меры полезности плана на основе полез-постей конечных вершин.
4. Правило выбора лучшего первого хода как текущего продолжения плана.
Следует отметить, что игровой подход к планированию исследован чрезвычайно мало, особенно с экспериментальной точки зрения. Тем не менее можно априори указать на некоторые принципиальные трудности, связанные с его использованием.
Во-первых, неясен конструктивный путь получения статических и вероятностных оценочных функций, составляющих основу вычисления меры полезности. По-видимому, наиболее реальный способ их получения состоит в произвольном выборе оценок (конечно, с учетом специфики задачи и доступной эвристической информации) и дальнейшей корректировке их на основе статистического обучения. Однако эта методика может быть использована в случаях, где процесс обучения не может привести к серьезным последствиям (например, к гибели ИИО). Кроме того, при этом необходимо использовать стратегии «забывания», с тем чтобы решения основывались не только на усредненных оценках и система была способна открывать новые факты и законы.
Во-вторых, в динамическом реальном мире может оказаться невозможным получить в результате вероятностного обучения более или менее стабильные оценки. Могут потребоваться более сложные, индуктивные методы обучения.

10.2. Планирование в процедуральных представлениях

Рассмотренный в предыдущих параграфах метод построения планов, основанных на декларативном представлении моделей в виде ппф в исчислении предикатов первого порядка и механизме редукции GPS в

качестве стратегии поиска решения, обладает рядом достоинств. К числу их прежде всего относятся:

1) универсальность формализма, позволяющая решать задачи в значительном разнообразии миров;

2) возможность формального обобщения планов и использования макрооператоров как для исполнения построенного плана, так и построения других, более сложных планов;

3) возможность представления задачи в иерархии абстрактных пространств с последовательной детализацией первоначально построенных набросков плана.

В настоящем параграфе мы проиллюстрируем на примере некоторые особенности построения планов в процедуральном представлении, используя ПОЯ QA4.

10.2.1. Построение простых планов

Рассмотрим следующую задачу. Отправитель и ящик BOX 1 находятся в одной комнате. Требуется повернуть выключатель. Эта цель может быть достигнута, если отправитель пододвинет ящик к выключателю, встает на него, а затем повернет выключатель.

Для решения этой задачи нам потребуются следующие операторы (мы описываем операторы в декларативном представлении):

F_6: CLIMBONBOX *(bx)* — отправитель встает на ящик *bx.*

Предусловия:

TYPE (*bx*, BOX), ONFLOOR, NEXTTO (Отправитель, *bx)* (ONFLOOR — на полу).

Список вычеркиваний:

At (Отправитель, $), ONFLOOR.

Список добавлений:

ON (Отправитель, *bx*)

F_7: CLIMBOFFBOX (*bx*)— отправитель слезает с ящика *bx.*

Предусловия:

TYPE (*bx*, BOX), ON (Отправитель, *bx*).

Список вычеркиваний:

ON (Отправитель, *bx).*

Список добавлений:

ONFLOOR.

F_8: TURNONLIGHT(*m*)— отправитель поворачивает выключатель *m.*

Предусловия:

TYPE (*m*, LIGHTSWITCH), ON (Отправитель, *bx*), NEXTTO (*bx*, *m).*

Список вычеркиваний:

STATUS(*m*, OFF) *(m* выключен).

Список добавлений:
STATUS (*m, ON) (m* включен),
Операторы GOTOB *(bx)* и PUSHB(*(bx, m)* описываются аналогично операторам F_4 и F_1 соответственно (п. 7.4.3).
ПС STRIPS дает следующее решение указанной выше задачи:
GOTOB (BOXl); CLIMBONBOX (BOX1); CLIMB-
OFFBOX (BOXl);
PUSHB (BOX1, LIGHTSWITCH1); CLIMBONBOX
(BOX1);
TURNONLIGHT(LIGHTSWITCHl).

(1)

Заметим, что второй и третий операторы последовательности (1) реализуют взаимоисключающие действия. Эти действия были бы исключены при подготовке обобщенного плана в виде макрооперaюра к исполнению (п.9.5).
Рассмотрим теперь описание оператора TURNONLIGHT в ПОЯ QA4:

```
(LAMBDA (STATUS ← M ON)
(PROG (DECLARE N)
(EXISTS (TYPE $ M LIGHTSWITCH))
(EXISTS (TYPE ← N BOX))
(GOAL DO(NEXTTO $N $M))
(GOAL DO (ON (Отправитель $N)))
($ DELETE ('(STATUS $M OFF)))
(ASSERT (STATUS $M ON))
($BUILD('(:$ TURNONLIGHTACTION $M))))).
```

(2) ,

Общая структура довольно прозрачна, особенно для читателя, знакомого с языком LISP. Оператор выбирает выключатель и ящик и требует, чтобы ящик был поставлен рядом с выключателем, а отправитель встал на этот ящик. Затем осуществляются соответствующие вычеркивания и добавления, после чего сам оператор присоединяется к последовательности действий, выполняемых отправителем.
Оператор может быть применен для целей, сопоставляющихся образцу (STATUS ← *M* ON). Здесь ←*M* обозначает переменную, которая может быть сопоставлена любому выражению, после чего *M* будет связана с этим выражением. Если мы в соответствии с (1) хотим повернуть выключатель LIGHTSWITCH 1, то цель выражается как (STATUS LIGHTSWITCH I ON). Эта цель сопоставляется с образцом под знаком LAMBDA, и *M* связывается с LIGHTSWITCH 1. Таким образом, образец (STATUS ←*M* ON) играет роль списка добавлений оператора F_8.

Предусловия F_8,, выражены в форме выражений под знаком EXISTS. Поиск выражений, сопоставляющихся с представлениями EXISTS, производится в базе данных. Здесь $M означает, что переменная *M* может быть сопоставлена только с уже присвоенными *M* значениями. Таким образом, сопоставление с третьей строкой даст выражение (TYPE LIGHTSWITCH I LIGHTSWITCH). Этот факт, если он истинен, хранится в базе данных под знаком ASSERT. В общем, модель мира в процедуральном представлении состоит из всех выражений ASSERT. Заметим, что в случае, если выражение, сопоставляющееся с EXISTS, не будет найдено в базе данных, то это вызовет сигнал о неудаче, включающий механизм возврата.

Второе выражение под знаком EXISTS будет истинным, если есть хоть один ящик. Для случая BOX1 переменная N будет связана с BOX1. Эта строка оператора выбирает ящик для последующих действий.

Следующее предусловие оператора выражено в виде цели (GOAL DO(NEXTTO $*N* $M)). Сначала производится проверка, нет ли выражения формы (NEXTTO $*N* $*M)* в базе данных (действие аналогично EXISTS). В случае положительного ответа цель достигнута и производятся соответствующие связывания *N* и *M* (заметим, что для этого, в силу префикса $, в базе данных должно быть выражение (NEXTTO B0X1 LIGHTSWITCH1)). Если такого выражения в базе данных нет, то цель рассматривается как подцель. Управление ПС в процедуральном представлении осуществляется специальной управляющей программой, получающей цели и подцели и выбирающей в соответствии с некоторой семантической информацией и рекомендациями пригодные для достижения подцели операторы.

Оператор снабжен рекомендациями относительно того, какие операторы следует использовать для достижения подцели. В данном случае имеется два класса операторов: F_4 принадлежит к классу GO, а F_1, F_6, F_7, F_8 — к классу DO.

Поэтому для установленной подцели (GOAL DO (NEXTTO *$N* $*M*)) МОЖНО использовать один из четырех упомянутых операторов. Эти операторы пригодны, однако применимым из них будет тот, чья форма под знаком λ-выражения будет сопоставляться с выражением, стоящим под знаком GOAL. Приведем теперь запись оператора F_1 в языке QA4 (Мы используем слегка измененную формулировку оператора F_1, исключая из его предусловий TYPE (by, OBJECT) и включая предикат без параметров ONFLOOR.)*:*

(LAMBDA (NEXTTO←*M* ←*N)*
(PROG (DECLARE X)
(EXISTS (PUSHABLE $*M*))
(EXISTS (ONFLOOR))

```
(EXISTS (INROOM $M ←X))
(EXISTS (INROOM $N $X))
(EXISTS (INROOM Отправитель $X))
(GOAL GO (NEXTTO Отправитель $M))
(MAPC (QUOTE (TUPLE (AT (Отправитель ←X))
                    (AT ($M ←X))
                    (NEXTTO (Отправитель ←X))
                    (NEXTTO ($M ←X))
                    (NEXTTO ($X  $M))))
                        $DELETE)
(ASSERT (NEXTTO $M $N))
(ASSERT (NEXTTO $N $M))
ASSERT (NEXTTO Отправитель $M)))
($ BUILD (' (:$ PUSHBACTION (TUPLE $M $N))))).
```

(3)

Очевидно, что оператор (3) применим для достижения подцели (GOAL DO (NEXTTO *$N $M)),* т. е. переменная *M* в (3) связывается с BOX1, а *N* с LIGHTSWITCH1. Оператор (3) устанавливает другую подцель класса GO, так что делается попытка применить F_4. Эта попытка будет успешной.

Вернемся к оператору (2). Третье его предусловие выражается в виде (GOAL DO (ON (Отправитель $*N*))). Эта подцель может быть достигнута с помощью F_6. Мы не будем в интересах краткости изложения описывать F_4 и F_6 в виде процедур QA4, поскольку общий стиль конструирования этих процедур, вероятно, достаточно ясен.

Мы также не будем описывать синтаксические подробности описания процессов вычеркивания и добавления в операторах (2) и (3). Они осуществляются выражениями под знаком DELETE и ASSERT соответственно. Последняя строка операторов, как указано в самом начале, присоединяет оператор с соответствующими подстановками параметров в общую последовательность действий, осуществляемых отправителем.

Следует обратить внимание на использование в описаниях операторов дополнительной информации, получаемой от пользователя. Об использовании классификации операторов в рекомендациях мы уже упоминали. Заметим, что выражения типа GOAL в (2) установлены в таком порядке, что отправитель не будет пытаться встать на ящик до того, как придвинет его к выключателю. Это позволяет избежать таких взаимоисключающихся последовательностей операторов, как в (1). Конечно, и в декларативной формулировке задачи можно было тоже наложить упорядочение на использование предусловий.

В настоящее время работы по созданию законченных ПС в процедуральных представлениях находятся в незавершенном виде, повторяя в значительной степени задачи того же класса, что и STRIPS .

10.2.2. Построение условных и циклических планов

Предположим, что нам задано множество связанных предикатов *P1, P2, . . ., Pn,* значения которых неизвестны во время планирования, однако могут быть установлены во время выполнения плана. Предикаты связаны в том смысле, что по крайней мере один из них должен принимать значение ИСТИНА во время выполнения плана. Тогда общая форма условного плана может быть представлена в виде

IF PI THEN Al ELSE
IF P2 THEN A2 ELSE

……

IF *P* (*n*—1) THEN *A* (*n*—1) ELSE *An.*

Заметим, что *Pn* не проверяется на истинность, так как он должен проверяться только в том случае, если *P1, P2, . . ., P(n*—1) принимают значение ЛОЖЬ. Но в этом случае *Pn* должен быть истинным.

Один частный пример условного плана рассмотрен в п.9.6.2 для случая многовыходных операюров, выходы которых управляются соответствующими вероятностями.

Рассмотрим более общий случай условного плана, где ветвления обуславливаются значениями предикатов *P1, P2, . . ., Pn,* истинность которых устанавливается некоторым источником действий, возможно, независимым от отправителя. Тем самым мы делаем первый шаг на пути к построению планов в динамическом мире, одновременно демонстрируя некоторые характерные черты построения относительно сложных планов в процедуральном представлении. Поскольку при построении условных планов значения предикатов не определены, то следует составить такую программу, которая создавала бы последовательность локальных баз данных, или контекстов, в которых все большее количество предикатов получало бы определенные значения, а затем строить планы в этой последовательности контекстов. Эта программа, называемая в дальнейшем ALTPLAN (ALTPLAN (alternative plan) — альтернативный план), воспринимает в качестве аргументов предикат, или условие, и цель и в случае, если это условие истинно, указывает, что план не является условным. В случае, если условие не определено, ALTPLAN создает новый контекст, в котором условие ложно, решает задачу в этом контексте и выдает план решения в качестве результата. Прежде чем перейти к

описанию процесса планирования в ПОЯ QA-4, введем некоторые дополнительные определения, касающиеся этого языка.

Структуры данных. Мы будем использовать два типа структур — тупль (TUPLE) и множество (SET). TUPLE — это выражение формы (TUPLE *e*1. . .*en),* где e_1, . . ., e_n— произвольные выражения, являющееся аналогом списка. Значение тупля есть тупль, элементами которого являются значения элементов исходного тупля. SET — это выражение формы SET (*e*l. . .*en),* причем порядок элементов и число вхождений одного и того же элемента безразлично. Например, SET *(ABC),* SET(*BCA*) и SET*(CAACB)* идентичны, Значением множества является множество, элементами которого являются значения элементов исходного множества.

Встроенные функции. Нам потребуется встроенная функция «равно» (EQUAL). Аргументом EQUAL является множество. EQUAL принимает значение ИСТИНА (TRUE), если значения всех элементов аргумента идентичны. Например, EQUAL (*A* $*X*) принимает значение TRUE, если *X* имеет значение *A*.

Переменные. Переменные — это идентификаторы, снабженные префиксами (нам потребуются префиксы ←, $, ⇜ и $$). Значения первых двух префиксов объяснены в п.10.2.1. В отличие от переменных с префиксами ← или $, сопоставляющихся одному элементу, переменные с префиксом ⇜ и $$ являются сегментными, т. е. могут сопоставляться некоторому фрагменту множества или тупля. В остальном префиксы ⇜ и $$ работают так же, как префикс ← и $ соответственно.

Функция квази-QUOTE обозначается ' и имеет формат ('*e*). Эта функция подставляет значения переменных в *e*, не вычисляя последнего. Например, если $X \Rightarrow 2$ (*X* имеет значение 2), то

('(PLUS $X 5)) $\Rightarrow$ (PLUS 2 5), но не 7.

Операция DENY (отрицать) имеет формат DENY (синтаксическая форма) и присваивает значение FALSE (ЛОЖЬ) выражению с данной синтаксической компонентой, т. е. делает это выражение ложным в модели мира.

Оператор SETQ имеет формат SETQ *(pe),* где *p*—образец, и сопоставляет образец с результатом вычисления выражения *e,* связывая переменные в образце с подвыражениями, которым они были сопоставлены.

Условные выражения. Мы будем использовать условные выражения IF (если) и ATTEMPT(пробовать).

Выражение IF имеет общую форму (IF *e*1. . .*en* THEN *e*'l . . .*e'm* ELSE *e*"l. . .*e"k).* Выражения e_1, . . ., e_n вычисляются по порядку. Если значение последнего выражения e_n FALSE, то вычисляются по порядку

e''_1,. . .,e''_k и значение e''_k выдается как результат условного выражения IF. Если значение e_n отлично от FALSE, то вычисляются по порядку e'_1,....,e'_m и значением IF является значение e'_m. Например, значением выражения

IF (EQUAL $X 4) THEN (SETQ ← *Y3)* (PLUS $X *$Y)*
ELSE (SETQ←*Y* 6) (PLUS 4$ *Y*)

будет 7, если X=4, и 10 в противном случае. Выражение ATTEMPT имеет общую форму

(ATTEMPT *e1 . ..en* THEN *e'1. . . e'm* ELSE *e"1 . . . e"k).*

Выражения e_1, . . ., e_n вычисляются по очереди, и если одно из них генерирует сигнал о неудаче, то управление передается в ELSE (если ELSE отсутствует, выдается сигнал FALSE). Если ни одно из e_1, . . ., e_n не выдает сигнала о неудаче, управление передается в THEN. Дальнейшие вычисления подобны вычислениям выражения IF.

Выражение RETURN (возврат) используется для выхода из программы PROG, имеет форму (RETURN *e)* и выдает в качестве результата программы выражение *e* с подставленными значениями переменной. Например, если BOX имеет значение BOX1, то результатом программы по выходу RETURN (NEXTTO Отправитель $BOX) будет (NEXTTO Отправитель BOX1).

Выражение (CONTEXT PUSH CURRENT) [PUSH CURRENT — протолкнуть текущий] создает новый контекст, дочерний по отношению к текущему в дереве контекстов.

Связывание выражений с контекстом $CONTEXT осуществляется приписыванием к выражению конструкции WRT $CONTEXT (WRT (with respect to) — относительно).

Используя приведенные выше определения, построим программу ALTPLAN:

```
ALTPLAN(LAMBDA(TUPLE←CONDITION←GOAL)
          (PROG (DECLARE NC Y ALT Z)
          (ATTEMPT (EXISTS $CONDITION)
          THEN (RETURN (TUPLE)))
          (EXISTS (UNCERTAIN (SET  $CONDITION<<— Y)))
          (SETQ←NC (CONTEXT PUSH CURRENT))
          (DENY (UNCERTAIN (SET $CONDITION
                              $$Y)) WRT $NC)
          (DENY $CONDITION WRT $NC)
          (ATTEMPT (SETQ (SET←Z)$$Y)
          THEN (ASSERT $Z WRT $NC)
          ELSE (ASSERT (UNCERTAIN $Y) WRT $NC)
          (SETQ ←ALT (GOAL $GOALCLASS  $GOAL WRT $NC)
          (RETURN $ALT))).                                    (4)
```

Первое выражение ATTEMPT проверяет, существует ли входное условие CONDITION в модели, и выдает в качестве результата пустой тупль (TUPLE), если это условие существует. Это означает, что альтернативный план не нужен. В противном случае EXISTS извлекает из базы данных множество неопределенных (UNCERTAIN) условий, связывая Y со всеми такими условиями, кроме входного. Затем программа создает новый контекст NC, отрицая в нем множество неопределенных условий как утверждения в текущем контексте. Кроме того, мы предполагаем, что в новом контексте NC условие CONDITION ложно. Второе выражение ATTEMPT проверяет, осталось ли в множестве неопределенных условий только одно такое условие. В случае, если условие одно, оно утверждается относительно нового контекста. В противном случае утверждается неопределенность меньшего множества условий. Затем программа решает задачу в новом контексте и выдает план.

Рассмотрим работу приведенной программы на следующем примере. Пусть отправитель хочет развлечься (HAVEFUN) и имеет для этого две возможности: если идет дождь (RAINY), то отправитель идет в кино (MOVIE), если же светит солнце (SUNNY), он идет на пляж (BEACH). В результате мы хотим получить план вида

IF SUNNY THEN BEACH ELSE MOVIE. (5)

Запишем программы действий отправителя — BEACH и MOVIE.

```
BEACH (LAMBDA HAVEFUN
(PROG (DECLARE ALT) (SETQ ←ALT ($ALTPLAN SUNNY   HAVE-
FUN))
 (IF (EQUAL (TUPLE) $ALT)
     THEN(RETURN BEACH)
     ELSE
     (RETURN ('(IF SUNNY THEN BEACH
     ELSE  $ALT))))));
MOVIE (LAMBDA HAVEFUN
          (PROG (DECLARE ALT)
          (SETQI←ALT ($ALTPLAN RAINY    HAVEFUN))
          (IF (EQUAL (TUPLE) $ALT)
           THEN (RETURN MOVIE)
            ELSE
           (RETURN ('(IF RAINY  THEN MOVIE
            ELSE $ALT))))))).
```

Эти программы аналогичны с точностью до условий, поэтому мы опишем работу одной из них, например BEACH. Программа вызывает ALTPLAN, запоминая его в ALT. Если ALTPLAN выдаст пустой тупль (условие SUNNY истинно), то программа BEACH выдаст план

BEACH. В противном случае ALTPLAN решает задачу в предположении, что SUNNY ложно, и программа BEACH выдает план IF SUNNY THEN BEACH ELSE $ALT.

Приведем протокол выработки условного плана (5) с помощью программ ALTPLAN, BEACH и MOVIE. Итак, отправитель хочет развлечься, так что общая цель может быть представлена в виде

(GOAL $HAVEFUN HAVEFUN), (6)

где переменная HAVEFUN указывает, что цель относится к классу целей (GOALCLASS) HAVEFUN. В нашем случае к этому классу относятся две программы действий — BEACH и MOVIE.

Общий план построения условного плана сводится к следующему. Общая цель вызывает, например, BEACH, которая вызывает ALTPLAN. ALTPLAN упрощает неопределенное условие и вновь вызывает BEACH. BEACH опять вызывает ALTPLAN. На этот раз ALTPLAN терпит неудачу, так что вызывается MOVIE. Поскольку условие RAINY истинно, план MOVIE передается через ALTPLAN в BEACH, который вкладывает ее в общий условный план. Ниже приводится подробный протокол.

1. Ввод в базу данных (ASSERT (UNCERTAIN (SET RAINY SUNNY))).
2. Ввод в базу данных (GOAL $HAVEFUN HAVEFUN).

Примечание. Этот ввод осуществляется извне пользователем.

3. $HAVEFUN $\Rightarrow$ (TUPLE BEACH MOVIE) — процедуры класса HAVEFUN.
4. HAVEFUN отсутствует в базе данных, неудача.
5. Вызов LAMBDA BEACH.
6. Вызов LAMBDA ALTPLAN.
7. Связывание переменной CONDITION с SUNNY, GOAL с HAVEFUN.
8. Проверка (EXISTS SUNNY), неудача.
9. Проверка (EXISTS(UNCERTAIN (SET SUNNY $\Leftarrow$ Y))), связывание Y с RAINY.
10. Создание нового контекста NC.
11. Отрицание (UNCERTAIN (SET SUNNY RAINY)) относительно нового контекста NC.
12. Отрицание SUNNY относительно нового контекста NC.
13. Определение того, что осталось только одно условие RAINY, связывание Z с RAINY.
14. Утверждение RAINY в новом контексте.
15. (GOAL (TUPLE BEACH MOVIE) HAVEFUN) устанавливается относительно нового контекста.

16. HAVEFUN отсутствует в базе данных, неудача части EXISTS механизма GOAL.
17. Вызов LAMBDA BEACH.
18. Вызов LAMBDA ALTPLAN.
19. Связывание переменной CONDITION с SUNNY, GOAL с HAVEFUN.
20. Проверка (EXISTS SUNNY). В этот момент SUNNY ложно (см. п. 12), неудача.
21. Проверка (EXISTS (UNCERTAIN (SET SUNNY ⇇ Y))), это множество отсутствует в NC (см. п. 11), неудача, возврат к точке ветвления (п. 17)
22. Вызов LAMBDA MOVIE.
23. Вызов LAMBDA ALTPLAN.
24. Связывание переменной CONDITION с RAINY, GOAL с HAVEFUN.
25. Проверка (EXISTS RAINY), RAINY истинно в NC (п. 14).
26. ALTPLAN выдает (TUPLE), переменная ALT в MOVIE связывается с (TUPLE).
27. MOVIE выдает константу MOVIE, цель (п. 15) удовлетворена, переменная ALT в программе ALTPLAN принимает значение MOVIE, возврат к программе BEACH (п. 5).
28. Переменная ALT в BEACH принимает значение MOVIE.
29. BEACH выдает план (' (IF SUNNY THEN BEACH ELSE MOVIE)).

Перейдем теперь к рассмотрению процесса построения циклических планов. К сожалению, **задача автоматического синтеза циклических планов, включающая в себя определение того, когда и какие части планов следует представлять в виде циклов**, еще далека от своего решения. Мы покажем лишь, каким образом построить циклический план, если его цикличность задана явным образом.

Другими словами, если нам задан предикат *P* и действие *A,* содержащие одну и ту же свободную переменную *X,* то задача состоит в том, чтобы построить план, содержащий действие *A* для всех объектов в множестве, определяемом предикатом *P.*

Интуитивно ясно, что задача построения циклического плана может быть решена в три этапа.

Во-первых, необходимо иметь метод перечисления объектов, удовлетворяющих *P.* Во-вторых, необходимо выработать план, который в предположении об истинности *P* будет выполнять действие *A.* Наконец, следует ввести свободную переменную в *P* и *A,* которая к этому моменту является константой в QA-4, соответствующей связанной переменной как в *P,* так и в *A*, после чего объединить новые формы для *P* и *A* в общий циклический план.

Указанные этапы построения циклического плана реализуются в приводимой ниже программе LOOPPLAN (циклический план):

```
LOOPPLAN (LAMBDA (FA ←X (IF ←P THEN ←A))
          (PROG (DECLARE NC BODY NV1 NV2)
          (EXISTS (GENERABLE $P))
          (SETQ←NC (CONTEXT  PUSH  CURRENT))
          (ASSERT $P WRT $NC)
         (SETQ ←BODY (GOAL  $GOALCLASS
          $A WRT $NQ)
          (SETQ (TUPLE ←NV1 ←NV2) ($GENVAR (TUPLE)))
           (SETQ_←P(SUBST  $P  (TUPLE  $X $NV1)))
          (SETQ←BODY(SUBST $BODY (TUPLE $X $NV2)))
        (RETURN (REPEAT (GOAL:  $FIND $P) (DO $BODY))))).
```

(7)

Эта программа требует ряда пояснений. Начнем с пояснений синтаксического характера. Форма, стоящая под знаком LAMBDA, содержит идентификатор FA(FA (for all) — для всех), читающийся как «для всех». SUBST — обычная функция языка LISP.

Выражение REPEAT (REPEAT —повторить.) имеет форму (REPEAT nde DO *e*l. . .*en),* где *nde* — недетерминистическое выражение, $e_1, \ldots, e_n$ — выражения, и в нашем случае означает: «повторять выполнение программы $BODY для всех целей класса $FIND, сопоставляющихся образцу $P».

Программа действует следующим образом:

1) Устанавливает перечислимость объектов, удовлетворяющих *P* (GENERABLE $*P).*

2) Создает новый контекст, утверждает в нем *P* и устанавливает цель *A* в этом контексте, вырабатывая план, выполняющий действия *A* для объектов, определяемых *P.*

3) После выполнения плана подставляет произвольную переменную вместо свободной константы в *P* и *A,* т. е. создает план-схему.

4) Выдает в качестве результата итеративный план выполнения действия *A* для всех целей класса $FIND, т. е. осуществляющих поиск объектов, сопоставляющихся со значением *P.*

Рассмотрим пример. Предположим, что отправитель должен перенести все ящики, находящиеся в комнате ROOM1, в комнату ROOM2, т. е. установим цель класса $DO:

```
(GOAL $DO(FA BOX (IF (INROOM BOX RCOM1)
THEN(INROOM BOX RQOM2)))),
```

(8)

и мы хотим построить план достижения этой цели. Предположим, что к классу $DO относятся программа LOOPPLAN и программа, передвигающая ящики,— MOVEBOX. Произвольно предположим

также, что поиск ящиков (класс $FIND) осуществляется программами LOCATE (обнаружить) и CHECKMAP (проверить по карте). Построение циклического плана может быть представлено следующим протоколом:

1 Ввод в базу данных (GOAL $DO (FA BOX (IF (INROOM (TUPLE BOX ROOM1)) THEN (INROOM (TUPLE BOX ROOM2))))).

2. $DO⇒ (TUPLE LOOPPLAN MOVEBOX).

3. Цель не найдена в базе данных, неудача.

4. Вызов LAMBDA LOOPPLAN.

5. Связывание переменных: *X* с BOX (свободная константа), *P* с (INROOM(TUPLE BOX ROOM1)), *A* с (INROOM (TUPLE BOX ROOM2)).

6. Проверка (EXISTS (GENERABLE (INROOM (TUPLE BOX ROOM1)))), устанавливается генерируемость.

7. Создание нового контекста NC.

8. Утверждение (INROOM (TUPLE BOX ROOM1)) в контексте NC.

9. Устанавливается (GOAL (TUPLE LOOPPLAN MOVEBOX) (INROOM (TUPLE BOX ROOM2))) относительно контекста NC.

10. Вызов LAMBDA MOVEBOX (программа MOVEBOX не приводится, предполагается, что ее образец (INROOM (TUPLE ←BOX ←ROOM)) сопоставляется с целью, переменная BOX связывается с BOX, переменная ROOM связывается с ROOM2, и программа выдает план (PUSHTO BOX ROOM2)).

11. Переменная BODY принимает значение (PUSHTO BOX ROOM2).

12. Переменная *P* принимает значение (INROOM (TUPLE ←*X* ROOM1)).

13. Переменная BODY принимает значение (PUSHTO *$X* ROOM2).

14. LOOPPLAN выдает результат (REPEAT (GOAL (TUPLE LOCATE CHECKMAP) (INROOM (TUPLE ←*X* ROOM1))) DO (PUSHTO $X ROOM2)).

10.3. Многоцелевое планирование

10.3.1. Общая постановка

В большинстве реальных применений ИИО должен решать одновременно множество задач, а не одну изолированную задачу. Очевидно, что такое планирование становится многоцелевым. В этой постановке следует формулировать цель для ПС в виде множества целевых ппф (выражений GOAL в процедуральном представлении), каждой из которых должен быть присвоен свой вес, или мера полез-

ности. При этом мы должны допустить существование нежелательных целей, описывающих недопустимые состояния мира, и присваивать им отрицательный вес. Кроме того, следует задать функционал, определяющий общую целенаправленность действий, и обеспечить возможность построения планов, которые минимизируют этот функционал. Таким образом, на каждом шаге планирования необходимо оценивать текущее состояние работы по достижению каждой из целей и принимать решения, над какой из целей работать, на основании вычисления функционала.

Поставленная таким образом задача может потребовать весьма сложных вычислений в каждом состоянии мира. Кроме того, в общем случае, полезности различных целей должны быть функциями времени и общего состояния процесса планирования. Мы рассмотрим в настоящем параграфе две частные задачи многоцелевого планирования, не требующие сложных манипуляций, — планирование с ограничениями и кооперация ИИО.

10.3.2. Планирование с ограничениями

Рассмотрим задачу планирования с двумя целями — основной и нежелательной. Мы должны построить план достижения основной цели, избегая при этом любого состояния мира, удовлетворяющего нежелательной цели. К решению этой задачи можно подойти двояким образом: включить ограничения, представленные ппф нежелательной цели, в предусловия оператора или допустить выполнение операторов без ограничений, но распознавать недопустимые состояния в процессе планирования и направлять поиск по другому пути. Включение нежелательных целей в предусловия нарушает общность и гибкость использования операторов, не говоря уже о дополнительных сложностях, вводимых в механизм работы с предусловиями.

Поэтому мы пойдем по второму пути, принимая следующий план рассуждений. При выборе пригодного для уменьшения различия оператора производится проверка, не явится ли он сам причиной появления недопустимого состояния. Если предусловия оператора имплицируют ппф нежелательной цели, то нет смысла использовать этот оператор, поскольку, устанавливая его предусловия в качестве подцели, мы заранее обрекаем эту часть плана на неудачу. Если список добавлений сам содержит ппф нежелательной цели, оператор также должен быть отвергнут, так как он приведет пространство в недопустимое состояние.

Теперь остается случай, когда ппф нежелательной цели будет удовлетворяться конъюнкцией выражений, уже имеющихся в модели

мира и не вычеркиваемых оператором, и выражений из списка добавлений оператора. В этом случае недопустимое состояние распознается путем доказательства его из текущей модели мира (после применения оператора). Далее должны быть выделены высказывания, использованные в этом доказательстве. Отрицание одного из тех выражений, которые явились поддержкой ппф нежелательной цели, но не были добавлены оператором, вводится временно как дополнительное предусловие оператора. После этого подцель доказательства предусловий этого оператора пересматривается в свете добавлений. Может потребоваться пересмотр целой цепи предшествующих вершин плана (см. рис. 1 п. 9.2), содержащих предусловия оператора в списках целей. Кроме того, если поддержка ппф нежелательной цели содержит несколько кандидатов, удовлетворяющих указанным выше условиям, создаются отдельные вершины для каждого из отрицаемых директив, которые добавляются в предусловия оператора.

Проиллюстрируем сказанное выше на примере (рис.1). Пусть основная цель

$$\text{INROOM(R1, BOX1)} \vee \text{INROOM(R1, BOX2)}, \quad (9)$$

а нежелательная цель

$$(\forall x \forall y)\,[\sim (\text{INROOM } (rx, by) \vee \text{INROOM } (rx, \text{W1})], \quad (10)$$

т. е. ни в одной комнате не должны стоять одновременно ящик и призма W1.

Для построения плана мы используем операторы GOTHRU и PUSHTHRU, определенные в п.9.2. ПС типа STRIPS решает применить оператор PUSHTHRU (BOXI, D3, R3, Rl), однако это приводит к недопустимому состоянию. Поддержкой доказательства (10) являются директивы

$$\text{INROOM(Rl, W1)}, \quad (11)$$

$$\text{INROOM (Rl, BOX1)}. \quad (12)$$

Поскольку (12) получено в результате применения PUSHTHRU, то мы добавляем в предусловия конкретного случая применения оператора PUSHTHRU отрицание (11). Тем самым создается подцель извлечения призмы W1 из R1, например, в R2. Эта подцель может быть достигнута применением PUSHTHRU (Wl, Dl, Rl, R2), что приводит опять к недопустимому состоянию. Поэтому создается подцель извлечения BOX2 из R2 в R3. В результате будет построен план

GOTHRU (D2, R3, R2); PUSHTHRU (BOX2, D2, R2, R3);
GOTHRU(D3, R3, Rl); PUSHTHRU(W1, Dl, Rl, R2);
GOTHRU(D2,[1] R2, R3); PUSHTHRU (BOX 1, D3, R3, Rl).

(13)

Заметим, что в случае использования макрооператоров необходима проверка всех промежуточных состояний мира, вырабатываемых

компонентами макрооператора. В случае обнаружения недопустимого состояния следует либо запланировать его исключение перед использованием макрооператора либо пытаться обойти его имея по-прежнему в качестве цели результат макрооператора.

Некоторые проблемы возникают и при использовании обобщенного макрооператора для выполнения плана с ограничениями. Во время выполнения обобщенного плана мы должны быть уверены, что процесс подстановки не приведет к недопустимому состоянию. Более того, если даже частный случай плана выполняется в соответствии с планом, то всегда имеется опасность, что в процессе выполнения может появиться новая информация, которая не мешает достижению основной цели, но вызывает недопустимые состояния мира. Так, очевидно, что если отправитель (рис.1) не знает, что INROOM (Rl, Wl), и обнаруживает наличие призмы, перемещая BOX1 в R1, необходимо перепланирование. Следовательно, ИС должна распознавать недопустимые состояния и использовать эту информацию для обращения к ПС.

Особый случай возникает, когда ограничения являются постоянным свойством мира, и, следовательно, недопустимые состояния могут возникнуть при решении любой задачи в этом мире. Тогда может оказаться более эффективным построить список частных случаев операторов, применение которых при определенных условиях запрещено. ПС и ИС просто проверяют каждое применение оператора на допустимость, причем ИС исключала бы недопустимые операторы, обращаясь к ПС для перепланирования.

10.3.3. Кооперация ИИО

Назовем *кооперацией ИИО* совокупность нескольких эффекторых устройств, объединенных общей планирующей системой, избегая при этом термина «многоцелевой ИИО» (этот термин может относиться и к ИИО, решающим одновременно несколько задач одним ИИО). Следует отличать понятия кооперации ИИО от нескольких независимых ИИО, каждый из которых снабжен своим решателем задач и может решить свою задачу вне кооперации с другими ИИО.

В противоположность этому, наши ИИО, как мы предполагаем, решают кооперативно одну задачу. Представляется целесообразным выяснить возможность независимого выполнения действий по решению этой задачи каждым ИИО, т. е. каким образом можно распараллелить общий план на субпланы, выполняемые отдельными ИИО.

Если планы представляются ТТ (п. 9.3.1), то задача сводится к разбиению исходной ТТ на подтаблицы, относящиеся к различным ИО, так, чтобы ИС могла независимо и параллельно управлять их действиями.

Рассмотрим ТТ, представленную на рис.3.

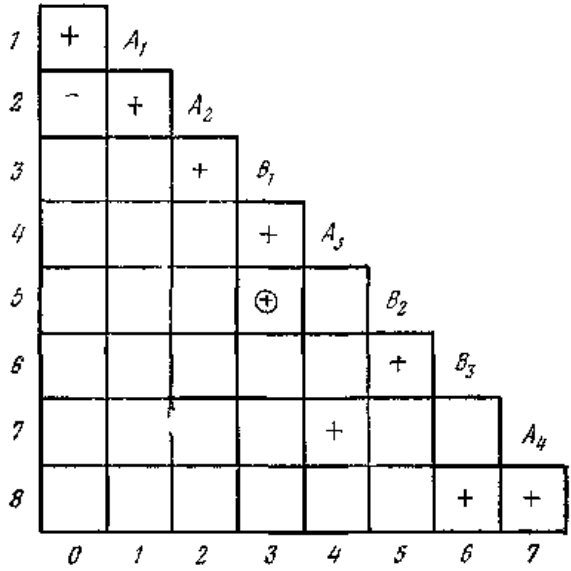


Рис.3. Исходная треугольная таблица.

Здесь A и B — различные ИИО, так что A_1, A_2, A_3, A_4— некоторые операторы, относящиеся к ИИО A, а B_1, B_2, B_3 — операторы, относящиеся к ИИО B. Знаком «+» обозначены отмеченные высказывания, $\oplus$ в $C_{3,5}$ просто отличает два отмеченных высказывания в 4-м столбце. Для каждого из ИИО все отмеченные высказывания могут быть разбиты на три класса:

1) Результаты действия B_i, которые являются предусловиями A_j или частью списка добавлений *TT* (класс 1).

2) Результаты действия B_j, которые являются предусловиями B_j (класс 2).

3) Предусловия B_i, которые получаются как результат действия A_j или из начальной модели ТТ (класс 3).

Высказывания класса 1 являются внешними выходами для подтаблицы ИО B и поэтому они должны быть помещены в ячейки $c_{i,n+1}$ подтаблицы, n — количество операторов субплана ИИО B.

Высказывания класса 2 являются внутренними для подтаблицы ИИО *B,* и они должны быть помещены в ячейки $c_{i,\,j}$ подтаблицы. Наконец, высказывания класса 3 являются внешними входами для подтаблицы ИИО *B,* и поэтому они помещаются в ячейки $c_{0,i}$, подтаблицы. Аналогичным образом может быть построена и подтаблица для ИИО *A*.

Обращаясь к нашему примеру (рис.3) и действуя, как описано выше, мы получим две подтаблицы для ИИО *A* (рис.4, *а)* и ИИО *B* (рис.4, *б).*

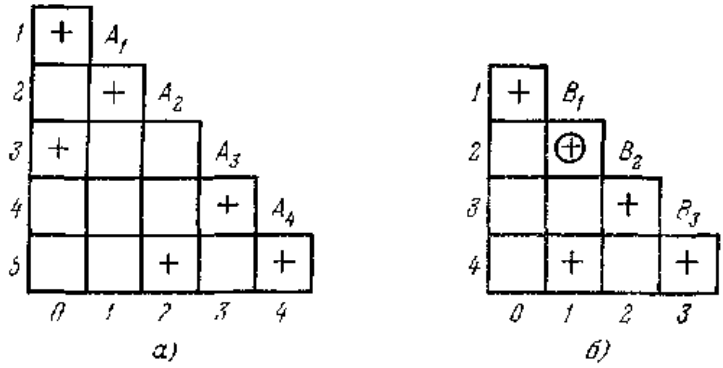


Рис.4. Подтаблицы для ИИО *A* и *B*.

Рассмотренный алгоритм может быть легко обобщен на любое число ИИО последовательным разбиением ТТ на подтаблицы A_1 и $A_2, A_3,\ldots, A_n$, затем подтаблицы $A_2, A_3, \ldots, A_n$ на A_2 и $A_3, A_4, \ldots, A_n$ и т. д.

10.4. Особенности представления планов в динамическом пространстве

В п.9.1.1 мы указали основные факторы, определяющие динамичность мира: наличие независимых от отправителя источников действия и явная зависимость элементов модели мира от времени. В связи с этим возникает вопрос: каковы адекватные средства представления такого мира в ПС для ИИО? Наиболее естественным решением этого вопроса является **описание мира в виде совокупности параллельно идущих, независимых, равноправных и взаимодействующих процессов, каждый из которых может некоторым образом влиять на мир, изменяя его.** Тогда ПС ИИО, работающего в динамическом мире, должна быть снабжена специальным процессором, в функции которого

входит управление процессами, в частности их инициацией, прерыванием и завершением, и обновление модели мирв в соответствующие моменты времени.

В общих чертах работа такого процессора может быть описана следующим образом. Взаимодействуя с моделью мира, процессор определяет условия, необходимые и достаточные для инициации тех или иных процессов. Если процесс должен быть инициирован, то процессор запускает его, определяет необходимые связки для переменных процесса, после чего следит за условиями его прерывания или естественного завершения, которые определяются другими процессами, находящимися под управлением процессора. В случае прерывания или естественного завершения процесса процессор производит изменения в модели мира, соответствующие достигнутым процессом результатам его действия. **Взаимодействие процессора с процессами и моделью мира осуществляется в специальном модельном времени аналогично тому, как это делается в языках моделирования.**

Результатом работы процессора является план взаимодействия процессов, необходимый для достижения заданной цели.

Описанная система может быть реализована как в декларативном, так и в процедуральном представлениях. В качестве процессора может выступать специальная мониторная программа. В настоящем параграфе мы рассмотрим особенности построения планирующей системы, работающей в динамическом мире и основанной на базовых представлениях системы STRIPS. Затем мы опишем процедуральный подход к построению децентрализованной системы взаимодействующих процессоров — **акторов**, основанный на двух основных принципах — неразделимости потока данных и управляющей информации и полной локализации всей обработки информации, включая управление взаимодействием акторов, в самих акторах.

Рассмотрим систему, состоящую из трех основных частей: **модели мира, множества описаний, или сценариев процессов, и упомянутой выше мониторной программы.** В соответствии с нашими представлениями о ПС, работающих в динамическом мире, мониторная программа включает в себя динамически изменяемое множество **блоков управления процессами (БУП).** Каждый БУП описывается обращением к определенному **сценарию процесса и множеством связок параметров процесса**. Заметим, что если одновременно протекает несколько идентичных процессов, то для каждого из них создается свой БУП, однако с обращением к одному и тому же сценарию. Таким образом, мониторная программа

децентрализует управление идущими процессами и, в то время как БУП производят соответствующие сценариям изменения в модели мира, другая часть мониторной программы определяет возможность запустить процессы, а также ожидает от БУП сигналов о прерывании или естественном завершении процесса и в этом случае ликвидирует соответствующий БУП. Эту часть мониторной программы назовем *супервизором.* Опишем структуру и дадим примеры сценариев процессов, а затем более подробно в понятиях сценариев процессов опишем ряд функций мониторной программы. При этом будем рассматривать два основных класса процессов.

Первый класс процессов — это процессы, происходящие мгновенно во времени (с точки зрения ПС). Такие процессы можно описывать с помощью стандартной формы операторов системы STRIPS, т. е. **наименования, предусловий и результатов в виде списков добавлений и списков вычеркиваний**. Имеется лишь два отличия. Во-первых, поскольку мы представляем динамический мир как совокупность равноправных процессов, то в сценариях, управляющих ИИО, необходимо включать в предусловия указание о процессе выбора ИИО данного сценария из множества альтернатив. Будем задавать это указание в виде (SELECT *rf),* где *r* —ИИО (в общем случае — один из ИИО, так чго *r* является параметром), *f* — сценарий процесса.

Указание предусловия SELECT делает всю совокупность предусловий необходимыми и достаточными условиями для начала процесса, в отличие от предусловий операторов STRIPS, где предусловие определяют только необходимые условия (ИИО может выбрать данный оператор, но не обязан этого делать). Таким образом, как только предусловия сценария процесса удовлетворены, процесс запускается.

Во-вторых, **допустимы два вида предусловий — предусловия, выраженные в исчислении предикатов, или символические предусловия,** которые мы использовали при рассмотрении системы STRIPS, **и предусловия, выраженные в виде отношений, заданных в области действительных переменных, или аналитические предусловия**. Аналитические предусловия необходимы для обеспечения взаимодействия в общем плане мгновенных процессов и процессов второго класса.

Второй класс процессов — это процессы, явно зависящие от времени (в том числе, непрерывные), т. е. процессы, которые воздействуют на модель мира постепенно. В сценариях таких процессов могут присутствовать результаты действий, выраженные как отношения и функции, заданные в области действительных переменных и принимающие значение также в области

действительных переменных (в отличие от списков вычеркивания и добавлений в обычных результатах действий). Для таких процессов мы будем различать **символические результаты, т. е. выражаемые в исчислении предикатов, и аналитические результаты, выраженные в виде указанных функций и отношений**. Кроме того, для процессов, моделирующих постепенные изменения мира, мы должны задавать условия их продолжения, нарушение которых приводит к их естественному завершению или прерыванию процесса мониторной программой. Эти условия также могут быть символическими и аналитическими.

Рассмотрим следующий пример: «ИИО должен повернуть кран и наполнить ведро водой». Введем сценарии процессов поворота крана (TURNVALVE) и наполнения ведра (FILLBUCKET).

Наименование сценария: TURNVALVE (*r, v, V^c)* —ИИО *r* поворачивает кран *v* так, чтобы установилась скорость потока V^c.

Предусловия:

символические: (SELECT *r* TURNVALVE (*r, v,* V^c));
(V_{max} *v* V^c_{max}); (AT *r n*); (AT *v n);*

аналитические: $0 \le V^c \le V^c_{max}$.

Результаты-М:

вычеркивания: *(V v* $);

добавления: *(V v* V^c*).*

Процесс, определяемый сценарием TURNVALVE, является мгновенным. В связи с этим его результаты, обозначенные буквой «М», представляют собой обычный список вычеркиваний и добавлений. Значение $, как и во всех определениях операторов, безразлично. Все параметры, связанные со скоростью потока *(V^c, V^c_{max}),* принимают численные значения; индекс наверху означает, что переменные связываются при запуске процесса и остаются в течение процесса неизменными.

Наименование сценария: FILLBUCKET *(b, v)* — ведро *b* наполняется из крана *v*.

Предусловия:

символические: *(V v* V^c*);* (AT *v n);* (AT *b n);*
(ORIENTATION *b* UP); *(C_b b C%)*
(C b C_0);

аналитические: $0 < V^c$;
$C_0 < C_b$

Результаты-П:

символические: (C *b* C^Y*);*

аналитические: $C^Y = C_0 + V^c \cdot \tau$.

Условия продолжения:

символические: *(V v V^c); (AT b n);*
(ORIENTATION *b* UP);
аналитические: $C^Y < C_b$.

Этот сценарий определяет постепенно изменяющий свои результаты процесс. В связи с этим его результаты снабжены буквой «П» и имеют символическую и аналитическую часть. Через С обозначено содержимое ведра, C_o— начальное содержимое, C_b— емкость ведра. Индекс «Y» у переменной *C* означает, что эта переменная изменяется самим процессом (в отличие от переменных с индексом С). Таким образом, символическая компонента результатов означает, что в результате процесса содержимое ведра станет C^Y, а аналитическая компонента — что C^Y определяется через начальное содержимое, скорость потока и длительность процесса τ. Предикат (ORIENTATION *b* UP) означает, что ведро должно быть поставлено отверстием вверх.

Условия продолжения указывают, что процесс может быть прерван, если изменится скорость потока, ведро не будет находиться под краном или будет перевернуто. Процесс придет к естественному завершению, если ведро наполнится $(C^Y \geq C_b)$.

Заметим, что, поскольку процесс наполнения независим от ИО, в его предусловиях отсутствует SELECT.

Рассмотрим теперь работу супервизора мониторной программы на примере наполнения ведра.

Каждый раз, когда супервизор обнаруживает, что предусловия процесса удовлетворены, он создает БУП и записывает в него все значения параметров, при которых удовлетворяются предусловия процесса, и время начала процесса τ_0. Далее, рассматривая символические компоненты предусловий и результатов-П, супервизор определяет отношения, которые будут изменяться процессом. В нашем случае таким отношением является (С ВКТ C_0), где ВКТ — конкретное ведро (значение параметра *b).* Все такие отношения удаляются из множества символических отношений в модели мира и добавляются к множеству аналитических отношений модели мира с заменой значения на переменную с индексом *Y.* В нашем примере будет удалено (С ВКТ C_o) и добавлено (С ВКТ C^Y). Каждое из таких новых аналитических отношений связывается указателем с БУП, который моделирует процесс, определяющий *Y*-переменные и отношения, в данном случае FILLBUCKET. Таким образом, процесс запущен. Теперь супервизор должен обеспечить наблюдение за условиями продолжения. Оно осуществляется различным образом для символической и аналитической компонент условий.

С каждым отношением символической компоненты условий продолжения связан список указателей к тем БУП, которые

моделируют процессы, продолжение которых зависит от этого отношения. Как только это отношение вычеркивается из модели мира, производится прерывание всех процессов, чьи БУП были связаны с этим отношением.

При создании БУП супервизор конструирует систему уравнений из аналитических компонент результатов — П и условий продолжения. Из этой системы могут быть определены условия прерывания процесса относительно одной из Y-переменных или времени. В нашем примере создается система

$$\begin{cases} C^Y = C_0 + V^c \cdot \tau, \\ C^Y < C_b, \end{cases}$$

из которой может быть определен критический момент естественного завершения процесса $t = \tau_0 + \tau$, где

$$\tau = \frac{C_b - C_0}{V^c}.$$

Время t записывается в БУП и может быть использовано супервизором.

Рассмотрим, как происходит прерывание при нарушении символического отношения $(V \vee V^c)$. Это условие может быть нарушено процессом TURNVALVE. Если скорость потока изменилась до нуля, то отношение $(V \vee V^c)$ вычеркивается из символической части модели мира. Супервизор проверяет список указателей этого отношения, находит указатель к БУП FILLBUCKET и прерывает этот процесс. При этом аналитическая компонента результатов должна быть преобразована в символические отношения в модели мира. Исходя из времени прерывания τ, вычисляется C^Y и вводится символическое отношение $(C$ ВКТ $C^Y_\tau)$, где C^Y_τ — значение C^Y в момент τ. Затем БУП FILLBUCKET ликвидируется.

Если скорость потока изменяется до некоторой новой положительной величины, то это изменение также прервет FILLBUCKET, но в данном случае он будет вновь запущен с другим параметром V^c и новым начальным содержимым C'_0, так как все его предусловия будут удовлетворены.

Нам осталось рассмотреть работу супервизора со временем. Как указывалось выше, с каждым БУП связано время прерывания, т. е. число, определяющее самый ранний момент времени, в который процесс должен быть прерван. Супервизор определяет самый ранний из таких моментов для всех действующих процессов, например T_0. Далее супервизор ищет среди всех сценариев процессов такой, который при некоторых значениях его параметров мог бы быть запущен раньше, чем в момент T_0. Если такого сценария нет, то супервизор

устанавливает модельное время в T_0 и выполняет прерывание. Если же таких сценариев несколько, то супервизор находит тот из них, который должен быть запущен раньше других, соотносит модельное время моменту запуска этого процесса, создает для него БУП и переходит к запуску следующего по времени процесса.

Описанный механизм осуществляет моделирование параллельных во времени, независимых и взаимодействующих процессов и действий. Очевидно, что **над этим механизмом следует надстроить планирующую систему, которая могла бы на основе поставленной цели и начальной модели мира определять последовательность совокупностей процессов, приводящую к цели оптимальным в некотором смысле образом.** Основной для построения такой ПС в значительной степени могут служить обычные ПС типа STRIPS. Однако наличие аналитических компонент предусловий, результатов и условий продолжения приводит к серьезным осложнениям. В частности, главной проблемой является необходимость решения систем уравнений, которые могут быть весьма сложными. Поскольку мы хотели бы, чтобы ПС была достаточно универсальной, она должна быть в этом случае снабжена решателем таких систем, обладающим весьма общими (а следовательно, и слабыми) методами, что существенно снизит эффективность ПС

Выходом из положения может быть **моделирование человекоподобного поведения.** Действительно, человек, осуществляя одновременно несколько действий в реальном мире, как правило не решает систем уравнений для точного определения моментов времени или переменных процессов, соответствующих завершению или прерыванию этих процессов. Хорошая хозяйка, готовя обед из нескольких блюд, может заниматься уборкой квартиры и делает это, изредка окидывая взглядом все поле своей деятельности, а отнюдь не производя аналитических выкладок.

Мы приходим к выводу о важнейшей роли взаимодействия процессов восприятия, приблизительной оценки и планирования действий в динамическом мире. Однако эта задача и связанная с ней задача планирования в динамическом мире еще далеки от своего решения.

В настоящее время развивается несколько другой подход к планированию и выполнению действий в динамическом мире. Главной особенностью его является децентрализованное взаимодействие специальным образом построенных процедур, или скелетов (фреймов). Этот подход представляет собой вполне естественное развитие процедуральных представлений в направлении локализации знаний и решения задач в процедурах.

Мы опишем основные идеи, связанные с построением одной из наиболее перспективных, на наш взгляд, **реализаций скелетов (фреймов) — системы взаимодействующих акторов**. Взаимодействие между акторами выражается в передаче сообщений, и это единственный вид внешнего поведения акторов.

Внутреннее поведение акторов определяется рядом механизмов:

— целью, которая проверяет удовлетворимость всех начальных условий актора, которому передано сообщение;

— рядом процедур, обеспечивающих связывание переменных и присвоение последним соответствующих значений;

— синхронизатором, определяющим, когда в действительности должен начать работать актор после передачи ему сообщения, а также предписывающим такой порядок действия акторов, при котором удовлетворяются их цели;

— монитором, воспринимающим все сообщения, переданные актору, и определяющим порядок их обработки;

— «банкиром», управляющим распределением ресурсов памяти и времени для актора.

Следует отметить, что каждый из указанных механизмов определен локально и работает только при вызове актора, что **обеспечивает максимальную степень параллелизма**. Кроме того, локализация, как уже отмечалось ранее, обеспечивает решение проблемы границ.

Рассмотрим схему передачи сообщения актору. Мы должны различать при этом актор, **посылающий** сообщение (**отправитель, источник**), и актор, **воспринимающий** его (**получатель, адресат**). В структуру сообщения входит собственно сообщение *M* и продолжение процесса *C,* которое должно быть активировано после выполнения действий получателем (отсутствие продолжения означает, что отправитель закончил свою работу). Итак, если отправитель *R* посылает получателю *T* сообщение *M* с продолжением *C,* то схема передачи сообщения может выглядеть следующим образом:

1) Вызов «банкира» *R,* санкционирующего расходы ресурсов отправителем.
2) Посылка сообщения «банкиром» *R* синхронизатору *T.*
3) Передача сообщения от синхронизатора Т мониторам *T.*
4) Посылка сообщения мониторами *T* к целям *T.*
5) Цели *T* посылают сообщение *M* продолжениям *T.*
6) Продолжение *T* осуществляет некоторую работу (в том числе передачу сообщения следующему получателю или, в случае успешного завершения работы, возврат к продолжению С).

Следует отметить ряд особенностей такой схемы.

Во-первых, передача сообщения актору-получателю сопровождается созданием нового актора, являющегося частным случаем получателя, управляемым сообщением. В частности, это предполагает повторное использование старого актора, если таковой окажется.

Во-вторых, передача сообщения включает в себя все необходимые управляющие функции, свойственные процедуральному сообщению, например, вызовы функций, повторения, вызовы параллельных процессов и т. д.

В-третьих, общение между акторами осуществляется децентрализованно, т. е. без каких-либо посредников.

В-четвертых, полная локализация позволяет, за счет отсутствия боковых эффектов, осуществлять одновременное общение актора с несколькими другими.

В-пятых, передача сообщений носит полностью ненаправленный характер, поскольку не предполагает заранее, что продолжение отправителя когда-либо получит сообщение (требуемый процесс может быть осуществлен другим способом или отвергнут как неудачный).

В заключение отметим ряд особенностей, связанных с накоплением и реорганизацией знаний в формализме акторов. Эти особенности в большой степени относятся к процедуральному представлению вообще.

Общей проблемой для системы ИИ является добавление информации так, чтобы уже накопленная информация могла остаться без изменений (конечно, если нет необходимости в изменениях). **В представлениях в логических исчислениях мы в большинстве случаев легко делаем это путем добавления аксиом. Процедуральные механизмы не позволяют делать этого так легко. Поэтому очень важно развивать гибкие механизмы накопления процедуральной информации.**

Формализм акторов пытается обеспечить следующие возможности:

1) Процедуральное вложение информации, т. е. средства, которыми информация может быть вложена в процедуры.

2) Консервативное развитие, т. е. структурирование информации в виде отдельных **боксов (сценариев) ориентированных знаний**. При этом имеется возможность создания новых боксов знания и связывания их с уже существующими боксами.

3) Модульная связность, т. е. возможность реорганизовать связи между боксами знаний. При этом модульная эквивалентность означает, что один бокс может быть заменен другим, если все связи между этим и остальными боксами знаний сохраняются теми же.

В настоящее время теория процедурального накопления и тем более обобщения знаний сформулирована недостаточно четко.

Можно выделить ряд исследуемых общих подходов в теории процедурального накопления и обобщения знаний. К ним относятся:
1) Процедуральная абстракция, сводящаяся к обобщению частных случаев (в стиле обобщения треугольных таблиц, п. 9.3.2). Грубо говоря, задача заключается в связывании с полученными частными случаями обобщенных образцов в определенных контекстах. Сюда же относится параметризация некоторых часто встречающихся процедур
2) Исключение абстрактно невозможных случаев Эта техника связана с процецуральной абстракцией и заключается в связывании альтернативе частными случаями и выявлением противоречий
3) Протокольная абстракция, заключающаяся в анализе и связывании протоколов выполнения процедур и преобразовании деревьев протоколов в графы путем исключения неразличимых с точки зрения образцов вершин.

11. Языковые формы общения интеллектуального икусственного объекта

11.1. Структура и задачи подсистемы языковых форм общения

В данном разделе описана структура и общие принципы построения подсистемы языковых форм общения интеллектуального икусственного объекта (ИИО).
Кажется естественным потребовать, чтобы ИИО в части языковых форм общения умела решать по крайней мере следующие задачи:
1. Понимать предложения естественного языка (возможно, ограниченного).
2. Выводить ответ на основании имеющихся у ИИО фактов.
3. Выражать ответ в ограниченном естественном языке.
4. Накапливать и корректировать свои знания на основании информации, воспринимаемой в процессе диалога.
На рис. 1 приведена структура подсистемы языковых форм общения интеллектуального икусственного объекта, удовлетворяющая перечисленным выше требованиям.

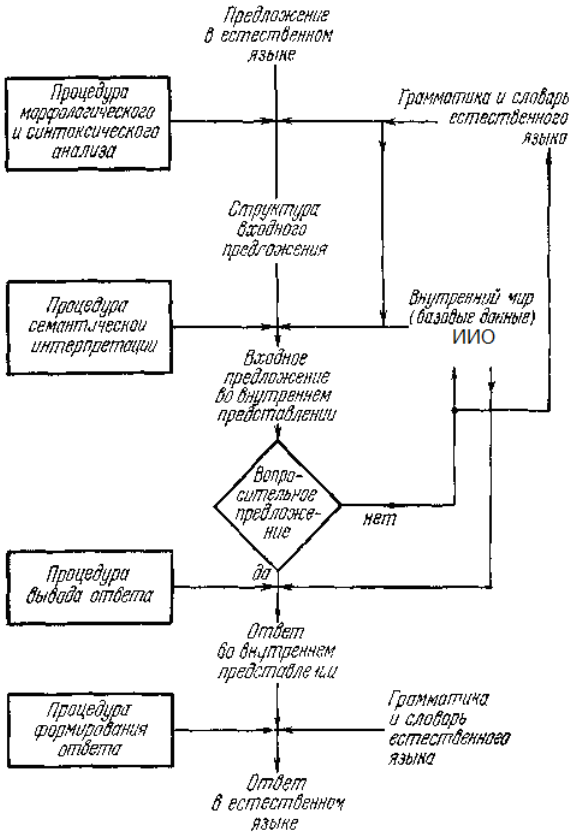


Рис. 1. Структура подсистемы языковых форм общения интеллектуального икусственного объекта

Подсистема, приведенная на рис. 1, совпадает по структуре и решаемым задачам с вопросно-ответными системами (ВОС) общего типа.

В дальнейшем в основном будет использоваться термин ВОС вместо термина «подсистема языковых форм общения интеллектуального икусственного объекта» Этим мы хотим подчеркнуть, что изложенный в разделе подход может быть использован не только при построении подсистем ИИ, но и при разработке изолированных информационно-поисковых или вопросно-ответных систем, воспринимающих ограниченный естественный язык.

Рассмотрим структуру подсистемы языковых форм общения ИИО (рис. 1), подразделяемую в соответствии с традициями ВОС на следующие основные этапы.

1. Синтаксический анализ. Синтаксис понимается в широком смысле с включением морфологии. Задача этапа — определить структуру входного предложения в соответствии с грамматикой языка и идентифицировать слова предложения со словарем системы.

2. Семантическая интерпретация. Задача этапа— понять входное предложение, т. е. однозначно выразить его в понятия внутреннего мира ИИО. Если входное предложение является *утверждением,* то требуется объединить его с базовыми данными (внутренним миром). Если входное предложение является *вопросом,* то требуется выделить из базовых данных информацию, необходимую для ответа на поставленный вопрос.

3. Вывод ответа. Задача этапа — вывести из выделенного подмножества базовых данных ответ на заданный вопрос. Ответ выражается в понятиях внутреннего представления.

4. Формирование ответа. Задача этапа — перевести ответ из внутреннего представления в естественный язык.

В данном разделе (теме) нашей основной задачей является выделение семантической информации входного предложения и установление взаимосвязи ее со знаниями ИИО. Решение указанной задачи в значительной степени не зависит от морфологии и частично от синтаксиса языка. Поэтому мы не будем рассматривать вопросы морфологии и будем касаться синтаксиса только в его связи с семантикой.

Системы, воспринимающие естественный язык, основываются на том положении, что для понимания языка необходимо не последовательное применение этапов обработки входного предложения (**синтаксис, семантика, вывод**), а **интегральное использование всех аспектов языка**. Так, например, для понимания синтаксиса может потребоваться обращение к семантике, к дедукции или к общим знаниям ИИО. Тем не менее мы, в целях методичности изложения, будем описывать этапы работы ВОС последовательно.

На наш взгляд, еще не существует полной общей теории вопросно-ответных систем, воспринимающих естественный язык. Выполнение ряда этапов ВОС (семантическая интерпретация, дедуктивный вывод) в значительной степени зависит от многих факторов и особенно от представления базовых данных в виде исчисления предикатов (системы с общим выводом, см. п.9.3.4) или в виде семантических сетей и процедурального представления (системы с ограниченной логикой, см. п.9.3.3). Указанное обстоятельство нашло отражение и в

изложении материала в даннои разделе. Мы будем описывать этапы семантической интерпретации и дедуктивного вывода отдельно для систем с **общим выводом (исчисление предикатов) и систем с ограниченной логикой (семантические сети).**

Прежде чем перейти к описанию вопросно-ответных систем, отметим, что обработка естественного языка является основной задачей не только ВОС, но и области научных исследований, называемой «машинным переводом» (МП). Несмотря на общность ряда этапов ВОС и МП, мы не будем проводить параллели между используемыми в них принципами. По нашему мнению, различие целей ВОС и МП вызывает (по крайней мере в настоящее время) различие в используемых методах.

Перечислим основные различия ВОС и МП.

1. Основной задачей ВОС по обработке языка является **преобразование текста в его смысл (при обработке входного предложения) и преобразование смысла в текст (при формировании ответа).**

При МП полного проникновения в смысл фразы может не требоваться. Действительно, если осуществляется перевод с языка Я1 на язык Я2 (рис. 2), то априори не видно причин, почему необходимо полное понимание смысла текста (путь Я1→С→Я2 на рис. 2), а не частичное (путь Я1 →С1→С2→Я2).

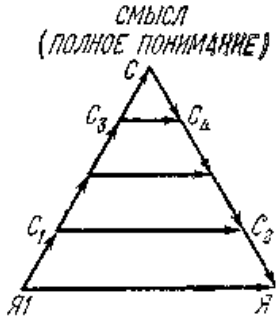


Рис. 2. Интерпретация задач машинного перевода и вопросно-ответных систем.

Использование подхода СМЫСЛА $\overrightarrow{\leftarrow}$ ТЕКСТ осуществляют и в МП.

2. При создании ВОС мы можем ограничить естественный язык **как по тематике, так и по грамматике**, и при этом не потеряем практической ценности работы. При МП требуется иметь дело с естественным языком. Попытки ограничиться определенной тематикой не делают задачу настолько простой, чтобы удалось получить

практически значимые результаты. Как следствие указанного фактора, в ВОС могут быть использованы более **формализованные лингвистические модели**, существенно упрощающие естественный язык и позволяющие получить эффективные алгоритмы обработки текста.

Методы, используемые при разработке большинства этапов ВОС, в существенной степени зависят от лингвистической модели, принятой для описания языка. Поэтому в очередном параграфе приведены сведения из лингвистики, необходимые для понимания последующего материала.

11.2. Формальные грамматики

11.2.1. Основные определения

Будем называть ***языком*** **конечное или бесконечное множество предложений, каждое из которых имеет конечную длину и построено с помощью операции соединения из конечного множества элементов**. Это определение включает как естественные, так и искусственные языки логики и программирования.

Чтобы точно определить язык, необходимо установить общие принципы, которые отделяют последовательности атомарных элементов, являющихся предложениями, от последовательностей, таковыми не являющихся. Это различие нельзя выразить просто списком, так как для всех систем, представляющих интерес, не накладывается ограничений на количество предложений и их длину. Традиционные грамматики естественных языков не содержат всей информации, необходимой для выполнения указанной задачи. Стремление к созданию более строгой лингвистической теории привело к разработке формальных грамматик, основоположником которых считается Н. Хомский. Он указал на возможность использовать для описания естественных языков некоторые исчисления, рассматривавшиеся ранее в теории алгоритмов, придав им удобную для этого форму (за ними закрепилось данное Н. Хомским название «порождающих грамматик»).

Следуя Хомскому, предъявим к лингвистической теории два требования:

1) порождать все те и только те предложения, которые составляют описываемый данной грамматикой язык;

2) определять метод, с помощью которого можно было бы дать единственное и единообразное структурное описание каждого

предложения, порожденного грамматикой. Было бы удобно, чтобы структурное описание получалось автоматически при порождении предложения.

11.2.2. Формальные грамматики

Введем понятия **цепочек и языков**, а затем определим понятие **порождающей грамматики**.

Пусть *V*— непустое конечное множество, которое мы будем называть *словарем (алфавитом),* а его элементы — *символами (буквами).* Произвольную конечную последовательность элементов ω будем называть *цепочкой* в словаре *V.* Пустую цепочку будем обозначать символом Λ. Число символов в цепочке будем называть *длиной цепочки* и обозначать |ω|. Над цепочками определяется *операция конкатенации.* Конкатенацией непустых цепочек $b_1 \ldots b_n$ и $c_1 \ldots c_p$ называется цепочка $d_1 \ldots d_{n+p}$, где $d_1 = b_1, \ldots, d_n = b_n, d_{n+1} = c_1, \ldots, d_{n+p} = c_p$. Конкатенацию цепочек ω и φ мы будем обозначать ωφ. Кроме того, конкатенация цепочки ω и Λ, равно как Λ и ω, считается по определению равной ω.

Если для каких-либо цепочек ω, φ, η_1, η_2 в словаре *V* имеет место равенство $\omega = \eta_1\varphi\eta_2$, то будем называть цепочку $\eta_1{*}\varphi{*}\eta_2$, где *с*имвол * не принадлежит *V, вхождением цепочки* φ в ω. Вхождение символов в цепочку будем называть ее *точками.* Если $\alpha = \eta_1{*}b{*}\eta_2$ и $\beta = \xi_1{*}c{*}\xi_2$— точки одной и той же цепочки $\omega = \eta_1{*}b{*}\eta_2 = \xi_1{*}c{*}\xi_2$ и если при этом $|\eta_1| < |\xi_1|$, то мы будем писать $\alpha < \beta$ или $\beta > \alpha$ и говорить, что α расположена левее β, а β — правее α. Для любых двух точек α и β цепочки ω таких, что $\alpha \le \beta$, мы будем называть множество точек ξ, удовлетворяющих неравенствам $\alpha \le \xi \le \beta$, *отрезком цепочки* ω.

Произвольное множество цепочек в словаре *V* мы будем называть *языком* в этом словаре. Естественно, что задание языков практической сложности перечислением всех цепочек, составляющих язык, нецелесообразно. Языки задаются с помощью формальных грамматик, порождающих все цепочки данного языка и только их.

***Порождающая грамматика* — это упорядоченная четверка**

$\Gamma = (V_т, V_н, S, P)$, где $V_т$ и V_H — непересекающиеся непустые конечные множества; *S* — некоторый элемент из V_H;

P —конечное множество правил вида $\varphi \to \psi$, где φ и ψ — произвольные цепочки словаря $V_T \cup V_н$ и символ → не входит в $V_T \cup V_H$.

Множества V_T и V_H называются соответственно *терминальным (основным)* и *нетерминальным (вспомогательным) словарями,* а их

элементы соответственно терминальными и нетерминальными символами грамматики Г. Объединение $V_T \cup V_H$ будем называть *полным словарем* грамматики Г.

S называется *начальным символом* Г. Это выделенный нетерминальный символ, обозначающий класс всех тех языковых объектов, для описания которых предназначена данная грамматика. Иногда символ S называют *целью грамматики.*

Отметим, что при изучении естественного языка в аспекте теории формальных грамматик терминальные символы интерпретируются как *словоформы* (**словоформами называют единицы языка, рассматриваемые одновременно в плане выражения (последовательность букв от пробела до пробела**) и в плане содержания (совокупность значений)) или *морфемы* (**морфемами называют наименьшие осмысленные единицы языка (корни, суффиксы и т. п.**)), нетерминальные символы — как названия *классов слов* и *словосочетаний,* а начальный символ— как *предложение.*

P называют *схемой грамматики* Г, а цепочки вида $\varphi \rightarrow \psi$ называют *правилами* Г.

Пусть $r=\varphi \rightarrow \psi$ — некоторое правило грамматики Г и $\xi_1 * \varphi * \xi_2$ — вхождение φ в цепочку $\omega=\xi_1\varphi\xi_2$ в словаре $V_T \cup V_H$. Говорят, что цепочка $\eta=\xi_1\varphi\xi_2$ получается из ω применением правила *r* к вхождению φ в цепочку ω. Если цепочка η получается из цепочки ω применением какого-либо правила Г, будем говорить, что η *непосредственно выводима* из ω в Г и будем писать $\omega \mid= \eta$.

Последовательность цепочек $D= (\omega_o, \omega_1, \ldots, \omega_n)$, $n \geq 1$, называется *выводом* ω_n *из* ω_o *в грамматике* Г, если для каждого *i*, $1 \leq i \leq n$, имеет место $\omega_{i-1} \mid= \omega_i$. Число *n* называется *длиной вывода* D. Если существует вывод цепочки η из ω в Г, то будем говорить, что η *выводима из* ω в Г, и писать $\omega \mid\!\!-\!\!- \eta$.

Множество цепочек в основном словаре грамматики Г, выводимых из ее начального символа, называется *языком, порождаемым грамматикой* Г, и обозначается *L(Г).*

Из приведенных выше определений видно, что **грамматика — это исчисление, т. е. разрешение производить некоторые операции** — в данном случае подстановки, **а не алгоритмы**, т. е. **предписание производить операции (директивы)**.

Грамматики, на правила которых не наложены никакие ограничения, способны порождать любые множества цепочек, порождаемые каким-либо автоматом. В теории алгоритмов такие множества называются *рекурсивно-перечислимыми.* Языки, порождаемые этими

грамматиками, образуют слишком широкий класс и не представляют интерес для лингвистики.

Естественно полагать, что для множества цепочек, составляющих некоторый естественный язык, у носителей языка есть распознающий алгоритм, т. е. **способ узнавания принадлежности каждой предъявленной цепочки к цепочкам данного языка**. Более того, этот алгоритм должен давать ответ довольно быстро. **Множества, для которых существуют распознающие алгоритмы, называются *рекурсивными.***

Мы будем рассматривать грамматики, порождающие рекурсивные множества.

При изучении формальных грамматик выделяют **три типа грамматик**, представляющих наибольший интерес как в теоретическом, так и в практическом смыслах. Эти грамматики задаются путем наложения последовательно усиливающихся ограничений на правила *P*.

Грамматика Г называется *грамматикой составляющих* или *непосредственно составляющих (**НС-грамматикой**),* если каждое ее правило имеет вид $\xi_1 A \xi_2 \rightarrow \xi_1 \psi \xi_2$, где ξ_1 и ξ_2— произвольные цепочки в словаре $V_T \cup V_H$, $A \in V_H$ и ψ — произвольная непустая цепочка в $V_T \cup V_H$. При применении НС-правила одно вхождение символа A заменяется на ψ в зависимости от наличия нужного контекста. Поэтому данные грамматики называют также *контекстными,* или *контекстно-чувствительными.*

Грамматика Г называется *бесконтекстной* или ***контекстно-свободной*** **(КС**), если все ее правила имеют вид $A \rightarrow \psi$, где A — нетерминальный символ, а ψ — произвольная непустая цепочка в $V_T \cup V_H$.

Грамматика Г называется *автоматной грамматикой* или ***A*-грамматикой** с конечным числом состояний, если каждое ее правило имеет вид $A \rightarrow aB$ или $A \rightarrow a$, где $a \in V_T$, A и $B \in V_H$.

В указанной последовательности классов порождающих грамматик, каждый последующий класс (в смысле общности правил грамматики) является частью предыдущего.

Языки, порождаемые грамматиками перечисленных классов, называются соответственно *языками непосредственно составляющих,* или *контекстно-чувствительными, бесконтекстными и автоматными.*

Как было указано ранее, задача грамматики состоит не только в том, чтобы породить предложения языка, но и приписать каждому из них **структурное описание**. В современной лингвистике наиболее

употребительны два способа описания синтаксической структуры предложения:

1) описание с помощью систем составляющих;
2) описание с помощью деревьев синтаксического подчинения.

Пусть *x* — произвольная непустая цепочка в словаре *V*. Множество С отрезков цепочки *x* называется *системой составляющих* этой цепочки, если оно удовлетворяет двум условиям:

1) множество *C* содержит отрезок, состоящий из всех точек цепочки *x,* и все одноточечные отрезки *x;*
2) любые два отрезка из С либо не пересекаются, либо один из них содержится в другом.

Элементы *C* называют *составляющими.* Одноточечные отрезки называют *точечными составляющими;* отрезок, состоящий из всех точек цепочки,— *полной составляющей;* полную и точечные составляющие называют *тривиальными.*

Для наглядности изображения системы составляющих иногда заключают в скобки каждую нетривиальную составляющую. Скобки можно не нумеровать, так как в силу пункта 2 определения системы составляющих каждой левой скобке можно однозначно указать соответствующую ей правую.

Если цепочка *x* интерпретируется как предложение естественного языка, то система составляющих может быть использована в качестве **способа выражения информации** о его синтаксической структуре. Такая **информация может представлять собой список словосочетаний, то есть тех «кусков» предложения, которые в каком-то интуитивном смысле являются «синтаксически связанными».** Эмпирические соображения позволяют сделать допущения, что словосочетания, во-первых, образуют отрезки и, во-вторых, не имеют «частичных пересечений», т. е. удовлетворяют п. 2 определения системы составляющих.

Таким образом, **нетривиальными составляющими являются словосочетания** (при подходящем выборе системы составляющих). Тривиальные составляющие добавлены для придания системе формальной законченности.

Среди многочисленных систем составляющих, которые имеет предложение естественного языка, лишь весьма немногие «правильны», то есть адекватно отражают синтаксическое строение предложения. При этом понятие «правильной» системы составляющих не абсолютно, оно зависит от соглашений лингвистического характера, отражающих определенные содержательные представления о синтаксической структуре предложения данного языка. Нетрудно видеть, что системе составляющих можно поставить в соответствие

дерево корнем которого служит **полная составляющая**, а висячими узлами — **точечные составляющие**. Это дерево называется **деревом составляющих**. Приведем пример (рис. 3) дерева составляющих для фразы

(Онегин, (добрый (мой приятель))),
(родился (на (брегах Невы))).

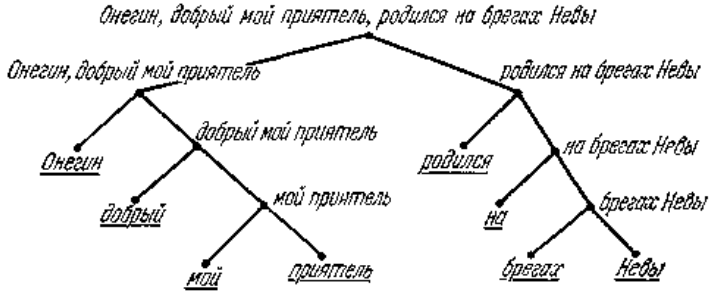


Рис.3. Пример дерева составляющих.

Построенная система составляющих указывает в предложении словосочетания разных «уровней» (составляющие разной высоты), но не вводит при этом никакой иерархии среди словосочетаний одного «уровня». Между тем, в предложениях естественного языка часто интуитивно ощущается «главенствование» некоторого словосочетания над другими, в нем не содержащимися.

Чтобы в некоторой степени отразить этот факт, вводят иерархизованную систему составляющих следующим образом. Пусть C — система составляющих цепочки *x*. Для каждой неточечной составляющей $A \in C$ выделим во множестве всех составляющих, непосредственно вложенных в *A,* какую-либо одну составляющую *A',* которую будем называть *главной.* Множество всех главных составляющих обозначим C′ и назовем *иерархизацией системы C.* Упорядоченную пару *(C, C')* назовем *иерархизованной системой составляющих.*

При описании предложений естественного языка с помощью системы составляющих *C* часто используют **способ введения в эту систему дополнительной информации** путем отображения *C* во множество всех подмножеств некоторого конечного множества, элементы которого называются *метками* и содержательно интерпретируются как символы синтаксических классов слов и словосочетаний. Упорядоченную тройку (C, *W,* φ), где *C*—система составляющих, *W*— множество меток и φ — отображение *C* в 2^W, называют *размеченной системой*

составляющих. Пусть множество меток W, состоит из элементов, указанных в таблице 1.

Таблица 1

Члены множества W	Содержательная интерпретация слов и словосочетаний
S	Предложение
VP_{xyvw}	Группа глагола } в роде x, числе y, времени
V_{xyvw}	Глагол } v и лице w.
NP_{xyz}	Группа существительного } рода x, в числе y
N_{xyz}	Существительное } и падеже z
A_{xyz}	Прилагательное в роде x, числе y и падеже z
PP	Предложная группа
$ПР$	Предлог «на»

В ней одновременно с перечислением меток приведена их содержательная интерпретация.

Тогда для приведенного ранее примера получим следующую «естественную» разметку (рис. 4).

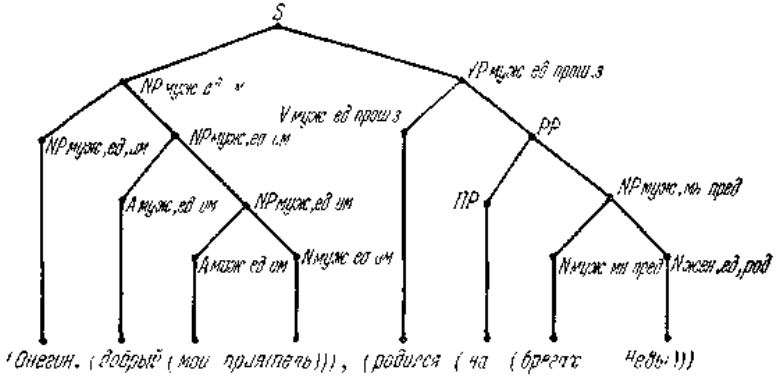


Рис. 4. Пример размеченного дерева составляющих.

Заметим, что вывод цепочки в НС-грамматике можно также представить в виде дерева. При этом, если каждому нетерминальному символу A, заменяемому в правиле грамматики $\xi_1 A \xi_2 \rightarrow \xi_1 \psi \xi_2$ на цепочку ψ, поставить в соответствие вершину, из которой исходят ребра к символам, образующим цепочку ψ, также интерпретируемым

как вершины, то выводу цепочки будет соответствовать дерево. Нетрудно видеть, что это дерево является деревом составляющих. В теории порождающих грамматик его часто называют *С-маркером*
Тот факт, что НС-грамматика при порождении терминальных цепочек одновременно дает их дерево составляющих, делает НС-грамматики особенно интересными с лингвистической точки зрения.
Определим теперь дерево подчинения.
Пусть *x* — произвольная непустая цепочка в словаре *V* и X — множество всех точек в *x*. Произвочьное бинарное отношение → на *X*, при котором граф <*X*; →> является деревом, будем называть *отношением синтаксического подчинения* или просто *отношением подчинения* для *x*. Само дерево <*X*; →> будем называть *деревом (синтаксического) подчинения* для *x*.
При графическом изображении дерева подчинения точки цепочки *x* помещают на горизонтальной прямой (рис. 5), и для всякой пары точек α, β, для которой α→β будем проводить из α в β стрелку, таким образом, чтобы все стрелки были по одну сторону от прямой.
На рис. 5 изображено несколько различных деревьев подчинения для одной и той же цепочки *abcdefg*.

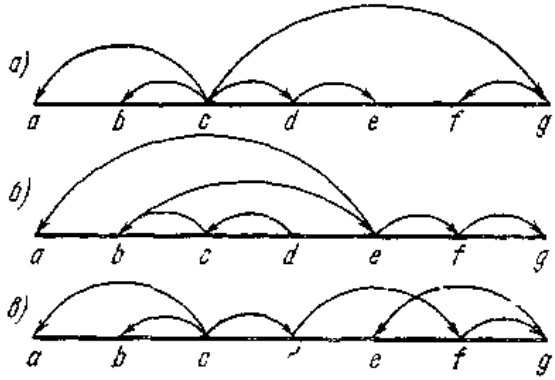
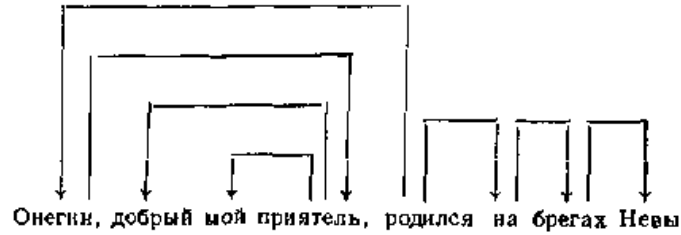


Рис. 5. Возможные деревья подчинения для цепочки *abcdefg*.

Деревья подчинения могут быть использованы как один из способов изображения синтаксической структуры предложения. Именно **информация о синтаксическом строении предложения может представлять собой набор сведений о «главенствовании» одних слов (точнее, вхождений слов) в предложении над другими**; задать такой набор — значит задать некоторый граф на множестве точек цепочки. Из интуитивных соображений вытекает, что этот граф можно считать деревом.

Приведем пример дерева подчинения для предложения из рис.4, соответствующего «естественным» представлениям о главенствовании одних слов над другими.



Понятие «правильного» дерева подчинения, так же как и системы составляющих, для естественного языка зависит, вообще говоря, от некоторых лингвистических соглашений.

Для анализа предложений естественного языка часто используются размеченные деревья подчинения, в которых, кроме подчинения, указывается его вид.

В классе всевозможных деревьев подчинения выделяют подкласс, который содержит подавляющее большинство «естественных» деревьев для предложений реальных языков. Это класс так называемых *проективных деревьев.*

Дерево подчинения $<X; \rightarrow>$ для цепочки x, а также соответствующее отношение подчинения $\rightarrow$, называется *проективным,* если для любых трех точек α, β, γ цепочки x из того, что $\alpha \rightarrow \beta$ и γ лежит между α и β, следует, что γ зависит от α (т. е. в дереве существует путь из α в γ).

Имеется еще один важный класс деревьев подчинения, являющийся расширением предыдущего: класс *слабо проективных деревьев.* Дерево подчинения $<X; \rightarrow>$ для цепочки x (и отношение $\rightarrow$) называется *слабо проективным,* если для любых четырех точек α, β, γ, δ цепочки x из $\alpha \rightarrow \beta$ и $\gamma \rightarrow \delta$ следует, что пары α, β и γ, δ не разделяют друг друга. *(Пары точек* α, β и γ, δ *разделяют друг друга,* если одна из точек γ, δ лежит, а другая не лежит между α и β.)

При графическом способе изображения слабая проективность равносильна возможности провести все стрелки так, чтобы никакие две из них не пересекались, а проективность, кроме того, обозначает, что корень дерева не лежит ни под какой стрелкой.

На рис. 5 дерево *а*) проективно, деревья *а), б)* слабо проективны, дерево *в*) непроективно.

Содержательный смысл условий проективности и слабой проективности обозначает, что слова, близкие синтаксически, близки и по положению в тексте.

В так называемой научной и деловой прозе, по крайней мере русской, естественные деревья подчинения подавляющего большинства предложений слабо проективны и даже проективны.

В рассматриваемом нами применении (ИИО) указанное ограничение не является существенным, но упрощает представление синтаксической структуры предложения.

Можно показать, что существуют правила перехода от систем составляющих к деревьям подчинения и обратно.

Рассмотрим вопрос о возможности использования введенных формальных грамматик для описания естественного языка. В литературе подробно освещен вопрос о недостаточности **А-грамматик и КС-грамматик** для описания естественных языков в полном объеме. Заметим, однако, что это не исключает их использования на определенных уровнях описания грамматики.

Что касается **НС-грамматик** и равных им по порождающей силе *неукорачивающих грамматик,* то есть грамматик с правилами $\varphi \rightarrow \psi$, удовлетворяющими тому требованию, что длина цепочки ψ не короче цепочки φ, то они достаточны (хотя и не обязательно удобны) для описания любых естественных языков в полном объеме. Это утверждение вытекает из следующих допущений:

1) любой естественный язык, точнее, множество его правильных фраз, есть легко распознаваемое множество, т. е. существует достаточно простой алгоритм распознавания правильности фраз. Данное допущение является практически очевидным, так как люди обладают таким алгоритмом;

2) алгоритм распознавания правильности фраз естественного языка должен обеспечивать такой процесс распознавания, при котором требуемый объем «оперативной памяти» сопоставим с длиной фразы, например, не превышает числа Mn, где n— длина фразы, а M — достаточно большая константа.

Последнее допущение подтверждается психологическими экспериментами. Автоматы, на работу которых накладывается такое ограничение, называются *линейноограниченными.* Как известно, ***множество цепочек, представимых линейноограниченным автоматом, есть контекстно-чувствительный язык.*** Таким образом, из наших допущений следует, что НС-грамматики в принципе способны описывать множество правильных фраз любого естественного языка.

Однако, приведенные выше рассуждения не гарантируют удобства описания любого естественного языка.

Отметим основные недостатки НС-грамматик:

1) с помощью НС-грамматик не удается естественно описывать фразы, содержащие непроективные конструкции;

2) НС-грамматика, как и любая порождающая грамматика в смысле данного нами определения, содержит только правила образования языковых выражений, но не содержит правил преобразования правильно построенных выражений.

В литературе **по математической лингвистике** кроме термина «порождающие грамматики» используется и термин ***«распознающие грамматики»***. Деление грамматик на порождающие и распознающие имеет историческое объяснение, хотя по существу порождающие грамматики наиболее интересных типов, в частности, всех типов рассмотренных нами, могут быть использованы также и для распознавания. При этом, если вместо распознавания говорить о «допускании», то все порождающие грамматики могут трактоваться как *допускающие*. ***Грамматика* Г *допускает язык L,* если для Г известна процедура, определяющая для любой цепочки *x* ($x \in L$) ее принадлежность к языку L, если же *x* не принадлежит L, то от процедуры не требуется никакого ответа.** При этом допускающая процедура для любой порождающей грамматики состоит просто в применении правил к данной цепочке справа налево, а не как обычно слева направо, до тех пор пока это возможно. Верно и обратное, что распознающие грамматики, например, категориальные грамматики, могут использоваться для порождения фактически опять же применением правила в обратном порядке.

Таким образом, формальные грамматики по существу нейтральны по отношению к порождению и допусканию, а также распознаванию для рассмотренных нами грамматик. Можно говорить просто о формальных грамматиках, рассматривая отдельно от самих грамматик аспект направления применения правил.

11.2.3. Трансформационные порождающие грамматики (ТПГ)

Владение языком предполагает умение не только построить правильную фразу, но и перейти от одной фразы к другим, либо полностью синонимичным ей, либо отличающимся от нее по смыслу на определенную «величину». Примером таких переходов является переход от утвердительного предложения к вопросительному или отрицательному, переход от активной формы к пассивной, выражение одной и той же мысли разными способами и т. д.

В НС-грамматике все такие предложения будут порождаться более или менее независимо и, следовательно, не будут находиться в каких-либо явных отношениях друг к другу.

Соответствующая задача была впервые четко сформулирована Н. Хомским. Выдвинутая им концепция приобрела широкую популярность под названием **«трансформационной грамматики».** Фактически, дело заключается во введении еще одного семантического уровня описания языка. В самом деле, **инвариантом всех трансформаций (преобразований) является смысл.** Таким образом, **теория трансформаций оказывается по существу теорией синонимии в языке.**

Отметим, что НС-грамматики описывают синтаксический уровень в широком смысле, т. е. с включением морфологии, а **трансформационные грамматики включают и семантические преобразования**. Поэтому, когда говорят о недостаточности НС-грамматик для описания языка, следует понимать, что это верно только в смысле неохвата НС-грамматиками семантического уровня. На своем, чисто синтаксическом уровне, НС-грамматики оказываются принципиально вполне достаточными.

Основные положения ТПГ изложены в ряде работ Хомского. Согласно этим представлениям, ТПГ состоит из **трех основных компонентов: *синтаксического, фонологического, семантического*.** По Хомскому, собственно порождающей частью теории является только синтаксическая компонента, где по правилам грамматики происходит порождение *глубинной и поверхностной структур.* **Семантическая и фонологическая компоненты интерпретируют** соответственно **глубинные структуры и поверхностные**. Таким образом, центральная роль отводится синтаксической компоненте, состоящей из двух субкомпонент: *базовой* (или просто *базы)* и *трансформационной.* База синтаксической компоненты служит для порождения *глубинных структур.* В содержательном отношении глубинная структура является представлением логического содержания порождаемого предложения в терминах **элементарных суждений**. Задача трансформационной субкомпоненты состоит в том, чтобы перейти от глубинной структуры к предложению в естественном языке *(поверхностной структуре)* либо полностью синонимичному с ней, либо отличающемуся от нее по смыслу на некоторую величину.

Формально глубинные структуры представляют собой деревья *(С-маркеры),* порождаемые *категориальными правилами* (категориальная компонента) и *лексиконом*. Категориальные правила представляются в виде правил НС-грамматики, не содержащих терминальные символы, т. е. порождаемые ими цепочки состоят только из категориальных символов *(N*— существительное, *V*— глагол, *D* — детерминатив и т. п.)

Бесконечная порождающая способность грамматики категориальной компоненты вытекает из того факта, что допускается введение начального символа *S* (предложения) в строки вывода. Таким образом, правила подстановки могут, по существу, включать одни базовые С-маркеры в другие.

Подстановка терминальных символов в порождаемые предложения осуществляется с помощью **трех видов правил**: правила *субкатегоризации,* правил *селекции* и правил *лексического включения.* Субкатегориальные правила и правила селекции комбинируют категориальные символы вместе с субкатегориальными в *комплексные символы,* осуществляя замены типа *N*→[+*N*, + Нарицательность, + Исчисляемость, + Одушевляемость, — Абстрактность]. Правила лексического включения подставляют в порождаемое дерево вместо правой часги приведенного выше правила лексическую единицу с подходящими набором признаков, например «мальчик» (см. табл. 2)

Таблица 2

Фрагмент лексикона трансформационной грамматики

C	*D*
[+*N*, +Нарицательность, +Исчисляемость, +Одушевленность, +Человечность]	мальчик
[*V*+, +Транзитивность, [+Абстрактность] *AUX_DET* [+Одушевленность]]	испугать
[+*M*,]	мочь
[+*N*, +Нарицательность, —Исчисляемость, +Абстрактность]	искренность

Итак, лексикон состоит из записей, связанных с трансформациями подстановки, которые вводят лексические единицы (лексемы) в цепочки, порожденные категориальным компонентом, образуя так называемый *обобщенный С-маркер.* Все контекстуальные ограничения в базе обеспечиваются этими трансформационными правилами лексикона *(лексическими трансформациями).*

Отметим, что даже простому предложению в глубинной структуре может соответствовать система из нескольких элементарных суждений. На рис. 6 приведена глубинная структура простого предложения «Невидимый бог создал видимый мир».

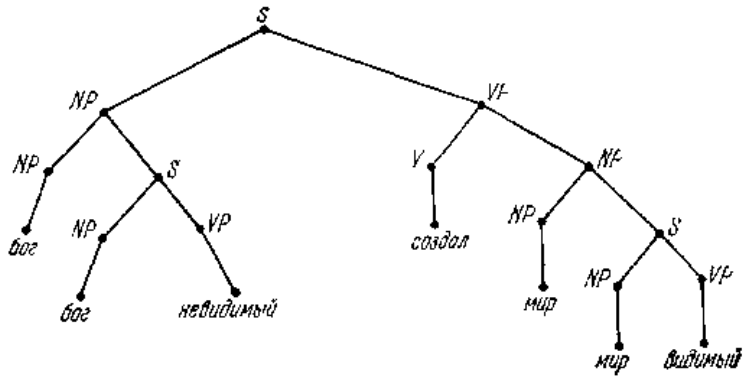


Рис. 6. Глубинная структура предложения «Невидимый бог создал видимый мир».

Как уже было указано ранее, задача трансформационной субкомпоненты состоит в том, чтобы из глубинной структуры, порожденной базой, получить поверхностную структуру, используя *грамматические (нелексические) трансформации.* Каждая трансформация представляется в виде правила, условие применимости которого задается в виде С-маркера.

Имея обобщенный С-маркер, мы строим трансформационный вывод, применяя нелексические трансформационные правила последовательно «снизу вверх». Другими словами, мы применяем последовательность правил к данной конфигурации только в том случае, если мы уже применили ее ко всем базовым С-маркерам, вставленным в эту конфигурацию.

Итак, мы определили ТПГ, которая включает систему правил структуры составляющих, порождающих деревьями систему трансформационных правил, отображающих деревья в деревья. Такая грамматика задает бесконечный класс конечных последовательностей деревьев *(P_1, . . ., P_n).* Каждая последовательность, называемая выводом, удовлетворяет следующим условиям:

P_1—дерево, порождаемое правилами категориального компонента;

P_n — дерево, представляющее поверхностную структуру порожденного предложения;

P_i (для любого i, кроме i=l) образовано применением к дереву P_{i-1} одного из трансформационных правил.

Отметим, что с каждой **лексической единицей** связана определенная лексическая трансформация, включающая ее в дерево.

В соответствии с требованиями теории Хомского подобные лексические трансформации предшествуют нелексическим трансформациям, и границей между двумя видами трансформаций является глубинная структура.

Полученная в результате трансформации поверхностная структура отображается с помощью фонологических правил на фонетические представления. Система таких правил образует *фонологическую компоненту* (рис. 7).

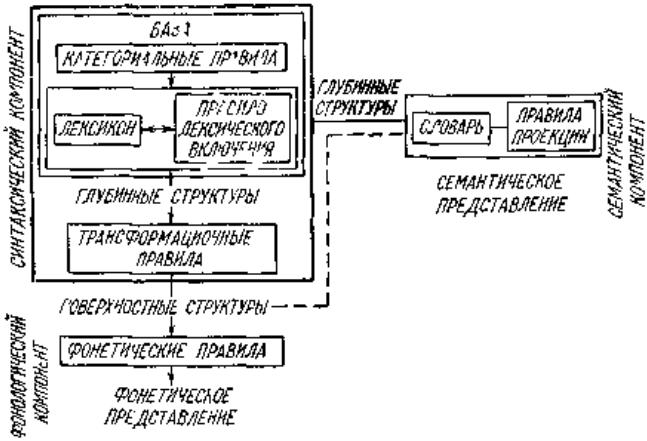


Рис. 7. Схема теории трансформационных порождающих грамматик («Стандартная теория»).

С другой стороны, глубинные структуры, после того как в них введены конкретные лексемы, отображаются с помощью **семантической компоненты на семантические представления.** Семантическая компонента состоит из *системы проекционных правил* и *словаря,* где каждая единица получает словарную статью, состоящую из стольких прочтений (readings), сколько у единицы значений.

Правила проекции идут снизу по дереву и сочетают прочтения единиц, образующих составляющие, учитывая при этом имеющиеся в прочтениях селекционные ограничения, запрещающие те или иные сочетания значений. **Получившийся в результате суммарный смысл и представляет собой семантическую интерпретацию, или прочтение данной глубинной структуры**

Катц и Постал показали, что возможна такая перестройка теории, при которой трансформации никогда не меняют смысла, и вся **смысловая**

информация предложения содержится уже в глубинной структуре. Поэтому семантической компоненте достаточно ее только интерпретировать.

Изложенная выше теория ТПГ называется **«стандартной теорией» Хомского.** В дальнейшем некоторые новые лингвисчические факты заставили Хомского признать в рамках «пересмотренной стандартной теории» зависимость семантического представления не только от глубинной, но и от поверхностной структуры (см. пунктир на рис. 7) Концепцию семантической компоненты, развитую в рамках «пересмотренной стандартной теории», называют *интерпретирующей семантикой* (ИС).

Существенный вклад в разработку теории глубинных структур в рамках интерпретирующей семантики внес Ч. Филмор. В отличие от Хомского Ч. Филмор считает, что основу предложения образует не **субъектно-предикатная**, а **предикатно-аргументная структура. Аргументами такой структуры Филмор считает имена, для которых может быть указан глубинный падеж.** При этом глубинные падежи выявляются в результате анализа так называемых скрытых категорий, проявляющихся в характере проведения трансформаций, перифразов и т. д. Падеж при таком понимании рассматривается как универсальное явление, присущее всем языкам независимо от того, имеется ли в них падеж в традиционном смысле или нет. Итак, ***глубинный падеж* — это обобщенное отношение между содержанием глагола и содержанием той или иной из его именных групп.** Ч. Филмор предлагает использовать **семь глубинных падежей**:

— агентивный *(A)* — одушевленный возбудитель лействия *(Джон* открыл дверь);

— инструментальный (*I*) — падеж неодушевленного предмета или силы, составляющих причину глагольного действия (состояния) *(Камень* разбил стекло.);

— дательный (D) — падеж одушевленного существа, затронутого глагольным действием (состоянием) *(Мальчик* получил удар в лицо.);

— фактитивный *(F)* — падеж предмета или существа, возникающего в результате действия (состояния) или входящего как часть в само глагольное действие (Мать варит *картошку.);*

— локативный *(L)* — место или пространственная ориентировка глагольного действия (состояния) (Джон идет *по улице.);*

— бенефактивный *(B)* — падеж пользователя (Мать варит картошку *для Джона.);*

— объективный (*O*) — немаркированный падеж, падеж любой вещи, представленной в виде имени, роль которой по отношению к

глагольному действию (состоянию) определяется из семантического толкования самого глагола (Джон открыл *дверь.).*

Филмор подчеркивает, что состав набора, а также характеристики и названия отдельных падежей не являются окончательными.

Глубинная структура по Ч. Филмору имеет следующий вид:

S→Aux+P, где S— предложение, *Aux* — модальный показатель (modality), *P*— пропозиция (proposition). Пропозиция может быть развернута в формулу следующего вида:

$$P \rightarrow V + C_1 + \ldots + C_n,$$

где *V*— глагол, $C_1 + \ldots + C_n$ — глубинные падежи.

С помощью подобных правил порождаются *претерминальные цепочки,* графическое представление которых дано на рис. 8.

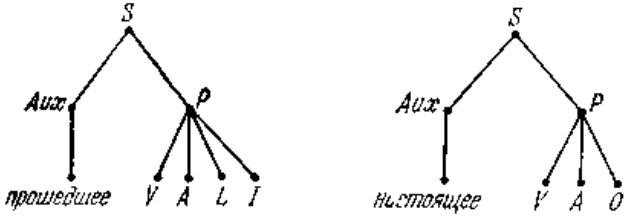


Рис. 8. Претерминальные цепочки падежной грамматики.

Отметим, что в данной грамматике, как и у Хомского, предполагается наличие *лексикона, правила лексического включения* и *трансформационных правил.*

Итак, по правилу лексического вывода, замена *комплексного символа C* претерминальной цепочки на некоторую словоформу *D* происходит при отождествлении этого символа с входной информацией *C* этой словоформы в словаре (табл. 3).

Таблица 3

Фрагмент лексикона в падежной грамматике

C	*D*
[+*A*]	*by NP*
[+*I*]	*with NP* (если во всей фразе уже есть *by*)
+[—*ALI*]	*smear* (*мазать*)
+[—*AOD*]	*show* (*показывать*)

Комплексный символ словоформы может содержать признаки двух родов: внутренние, характеризующие внутреннюю структуру

словоформы [+X, +Y], и внешние, указывающие на признаки окружающих слов (с прочерком на месте данного слова) [__X]. В табл. 3 приведен фрагмент лексикона в падежной грамматике.
Пользуясь базовым компонентом грамматики, можно вывести лексически заполненные предложения, типа приведенного на рис. 9.

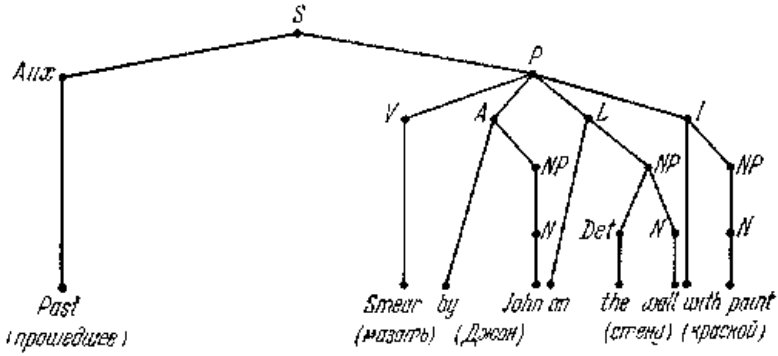


Рис. 9. Претерминальная цепочка падежной грамматики с лексическим заполнением.

К окончательному виду подобные предложения приводятся применением *общих и частных трансформационных правил.* Общие правила определяют выбор глубинных падежей на роль поверхностного субъекта и способы субъектного оформления слова. Частные правила определяют особенности состава и поверхностного оформления глубинных падежей у индивидуальных глаголов, если эти особенности не соответствуют типовым словарным предписаниям или общим трансформационным правилам.
Общие основания интерпретирующей семантики (ИС) были подвергнуты критике со стороны представителей *порождающей семантики* (ПС). Следует однако иметь ввиду, что и ИС, и ПС опираются на общую теорию трансформационных порождающих грамматик. Однако в ПС роль активного творческого элемента в процессе порождения высказывания принадлежит не синтаксису, а семантике. Соответственно этому, базовая компонента в модели порождающей семантики представляет собой **правила образования семантических представлений предложений**. Затем эти представления преобразуются с помощью трансформационных правил в поверхностные структуры. Модель ПС кроме того содержит также лексикон, заменяющий комбинации элементарных семантических смыслов лексемами. Схема лингвистической теории ПС представлена на рис. 10.

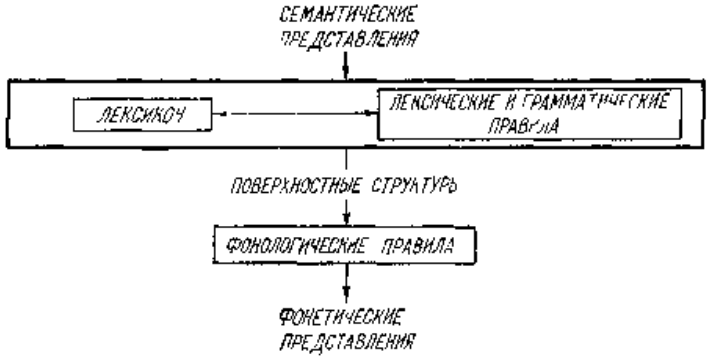


Рис. 10. Схема лингвистической теории в порождающей семантике

Следует отметить, что основные расхождения ПС и ИС касаются непризнания ПС уровня глубинных структур.

Следует отметить, что при практической реализации указанные различия между ПС и ИС сглаживаются, и с точки зрения практики нам кажется целесообразно говорить о **двух уровнях языка: уровне глубинных структур в широком смысле и уровне поверхностных структур. На первом уровне структуры характеризуются чисто семантическими свойствами, а на втором семантика осложнена синтаксическими категориями и преобразованиями.** Первый уровень не зависит от конкретного языка, а второй воплощается в конкретном естественном языке. Что же касается необходимости введения промежуточного уровня между глубинным и поверхностным, то он может использоваться просто как удобный технический прием, облегчающий переход от поверхностного уровня к глубинному и обратно.

На этом мы закончим рассмотрение лингвистических аспектов, используемых при построении вопросно-ответных систем (ВОС) и перейдем к вопросно-ответным системам (ВОС).

11.3. Классификация вопросно-ответных систем, понимающих естественный язык

Вопросно-ответные системы, понимающие естественный язык, по способам представления и использования знаний можно условно разбить на четыре типа:

1) системы, использующие форматы частного вида;
2) системы, основанные на запоминании текста;
3) системы с ограниченной логикой;
4) системы с общим выводом.

11.3.1. Системы, использующие форматы частного вида

К этому классу относятся наиболее ранние программы. Они обычно используют два частных формата — один для представления знаний, хранимых в системе, а другой для представления предложений в языке. Такие программы исходят из предположения, что **необходимой (требуемой) информацией** в предложении является только та, которая соответствует их **частным форматам**. Хотя программы данного типа могут иметь усложненные механизмы для использования этой информации, они создаются для частных целей и не обладают в управлении информацией гибкостью, которая бы могла позволить использовать программы для других целей.

11.3.2. Системы, основанные на запоминании текста

Стремление преодолеть ограничения рассмотренных выше программ привело к созданию систем, использующих текст в естественном языке во всем его разнообразии и общности как основу для представления знаний в системе. Запомненный текст снабжается различного рода схемами индексирования, предназначенными для упрощения поиска запрашиваемых предложений. Задача системы состоит в выдаче одного или нескольких предложений из знаний системы, имеющих отношение к запросу. Существуют разнообразные методы для поиска уместных предложений и выбора тех, которые наиболее удовлетворяют запросу.
Данный подход имеет ряд трудно разрешимых проблем. К ним относятся:
1) невозможность получения ответа на любой вопрос, который требует некоторых выводов из более чем одной части запомненной информации;
2) качество ответов в большей степени зависит от формы, в которой текст и вопрос определены в естественном языке, чем от смысла текста и запроса.

11.3.3. Системы с ограниченной логикой

Системы данного вида разрабатываются с целью устранения недостатков систем, основанных на запоминании текста в естественном языке. В первую очередь в этих системах в качестве базовых знаний вместо предложений в естественном языке используется более формальная нотация, преследующая цель представить семантические отношения между данными. Раз знания записаны в этом виде, система должна уметь переводить входные предложения в естественном языке в формат внутреннего представления, т. е. выполнять семантическую интерпретацию.

Основным недостатком ранних систем с ограниченной логикой был тот, что сложная информация выражалась в них в форме программ и для введения новых объектов требовалась разработка новых программ и их связь со старыми программами. В свою очередь каждое изменение в программе могло привести (и на практике обычно приводило) к изменению других программ. В результате система росла, теряла стройность, обозримость, и экспериментировать с такой системой становилось практически невозможно. Выход из указанных затруднений был найден за счет разработки новой техники программирования, способной использовать **процедуральную информацию**, но в то же самое время выражающую эту информацию способом, не зависящим от специфики программы или **темы диалога.**

Типичным примером систем подобного рода, называемых *системами с процедуральной дедукцией,* является система Винограда. Система выполнена в языках PLANNER, PROGRAMMER (модификация LISP) и LISP. Система отвечает на вопросы, выполняет команды и принимает информацию в процессе ведения диалога на английском языке. Система состоит из разборщика грамматики английского языка, выполненного в PROGRAMMERS, программ семантического анализа, выполненных в LISP, и общей системы принятия решений, выполненной в PLANNER'e. Система включает в себя также в виде теорем, записанных в языке PLANNER, детальную модель простого мира игрушек и упрощенную модель ее собственной разумности. Факты о текущей сцене представлены в виде утверждений PLANNER'a.

Система организует **знания как набор программ, называемых «специалистами».** Каждая из этих программ несет в себе частные знания системы о структуре языка и окружающем мире. Так, например, существуют **синтаксические специалисты**, которые анализируют различные типы фраз. Кроме того, существуют **семантические специалисты**, выполняющие различные функции. Они выбирают

значения индивидуальных слов в зависимости от контекста, определяют, какие комбинации слов имеют смысл, определяют виды ссылок. Семантические специалисты могут вызывать знания о мире, о ситуации, о беседе.

Каждый из специалистов может потенциально использовать любую информацию, собранную о предложении, контексте или ситуации реального мира другими специалистами. Эта гибкость необходима для управления английским языком, но она создает самостоятельную проблему, так как часный специалист интересуется частными аспектами процесса понимания языка и имеет специальную организацию информации. Для решения этой проблемы в системе используются развитые описывающие структуры и явное описание всех свойств текущей структуры.

11.3.4. Системы с общим выводом

Попытки увеличить дедуктивную мощность систем с ограниченным выводом привели к идее **выражать знания в некоторой математической нотации (например, в исчислении предикатов) и затем использовать результаты в области математической логики по формальному доказательству теорем. В таких системах вопрос, заданный в естественном языке, представляется в виде формулы исчисления предикатов и трактуется как теорема, которая должна быть доказана**.

Стимулом к развитию таких систем послужил разработанный Робинсоном **принцип резолюции**, являющийся полной универсальной процедурой доказательства для исчисления предикатов первого порядка. **На основании работы Робинсона стало возможным создание эффективных автоматических программ, доказывающих теоремы и обладающих двумя важными свойствами. Первое свойство состоит в том, что процедура доказательства в ней универсальна, т. е. не учитывает специфику данной области. Область задается множеством утверждений (аксиом). Второе свойство состоит в том, что если доказательство возможно с использованием правил исчисления предикатов, то процедура гарантирует, что оно будет найдено, хотя, возможно, за очень длительное время.**

Эти свойства удобны, но плата за их использование велика: снижение эффективности системы.

Подход к вопросно-ответным системам с использованием универсальной процедуры доказательств был развит в ряде разработок. В этих системах **информация представляется единым образом**, не

зависящим от особенностей частной программы. Это дает возможность пользователю описывать знания в нейтральном виде, не приспосабливая их к особенностям и частностям вопросно-ответной системы, и гарантирует то, что система будет применима к любой области, представимой в исчислении предикатов.
Использование исчисления предикатов имеет и серьезные недостатки. Представление сложной информации в нейтральной форме игнорирует важный вопрос, как представлять знания. Человек не хранит в голове точно определенное множество логических аксиом, из которых он выводит знания с помощью процедуры доказательства. **Скорее у него есть масса эвристик и процедур различной степени общности для решения и представления различных задач.** Игнорирование этого факта приводит к неэффективности при представлении в исчислении предикатов объектов реальной сложности. К недостаткам использования исчисления предикатов относится и то, что определенные свойства естественного языка плохо представимы в двузначной логике. Устранить указанные недостатки можно, используя **размытые множества и модальные логики.**

12. Синтаксический и семантический анализ предложений

12.1. Синтаксический анализ

Задача этапа синтаксического анализа (СА) предложения, записанного в некотором языке, заключается в построении структуры предложения, определяемой грамматикой этого языка. Так, например, при записи предложения в языке, определяемом КС-грамматикой, задача СА заключается в получении дерева составляющих (дерева зависимостей) исходного предложения При обработке языка, описываемого трансформационной грамматикой, задачей СА уже будет не получение **поверхностной структуры предложения, а построение его глубинной структуры**. В связи с тем, что для **удобства и компактности описания естественных языков используются трансформационные грамматики** или их модификации, мы будем считать задачей СА получение глубинной структуры предложения, выражаемой, например, и **виде дерева составляющих или в виде формул исчисления предикатов**. Однако, для введения в историю

вопроса и определения используемой в дальнейшем терминологии, мы приведем краткую характеристику ранних анализаторов.

12.1.1. Синтаксические анализаторы КС-языков

Когда создавались первые *синтаксические анализаторы (разборщики)* предложений естественного языка, не существовало теории синтаксиса, приемлемой для использования на машинах. Анализаторы представляли набор подпрограмм, который постепенно развивался по мере того, как грамматика разширялась и управляла все более сложными предложениями. Анализаторы имели те же недостатки, что и любые программы, конструируемые таким образом. В результате анализаторы становились все сложнее и все труднее становилось понимать их внутренние взаимосвязи. Неудачные попытки в области раннего машинного перевода ясно показали, что задача обработки естественного языка без лучшего понимания основ лингвистики и математических свойств грамматики является преждевременной.

В последующих исследованиях по обработке естественного языка на ЭВМ можно выделить **два подхода.**

В первом подходе игнорируется синтаксис в его традиционном понимании, и для получения **информации о предложении используется процесс сопоставления по образцу**. Системы этого типа не предпринимали попытки полного синтаксического анализа входного предложения. Они или ограничивали входной язык небольшим множеством фиксированных форматов, или ограничивали понимание предложений в результате игнорирования синтаксиса.

Во втором подходе берется упрощенное подмножество естественного языка, которое может быть описано в хорошо изученной формальной грамматике, например, такой, как некоторая вариация КС-грамматики.

Не останавливаясь детально на существе ранних алгоритмов разбора, воспринимающих КС-грамматики, отметим, что они разделяются по **двум основным параметрам**:

1) направлению разбора «сверху-вниз» или «снизу-вверх»;

2) последовательному или параллельному методу генерации деревьев разбора.

Деление алгоритмов по направлению разбора основывается на делении **грамматик на порождающие и распознающие**. **Алгоритмы «сверху-вниз» теоретически основываются на идее использования порождающей грамматики** при генерации всех возможных предложений языка, пока не будет порождено предложение, соответствующее входному. Такой подход при прямолинейном

использовании требует слишком много времени, но существуют способы, повышающие эффективность этого метода.

Алгоритмы «снизу-вверх» основываются на распознающих грамматиках. Алгоритмы пытаются объединить различным образом элементы входной строки до тех пор, пока не будет найдено дерево, покрывающее всю входную строку.

Алгоритмы разбора делятся на последовательные или параллельные в зависимости от того, строят они одно дерево разбора (если оно не соответствует входному предложению, то строится новое дерево, до тех пор пока не будет получен правильный разбор) или одновременно все возможные деревья разбора.

Последовательные алгоритмы являются медленными, но требуют мало памяти. К числу наиболее интересных параллельных алгоритмов относится алгоритм Эрли, который рассматривает одновременно (параллельно) все возможные анализы и получает все варианты разбора входной строки за время, пропорциональное кубу длины входной строки. Коэффициент пропорциональности определяется рассматриваемой КС-грамматикой и не зависит от вида строки. Для грамматик специального вида данный алгоритм автоматически достигает времени разбора, пропорционального квадрату длины (или длине) входной строки.

Как было указано ранее, контекстно-свободные грамматики не могут описать естественный язык в полном объеме.

12.1.2. Анализаторы языков, описываемых трансформационными грамматиками

Все более поздние попытки по расширению мощности грамматики, описывающей естественный язык, так или иначе связывались с **трансформационными грамматиками**. Из определения трансформационной грамматики следует, что **она ориентирована на генерацию предложений (синтез), а не на распознавание (анализ).** Хотя существует алгоритм, использующий трансформационную грамматику для анализа предложений, он слишком неэффективен, и вопрос о его практическом использовании даже не встает. Алгоритм работает по принципу «анализ через синтез», т. е. генерирует все возможные предложения и ищет среди них анализируемое.

Две попытки разработать более практичные алгоритмы трансформационного распознавания привели к созданию алгоритмов, или требующих больших затрат времени, или теряющих свойство полноты. Оба алгоритма пытались анализировать предложения, применяя трансформации в обратном порядке.

12.1.3. Анализ естественных языков, описываемых расширенными сетями переходов

Далее были разработаны системы, воспринимающие входной язык, близкий к естественному (английскому). Эти системы **используют понятие *расширенной сети переходов* (augmented transition network) и основываются на трансформационной грамматике.** Их задача выработать глубинную структуру предложения, анализируя его поверхностную структуру.

Введем понятия расширенной и *рекурсивной сети переходов,* являющиеся основными в перечисленных выше анализаторах и используемые ими для задания грамматики.

Рекурсивная сеть переходов (РСП) есть направленный граф с помеченными вершинами (состояниями) и дугами. Выделяется **начальное состояние и множество конечных состояний** (рис. 1).

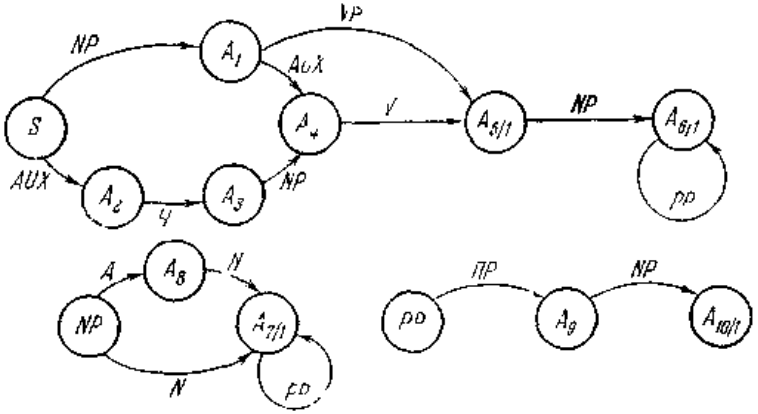


Рис. 1. Фрагмент грамматики, представленный в виде рекурсивной сети переходов. S — начальное состояние; A_5, A_6, A_7, A_{10}— конечные состояния; AUX — вспомогательный глагол; Ч — частица; PP — предложная группа существительного; ПР — предлог; AUX, Ч,V, A, N, ПР лексические (терминальные) категории.

Метки на дугах могут быть как терминальными символами, так и наименованиями состояний.

Дуга, помеченная наименованием состояния, интерпретируется следующим образом:

1) наименование состояния на конце дуги запоминается в стеке;

2) управление передается к состоянию, которым помечена дуга (без изменения обрабатываемого символа входной строки).

Будем в дальнейшем называть операции, перечисленные в пунктах 1 и 2, *погружением.*

Если текущее состояние является конечным, то происходит выталкивание последнего запомненного в стеке состояния и передача управления в это состояние. Критерием приемлемости входной строки является попытка вытолкнуть пустой стек, когда последний символ входного предложения обработан. Наименования состояний, появляющихся на дугах в этой модели, соответствуют наименованиям нетерминальных конструкций, которые могут быть обнаружены во входных предложениях.

На рис. 1 приведен пример рекурсивной сети переходов для подмножества русского языка. Конечные состояния имеют вид $A_{i/1}$. Приведенная РСП принимает, например, такие предложения, как: «Большой Джон будет строить красивый дом у белой реки», «Будет ли Михаил строить кооператив в Москве?» и т. п.

Нетрудно установить, что **РСП эквивалентна недетерминированному автомату с магазинной памятью и имеет мощность контекстно-свободной грамматики.** Как было уже указано, КС-грамматика не является адекватным механизмом для описания естественного языка. Увеличим возможности РСП, определив понятие **расширенной сети переходов** (РАСП), которая способна выполнять трансформационные преобразования, не вводя понятия **трансформационной компоненты**. Основное свойство, которое трансформационная грамматика добавляет к КС-грамматике — это способность передвигать, копировать и устранять фрагменты синтаксической структуры (так, что их положение в глубинной структуре отличается от положения в поверхностной структуре) и совершать эти действия в зависимости от контекста. Мы можем ввести эквивалентные свойства в РСП, добавив к каждой дуге:

1) произвольные *условия* (переход в состояние, указываемое дугой, разрешается только при выполнении этих условий);

2) произвольные *действия* (действия выполняются, если осуществляется переход к состоянию, на которое указывает дуга).

Будем называть это расширение РСП *расширенной сетью переходов* (РАСП).

РАСП строит частичное структурное описание предложения при переходе из одного состояния сети в другое. Куски этого частичного описания хранятся в **регистрах (со стековой структурой**), и их содержимое автоматически проталкивается (погружается), когда рекурсивное применение вызывает переход к более нижнему уровню.

Действия, связанные с дугами, изменяют содержимое этих регистров в терминах их предыдущего содержимого, содержимого других регистров, текущих входных символов и (или) результатов нижних уровней вычисления. Кроме того, регистры могут использоваться для хранения указателей (флагов), отражающих особенности прохождения через сеть. Указатели могут опрашиваться *условиями,* связанными с дугами.

Каждое конечное состояние связывается одним или несколькими условиями, при выполнении которых осуществляется переход на предыдущий уровень. Каждому из этих условий ставится в соответствие функция, результат вычисления которой возвращается как итог при переходе на верхний уровень.

Чтобы рассмотрение РАСП было более конкретным, приведем в табл. формальное определение языка, в котором РАСП может быть представлена.

Таблица 1

Формальное определение языка для описания расширенных сетей переходов

```
<сеть переходов> → (<множество дуг> <множество дуг> *)
<множество дуг> → (<состояние> <дуга> *)
<дуга> → (КАТ <имя категории> <условие> <действие> *<переход>) |
         (ПОГР <состояние> <условие> <действие> * <переход>) |
         (УСЛ<произвольн.метка><условие><действие>*<переход>) |
         (ВЫТ <форма> <условие>)
<действие> → (ПРТЕК <регистр> <форма>) |
             (ПРПОГР <регистр> <форма>) |
             (ПРВЫТ <регистр> <форма>)
<переход> → (ПУ <состояние>) |
            (ПП <состояние>)
<форма> → (ЗНАЧ <регистр>) |
          Δ |
          (СВОЙСТВО <свойство>) |
          (ДЕР <фрагмент> <регистр> *) |
          (СПИСОК <форма> *) |
          (ОБЪЕДИН <форма> <форма>) |
          (ИМЯ <произвольная структура>)
```

Определение дадим в форме расширенной бэкусовской нотации. Будем обозначать вертикальной чертой | альтернативные варианты, операцию * будем использовать как индекс, присутствие которого указывает на возможность вхождения конституента произвольное

число раз. Нетерминальные символы грамматики будем представлять в виде текста из строчных букв, заключенного в угловые скобки, а терминальные (исключая *, Δ, |) заглавными буквами, не заключенными в скобки. Символом Δ будем обозначать наименование регистра, содержащего текущее слово входного предложения.

Дадим пояснения к определениям. Первая строка в таблице указывает, что сеть переходов есть множество дуг (или несколько таких множеств), заключенное в круглые скобки. Множество дуг состоит из наименования состояния, за которым следует любое число дуг. Мы будем предполагать, что РАСП представляется в виде списочной структуры. Тогда сеть переходов есть список, элементы которого — множество дуг. Множество дуг в свою очередь есть список, элементами которого являются наименование состояния и дуги.

Дуги представляются в виде списков четырех разновидностей. Условия и действия, связанные с конечными состояниями, представляются как псевдодуги. Первый элемент списка (дуга) указывает тип дуги (КАТ — задает категорию терминального наименования, которым помечена дуга, ПОГР — указывает, что данный тип дуги вызывает операцию погружения, УСЛ — задает условие, ВЫТ — указывает на необходимость выполнения операции выталкивания стека). Третий элемент определяет произвольное условие, при выполнении которого осуществляется переход к состоянию, указанному в пятом элементе. Четвертый элемент определяет действие, осуществляемое при переходе к состоянию, на которое указывает дуга.

Опишем более подробно типы дуг. Дуга типа КАТ — дуга, которой следуют, если текущее входное слово является лексической (терминальной) категорией, указанной во втором элементе дуги.

Дуга типа ПОГР используется для обработки дуг, помеченных наименованием состояния (что вызывает операцию погружения)

Дуга типа УСЛ (условие) используется для задания произвольного условия, определяющего возможность следования к состоянию, указываемому дугой.

Дуга типа ВЫТ (выталкивание) является пустой дугой и предназначена для того, чтобы иметь возможность связать с конечными состояниями определенные условия (выполнение которых необходимо для подъема на более верхний уровень) и действия (возвращаемые на верхний уровень). Представления условий и действий в виде информации, связанной с пустой дугой, дает возможность упорядочить выбор ВЫТ по отношению к другим дугам данного состояния.

Состояние, к которому осуществляется переход, указывается в пятом элементе дуги. Переходы бывают двух видов:

1) с выбором в качестве текущего обрабатываемого символа очередного символа входной строки (предложения);

2) без изменения текущего обрабатываемого символа. По аналогии с программированием первый переход будем называть передачей управления (ПУ), а второй—переходом к подпрограмме (ПП).

Действия и *формы,* встречающиеся в сети, представляются в виде «польской нотации», нотации, в которой функция представляется как список, заключенный в скобки, первым элементом его является наименование функции, а остальными элементами — аргументы функции.

Мы определим **три типа *действия*,** которые присваивают значение *формы* наименованию регистра. Действия различаются по тому, на каком уровне они выполняют присваивание (ПРТЕК—присваивание на текущем уровне, ПРПОГР — присваивание на уровне погружения, т. е. уровне более глубоком, чем текущий, ПРВЫТ — присваивание на уровне выталкивания).

Формы, так же как и условия, можно записывать в виде функций языка программирования (например, LISP). Приведенные в табл. 1 типы форм составляют базовое множество, достаточное для того, чтобы проиллюстрировать основные свойства РАСП.

ЗНАЧ есть функция, чьим значением является содержимое указанное в форме регистра. Значением формы Δ является текущее слово входного предложения. В случае ПОГР значением формы Δ является уровень, получаемый при возврате из ПОГР.

СВОЙСТВО есть функция, которая определяет значение свойства, указанного в форме для текущего слова входного предложения.

Форма ДЕР (дерево) используется для построения фрагмента или полного дерева (структуры) разбора. Фрагмент задается использованием скобок, наименований символов и указателей параметров, изображаемых знаком + (см. ниже пример). Значением формы ДЕР является фрагмент дерева, в которое на место параметров подставлены значения регистров, указанных в форме. Параметры ставятся в соответствие регистрам следующим образом: содержимое первого регистра замещает первый символ +, содержимое второго регистра — второй символ и т. д. Кроме того, форма Δ (если она присутствует в ДЕР) заменяется на соответствующее ей значение. Три оставшиеся формы предназначены также для построения структур. Форма СПИСОК — создает список из значений аргументов, ОБЪЕДИНИТЬ — объединяет два списка в один, ИМЯ — производит

как значение (невычисленные) аргументы формы. Отметим, что три последних формы являются встроенными функциями языка LISP (LIST, APPEND, QUOTE).

При выполнении анализатора для конкретной грамматики для создания более гибкой системы (чем описанная) могут быть введены дополнительные форматы дуг, увеличена гибкость функции ДЕР и т. п. Вообще целесообразно **множество условий, действий и форм оставить открытым для расширений в ходе эксперимента**. Заметим, что формат дуг и действий вместе с произвольными выражениями LISP для изображения условий и форм обеспечивает модель, эквивалентную по мощности машине Тьюринга и, следовательно, полную в теоретическом смысле.

В табл. 2 приведен пример записи во введенном формализме фрагмента грамматики, введенной на рис. 1.

Таблица 2

Фрагмент грамматики, представленной в расширенной сети переходов

```
((S (ПОГР NP | Усл
      (ПРТЕК S Δ)
      (ПРТЕК ТИП (ИМЯ ПОВЕСТВ))
      (ПУА1))
      (КАТ AUX Усл
      (ПРТЕК AUX Δ)
      (ПРТЕК ТИП (ИМЯ ВОПР))
      (ПУА2)))
 (A1  (ПОГР VP | Усл
      (ПРТЕК AUX ø)
      (ПРТЕК VP Δ)
      (ПУА5))
      (КАТ AUX Усл
      (ПРТЕК AUX Δ)
      (ПУА4)))
 (A2  (КАТ Ч Усл
      (ПУА3)))
 (A3  (ПОГР NP | Усл
      (ПРТЕК S Δ)
      (ПУА4)))
 (A4  (КАТ V | Усл
      (ПРТЕК V Δ)
      (ПУА5)))
 (A5  (ВЫТ (ДЕР (S+++| (VP+)) тип, S, AUX, Δ) Усл)
      (ПОГР NP | Усл
      (ПРТЕК VP (ДЕР (VP (V+)Δ) V))
      (ПУА6)))
 (A6  (ВЫТ (ДЕР (S++++), тип, S, AUX, VP) Усл)
      (ПОГР PP | Усл
      (ПРТЕК VP (ОБЪЕДИН (ЗНАЧ VP (Список Δ)))
      (ПУА6))))
```

Рассмотрим действие РАСП, введенной в табл. 2 для предложения «Будет ли Джон строить дом?»

В качестве пояснения приведем содержимое регистров в соответствующих состояниях:

S: AUX:=будет;
ТИП: = ВОПР;
A2:
A3: *S:=(NP* Джон);
A4: *V* =строить;
A5: *VP:=(VP{V* строить)(*NP* дом));
A6: (S ВОПР(*NP* Джон) будет *(VP(V* строить) *(NP* дом))).

Полученную в состоянии А6 списочную структуру, соответствующую глубинному представлению исходного предложения, изобразим на рис. 2.

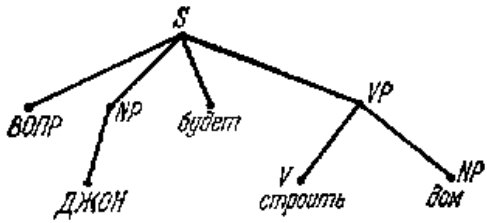


Рис. 2. Глубинная структура предложения.

В анализаторах НС-грамматик полученное структурное описание предложения соответствует алгоритму разбора, т. е. построение структуры происходит одновременно с применением грамматических правил. В РАСП эти процессы разделены. Это дает возможность некоторым конституентам, найденным в ходе разбора, появляться в окончательной структуре несколько раз или ни разу, и их место может отличаться от места в поверхностной структуре. Кроме того, структурное описание, присвоенное некоторой конституенте в ходе разбора, может быть впоследствии изменено. Эти свойства плюс способность вставлять проверку произвольных «условий» позволяет строить глубинную структуру в то время, когда анализатор выполняет переходы, соответствующие поверхностной структуре предложения.

Перечислим основные свойства РАСП, которые делают их привлекательными для использования в качестве модели естественного языка.

1. Наглядность. НС-грамматики являлись очень популярной моделью грамматики, несмотря на их неадекватность для управления

некоторыми свойствами естественного языка, во многом благодаря их наглядности. Теория трансформационных грамматик описывает практически все свойства естественного языка, но в том виде, в котором она существует, эта теория потеряла наглядность НС-грамматик. В трансформационной грамматике (ТГ) эффект отдельного правила связан с его взаимодействием с другими правилами, и требуется сложнейший анализ для определения эффекта и целей данного правила. РАСП, обеспечивая мощность ТГ, во многом сохраняет наглядность НС-грамматик.

2. Генеративная мощность. Как уже указано выше, РАСП имеет мощность машины Тьюринга, и при этом операции, выполняемые в РАСП, являются «естественными» для анализа языка. РАСП в отличие от ТГ (ориентированной на генерацию) с одинаковым успехом может использоваться как для генерации (формирования ответа), так и для распознавания (синтаксического анализа) предложений.

3. Эффективность представления. РАСП в отличие от НС-грамматик имеет средства для явного указания общих частей многих правил, что дает более эффективное представление. Объединение общих частей позволяет не только более компактно представлять грамматику, но и устраняет излишнюю обработку при разборе предложения (за счет уменьшения количества сопоставлений при определении применимости правил в течение разбора).

Отметим, что в РАСП возможно объединение подобных частей.

Пусть требуется представить в виде сети следующие подобные правила: *S→ABCDK, S→PBCTK*. На рис. 3 приведен пример объединения подобных частей этих правил.

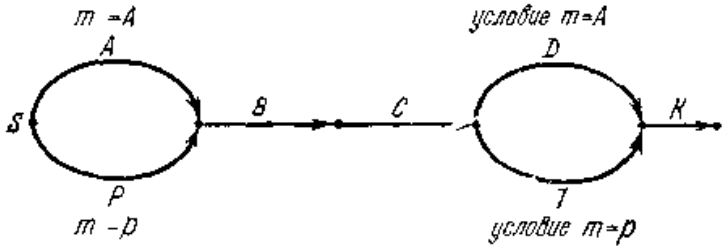


Рис. 3. Объединение подобных частей в расширенной сети переходов.

Для обоих правил строится единая сеть. Правила различаются благодаря действиям на дугах *A* и *P* и условиям на дугах *D* и *T*. Процесс объединения подобных частей делает грамматику более компактной, но увеличивает время разбора (за счет внесения дополнительных действий и условий).

4. Эффективность анализа. Как уже указывалось, эффективность вытекает из объединения общих частей и способности изменять построение фрагмента дерева (без повторного просмотра входного предложения) или откладывать его построение.

5. Гибкость экспериментирования. Гибкость достигается за счет того, что множество операций оставлено открытыми и может пополняться в ходе экспериментирования. Следует отметить, что явное построение структуры предложения (с помощью действий) позволяет использовать РАСП не только для построения глубинных структур предложения, но и для других видов представления таких, как грамматики зависимостей, падежные грамматики.

Рассмотрим **алгоритмы разбора**, которые можно использовать при представлении грамматики в виде РАСП. Как мы уже указывали, **РСП эквивалентна автомату с магазинной памятью**. Поэтому многие существующие алгоритмы для НС-грамматик могут быть непосредственно сведены к РСП. Кроме того, один из наиболее эффективных алгоритмов для КС-грамматик (п.12.1.1) может быть приспособлен с небольшими модификациями для использования в РСП.

Использование алгоритма Эрли для РАСП (а не для РСП) является более сложным делом. Действительно, для переходов, зависящих от содержимого регистров, трудно определить эквивалентные конфигурации и объединить их при дальнейшей обработке. Кроме того, использование регистров и действий при построении структур усложняет задачу выбора представления для объединенных конфигураций. Поэтому прямолинейно расширить алгоритм Эрли не удается. Однако, если в РАСП делать явные различия между **флаговыми регистрами (т. е. регистрами, содержащими только условия, выбранные из конечного словаря)** и **регистрами содержания (содержащими произвольные структуры)**, и ограничить условия и действия, приписанные дугам, таким образом, что:

1) действия могут ссылаться только на флаговые регистры и символы во входной строке;

2) на действия и условия наложить ограничения по времени обработки,

3) существует только один регистр содержания, изменяемый функцией ДЕР или конечным состоянием (содержимое его не анализируется),

тогда можно построить версию алгоритма Эрли с указанными ранее временными характеристиками. Однако если мы ослабим перечисленные условия, то временная граница превзойдет *(Kn^3)*

(например, одно условие и действие на дуге может потребовать больше, чем n^3 шагов).

Для многих применений нет необходимости получать представления всех возможных разборов входного предложения. В частности, в нашем случае более важно выбрать «наиболее вероятный» разбор в данном контексте, но сделать это наиболее быстрым образом. В таких применениях последовательный подход (с подходящим механизмом для выбора анализа, рассматриваемого первым) более предпочтителен, чем параллельный, так как он в большинстве случаев позволяет избежать необходимости следовать всем альтернативам. Повторный анализ тех же самых подстрок входной строки при альтернативных разборах можно устранить путем запоминания предыдущих результатов.

Успех последовательного подхода во многом зависит от качества механизма, осуществляющего выбор «наиболее вероятного» разбора. В РАСП некоторые из таких механизмов могут быть добавлены естественным образом. Упорядочением выбора дуг, исходящих из данного состояния, можно навязывать соответствующий порядок анализа. При создании грамматики можно подбором упорядочения дуг пытаться выделить наиболее вероятные ситуации. Кроме того, повторением в РАСП одной и той же дуги в виде нескольких дуг с разными условиями возможно сделать упорядочение этих дуг зависящим от особенностей обрабатываемого предложения, в частности, сделать зависящим от семантических свойств слов, встречающихся в предложении.

Заканчивая рассмотрение **методов синтаксического анализа**, укажем, что **алгоритмы разбора** могут рассматриваться как **процедуры поиска на графе**. Процедура поиска начинается с пустого дерева разбора. Каждый узел дерева поиска соответствует частному разбору предложения, а дуга соответствует добавлению нового уровня к дереву разбора. Различные методы синтаксического анализа отличаются способом, которым они добавляют новые дуги к дереву разбора. Алгоритмы «сверху—вниз» добавляют дуги на вершину дерева разбора и работают в направлении «вниз», в то время как процедуры «снизу — вверх» добавляют дуги в основание дерева и работают в направлении «вверх». Последовательные алгоритмы разбора соответствуют поиску «сначала в глубину», а параллельные — поиску «сначала в ширину». Однако хорошо известно, что **большей эффективностью обладают стратегии, управляемые эвристиками**.

Приведем примеры подобных эвристик:

1) избегать выполнения одного и того же частного разбора (т. е. разбора подстроки входного предложения) более одного раза;

2) избегать определенных разборов, которые не могут привести к успеху (определяются просмотром вперед, разделением предложения на части и т. п.);
3) устранять семантически нелепые анализы (проверкой части анализа на соответствие базовым данным, обычно при анализе «снизу—вверх»);
4) использовать оценочные функции и другие механизмы управления поиском наиболее перспективного пути.

12.2. Семантическая интерпретация

12.2.1. Общие сведения о семантической интерпретации

Входом для этапа **семантической интерпретации** (СИ) является глубинная структура обрабатываемого предложения, точнее **структуры**, так как **на этапе синтаксического анализа не удается (в общем случае) получить однозначную интерпретацию предложения. Задачей СИ явтяется понять входное предложение. Под словом «понять» мы подразумеваем не только выражение смысла предложения в известных нам понятиях, но и установление связи данного предложения с известными нам фактами.** Поэтому мы определяем задачи этапа семантической интерпретации следующим образом:
— получить **однозначное представление входного предложения** (интерпретировать его) в терминах базовых данных;
— **объединить входное предложение** с базовыми данными.
Этап СИ может выполняться как параллельно с процессом синтаксического анализа, так и после его завершения. В первом случае подпрограммы интерпретации добавляются к некоторым или всем правилам грамматики и выполняются во время применения грамматического правила при синтаксическом анализе. Достоинством этого подхода является то, что результат семантической интерпретации может направлять синтаксический анализ.
Второй подход имеет то преимущество, что глубинная структура, полученная на уровне синтаксического анализа, может просматриваться многократно и в произвольном порядке, что облегчает задачу СИ.
Следует отметить, что еще **не существует общей теории процесса семантической интерпретации**. Процессы СИ существенно зависят

от ряда факторов и в первую очередь от внутреннего представления базовых данных, определяющего общность или ограниченность вывода в системе.

В данном параграфе мы сначала опишем общую постановку задачи СИ, а затем рассмотрим ее конкретные реализации отдельно для **представления в виде семантических сетей (системы с ограниченной логикой) и в виде формул исчислении предикатов (системы с общим выводом).**

В связи с распространенностью в естественных языках явления многозначности слов, словосочетаний, предложений, раскрытие многозначности при обработке текстов на ЭВМ приобретает особое значение. Раскрытие многозначности некоторых предложений невозможно без ссылок на контекст или без обращения к общим знаниям о мире. Например, предложение «Он пошел в парк с девушками» может быть понято как:

1) Он и девушки пошли в парк, или 2) Он пошел в парк, где есть девушки. Вероятно, двузначность данного предложения может быть раскрыта ссылкой на контекст. Однако только общие знания о мире могут раскрыть двузначность следующих выражений: «Гастроли балета на льду» и «Гастроли балета на Кавказе».

Значительно большее количество предложений является многозначным для машины, чем для человека, так как ЭВМ имеет более простые алгоритмы для раскрытия многозначности и меньшие знания о контексте и мире. Например, приведенные выше предложения (о гастролях) будут двузначными для ЭВМ и скорее всего однозначными для всех людей. Существующие системы распознают только подмножество естественного языка, и это дает возможность раскрыть некоторые из многозначностей. **Оставшиеся многозначности устраняются обычно одним из двух способов: 1) *методом семантических категорий;* 2) *ссылками на базовые данные.***

Семантические категории расширяют традиционную грамматическую классификацию слов введением подклассов. Отсутствие подобных подклассов приводит к катастрофическому возрастанию количества интерпретаций. Например, **если слова A, B, C и D имеют по три значения, то предложение, содержащее все эти слова, может иметь 3×3×3×3=81 интерпретацию. Человек не строит все интерпретации, он выделяет наиболее разумные.** Известно, что слово «коса» может обозначать следующие объекты: сельскохозяйственный инструмент, выступающий мыс, длинные волосы, уложенные определенным образом. С другой стороны, «зеленый» может обозначать: цвет, неспелый, неопытный. Однако,

когда человек видит фразу «зеленая коса» он не рассматривает 9 интерпретаций, так как он знает, что «зеленый» в смысле «неспелый» применяется только к овощам и фруктам, а в смысле «неопытный» только к людям. Таким образом, **чтобы ввести эту информацию в ЭВМ, необходимо расширить традиционную грамматическую классификацию слов на классы**. Кроме обычных классов: **существительное, прилагательное и т. д., вводят классы «одушевленный», «неодушевленный», «человеческий», «абстрактный», «физический» и т. п** Используя подобную информацию, удается выделить бессмысленные комбинации слов при выборе интерпретации.

Типичной системой, использующей такую информацию, является программа, разработанная Глазерфельдом. Анализ предложения осуществляется «снизу— вверх». В начале анализа каждому слову в предложении присваивается список «индексных категорий» из словаря (соответствуют описанным выше классам и подклассам). В течение анализа, каждой более высокой вершине дерева разбора назначается своя **«индексная категория»** на основании «индексных категорий» составляющих данной вершины. Назначение категории осуществляется специальной процедурой — **«реклассификации».**

Недостатком метода семантических категорий является тот факт, что с их помощью не могут быть раскрыты все многозначности. Например, они не могут раскрыть двузначность предложения «Он пошел в парк с девушками», так как для раскрытия двузначности необходима **контекстуальная информация**. Другим их недостатком является увеличение и усложнение словаря в связи с детальной классиффикацией слов. При **методе семантических категорий информация в словаре будет частично дублировать информацию в базовых данных системы.** Желание объединить эту информацию приводит к методу раскрытия многозначности путем ***ссылки на базовые данные***. Например, в приведенном предложении семантический интерпретатор можем сформулировать вопрос к базовым данным: «Идут ли к ним девушки»? Ответ на этот вопрос может определить выбор интерпретации. В общем случае **метод раскрытия многозначности заключается в том, что все имеющиеся интерпретации сопоставляются с базовыми данными и устраняются интерпретации, противоречащие базовым данным**. Однако и после этого может остаться более одной интерпретации. Это наиболее трудный случай.

Можно указать несколько подходов к решению этой задачи:

1) выбрать интерпретацию, которая соответствует части базовых данных с наибольшими связями между вершинами;

2) выбрать интерпретацию, при объединении которой с базовыми данными к ним будет добавляться минимальное количество новых вершин и дуг;
3) выбрать наиболее разумную интерпретацию, используя некоторую «вероятностную» меру «разумности»;
4) запомнить все интерпретации в базовых данных и пытаться устранять многозначность за счет поступления новой информации (в частности, запрашивая необходимую информацию).

Отметим, что при обработке естественного языка **процессы СИ, представления фактов, выполнения выводов и формирования ответа** на вопросы могут быть существенно упрощены, если идентичные предложения, выражаемые различными поверхностными структурами, будут представлены **единой концептуальной конструкцией**. Одной из причин разнообразия поверхностных структур при единстве смысла является разнообразие «поверхностных глаголов», описывающих одну и ту же ситуацию. Шенк исходит из существования «глубинных» (канонических) глаголов, унифицирующих в глубинных структурах смысл многих поверхностных глаголов.

До сих пор мы, рассматривая вопросы семантической интерпретации, по умолчанию предполагая, что на вход системы поступают отдельные предложения. Задача значительно усложняется, если требуется понять смысл связанного текста. Перейдем к рассмотрению процессов СИ с учетом структуры представления базовых данных.

12.2.2. Семантическая интерпретация в системах с ограниченной логикой

При описании процесса СИ с ограниченной логикой мы будем основываться на реальной вопросно-ответной системе, воспринимающей естественный (английский) язык. Выбор реальной ВОС преследует цель показать состояние практики в решении одного из наиболее сложных этапов по обработке естественного языка— **семантической интерпретации**. Несмотря на то, что описываемая система воспринимает английский язык, мы там, где это возможно, будем для удобства читателей приводить примеры на русском языке. В случаях, где примеры отражают специфику английского языка, мы будем приводить английский текст, а в скобках давать его русский перевод. В рассматриваемой системе входом для СИ является каноническая структура, полученная в результате обработки ограниченного естественного языка вариантом расширенной сети переходов (РАСП). Действия РАСП были подробно описаны в п.

11.4.3, поэтому мы сейчас остановимся только на механизме, позволяющем осуществлять перевод входного текста в каноническую структуру. Основой этой операции является включение в словарь определенной информации (табл.3).

Таблица 3

Фрагмент словаря

L-BUY (ПОКУПАТЬ) WORD CLASS (КЛАСС СЛОВА) CANNON-VB (КАНОН. ГЛ) INF (ИНФИНИТИВ) SG3 (ЕД. ЧИСЛО 3 ЛИЦО) PAST (ПРОШЕДШЕЕ) -EN -ING	 VERB (ГЛАГОЛ) EXCHANGE (ОБМЕНЯТЬ) BUY BUYS BOUGHT BOUGHT BUYING
P-RULES (А-Правила)	(OK (HUMAN ORGANIZATION) BUYER) (БП (ЧЕЛОВЕК ОРГАНИЗАЦИЯ) ПОКУПАТЕЛЬ) (OK (PHYSOBJ) THINGBT) (БП (ФИЗИЧ. ОБЪЕКТ) ПОЛУЧ.) (FROM (HUMAN ORGANIZATION) SELLER) (У (ЧЕЛОВЕЧ. ОРГАНИЗАЦИЯ) ПРОДАВЕЦ) (FOR (MONEY) THINGGIVEN) (ЗА (ДЕНЬГИ) ОТДАННОЕ) (AT (PLACE) LOC) (В (МЕСТО) МЕСТО) (IN (PLACE) LOC) (OK (DAY TIME) TIME) (IN (DAY TIME) TIME))
G-RULES (Ф-правила)	((BUYER ACTIVE THINGBT (FROM SELLER) (FOR THINGGIVEN)) (THINGBT PASSIVE(FROM SELLER) (FOR THINGGIVEN)))
L-COST (СТОИТЬ) G-RULES	 ((THINGBT ACTIVE (BUYER) (THINGGIVEN)))

Продолжение табл. 3

L-PAY (ПЛАТИТЬ) G-RULES	((BUYER ACTIVE (SELLER) (THINGGIVEN) (FOR THINGBT)))
L-WHEN (КОГДА) WORD CLASS SEM	Q WORD (ВОПР. СЛОВО) (DAY DAYPART) (ДЕНЬ, ЧАСТЬ ДНЯ)
EXCHANGE SURF-VB (ПОВЕРХН. ГЛ.)	(L-BUY L-SELL L-PAY L-COST)

В словаре каждый поверхностный глагол содержит основной вход, представляемый в виде L-«ГЛАГОЛ», где L— признак основного входа, а «ГЛАГОЛ» — конкретный глагол в инфинитиве. Список свойств поверхностного глагола содержит различную информацию: **ссылку на каноническую форму данного поверхностного глагола, различные формы глагола, А-правила и Ф-правила.** А-правила используются при анализе входного предложения и соотносят поверхностные (предложные и беспредложные — обозначаются БП) существительные, используемые с данным поверхностным глаголом, глубинным структурам (например, в смысле Филмора), связанным с соответствующим каноническим глаголом. А-правила представлены в виде **списка триплетов**; первый элемент в каждом триплете указывает предлог (или отсутствие его), с которым используется поверхностное существительное. Второй элемент триплета есть список семантических категорий (о назначении семантических категорий см. п.12.2.1), к одной из которых должно относиться поверхностное существительное для того, чтобы удовлетворять глубинному падежному отношению (deep case relation), задаваемому третьим элементом триплета. Поясним структуру А-правила на примере: JOHN BOUGHT THE CAR FROM MARY (ДЖОН КУПИЛ АВТОМОБИЛЬ У МЭРИ).

Как видно из табл.3, поверхностному глаголу BUY (ПОКУПАТЬ) соответствует канонический глагол EXCHANGE (ОБМЕНИВАТЬ). Глагол ОБМЕНИВАТЬ определяется следующими глубинными отношениями (падежами):

ПОКУПАТЕЛЬ (BUYER); ПРОДАВЕЦ (SELLER); ПОЛУЧЕННОЕ (THINGBT); ОТДАННОЕ (THINGGIVEN); МЕСТО (LOC); ВРЕМЯ (TIME).

Из табл. 3 мы можем увидеть, что глагол BUY (ПОКУПАТЬ) для выражения глубинного отношения SELLER (ПРОДАВЕЦ) использует предлог FROM (У) и, кроме того требует, чтобы семантическая категория поверхностного существительного относилась к классу ЧЕЛОВЕЧЕСКИЙ или ОРГАНИЗАЦИЯ. Очевидно, что слово МЭРИ в рассматриваемом примере удовлетворяет обоим условиям и, следовательно, выражает отношение SELLER.

Предполагается, что в словаре каждое существительное и вопросительное слово (Q WORD) имеет специальное свойсво (SEM), указывающее семантическую категорию, к которой существительное или вопросительное слово принадлежит. Результатом разбора входного предложения с применением РАСП являются две (в случае вопроса три) вспомогательные структуры. Например, для предложения WHO BOUGHT A CAKE AT THE NEW BAKERY FROM THE BAKER? (КТО КУПИЛ ПИРОЖНОЕ В НОВОЙ БУЛОЧНОЙ У БУЛОЧНИКА?) мы получим три вспомогательные структуры:

Глагольная составляющая:

(CANON-VB EXCHANGE MODAL (TENSE PAST MOOD
INTERROG CASE AFFIRM))
(КАНОН. ГЛ. ОБМЕНЯТЬ МОДАЛ. (ВРЕМЯ ПРОШ.
НАКЛ. ИЗЪЯВИТ. ПАДЕЖ УТВЕРД)).

Составляющая существительных:

(OK(PHYSOBJ) (TOK L-CAKE DET INDEF NBR S))
(AT(PLACE) (TOK L-BAKERY DET DEF NBR S
MOD (AGE L-NEW)))
(FROM (HUMAN) (TOK L-BAKER DET DEF NBR S))
(БП (ФИЗИЧ. ОБ.) (ОБОЗН. L-ПИРОЖНОЕ
АРТИКЛЬ НЕОПР. ЧИСЛО ЕДИНСТВ.))
(В(МЕСТО)(ОБОЗН. L-БУЛОЧНАЯ АРТИКЛЬ ОПРЕД.
ЧИСЛО ЕДИНСТВ. МОД.(ВОЗРАСТЬ-НОВЫЙ)))
(У (ЧЕЛОВЕЧ.) (ОБОЗН. L-БУЛОЧНИК АРТИКЛЬ
ОПРЕД. ЧИСЛО ЕДИНСТВ.)).

Составляющая вопросительного слова:

(OK(HUMAN) (TOK L-WHO))
(БП (ЧЕЛОВЕЧ.) (ОБОЗН. L-КТО)).

Глагольная составляющая указывает на канонический глагол, определяющий предложение, и на характеризации глагола (модальность МОДАЛ). Составляющие существительного определяют предложные и беспредложные (БП) существительные в порядке их появления во входном предложении. **Каждая составляющая представляет собой триплет. Первый элемент триплета** указывает синтаксическую форму (предлог или его отсутствие), в которой существительное появилось в предложении. **Второй элемент** определяет семантическую категорию существительного (скопированную из словаря). **Третий элемент** является списком, описывающим свойства данного существительного из входного предложения. Составляющая вопроситечьного слова представляется в виде триплета со структурой, аналогичной триплету существительного.

Окончательная стадия отображения входного предложения в каноническую структуру состоит в сопоставлении составляющих существительных и вопросительного слова с А-правилами поверхностного глагола входного предложения. Сопоставление заключается в поиске для каждой составляющей существительного А-правила, у которого первый элемент триплета совпадает с первым элементом составляющей, а вторые элементы образуют непустые пересечения. Глубинное отношение, указываемое при этом третьим элементом А-правила, связывается с третьим элементом составляющей (если оно еще не было связано с ранее сопоставляемой составляющей). Когда заканчивается процесс сопоставления составляющих существительных, переходят к сопоставлению составляющей вопросительного слова. Для вопросительного слова сопоставление делается аналогичным образом среди оставшихся несопоставленными отношений. Отличие состоит в том, что результирующей структуре назначается псевдоотношение Q (QUESTION— ВОПРОС), а подходящие отношения (в нашем примере это одно отношение— BUYER) собираются в список ARGS (АРГУМЕНТЫ).

Таким образом, **<u>каноническая структура</u>, полученная для рассматриваемого входного предложения, имеет вид**:

```
(CANNON-VB EXCHANGE MODAL (TENSE   PAST
                MOOD INTERROG CASE AFFIRM)
SELLER (TOK L-BAKER DET DEF NBR S)
(THINGBT (TOK L-CAKE DET INDEF NBR S)
 LOC (TOK L-BAKERY DET DEF NBR S MOD (AGE
                L-NEW))
Q (ARGS (BUYER) TOK L-WHO))
(КАНОН. ГЛ. ОБМЕН ЯТЬ МОДАЛ. (ВРЕМЯ ПРОШ.
        НАКЛ. ИЗЪЯВИТ.    ПАДЕЖ   УТВЕРД.)
```

ПРОДАВЕЦ (ОБОЗН. L-БУЛОЧНИК АРТИКЛЬ
ОПРЕД. ЧИСЛО ЕД)
ПОЛУЧЕННОЕ (ОБОЗН. L-ПИРОЖНОЕ
АРТИКЛЬ НЕОПР. ЧИСЛО ЕД)
МЕСТО (ОБОЗН. L-БУЛОЧНАЯ АРТИКЛЬ ОПРЕД.
ЧИСЛО ЕД. МОД. (ВОЗРАСТ L-НОВЫЙ))
ВОПРОС (АРГ. (ПОКУПАТЕЛЬ) ОБОЗН. L-КТО)).

Изложенный способ отображения входного предложения в каноническую структуру рассчитан на простые активные предложения без вложенных конструкций. Подход может быть расширен в направлении включения:

1) более сложных А-правил, явно выражающих упорядоченность существительных в списке составляющих;
2) более сложного алгоритма сопоставления, распознающего относительные предложения;
3) более разнообразных синтаксических форм и семантических категорий;
4) включение пассивных конструкций и вложенных предложений.

Перейдем теперь к вопросу объединения информации входного предложения с базовыми данными. **Базовые данные, представленные в виде семантической сети, хранят информацию об объектах и отношениях между объектами.**

На рис. 4. приведен пример текущего состояния семантической сети.

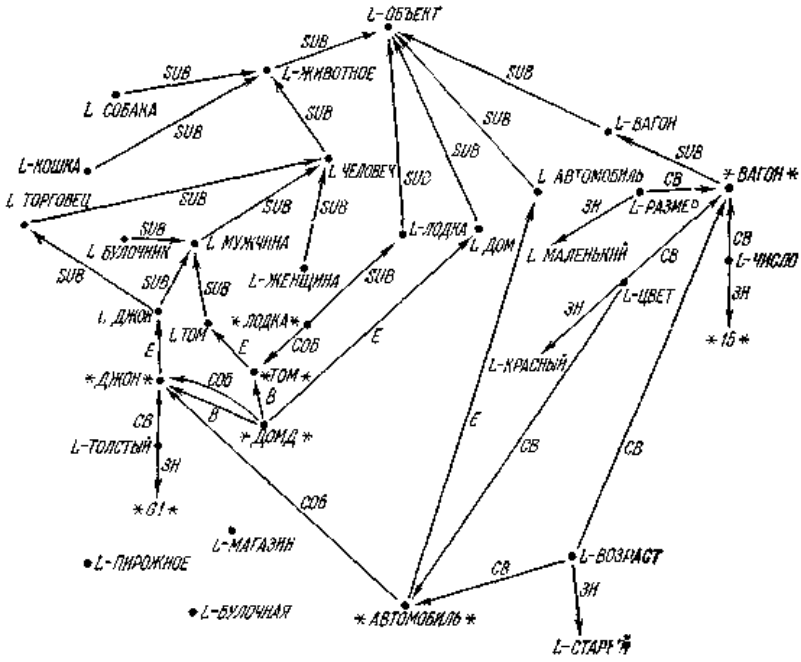


Рис. 4. Пример текущего состояния семантической сети.
SUB — ПОДМНОЖЕСТВО; Е — ЭЛЕМЕНТ; СОБ — СОБСТВЕННОСТЬ; В — НАХОДИТЬСЯ В; СВ — СВОЙСТВО; ЗН — ЗНАЧЕНИЕ.

В дальнейшем при описании сетей мы будем наравне с термином «вершины» использовать термин «узел». *Фактуальные вершины* будем представлять узлами, наименования которых начинаются и заканчиваются звездочкой (*), а *общие вершины*— узлами, наименованиям которых предшествует буква L. Рассмотрим узел *ДЖОН*, являющийся элементом L-ДЖОН и L-ТОРГОВЕЦ. L-ДЖОН является обозначением множества всех Джонов и подмножеством L-МУЖЧИНА (множества всех мужчин). L-ТОРГОВЕЦ есть множество всех торговцев. Следовательно, *ДЖОН* является узлом, представляющим конкретного человека, являющегося торговцем с именем Джон.

Этот Джон владеет конкретным домом *ДОМД* (что выражается отношением СОБСТВЕННОСТЬ). Джон находится в этом доме. Джон владеет старым красным автомобилем.

Для представления событий, происходящих во времени, используется понятие временной оси (рис. 5).

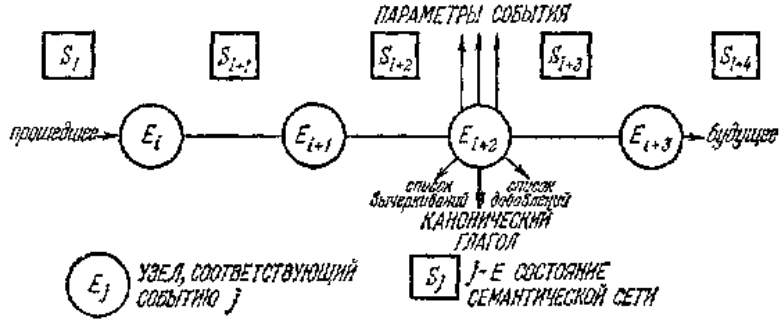


Рис. 5. Абстрактная временная ось.

Каждый узел на временной оси соответствует событию (предложению в естественном языке). В начальном состоянии сеть содержит только некоторую информацию из словаря. **Состояние сети изменяется в соответствии с последовательностью событий, происходящих на временной оси.** С каждым событием связывается список отношений (механизм связи будет определен ниже), имеющих место до совершения события и более неудовлетворяющихся после совершения события (так называемый **список вычеркивания**) и список отношений, которые не имели место до рассматриваемого события (или по крайней мере не было известно, что эти отношения выполняются), но которые удовлетворяются после выполнения события (**список добавлений**). Отметим, что появление события на временной оси изменяет не только отношения, существующие в семантической сети, но и может привести к добавлению новых узлов, если во входном предложении были объекты, отсутствовавшие в сети.

Рассмотрим в качестве примера изменения, которые вызовет в сети (см. рис. 4) событие: ДЖОН ОБМЕНЯЛ С ТОМОМ СВОЙ АВТОМОБИЛЬ НА ЛОДКУ ТОМА. Отношения (СОБСТВЕННОСТЬ *ДЖОН**АВТОМОБИЛЬ*) и (СОБСТВЕННОСТЬ *ТОМ** ЛОДКА*), существовавшие в сети на рис. 4 больше не являются истинными и должны быть заменены на отношения (СОБСТВЕННОСТЬ *ДЖОН ** ЛОДКА*) и (СОБСТВЕННОСТЬ * ТОМ* «АВТОМОБИЛЬ*). Указанные изменения в сети изображены на рис. 6 (состояние 2).

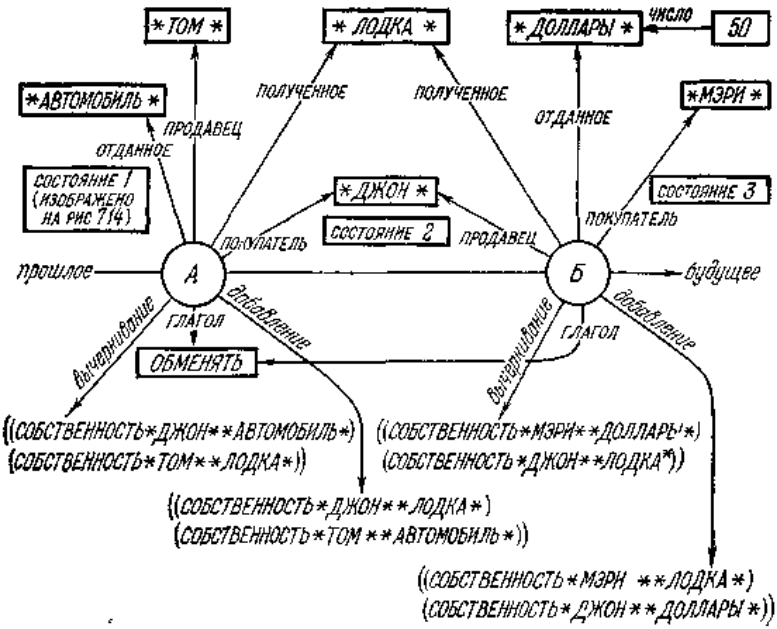


Рис. 6. Пример состояния временной оси при поступлении двух событий (вспомогательные узлы для простоты не показаны).

Так как **события преобразуют одно состояние сети в другое, можно рассматривать их как операторы**. Следуя этой трактовке, канонический глагол можно рассматривать как процедуральное событие (или оператор), которое преобразует состояние сети (являющееся неявным параметром) и множество явных параметров в новое состояние сети. Процедуральные события представляются аналогично операторам, используемым в системе STRIPS (см. п. 10.2). Неотъемлемой частью любого оператора STRIPS является список предусловий, который используется пользователем для определения законной последовательности действий. Так как в рассматриваемой системе предполагается, что события следуют в хронологической последовательности, то исчезает необходимость в списке предусловий для определения последовательности событий (операторов). Существует однако следующая проблема. Пусть на вход поступили предложения: ДЖОН КУПИЛ ЧАСЫ В МАГАЗИНЕ, ЗАТЕМ ДЖОН КУПИЛ ПИРОЖНОЕ В НОВОЙ БУЛОЧНОЙ. Заметим, что Джон покупал часы в магазине, а в следующем событии он покупал пирожное в булочной. До того как событие по покупке пирожного

может иметь место, Джон должен перейти из магазина в булочную (предусловие для события в булочной). Таким образом, мы видим, что необходимость в удовлетворении предусловиям остается даже при заданной последовательности событий. Предусловия определяют неспецифицированные вспомогательные события, которые необходимы для выполнения специфицированных (указанных в тексте) событий.

Сущность этих неспецифицированных вспомогательных событий в значительной степени определяется предусловиями специфицированных событий. Так, в приведенном примере до того, как ДЖОН КУПИЛ ПИРОЖНОЕ В НОВОЙ БУЛОЧНОЙ необходимо выполнить некоторое событие, устраняющее отношение (В*ДЖОН ** МАГАЗИН*) и добавляющее отношение (В*ДЖОН ** БУЛОЧНАЯ *). (В общем случае событию ОБМЕНЯТЬ должно предшествовать событие, переносящее участников обмена в место обмена и т. д.)

Таким образом, процедуральное событие приводит к вызову вспомогательного события и созданию на временной оси соответствующих им двух узлов. Процедуральное событие подставляет списки добавления и вычеркивания неспецифицированного события, которое преобразует текущее состояние сети в состояние, удовлетворяющее предусловиям главного события. То, что такое событие должно существовать, следует из заданной последовательности специфицированных событий.

Заметим, что описанный способ обработки процедурального события является одним из возможных вариантов представления события в виде скелета (фрейма).

Рассмотрим на примере, как осуществляется вызов и обработка процедурального события. Пусть входное предложение имеет вид:

AT THE NEW BAKERY JOHN BOUGHT A CAKE
FROM THE BAKER
(В НОВОЙ БУЛОЧНОЙ ДЖОН КУПИЛ ПИРОЖНОЕ
У БУЛОЧНИКА).

После этапа синтаксического разбора и приведения к канонической структуре оно примет вид:

(КАНОН. ГЛ. ОБМЕНЯТЬМОДАЛ. (ВРЕМЯ ПРОШ.
НАКЛ. ИЗЪЯВИТ. ПАДЕЖ УТВЕРДИТ.)
ПРОДАВЕЦ (ОБОЗН. L-БУЛОЧНИК АРТИКЛЬ
ОПРЕД. ЧИСЛО ЕД.)
ПОКУПАТЕЛЬ (ОБОЗН. L-ДЖОН ЧИСЛО ЕД.)
ПОЛУЧЕННОЕ (ОБОЗН. L-ПИРОЖНОЕ АРТИКЛЬ
НЕОПР. ЧИСЛО ЕД.)
МЕСТО (ОБОЗН. L-БУЛОЧНАЯ АРТИКЛЬ

ОПРЕД. ЧИСЛО ЕД. МОД. (ВОЗРАСТ L-НОВЫЙ))).

Канонический глагол указывает, какая процедура должна быть вызвана (табл.4).

Таблица 4

Представление процедурального события «обменять»

```
ПРОЦЕДУРА ОБМЕНЯТЬ (продавец, покупатель, полученное,
                                        отданное, место);
СПИСОК ВЫЧЕРКИВАНИЙ ВСПОМОГАТЕЛЬНОГО СОБЫТИЯ:
        ((В продавец *) (В покупатель *) (В полученное *)
         (В отданное *) (СОБСТВЕННОСТЬ * полученное)
         (СОБСТВЕННОСТЬ * отданное));
СПИСОК ДОБАВЛЕНИЙ ВСПОМОГАТЕЛЬНОГО СОБЫТИЯ:
        ((В продавец место) (В покупатель место)
         (В полученное место) (В отданное место)
         (СОБСТВЕННОСТЬ продавец полученное)
         (СОБСТВЕННОСТЬ покупатель отданное));
СПИСОК ВЫЧЕРКИВАНИЙ ГЛАВНОГО СОБЫТИЯ:
        ((СОБСТВЕННОСТЬ покупатель отданное)
         (СОБСТВЕННОСТЬ продавец полученное));
СПИСОК ДОБАВЛЕНИЙ ГЛАВНОГО СОБЫТИЯ:
        ((СОБСТВЕННОСТЬ покупатель полученное)
         (СОБСТВЕННОСТЬ продавец отданное)).
```

Параметры процедуры (возможно не все) в явном виде вырабатываются на стадии разбора. В приведенном примере это: продавец, покупатель, полученное, отданное, место. Остальным параметрам процедуры, не получившим на стадии разбора конкретных значений, присваивается значение NIL. Затем с каждым параметром должен быть связан некоторый узел сети. Если подходящего узла в сети не существует, то такой узел должен быть создан. Эту функцию выполняет специальная **программа НИС (найти или создать),** чьими аргументами являются параметры выбранной процедуры. Например, если артикль параметра является неопределенным, то НИС просто создает новый узел в сети, удовлетворяющий списку свойств параметра. Так, для А САКЕ, упомянутого во входном предложении, НИС создает новый узел, не заботясь о том, связан он или нет с некоторым понятием ПИРОЖНОЕ, существующим в сети. (Впоследствии необходимо определить различные узлы, выражающие одну и ту же сущность. Объединение этих узлов выполняется специальной функцией.) Если у параметра определенный (или специфированный) артикль, то НИС пытается найти в сети

соответствующий этому параметру узел. Если найдено больше одного узла, то выбирается тот, который упоминался последним. Если не найдено ни одного узла, то создается новый узел.

Значением свойства ОБОЗН, указанного в списке свойств параметра, является наименование узла, представляющего множество объектов. Значением параметра является элемент из этого множества. В нашем примере для параметра ПРОДАВЕЦ наименованием множества является L-БУЛОЧНИК (именующий множество всех булочников). Следовательно, значением параметра ПРОДАВЕЦ должен быть элемент из множества L-БУЛОЧНИК. Заметим, что слова типа L-БУЛОЧНИК, L-ДЖОН, L-ПИРОЖНОЕ введены и в словарь (для целей разбора), и в сеть перед началом функционирования системы. Кроме того, в сеть введены и определенные примитивные отношения между подобными словами, например, (ПОДМНОЖЕСТВО L-ДЖОН L-МУЖЧИНА). Всякий раз, когда упоминается некоторый *ДЖОН*, он становится элементом множества L-ДЖОН, являющегося в свою очередь подмножеством множества L-МУЖЧИНА.

Параметры указанного входного предложения будут обработаны следующим образом:

1) Программа НИС ищет в сети *x* такое, что для него справедливо отношение (ЭЛЕМЕНТ *x* L-БУЛОЧНИК). Не найдя такого *x,* программа создает узел * БУЛОЧНИК *, и этот узел становится значением параметра ПРОДАВЕЦ. (В течение этого процесса отношение (ЭЛЕМЕНТ * БУЛОЧНИК * L-БУЛОЧНИК) вводится в сеть).

2) Программа НИС ищет такое *x,* что (ЭЛЕМЕНТ *x* L-ДЖОН). Она находит узел «ДЖОН», который и становится значением параметра ПОКУПАТЕЛЬ.

3) В связи с тем, что у параметра ПОЛУЧЕННОЕ неопределенный артикль, НИС создает новый узел *ПИРОЖНОЕ* такой, что для него выполняется отношение (ЭЛЕМЕНТ * ПИРОЖНОЕ* L-ПИРОЖНОЕ), и он является значением параметра ПОЛУЧЕННОЕ.

4) В связи с присутствием у параметра МЕСТО модификатора (прилагательного) работа программы НИС усложняется. НИС ищет *x* такое, что (ЭЛЕМЕНТ *x* L-БУЛОЧНАЯ) и (ВОЗРАСТ НОВЫЙ *x).* Не найдя такого узла, НИС создает его (*БУЛОЧНАЯ*) и делает значением параметра МЕСТО.

Как только значения параметров определены, система вызывает процедуру ОБМЕНЯТЬ для того, чтобы обработать поступившее событие. Важно отметить, что отношения, вводимые в сеть в процессе определения значений параметров, не вводятся на списки вычеркивания и добавления узлов временной оси.

Итак, обращение к процедуральному событию (табл.4) имеет вид: ОБМЕНЯТЬ (* БУЛОЧНИК ** ДЖОН ** ПИРОЖНОЕ * NIL * БУЛОЧНАЯ *), где NIL присваивается параметру процедуры ОТДАННОЕ, отсутствующему в поступившем событии. Эффект, вызываемый обращением к процедуре, представлен на рис. 7.

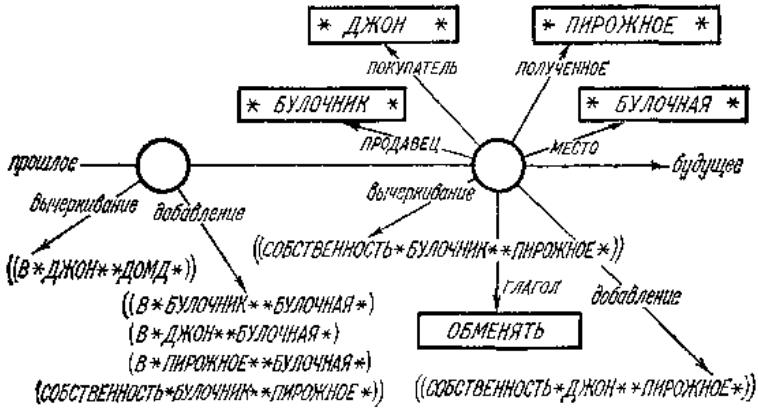


Рис. 7. Узлы временной оси, созданные процедуральным событием.

Рассмотрим этапы выполнения процедуры:

1) На временной оси создается узел, соответствующий вспомогательному событию. Так как природа события неизвестна, с узлом не связывается ни канонический глагол, ни параметры события.

2) Рассматривается список вычеркиваний вспомогательного события (табл. 4). Первым рассмотрим триплет (В ПРОДАВЕЦ *). Делается попытка сопоставить это отношение с отношениями, существующими в сети. Звездочка, стоящая на месте третьего элемента триплета, позволяет устанавливать соответствие с любым объектом сети. Параметру «продавец» при вызове процедуры сопоставлен узел *БУЛОЧНИК*. Итак, нам надо искать отношение в виде (В*БУЛОЧНИК*—). Такого отношения в сети нет, и поэтому никакие действия не предпринимаются. Следующим рассматривается отношение (В ПОКУПАТЕЛЬ*). Подстановка фактических параметров (при вызове процедуры) преобразует его в (В* ДЖОН * —). Этому отношению в сети (рис. 4) соответствует отношение (В*ДЖОН**ДОМД*). Следовательно, (так как мы рассматриваем список вычеркиваний) отношение (В * ДЖОН * * ДОМД *) вычеркивается из сети и устанавливается в список вычеркиваний узла, представляющего на временной оси вспомогательное событие. Остальным отношениям, перечисленным в списке вычеркиваний

процедуры, не удается сопоставить отношения сети, и поэтому они не вызывают никаких действий.

3) Рассматривается список добавлений вспомогательного события (табл.4). Первое рассматриваемое отношение (В ПРОДАВЕЦ МЕСТО) после подстановки фактических параметров преобразуется в отношение (В*БУЛОЧНИК ** БУЛОЧНАЯ *). Это отношение добавляется к сети и к списку добавлений вспомогательного узла временной оси. Аналогичным образом добавляется следующие отношения (В * ДЖОН ** БУЛОЧНАЯ *), (В * ПИРОЖНОЕ ** БУЛОЧНАЯ *) и (СОБСТВЕННОСТЬ*БУЛОЧНИК ** ПИРОЖНОЕ*). Триплеты, содержащие параметр «отданное», не вызывают никаких действий, так как этот параметр неопределен поступившим событием.

4) На временной оси устанавливается узел, соответствующий главному событию. С данным узлом связываются параметры процедуры ОБМЕНЯТЬ и канонический глагол ОБМЕНЯТЬ.

5) Рассматривается список вычеркиваний главного события (табл.4) и вырабатывается отношение (СОБСТВЕННОСТЬ * БУЛОЧНИК ** ПИРОЖНОЕ *), устраняемое из сети и заносимое в список вычеркиваний главного события.

6) Обрабатывается список добавлений главного события и добавляется отношение (СОБСТВЕННОСТЬ * ДЖОН * * ПИРОЖНОЕ*).

Описанной процедурой первоначальное состояние сети (рис. 4) преобразуется во вспомогательное состояние, в котором Джон и булочник находятся в булочной, и булочник владеет пирожным. Затем это вспомогательное состояние преобразуется в состояние, в котором Джон и булочник остаются в булочной, но владельцем пирожного становится Джон.

Итак, мы рассмотрели основные этапы семантической интерпретации в системах с ограниченной логикой при **представлении базы данных в виде семантической сети**. Перейдем теперь к описанию СИ при **представлении базовых данных в исчислении предикатов**.

12.2.3. Семантическая интерпретация в системах с общим выводом

Описание СИ систем с общим выводом дадим на базе вопросно-ответной системы. Основанием для выбора нами этой системы является следующее:

1) законченность системы (т. е. в системе реализованы все задачи диалога);

2) система разработана для ИИО;

3) система позволяет продемонстрировать альтернативные подходы в области грамматики, внутреннего представления и взаимосвязи этапов семантической интерпретации и синтаксического анализа (СА).

В связи с тем, чго в данной системе этапы СИ и СА выполняются параллельно (а не последовательно), описанию СИ необходимо предпослать краткие сведения о синтаксическом анализе

Синтаксический анализ системы основан на трансформационной грамматике (п. 10.2.3) подмножества английского языка, включающего повелительные, повествовательные и вопросительные предложения.

Задачи трансформационных правил состоят в преобразовании пассива в актив, вопросительных предложений в повествовательные, отображении множественных форм существительных и глаголов в единые формы.

Так, например, при обработке предложения с пассивной конструкцией «THE TALL BOX WAS PUSHED BY JOHN» (высокий ящик толкался Джоном), задачей трансформационной компоненты является определение необходимости применения трансформации (пассив→актив), которая переведет данное предложение в активную форму: «JOHN PUSHED THE TALL BOX» (Джон толкал высокий ящик). Затем к работе приступает базовая компонента. Анализатор базовой компоненты записывается в виде правил продукции, имеющих следующий вид:

*L*1*:* α |→β | γ * *L*2*;*

где *L*1 и L2— метки, α и β есть строки, → указывает операцию замещения, | — разделитель, γ есть последовательность семантических продукций, * указывает операцию «ЧТЕНИЕ» (выбора очередного слова из входной строки и размещение его на вершине стека синтаксического анализатора), ; — знак, разъединяющий продукции. *L*1, →, β, γ,* являются необязательными символами, в то время как |, α, *L*2 являются обязательными для каждой продукции. Рассмотрим, как осуществляется выполнение продукции. Предполагается, что в начале работы в верхний уровень синтаксического стека устанавливается первый (левый) символ входного предложения. Пусть управление передано продукции, помеченной меткой *L*1. Символ α правила *L*1 сопоставляется с верхними уровнями стека. Если сопоставление закончилось успешно, то:

1) часть стека, соответствующая α, заменяется на β (свободный класс переменных становится связанным в смысле исчисления предикатов);

2) выполняется последовательность γ (последовательность семантических продукций правила);

3) если * присутствует в продукции *L*1, то выполняется операция «чтения»;
4) совершается переход к продукции, с меткой *L2.*

Если сопоставление строки α правила *L*1 со стеком неудачно, то управление передается следующей продукции в последовательности. Операция сопоставления понимается в смысле сопоставления с образцом. **Образец (α-строка правила) может содержать терминальную константу, класс переменных, определенных в терминах терминальных констант или в терминах булевской комбинации других классов.** Образец может иметь вид $1, который сопоставляется с произвольной одиночной составляющей. Образец $ сопоставляется с любым числом произвольных составляющих.

Продукции трансформационной компоненты имеют такую же форму, как продукции базовой компоненты, за исключением того факта, что сопоставление осуществляется не со стеком, а с самим предложением (просмотром его слева направо). Любой элемент образца, взятый в кавычки, указывает на то, что сопоставление выполняется на уровне букв частного слова, а не на лексическом уровне. Таким образом могут быть выполнены проверки на множественное число и на стандартные суффиксы и префиксы.

Семантические продукции идентичны синтаксическим по форме и выполнению. Отличие состоит в том, что в семантических продукциях не используется операция * и, кроме того, семантические продукции выполняются на семантическом стеке.

В табл. 5 приведены примеры предложений ограниченного английского языка, используемые при обращении к отправителю.

Таблица 5

Образцы перевода предложений естественного языка в формулы исчисления предикатов

Предложения	Формулы
Повествовательные	
1) Все люди смертны	$\forall x$ (ЕСТЬ (x, человек) → ЕСТЬ (x, смертен))
2) Существуют высокие представители мужского пола, не являющиеся мальчиками	$\exists x$ (ЕСТЬ (x, высокий) ∧ ЕСТЬ (x, мужчина) ∧ ~ЕСТЬ (x, мальчик))
3) У Джона две руки	ИМЕТЬ (Джон, рука, 2)
4) Любой зеленый объект является высоким, узким ящиком	$\forall x$ (ЕСТЬ (x, зеленый) → ЕСТЬ (x, высокий) ∧ ЕСТЬ (x, узкий) ∧ ЕСТЬ (x, ящик))
5) Высокий ящик толкался Джоном	$\exists s, x, y, z$ (РАВНО (S, ТОЛКАТЬ (ДЖОН, $x, y, z, S_{нач}$)) ∧ ЕСТЬ (x, высокий) ∧ ЕСТЬ (x, ящик) ∧ ВРЕМЯ (S, прошл.))
Вопросительные	
6) Является ли Джон мужчиной?	ЕСТЬ (Джон, мужчина)
7) Сколько пальцев имеет Джон?	$\exists x$ (ИМЕТЬ (Джон, палец, x))
8) Есть ли слева от Вас какие-нибудь ящики?	$\exists x$ (ЕСТЬ (x, ящик) ∧ СЛЕВА (x, Отправ))
9) Когда вы будете толкать ящик?	$\exists s, t, x, y, z$ (РАВНО (S, ТОЛКАТЬ (Отправ., $x, y, z, S_{нач}$)) ∧ ЕСТЬ (x, ЯЩИК) ∧ ВРЕМЯ (s, t) ∧ БУДУЩЕЕ (t))
Повелительные	
10) Остановиться 11) Повернуться кругом 12) Двигаться на 10 футов	Осуществляется обращение к подпрограммам, реализующим указанные стандартные действия.
13) Идти к большой красной призме	$\exists s, x$ (У (Отправ., x, s) ∧ ЕСТЬ (x, большой) ∧ ЕСТЬ (x, красный) ∧ ЕСТЬ (x, призма))
14) Толкать черный ящик на вершину платформы	$\exists s, x, y, z$ (ТОЛКАТЬ (x, s) ∧ ЕСТЬ (x, черный) ∧ ЕСТЬ (x, ящик) ∧ НА (x, y) ∧ ЕСТЬ (y, вершина) ∧ ПРИНАДЛЕЖИТ (y, z) ∧ ЕСТЬ (z, платформа))
15) Собрать все кубы в центре комнаты	$\forall x \exists s, y, z$ (В (x, y, s) ∧ ЕСТЬ (x, куб) ∧ ЕСТЬ (y, центр) ∧ ПРИНАДЛЕЖИТ (y, z) ∧ ЕСТЬ (z, комната))
16) Исследовать комнату Джона	$\exists s, x$ (ИССЛЕДОВАТЬ (x, s) ∧ ЕСТЬ (x, комната) ∧ ПРИНАДЛЕЖИТ (x, Джон))

Для удобства читателей предложения переведены на русский язык. В таблице 5 приведены формулы исчисления предикатов, соответствующие предложениям естественного языка.

Повествовательные предложения (примеры 1-5 табл.5) переводятся в аксиомы системы, вопросительные предложения (примеры 6—9 табл.5) переводятся в утверждения, которые должны быть доказаны вопросно-ответной системой. Простые повелительные предложения (примеры 10-12 табл.5) непосредственно переводятся в наименования стандартных программ, вызывающих выполнение отправителем определенных действий. Сложные повелительные предложения (примеры 13-16 табл.5) рассматриваются системой как утверждения, возможность выполнения которых должна быть определена, т. е. они аналогичны вопросам.

Большинство предикатов, введенных в примерах, очевидны, однако, для ясности рассмотрим один пример (уже упоминавшийся ранее): «Высокий ящик толкался Джоном» (THE TALL BOX WAS PUSHED BY JOHN). Синтаксическая компонента, получив данное предложение, определяет, что к предложению необходимо применить трансформацию, переводящую его в активную форму «Джон толкал высокий ящик» (JOHN PUSHED THE TALL BOX). Затем базовая компонента распознает, что глагол «толкать» использован в предложении в прошедшем времени, и устанавливает в соответствующее значение предикат ВРЕМЯ. Прилагательное «высокий» и существительное «ящик» представляются в предикате ЕСТЬ. **Полученная в результате формула может быть проинтерпретирована примерно так. Существуют состояние S, объект *x* и координаты *y* и *z* такие, что S есть состояние, полученное из начального ($S_{нач}$) применением к нему оператора (Джон толкает объект *x* из координат *y* и *z*), причем *x* относится к классу высоких объектов и к классу ящиков. Кроме того, состояние S имело место в прошлом.**

Способ представления предложений в виде формул исчисления предикатов не является единственно возможным.

Сэндуолл указывает три возможных способа. Проиллюстрируем их на примере предложения «Джон продал лодку Мэри».

1) Глагол используется как предикатный символ. После трансляции входное предложение будет иметь вид ПРОДАЛ (Джон, Лодка, Мэри), где ПРОДАЛ является трехместным отношением, а Джон, Лодка, Мэри — константы. Если мы хотим указать дополнительные сведения о происходящем событии (например, время, место), то это может быть сделано введением предиката, содержащего большее количество аргументов.

2) Главным отличием этой нотации является использование небольшого множества базовых отношений и получение на их основе более сложных отношений благодаря введению понятия «события».

Рассматриваемое предложение после трансляции может иметь вид:

(∃ еПОДЛЕЖАЩЕЕ (Джон, е) ∧СКАЗУЕМОЕ (продал, е) ∧ ДОПОЛНЕНИЕ (лодка, е) ∧ КОСВ.

ДОПОЛН. (Мэри, е) ∧ РЕАЛЬНОСТЬ (е)),

где «е» интерпретируется как событие, в результате которого «Джон продал лодку Мэри». Первые четыре отношения используют для того, чтобы описать данное событие, а предикат РЕАЛЬНОСТЬ используется для того, чтобы определить, что указанное событие фактически имеет место. В данной нотации можно представить и более сложные предложения, например: «Джон верит, что Петр продал лодку Мэри».

(∃ е ∃ р ПОДЛЕЖАЩЕЕ (Джон, е) ∧ СКАЗУЕМОЕ (верящий, е) ∧ ДОПОЛНЕНИЕ (р, е)
∧ РЕАЛЬНОСТЬ (е) ∧ ПОДЛЕЖАЩЕЕ (Петр, р) ∧
СКАЗУЕМОЕ (продавший, р) ∧ ДОПОЛНЕНИЕ
(Лодка, р) ∧ КОСВ. ДОПОЛН. (Мэри, р)),

где «р» интерпретируется как событие: «Петр продал лодку Мэри».

Предикаты, используемые для описания события, существенно зависят от конкретной системы, так, если в системе используются глубинные падежи (см. п. 10.2.3), то вместо приведенных в примере предикатов ПОДЛЕЖАЩЕЕ, СКАЗУЕМОЕ, ДОПОЛНЕНИЕ и т. д. будут использоваться предикаты, отражающие глубинные отношения.

3) Третий способ отличается тем, что отношения между словами предложения выражаются не предикатами, а **функциональными символами.**

ЕСТЬ (Джон, Кому Юбъект (продавший, лодку), Мэри)).

Данное выражение можно представить в виде бинарного дерева (рис. 8), отражающего структуру предложения.

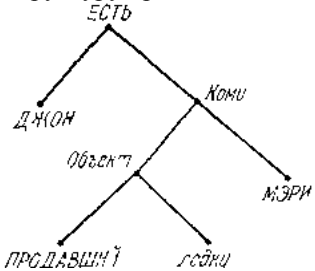


Рис 8. Представление структуры предложения в виде бинарного дерева.

В данном методе (так же как и в предыдущем) используются объекты (Джон, лодка, Мэри) и свойства (продавший). Отношение ЕСТЬ определено на **двух аргументах: первый — объект, второй — свойства**. Функции «Объект», «Кому» являются двухместными. **Первый аргумент этих функций должен быть свойством, второй объектом, а результат является свойством**. Подобным образом могут быть определены более сложные высказывания, содержащие кванторы, сравнительную степень прилагательных, придаточные предложения и т. п.

Первый метод широко используется. Причины его популярности вызваны тем, что для людей является естественной идентификация **понятия «предикат» из традиционной грамматики с понятием «предикат» в исчислении предикатов**. Однако этот метод имеет и очевидные недостатки. В нем затруднено выражение ссылок на предшествующий контекст, необходимо введение большого числа предикатов и функций при описании объектов реальной сложности.

Второй метод используется в нескольких вопросно-ответных системах. Этот метод близок к падежной грамматике Филмора (см. п. 10.2.3). Этот подход позволяет одному предложению ссылаться на другое и не требует введения большого числа предикатов.

Третий метод имеет некоторое сходство с глубинными структурами трансформационных грамматик (см. п. 10.2.3). Можно найти большое сходство между деревом на рис. 8 и традиционным С-маркером для предложения в естественном языке.

Второй и третий методы имеют преимущества перед первым и много общего между собой. Они вводят относительно немного функций и отношений, которые являются базовыми для **отражения лингвистического и концептуального состава языка.**

Итак, можно подытожить, что исчисление предикатов обладает достаточной выразительной мощностью для представления «глубинной структуры» предложений естественного языка.

Следует отметить различие в понимании термина «глубинные структуры» в лингвистике и при использовании в качестве глубинных структур формул исчисления предикатов. **В лингвистике глубинная структура есть результат синтаксического анализа входного предложения до его объединения с базовыми данными, а в исчислении предикатов — после объединения.**

Основным достоинством использования исчисления предикатов в качестве глубинных структур является существование для них универсальных вычислительных методов.

Для реализации конкретной системы с использованием исчисления предикатов необходимо решить две задачи:

1. Сформировать каноническое множество предикатов и функций, достаточных для описания среды, в которой будет функционировать ИСФ. Множество должно быть достаточно малым, но полным.
2. Сформулировать аксиомы, выражающие свойства понятий, введенных в пункте 1.

В таблице 6 приведен экспериментальный список основных понятий, используемых для описания существенных свойств действий.

Таблица 6

Каноническое множество предикатов, используемых для описания действия

Предикаты	Вопросы
1. АГЕНТ (*x*, *y*)	Кто совершил действие?
2. ДЕЙСТВИЕ (*x*, *y*)	Что он сделал?
3. ОБЪЕКТ (*x*, *y*)	Для кого или чего он это сделал?
4. Описание окружения:	В каком контексте он это сделал?
ВРЕМЯ (*x*, *y*)	Когда он это сделал?
МЕСТО (*x*, *y*)	Где он это сделал?
ОБСТОЯТЕЛЬСТВО (*x*, *y*)	При каких обстоятельствах он это сделал?
5. Модальность:	Как он это сделал?
СРЕДСТВО (*x*, *y*)	С помощью какого инструмента или метода он это сделал?
СПОСОБ (*x*, *y*)	С помощью чего он это сделал?
6. Причинность:	Почему он это сделал?
ПРИЧИНА (*x*, *y*)	Какая причина побудила его сделать это?
ЦЕЛЬ (*x*, *y*)	С какими намерениями он это сделал?
СОСТОЯНИЕ (*x*, *y*)	В каком состоянии (настроении) он это сделал?

Для того, чтобы назначение конкретного предиката было более понятно, рядом с предикатом приведены вопросы в естественном языке. Этот список в принципе не полон, но для заданных целей он достаточен. Аксиомы ИИО должны описывать, по крайней мере, информацию трех видов.

1. Геометрические отношения окружающей среды.

2. Правила, описывающие ограничения на возможности ИИО воспринимать окружающую среду и и выполнять манипуляции в ней.
3. Информацию, извлеченную из повествовательных предложений естественного языка, полученную в ходе диалога ИИО с человеком.

Следует отметить, что формулировка аксиом является довольно сложной задачей**. Можно предложить следующие неформальные шаги при выполнении аксиоматизации.**

1. Записать наблюдения об окружении, которое мы хотим описать. Выразить все в корректном исчислении предикатов.

2. Проверить на непротиворечивость.

3. Проверить на избыточность, т. е. проверить, не выводимы ли некоторые аксиомы из других аксиом?.

Следует отметить, что наличие избыточности может ускорить вывод, поэтому добиваться отсутствия избыточности не всегда целесообразно.

4. Проверить на полноту, т. е. проверить, введены ли все аксиомы, необходимые для вывода. Общий метод осуществления проверки на полноту состоит в подборе проверяющих примеров (теорем, которые должны быть выводимы в предлагаемой аксиоматике) и проверке вручную, что каждая из них может быть доказана.

Проверяющие примеры могут быть «позитивные» и «негативные». Позитивные примеры — это те, которые используются при прямом выводе. Их задача — показать, что то, что должно быть выводимо, выводится. Негативные примеры должны показывать, что то, что не должно быть выводимо, выводится.

Следует отметить, что **предложенные шаги имеют много общего с отладкой программы на вычислительных машинах**. Однако аксиомы проще записывать, отлаживать и объединять в общую систему.

Остановимся теперь на вопросе объединения некоторого факта (повествовательное предложение), переведенного в формулу исчисления предикатов, с базовыми данными системы.

Объединить новый факт (*Ф*) с базовыми аксиомами *(А)* можно только в том случае, если этот факт не противоречит аксиомам.

Заметим, что если исходное предложение имеет после синтаксического и семантического этапа более одной интерпретации, то из них выбирается та, которая не противоречит базовым аксиомам (если такая интерпретация существует) .

Проверка на непротиворечивость состоит в определении невыполнимости множества $A \bigcup \sim\Phi$, где A — исходное множество аксиом, $\sim\Phi$ — отрицание объединяемого факта. Практически во всех системах дедуктивного вывода, основанных на исчислении

предикатов, для определения невыполнимости используются **модификации принципа резолюции.**

При определении невыполнимости возможны три исхода.

1. Множество формул $A \bigcup \sim\Phi$ невыполнимо (получен пустой дизъюнкт), что обозначает выводимость формулы Φ из базовых аксиом.

2. Множество формул $A \bigcup \sim\Phi$ выполнимо ($R_{i+1}=R(R_i(A))$), что означает невыводимость формулы Φ из аксиом.

3. Процедура резолюции не дает ответа за время, отведенное на доказательство (из-за зацикливания или из-за ограниченности времени).

В первом случае утверждение Φ, несмотря на его выводимость из A, иногда целесообразно добавить к A, так это может ускорить получение некоторых выводов.

Второй случай обозначает, что Φ невыводима в A. Известно, что

если замкнутая формула Ф теории А невыводима в А, то теория А', полученная из А добавлением ~Ф в качестве аксиомы, непротиворечива.

На основании этого утверждения формула $\sim \Phi$ может быть добавлена к A.

Если произошел третий исход, то мы можем попытаться проверить выводимость ~Ф из A (т. е. установить невыполнимость $A \bigcup \Phi$). При этом опять же возможны три указанных исхода. Если имеет место первый случай, то это значит, что формула $\sim \Phi$ выводима из A и, следовательно, Φ противоречит A. В этом случае вопрос о раскрытии противоречия должен решать человек.

Второй случай указывает на то, что утверждение Φ должно быть добавлено к A.

Если имеет место третий случай, то это обозначает, что система не в состоянии установить (либо в принципе, либо в отведенное время), противоречит ли Φ исходному множеству A. Вероятно, наиболее разумное решение — выдать сообщение об этом факте и ждать указаний о присоединении (или неприсоединении) формулы Φ к исходному множеству аксиом.

12.3. Вывод ответа

12.3.1. Доказательство и извлечение ответа в системах с общим выводом

Входом для этапа дедуктивного вывода является некоторая **формула *Ф* исчисления предикатов, полученная из вопросительного или повелительного предложения исходного языка и рассматриваемая как теорема, подлежащая доказательству.** Задачей этапа является определение, следует ли данная теорема из базового множества аксиом *(А),* то есть определение невыполнимости множества $A \cup \sim Ф$. Как уже было указано в п. 10.5.3, при определении невыполнимости по методу резолюции возможны три исхода:

1) $A \cup \sim Ф$ невыполнимо, следовательно, *Ф* выводима из *А;*
2) $A \cup \sim Ф$ выполнимо, т. е. *Ф* не следует из *А*;
3) процедура не дает ответа.

Первый исход соответствует утвердительному ответу на поставленный вопрос. Второй исход соответствует ответу «для вывода недостаточно информации». Третий исход соответствует ответу «не знаю». В случае третьего исхода мы можем попытаться получить иной ответ на поставленный вопрос, либо увеличив время, отведенное системе на доказательство, либо выводя следование $\sim Ф$ из *А*, путем определения невыполнимости множества формул $A \cup Ф$. В последнем случае опять возможны три исхода. Если установлена невыполнимость $A \cup Ф$, т.е. $\sim Ф$ следует из *А*, на поставленный вопрос надо отвечать отрицательно. Выполнимость $A \cup Ф$, как и прежде соответствует ответу «для вывода недостаточно информации». В связи с ограниченностью времени, отводимого на доказательство, и неразрешимостью исчисления предикатов, процедура доказательства может не дать ответа (третий исход). Часто одно из доказательств будет найдено, и ответ на вопрос получен.

Остановимся теперь на том, как извлечь из доказываемой теоремы информацию, являющуюся развернутым ответом на вопрос.

Для извлечения ответа требуется **преобразовать граф опровержения с пустым дизъюнктом в корне в граф, называемый *модифицированным графом,* у которого в корне находится некоторое утверждение, могущее служить ответом. Преобразование состоит в том, что каждое предложение, возникающее из отрицания теоремы (часто называемой**

предположением), превращается в тавтологию (путем добавления к нему отрицания этого предложения). Затем в соответствии с первоначальной структурой графа опровержения выполняются те же самые резолюции, что и раньше, до тех пор пока в корне не будет получено некоторое утверждение.

Так как при указанном преобразовании каждое предложение, возникающее из отрицания предположения, превращается в тавтологию, то модифицированный граф представляет собой доказательство того факта, что утверждение, расположенное в его корне, **логически следует из аксиом и тавтологий**. Поэтому оно также следует только из одних аксиом.

Приведем простейший пример, поясняющий принцип извлечения ответа.

Пример 1. Пусть высказаны следующие утверждения: «Куб номер 2 находится всегда в том же месте, где куб номер 1» и «Куб номер 1 находится в месте *а»*. И задан вопрос: «Где находится куб номер 2?».

Указанные предложения после перевода их в исчисление предикатов примут вид аксиом:

$\forall x$ (В(Куб1, x)→В(Куб2, x)) аксиома 1

В (Куб1, *а)* аксиома 2

и предположения: $\exists x$В(Куб2, x), которое должно быть выведено из имеющихся фактов.

В приведенном примере предикату *В* придана интерпретация «находится в определенном месте».

На рис. 9 приведен граф опровержения для нашего примера.

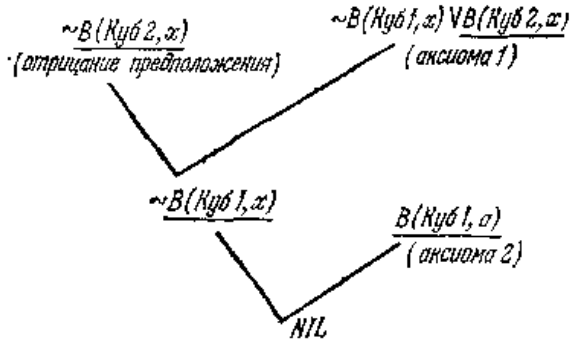


Рис. 9. Граф опровержения для примера 1.

Граф опровержения строится обычным путем.

Сначала отрицается формула, которую предстоит доказать. В рассматриваемом случае это приведет к формуле $\forall x(\sim$В (Куб2, $x))$. Затем это отрицание добавляется к множеству аксиом *(S),* и все члены этого расширенного множества преобразуются в форму предложений.
Далее с помощью принципа резолюции показывается, что это множество неудовлетворимо.
Будем выделять литеры, подвергающиеся унификации при образовании каждой резольвенты в графе опровержения. На рис. 9 эти литеры подчеркнуты. Подмножество литер в предложении, подвергающееся унификации в процессе построения графа опровержения, назовем *множеством унификации.* Каждая резольвента в модифицированном графе опирается на множества унификации, которые в точности соответствуют множествам унификации исходного графа опровержения.
Для извлечения ответа из этого графа построен модифицированный граф доказательства (рис. 20).

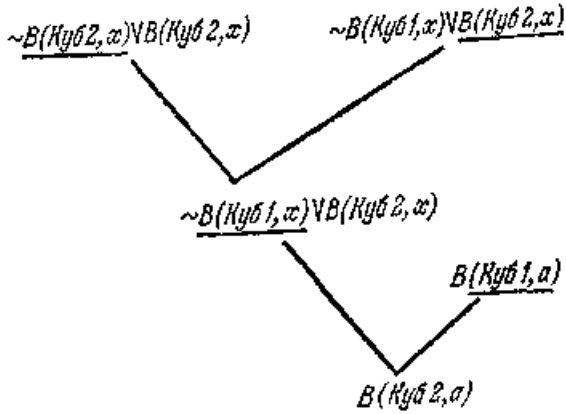


Рис. 20. Модифицированный граф доказательства для примера 1.

Смысл модификации графа состоит в том, чтобы извлечь тот частный случай переменной, относящейся к квантору существования, который служит ответом на вопрос. В нашем примере мы в качестве ответа получаем предложение В (Куб2, *а),* которое переводится в обычную форму исчисления предикатов (из формы предложений). Затем эта формула переводится в естественный язык, например, в виде предложения «Куб номер 2 находится в месте *а*».
Хотя указанный метод и прост, но у него есть несколько тонких моментов, которые мы разъясним на примерах.

Пример 2. Покажем, как преобразовать в тавтологию более сложные предложения, возникающие при отрицании предположения. Пусть в виде предложений дано следующее множество аксиом:

$$\sim A(x) \vee F(x) \vee G(f(x)),$$
$$\sim F(x) \vee B(x),$$
$$\sim F(x) \vee C(x),$$
$$\sim G(x) \vee B(x),$$
$$\sim G(x) \vee D(x),$$
$$A(g(x)) \vee F(h(x)).$$

Требуется доказать, исходя из этих аксиом, предположение $(\exists x \exists y((B(x) \wedge C(x)) \vee (D(y) \wedge B(y))))$.
Отрицание предположения переводит его в два дизъюнкта:

$$\sim B(x) \vee \sim C(x),$$
$$\sim B(x) \vee \sim D(x).$$

На рис. 21 приведен граф опровержения для примера 2.

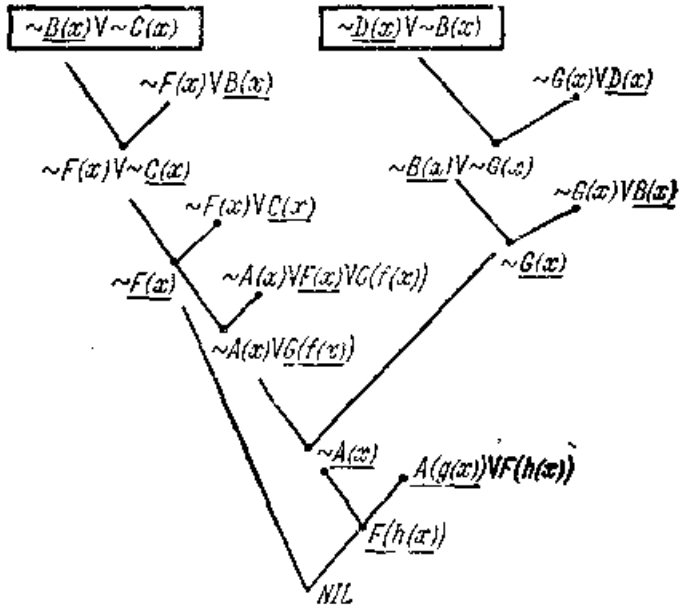


Рис. 21. Граф опровержения для примера 2.

Для преобразования графа следует превратить предложения, соответствующие отрицанию предположения, в тавтологии, добавив к ним их отрицания. Отрицания в данном случае (рис. 22**) не являются предложениями (дизъюнктами), так как в них содержатся конъюнкции.** Однако мы будем обращаться с этими конъюнкциями как с единой литерой и действовать формально, как будто наша формула является дизъюнктом. Мы можем себе это позволить, так как ни один из элементов рассматриваемой конъюнкции не может оказаться ни в одном множестве унификации.

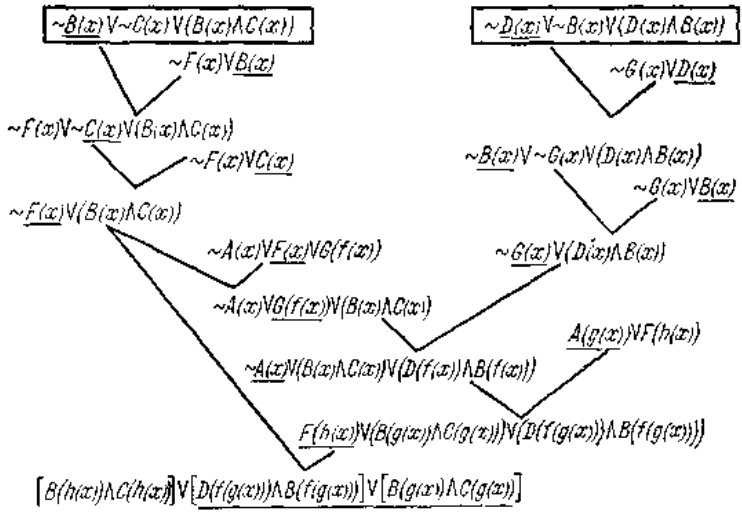


Рис. 22. Модифицированный граф доказательства для примера 2.

На рис. 22 приведен модифицированный граф. Мы видим, что ответное утверждение, расположенное в корневой вершине имеет форму, подобную форме предположения.

В общем случае, если предположение высказано в дизъюнктивной нормальной форме, то утверждение, получаемое в процессе извлечения ответа, представляет собой дизъюнкцию выражений, каждое из которых имеет форму либо всего предположения, либо одного или нескольких дизъюнктов этого предположения. Потому мы и говорим, что такое утверждение можно использовать в качестве «ответа» на вопрос, представляемый исходным предположением.

В случае, когда предположение, которое следует доказать, содержит переменные, относящиеся к квантору всеобщности, возникают дополнительные трудности. При отрицании такие переменные,

переходят в переменные, относящиеся к квантору существования, а это приводит к необходимости введения **сколемовых функций**. Возникает вопрос: «как интерпретировать эти функции, если они появляются в качестве термов в ответном утверждении»?.

Если мы будем применять описанный выше метод модификации к предположениям, содержащим квантор всеобщности, то мы будем получать ответы частного вида.

Пример 3. Пусть задана аксиома ($\forall x \forall u(P(x, u, x) \vee P(a, u, u))$) и требуется доказать предположение ($\exists w \forall v \exists y P(w, v, y)$).

На рис. 23, *а* представлен граф опровержения для примера 3. Граф доказательства, построенный по указанному ранее способу, приведен на рис. 23, *б*. Ответ, полученный в графе доказательства, имеет частный вид.

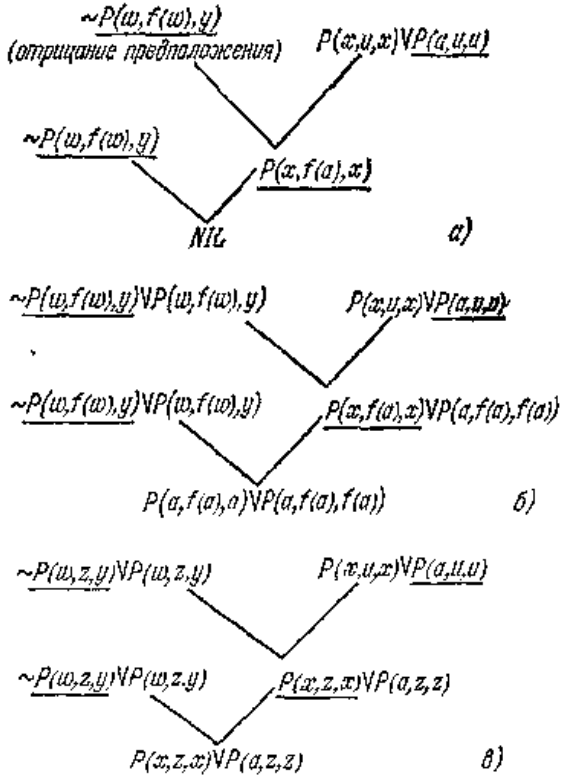


Рис. 23. Графы доказательств для примера 3.

Действительно, если из некоторого множества аксиом A мы можем доказать формулу $P(f(x))$, используя принцип резолюции, то это значит, что $A \rightarrow (\forall x P(f(x)))$ является теоремой. Если f не встречается в A, то формула $A \rightarrow (\forall f \forall x P(f(x)))$ также доказуема и, следовательно, $A \rightarrow (\forall z P(z))$ есть теорема. Применив указанные действия к ответу на рис. 23, *б* получим $(\forall z(P(a, z, a) \vee P(a, z, z)))$. Однако и этот ответ не является наиболее полным. Можно показать, что в процессе извлечения ответа всегда можно заменять сколемовые функции, возникающие при отрицании предположения, новыми переменными.

В модифицированном доказательстве в эти новые переменные не будет делаться никаких подстановок, так что они пройдут через доказательство без изменения и появятся в окончательном ответном утверждении.

На рис. 23, *в* приведен соответствующий граф доказательства. Ответ имеет вид $(\forall x \forall z(P(x, z, x) \vee P(a, z, z)))$.

В заключение перечислим основные этапы извлечения ответа.

1. Построение графа опровержения и выделение в нем множества унификации.
2. Подстановка новых переменных на место сколемовых функций, появляющихся в предложениях, образованных из отрицания предположения.
3. Преобразование предложений, полученных из отрицания предположения, в тавтологии.
4. Построение модифицированного графа доказательства, следуя структуре исходного графа опровержения. При образовании каждой резольвенты модифицированного графа используется множество унификации, определяемое множеством унификации графа опровержения.
5. Предложение, находящееся в корневой вершине модифицированного графа, представляет собой ответное утверждение, извлеченное в ходе описываемого процесса.

Очевидно, что ответное утверждение, построенное предложенным способом, зависит от опровержения, из которого оно было извлечено. Для одной и той же задачи может существовать несколько различных опровержений, и обычно у нас нет возможности установить, будет ли ответное утверждение, построенное на его основе, наиболее общим. **Эта трудность представляет, вероятно, лишь теоретический интерес, так как ответы, получаемые работающими программами, являются вполне удовлетворительными.**

Итак, мы рассмотрели способ получения ответа в системах с общим выводом, теоретически не накладывающих ограничений на тематику диалога. Однако в связи с тем, что существующие методы семантической интерпретации **требуют заранее задать множество предикатов и функций, определяющих тематику диалога**, и в связи с ограниченностью времени и памяти, выделяемых на получение ответа, системы с общим выводом на практике не обладают универсальностью. Как следствие этого факта, системы с ограниченной логикой, не обладающие универсальным выводом, оказываются на практике конкурентоспособными по сравнению с системами с общим выводом. В следующем параграфе мы рассмотрим на примере системы с ограниченной логикой, введенной в п. 11.2.2, методы извлечения ответа из семантической сети.

12.3.2. Вывод ответа в системах с ограниченной логикой

Системы с ограниченной логикой не имеют универсальных процедур вывода, и их тематика задается при создании систем разработкой процедур, отвечающих на требуемые типы вопросов. Так, например, система, описанная в п. 12.2.2, отвечает на вопросы трех типов. В указанной системе базовые данные представлены в виде семантической сети, отражающей текущее состояние внутреннего мира системы, и в виде последовательности событий на временной оси, отражающей факты, сообщенные системе (рис. 15).

Для ответа в момент E_j на вопрос об отношениях, существовавших между узлами семантической сети в некоторый прошедший момент E_k, $k<j$, система должна вернуть сеть в состояние, соответствующее событию E_k. Возврат осуществляется путем движения по временной оси в направлении от события E_j к событию E_k. При этом, при прохождении через каждое событие E_i, $k \le i < j$—1, отношения, перечисленные в списке добавлений этого события, устраняются из семантической сети, а отношения, перечисленные в списке вычеркиваний этого события, добавляются к сети. После ответа на вопрос семантическая сеть возвращается в состояние, соответствующее событию E_j.

Приведем примеры типов вопросов, на которые умеет отвечать система. Примером первого типа вопросов является: КАКОЙ МУЖЧИНА КУПИЛ ПИРОЖНОЕ В БУЛОЧНОЙ? Ответ на этот вопрос приводит к исследованию временной оси. Для приведенного примера осуществляется поиск по временной оси в направлении прошлого до тех пор, пока не будет найдено событие, определяемое

каноническим глаголом ОБМЕНЯТЬ. При этом параметр МЕСТО должен иметь значение *БУЛОЧНАЯ*.

Когда такой узел найден, проверяется, является ли значением параметра ПОЛУЧЕННОЕ элемент из множества L-ПИРОЖНОЕ. Если условие выполняется, то проверяется, является ли значением параметра ПОКУПАТЕЛЬ элемент из множества L-МУЖЧИНА. Если и это условие выполняется, то узел, являющийся значением параметра ПОКУПАТЕЛЬ, принимается в качестве основы для ответа на вопрос. Ответ формируется либо на основании этого узла (это соответствует неполному ответу), либо на основании узла-события, являющегося родителем данного узла. Подробности формирования ответа приведены в п. 12.4.

Примером вопроса второго типа является предложение: ЧТО БЫЛО СОБСТВЕННОСТЬЮ БУЛОЧНИКА ДО ТОГО, КАК ДЖОН КУПИЛ ЧТО-ТО В БУЛОЧНОЙ? В результате разбора предложение будет разделено на две части. Часть, соответствующая событию «ДЖОН КУПИЛ ЧТО-ТО В БУЛОЧНОЙ» обрабатывается аналогично вопросу первого типа, т. е. сеть возвращается в состояние, непосредственно предшествующее этому событию. Затем система ищет в полученной сети такое *x,* что для него справедливо отношение (СОБСТВЕННОСТЬ * БУЛОЧНИК * *x).* Если такое *x* обнаруживается, то оно используется при формировании ответа на вопрос.

Примером вопросов третьего типа является предложение «ЧТО ЯВЛЯЛОСЬ СОБСТВЕННОСТЬЮ БУЛОЧНИКА ДО ТОГО, КАК ПИРОЖНОЕ СТАЛО СОБСТВЕННОСТЬЮ ДЖОНА?». Для того чтобы ответить на вопрос этого типа, сеть возвращается назад до тех пор, пока не будет получено состояние, в котором истинны отношения (СОБСТВЕННОСТЬ * ДЖОН* *x)* и (ЭЛЕМЕНТ *x* L-ПИРОЖНОЕ). Затем производится движение по временной оси в прошлое до тех пор, пока одно из этих отношений не перестанет быть истинным, а отношение (СОБСТВЕННОСТЬ x БУЛОЧНИК * *x)* не становится истинным для некоторого *x.* Если такое *x* найдено, то оно используется для ответа на вопрос.

На этом мы закончим рассмотрение методов получения ответов на вопросы. Напомним, что ответы, полученные на данном этапе, выражены во внутреннем представлении системы, и поэтому нашей очередной и заключительной задачей является перевод их во внешнее представление.

12.4. Формирование ответа в ограниченном естественном языке

Задача данного этапа заключается в переводе смысла некоторого высказывания, выраженного во внутреннем языке системы, в предложение, составленное по правилам выходного языка.

Методы, используемые на данном этапе, подобны методам на этапе синтаксического анализа с той только разницей, что здесь мы на основании грамматики **порождаем предложение, а не распознаем его**. В частности, представление в виде расширенных сетей переходов, используемое при синтаксическом анализе, с успехом применяется в большинстве систем при формировании ответа.

Поясним детали, возникающие при формировании ответа, на примере системы, приведенной в п. 12.2.2.

В результате работы этапа дедуктивного вывода, описанного в предыдущем параграфе, происходит выбор узла в семантической сети, определяющего смысл формируемого высказывания. **Управление формированием ответа осуществляется на основе словарной информации и грамматики выходного языка, представленной в РАСП.** В словаре у каждого **канонического глагола есть свойство, значением которого является список,** содержащий соответствующие ему поверхностные глаголы. Кроме того, каждый поверхностный глагол имеет *Ф*-правила (см. табл.3), соотносящие глубинные структуры синтаксическим образам поверхностного глагола.

Проиллюстрируем эти свойства на примере сети, приведенной на рис. 16. Пусть выбран узел, отмеченный на рисунке буквой Б. Событие основывается на каноническом глаголе ОБМЕНЯТЬ, которому в словаре соответствуют несколько поверхностных глаголов (купить, продать, заплатить, стоить). Выбор одного из этих глаголов может быть осуществлен любым желаемым образом (возможен случайный выбор). Предположим, что выбран глагол КУПИТЬ. С глаголом «купить» в словаре связано два *Ф*-правила. Пусть выбрано первое правило

(BUYER ACTIVE THINGBT (FROM SELLER) (FOR
THINGGIVEN))
(ПОКУПАТЕЛЬ АКТИВНЫЙ ПОЛУЧЕННОЕ (У
ПРОДАВЦА) (ЗА ОТДАННОЕ)).

Это правило становится управляющим «предложением», которое должно быть разобрано генерирующей грамматикой.

Первый элемент в правиле указывает, что ему в соответствие должен быть поставлен узел, связанный с событием обменять (узел Б на рис.

16) глубинным отношением ПОКУПАТЕЛЬ. Этим узлом является *МЭРИ*. Итак, первым словом выходного предложения является МЭРИ. Второй элемент указывает на активный залог предложения. Пусть сформирован глагол КУПИЛ (BOUGHT). Третий элемент правила ПОЛУЧЕННОЕ указывает, что существительное должно быть сформировано на основании узла, удовлетворяющего отношению ПОЛУЧЕННОЕ в состоянии Б. Этому узлу соответствует слово ЛОДКА (THE BOAT).

Очередным элементом является (У ПРОДАВЦА). Наличие скобок в элементе указывает, что включение в выходную строку соответствующего ему существительного возможно, но не обязательно. В нашем случае результатом включения будет У ДЖОНА (FROM JOHN). Последний элемент в правиле (FOR THINGGIVEN) будет связан с узлом *ДОЛЛАРЫ* и приведет к генерации в выходной строке конструкции ЗА 50 ДОЛЛАРОВ (FOR 50 DOLLARS). Итак Ф-правило разобрано полностью, и на выходе получено предложение: MARY BOUGHT THE BOAT (FROM JOHN) (FOR 50 DOLLARS) (МЭРИ КУПИЛА ЛОДКУ (У ДЖОНА) (ЗА 50 ДОЛЛАРОВ)). Напомним, что конструкции в скобках не являются обязательными в выходном предложении. Не вдаваясь в детали, отметим, чго, выбрав в качестве поверхностного глагола PAY (ПЛАТИТЬ), мы бы сформировали предложение MARY PAID JOHN 50 DOLLARS FOR THE BOAT (МЭРИ ЗАПЛАТИЛА ДЖОНУ 50 ДОЛЛАРОВ ЗА ЛОДКУ).

Выбрав глагол COST (СТОИТЬ), мы бы получили предложение THE BOAT COST MARY 50 DOLLARS (ЛОДКА СТОИЛА МЭРИ 50 ДОЛЛАРОВ). В разговорной речи большинство ответов на вопросы не являются полными предложениями. Грамматика, основанная на РАСП, позволяет начинать работу с любого узла в РАСП, что приводит к возможности генерировать не обязательно все предложение, а например, только группу существительного.

Рассмотрим пример на рис. 16. Простейшим ответом на вопрос «WHO SOLD THE BOAT?» (КТО ПРОДАЛ ЛОДКУ?) является ответ «ДЖОН».

Система работает таким образом, что если в качестве узла, на базе которого строится выходное предложение, выбран узел на временной оси, то ответ является полным предложением. **Если выбран узел в сети, соответствующий существительному (например, *ВАГОН*, *ДЖОН* на рис. 14), то это существительное может быть использовано для получения неполного ответа.** Узлу *ВАГОН* соответствует конкретный объект в реальном мире, известный как «вагон» (точнее множество из пятнадцати вагонов), с которым связаны

значения: СТАРЫЙ, 15, КРАСНЫЙ, МАЛЕНЬКИЙ, являющиеся свойствами; ВОЗРАСТ, ЧИСЛО, ЦВЕТ, РАЗМЕР. Утверждение об узле *ВАГОН* во многом подобно узлу-событию. Но между этими узлами существует важное различие: **события происходят, совершаются, а утверждения являются статическими, неизменными до тех пор, пока не произойдет событие, изменяющее их истинность.**

Следует отметить, что случайная генерация модификаторов слова *ВАГОН* приводит к построению неправильных конструкций типа: КРАСНЫЙ, 15 ВАГОНОВ. Необходимо осуществить упорядочение модификаторов. Это может быть выполнено управляющей строкой словаря, руководящей генерацией группы существительного способом, аналогичным используемому при управлении генерацией предложения. Приемлемой строкой управления является (ЧИСЛО, РАЗМЕР, ВОЗРАСТ, ЦВЕТ).

В заключение отметим, что использование в качестве внутреннего представления исчисления предикатов не оказывает существенного влияния на изложенный метод формирования ответа. Справедливость этого замечания становится очевидной, если однозначно сопоставить каждой формуле исчисления предикатов некоторый граф.

13. Компьютеризация науки искусстеного интеллекта, ее проблемы и следствия

Развитие современных теорий предполагает анализ и осмысление фундаментальных изменений, происходящих в науке, культуре и образовании в связи с широким внедрением компьютерных технологий и персональных компьютеров. Обращение к этой проблеме будет осуществлено лишь в той мере, в какой позволит рассмотреть новые возможности изучения знания и «знания о знании», а также выявить новые способы описания присутствия человека и социокультурной составляющей в разных формах «представления знания». Реализовать это возможно, опираясь на исследования в области когнитивной науки – базовой науки искусственного интелекта, где **знание и информация являются главным предметом**. Представление знания, как оно исследуется в когнитивной науке, не только предполагает предметное его содержание, но и определяет интерпретативную деятельность субъекта, социокультурную обусловленность его знания и поведения, а также фиксирует другие связи и отношения, представленные в традиционных эпистемологических структурах лишь опосредованно.

13.1. Эпистемология и когнитивная наука

Когнитивная наука (когнитология) сформировалась в 60-70-х годах XX века (Гарвард, США) в качестве дисциплины, исследующей методом компьютерного моделирования функционирования знаний в интеллектуальных системах. Когнитивную науку отличают междисциплинарность, использование компьютерной метафоры и исследование познания. Центральным для всей проблематики когнитивной науки является обращение к компьютеру, служащему самой наглядной и самой убедительной моделью того, как формируется, структурируется и «работает» знание, а также имитируются различные когнитивные процессы (например, обучения или получения экспертного знания и т. п.). Феномен знания исследуется в аспектах его получения, хранения, переработки, коммуницированния (передачи), выясняется, какими типами знания и в какой форме обладает человек, как «представлено» знание в его голове и как он его использует.

Важную роль играет лингвистика, которая выступает для когнитивной науки как важный источник материала об устройстве когнитивных структур. По отношению к искусственному интеллекту когнитология является своего рода «теорией интеллектуальных машин и механизмов», т. е. сконструированных человеком компьютерных устройств и лишь через них — их естественных прообразов - людей познающих. Это объясняет различную природу эксперимента в психологии и когнитивной науке и определяет существование в последней компьютерной метафоры.

Традиционные проблемы гносеологии, эпистемологии, философии и методологии науки получили новое видение и интерпретацию. «Когнитивизм знаменовал появление новой парадигмы научного знания, и с ним в историю науки пришло новое понимание того, как следует изучать знание, как можно подойти к проблеме непосредственно не наблюдаемого — прежде всего к проблеме внутреннего представления мира в голове человека...» (Кубрякова Е.С., Демьянков В.З. и др. Краткий словарь когнитивных терминов. М., 1996. С. 61). Эксплицитно выраженные знания составляют лишь незначительную часть общей базы знаний человека. Согласно современным подходам, такая база есть самоорганизующаяся и саморегулируемая система. Она включает следующие компоненты:

- языковые знания — грамматика (с фонетикой и фонологией), дополненная знанием композиционной и лексической семантики;

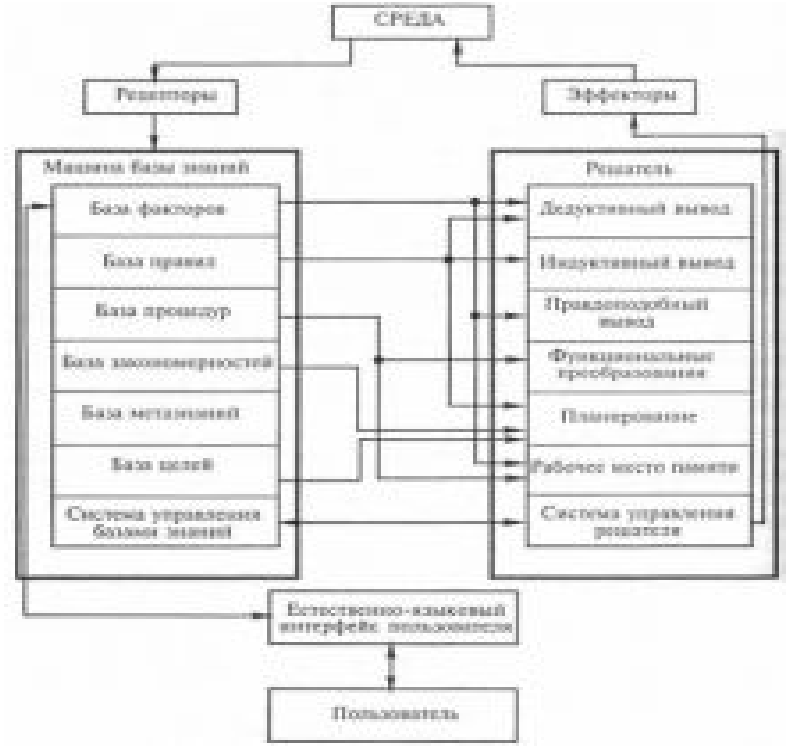
- знание об употреблении языка;
- знание принципов речевого обучения;
- внеязыковые знания — о контексте описываемой ситуации, об адресате коммуникации (в том числе знание о поставленных адресатом целях и планах, его представления о говорящем, об окружающей обстановке, знание своих умений);
- общефоновое знание, т. е. личностная картина мира.

При соотнесении эпистемологии и когнитивной науки необходимо различать **знание и информацию**, что упрощенно можно свести к формуле: **информация — это знание минус человек; информация — знаковая оболочка знания.** Под **компьютерным представлением знания принято понимать информацию, хранимую в компьютере, формализованную в соответствии с определенными структурными правилами, которые компьютер может автономно использовать при решении проблем с помощью заложенных в нем алгоритмов типа логического вывода.** Информационная модель знания (как записанная в компьютере, так и вербализованная в тексте) является лишь намеком на представленное знание, по которому человек способен творчески воссоздать само знание. **Следует отметить принципиальное отличие той информации, которая служит для получения знаний человеком, от информации, изучаемой в теории информации**. Когнитивное знание открывает человеку дополнительные возможности размышления и действия, увеличивает его свободу. **Информация как управляющий сигнал уменьшает неопределенность допускаемых состояний управляемой системы. Знание — личное достояние знающих, перенимающих его друг у друга как образцы действия в процессах познания.** Этого нельзя сказать об информации, которая в противоположность знанию не является достоянием конкретной личности, она равно доступна всем, хотя возможности превратить ее в знание у каждого свои, опирающиеся на личный опыт и способности.

Такое различие создается исключительно присутствием человека, способного извлечь из информации, записанной на бумаге или закодированной в компьютере, нечто, позволяющее реализовать человеческую свободу выбора. Пользователь получает представление о ряде возможных точек зрения, соответственно, возникает та самая неопределенность, которая является необходимой предпосылкой для выбора.

Отличие традиционной гносеологии от разделов теории познания, имеющих дело с использованием компьютеров, состоит в том, что первая

концентрируется на процедуре описания, обращаясь к высказываниям и правилам для получения знания. **«Компьютерная» теория познания делает центром своего внимания регуляцию, обращается к нормативным предложениям, использует знания для продуцирования правил.** Сегодня развитие теории познания классическими гносеологическими средствами не всегда возможно, изменяются инструментарий гносеолога, требования к его профессиональной подготовке. Философия становится дисциплиной, сопричастной экспериментальной деятельности, осуществляемой при разработке программ искусственного интеллекта. Выяснилось, что именно в этой сфере возможна проверка самых тонких и абстрактных гипотез о природе человеческого разума. Сегодня здесь на первый план вышла проблема порождения знания, и это потребовало пересмотра базовых концепций ИИ. По ходу поиска обнаружилось, что идеи Локка Лейбница, Канта, Гуссерля, Хайдеггера — это концептуальные модели, которые могут быть «экспериментально» проверены в рамках программы ИИ, что позволяет по-новому решать философские споры о природе разума и познания. Так, по существу, экспериментально была доказана несостоятельность представления Локка о душе как «чистой доске» (tabula rasa), на которой только опыт записывает какое-либо содержание. Выяснилось, что возможности универсальных распознающих устройств, не имеющих тех эмпирических знаний-предпосылок, которыми обладает любой человек, ограничены. На самом деле «чистая доска», на которую записываются данные опыта, — те «образы», которые необходимо распознать иногда только по намеку, как это делает человек, — должна иметь весьма сложную структуру, включающую множество априорных знаний о мире. При разработке программ ИИ экспериментально подтвердилась также огромная роль скрытых, неявных знаний, не выраженных в языке, но хранящих в себе жизненный опыт. **В литературе достаточно определенно высказывается мнение о том, что в настоящее время эксперименты на компьютерах, а не на людях — самый верный шаг на пути проверки гипотез о мышлении и познании, различных аналогов мыслительной деятельности.**



Такие эксперименты, разумеется, не могут рассматриваться в качестве полного доказательства предлагаемых гипотез, но принимаются как серьезный аргумент в их пользу. Программа искусственного интеллекта как своего рода «экспериментальная философия» делает фигуру эпистемолога столь же необходимой для компьютерной эпохи, как и математика-программиста. Эпистемология впервые за всю историю получает прямой выход в сферу конструктивной инженерной и технологической деятельности. Меняется характер связи эпистемологии с практикой.

За относительно короткий период своего существования компьютеры стали орудием труда миллионов людей. Без преувеличения можно сказать, что результатами, полученными с помощью вычислительных машин, пользуется все человечество, несмотря на то, что большинство компьютеров как средство решения задач пока не очень удобны для многих работников. Прежде всего это относится к задачам, решение которых носит человеко-машинный характер. Отличительной

особенностью процесса решения таких задач является так называемый режим разделения времени (оперативный режим,- режим *on-line),* при котором компьютер реагирует на отдельные события реальной окружающей среды столь быстро, что позволяет воздействовать на их ход. При этом события регистрируются посредством данных, вводимых с терминалов; данные, относящиеся к отдельному событию, обрабатываются до полного или частичного завершения операций над ними независимо от других входных сообщений.

В ряде случаев режим разделения времени (РРВ) имеет неоспоримые преимущества по сравнению с пакетным (*off-line* или *batch*): пользователю не нужно программировать различные ветви алгоритма в зависимости от промежуточных результатов счета; не нужно просчитывать все допустимые варианты, требующие значительного времени и ресурсов; создаются психологически важные предпосылки для «пробных ходов» при разработке алгоритма решения сложной задачи и т. п. В целом, как правило, РРВ значительно сокращает общее время решения задач самых различных классов, а также создает более благоприятные условия для пользователей низкой квалификации. Здесь прежде всего имеется в виду возможность проверки и исправления «на месте» текста формируемой программы, вводимых пользователем исходных данных и т. п.

Расширение областей применения вычислительной техники, массовое привлечение к решению задач с помощью ЭВМ пользователей-непрофессионалов требуют развития режима разделения времени за счет создания обслуживающей и обучающей среды, помогающей пользователю уточнить формулировку его задачи, разработать метод и алгоритм ее решения, построить программу, не изучая детально язык программирования и т. п. Компьютер должен снабжать пользователя необходимыми сведениями по интересующей его задаче, по возможным средствам ее решения, а также эффективно обучать пользователя применению этих средств.

В условиях постоянного развития информационных и алгоритмических возможностей ЭВМ указанные обслуживание и обучение, осуществляемые компьютером, необходимы не только для пользователей-новичков, но и для профессиональных программистов, желающих эффективно применять в своей работе новые программы и данные.

Реализация автоматизированного обслуживания и обучения пользователей позволяет развить режимы пакета и разделения времени в новую технологию решения задач с помощью компьютеров, при которой компьютер будет играть более активную и интеллектуальную роль во всем процессе взаимодействия с человеком, будет выступать как

«помощник» и постоянный «учитель» пользователей различных категорий.

Реализация новой технологии решения задач позволит существенно повысить эффективность взаимодействия человека и ЭВМ, облегчить процесс использования автоматизированных систем переработки информации.

Как показывает имеющийся опыт, даже простейшая обучающая программа для пользователей некоторого пакета прикладных программ:

1) существенно облегчает и ускоряет эксплуатацию и внедрение этого пакета;
2) освобождает разработчиков пакета от подготовки пользователей, оставляя за разработчиками только вводное ознакомление и консультации по тому материалу, который не может быть усвоен с помощью обучающей программы;
3) облегчает работу подразделения сопровождения, которое может не вникать во все тонкости конкретного программного продукта, обеспечивая только его содержание в соответствии с инструкцией по эксплуатации.

Рассмотрим гипотетический вариант работы на оснащенной комплексом обслуживающих и обучающих средств ЭВМ пользователя, который первый раз обратился к машине со своей задачей. Контакт с машиной начинается с ознакомления пользователя с элементарными манипуляциями на абонентском пункте, имеющем, к примеру, алфавитно-цифровой термирал. Обучающая программа разъясняет пользователю назначение клавиш, а также показывает, какие основные действия ему придется выполнить для решения своей задачи. После этого (автоматически или по команде пользователя) подключается некая программа —РЕДАКТОР формулировки задачи. При этом, по желанию пользователя, редактирование может вестись с обучением или без обучения. После работы с РЕДАКТОРОМ пользователь может вызвать автоматизированную информационную систему для того, чтобы попытаться найти готовую программу решения своей задачи. Если такая программа имеется, пользаватель должен подготовить исходные данные (самостоятельно или с подсказкой со стороны машины) и, если результат решения ему не нужен немедленно, перевести свою задачу в пакетный режим.

Если готовая программа не найдена, пользователю предлагается написать ее либо самостоятельно, либо с помощью программы — РАЗРАБОТЧИКА алгоритмов. Разрабатывание алгоритма может вестись с пошаговым выполнением (для этого вызывается соответствующий транслятор или интерпретатор) либо с выполнением всей программы. В этом последнем случае задача также может быть переведена в пакетный

режим.
В процессе программирования или разрабатывания алгоритмов пользователь может требовать от машины разъяснений, заказать себе более или менее развитый режим обучения и тренировки на своих же задачах. Он может также получать от ЭВМ справки по встречающимся объектам и средствам решения его задачи.
По мере освоения машинных средств решения своего класса задач пользователь может отказаться от помощи машины в программировании и разрабатывания алгоритмов и попытаться программировать самостоятельно. Вначале, вероятнее всего, он будет это делать в режиме разделения времени, но по мере овладения языком программирования и приобретения устойчивых навыков в работе с этим языком он все чаще и чаще будет обращаться к пакетному режиму. При этом по автоматизированному каталогу алгоритмов и программ он будет следить, не появилась ли готовая программа решения его очередной задачи.
К РЕДАКТОРУ или РАЗРАБОТЧИКУ алгоритмов пользователь будет обращаться в тех случаях, когда ему не захочется размышлять над очередной задачей или когда его устраивает рутинная процедура ее решения, а к обучающей или справочной системе — в тех случаях, когда он что-то забыл либо хочет освоить некоторые новые средства решения задач своего класса.
Рассмотренный пример иллюстрирует желаемое для пользователя сочетание режима разделения времени с автоматизированным обслуживанием и обучением пользователя и с пакетным режимом. Эти дополнительные удобства для пользователя должны обеспечивать в конечном счете более высокую эффективность всего процесса решения задач с точки зрения времени, стоимости, надежности решения и т. п.
Переходы от одного режима к другому, основанные на управлении ресурсами вычислительной системы, исходя из характеристик и запросов пользователей, не позволяют заранее определить, какие, например, части программы решения задачи будут построены пользователем самостоятельно, а какие — под управлением машины. Такое динамическое распределение равноценных функций, а также постоянное «доучивание» партнера напоминает диалог между людьми при решении одной и той же задачи. Здесь ЭВМ может «претендовать» на роль равноценного партнера, имеющего определенные знания о предмете диалога, а также умеющего обеспечивать выполнение такого сравнительно интеллектуального действия как, например, помощь человеку в создании программы решения его задачи.

13.2. Процесс решения задачи с и использованием компьютера

Уточним прежде всего понятие взаимодействия человека с вычислительной машиной, для чего введем два вспомогательных определения.

Предложением назовем последовательность символов (слов), заканчивающуюся некоторым символом (например, точкой),, отождествляемым с концом предложения.

Сообщением обычно называют цепочку предложений (возможно одно предложение или отдельный символ), заканчивающуюся символом конца сообщения. Предложения и символы, составляющие отдельное сообщение, обычно принадлежат некоторому языку.

Взаимодействие человеку с вычислительной машиной будем рассматривать как процесс обмена сообщениями между человеком и ЭВМ, обусловленный необходимостью последовательного и (или) параллельного выполнения человеком и машиной операций по решению какой-либо задачи.

При взаимодействии с пользователем системы разделения времени (СРВ) обычно обеспечивают:

1) непосредственный контакт между пользователем и системой, т. е. прием и выдачу разнообразных сообщений посредством локального или удаленного терминала;
2) немедленную обработку системой принятых сообщений, даже если эта обработка выполняется не полностью; эта функция обычно включает проверку синтаксической правильности сообщения, исполнение команд, содержащихся в сообщении, и вывод результатов;
3) поиск необходимых пользователю данных и (или) программ, т. е. управление библиотекой программ и данными;
4) возможность практически одновременно обслуживать нескольких пользователей в условиях, когда потребность в обслуживании непредсказуема.

Кроме того, программные средства СРВ обычно обеспечивают выполнение следующих функций, направленных на поддержание целостности и эффективности системы: планирование прохождения задач; обработка ошибок; текущий контроль за работой системы; закрытие, восстановление и повторный пуск системы и др.

С целью конкретизации процедур обслуживания пользователей выделим в процессе решения задачи с помощью ЭВМ следующие основные действия:

А) образование формулировки (условия) задачи;

Б) конструирование программы, включающее:
Б0) уточнение формулировки задачи, т. е. описание данных и требования в некоторой стандартизованной форме;
Б1) поиск аналогичной формулировки задачи с готовой программой ее решения;
Б2) разработка плана решения задачи, т. е. определение путей, метода ее решения и т. п.;
БЗ) описание и ввод плана или его фрагментов в терминах входного языка программирования;
В) проверка синтаксической и семантической правильности программы или ее фрагмента;
Г) осуществление решения:
Г1) подготовка и ввод исходных данных;
Г2) исполнение программы или ее фрагмента («счет»);
ГЗ) вывод результатов;
Д) проверка правильности результатов решения задачи, итоговый или конечный контроль.
Представление процесса решения задачи на ЭВМ в виде последовательности действия А—Д — очень грубое приближение к реальному процессу. Гораздо больше соответствует действительности выделение контуров типа, показанного на рис. 1:
контур КД — изменение описания алгоритма после синтаксической и семантической проверки программы или ее фрагмента (коррекция программы на уровне входного языка);
контур К2 — изменение формулировки (условия) задачи и переконструирование алгоритма после неудовлетворительного счета с введенными исходными данными.
Действие Е, замыкающее обратные связи на рис. 1 — коррекция (изменение, дополнение, редактирование) — основывается на выполнении системой человек — ЭВМ действий контроля В и Д.

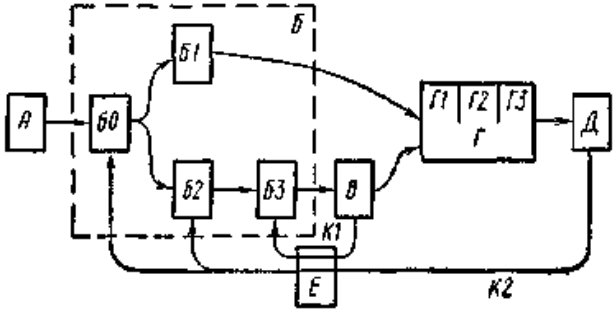


Рис 1.

Две основные «технологии» решения задач с помощью ЭВМ — пакетный режим и режим разделения времени — различаются между собой тем, как соотносятся во времени группа действий Б с группой действий В, Г и Д.

При пакетном режиме группа Б отделена от действий В, Г и Д более или менее значительным интервалом времени, который не зависит от особенностей решения конкретной задачи, а определяется порядком выдачи результатов проверки в счете, установленным на данной ЭВМ для всех задач. Если этот интервал составляет часы, то пользователь стремится написать полную и, по возможности, синтаксически правильную программу решения своей задачи, не обращаясь к ЭВМ с целью проверки фрагментов программы и пробного счета по этим фрагментам. При этом зачастую нарушается естественный, удобный для человека ход решения задачи (это происходит потому, что первоначальная задача принудительно разбивается на две, в значительной мере самостоятельные, подзадачи: 1) уточнение формулировки и конструирование алгоритма и программы (действие Б) и 2) проверка и выполнение полученной программы (действия В, Г и Д)), а многие задачи вообще не могут быть решены (особенно неопытным пользователем).

При режиме разделения времени пользователь имеет непосредственную связь с вычислительной машиной через индивидуальный терминал или абонентский пункт — телетайп, дисплей — и должен получать результаты выполнения действий В и Г достаточно быстро, как правило, через интервал времени, не нарушающий естественный ход его мысли. Именно при режиме реального времени контуры К1 и К2 обмена сообщениями, выделенные на рис. 1, имеют ярко выраженный, «интенсивный» характер, что обеспечивает, прежде всего, возможность пошагового конструирования и выполнения алгоритма, оперативной проверки возникающих гипотез и вариантов решения.

В соответствии с выделенными действиями по решению задачи с помощью ЭВМ будем различать процедуры обслуживания пользователей СРВ при подготовке и вводе данных в готовые программы; программировании, т. е. описании и вводе плана решения задачи в терминах входного языка программирования; конструировании программы, включая уточнение формулировки задачи.

Автоматизация обслуживания пользователей СРВ заключается либо в передаче соответствующего действия вычислительной машине, либо в совместном выполнении такого действия и пользователем и ЭВМ. Первый подход развивается многими исследователями в рамках проблематики искусственного интеллекта. Здесь разработаны системы запрос — ответ, которые позволяют пользователям получать

необходимые результаты, обращаясь к готовым программам и базам данных на языке, близком к естественному

13.3. Основные виды систем разделения времени

Получили распространение следующие основные типы СРВ:
1) системы запрос — ответ или автоматизированные информационные системы (АИС), осуществляющие поиск, обработку необходимых пользователю данных по готовым программам;
2) системы сбора данных и обновления файлов, включая постоянное и временное обновление;
3) специализированные («диалоговые») системы программирования;
4) управления технологическими процессами;
5) «полуавтоматизированного» решения сложных задач в ситуации, когда в определенных узловых пунктах решения выбор дальнейшего направления может быть предоставлен пользователю.
Наиболее полно действия по решению задач с помощью ЭВМ, выделенные в п.13.2, представлены в специализированных «диалоговых» системах программирования. Число таких систем, разработанных к настоящему времени, весьма обширно и разнообразно. Среди них имеются как «разговорные» реализации традиционных языков программирования, так и системы, реализующие специальные языки оперативного взаимодействия.
Общую структуру СРВ второго типа можно представить с помощью рис. 2 .

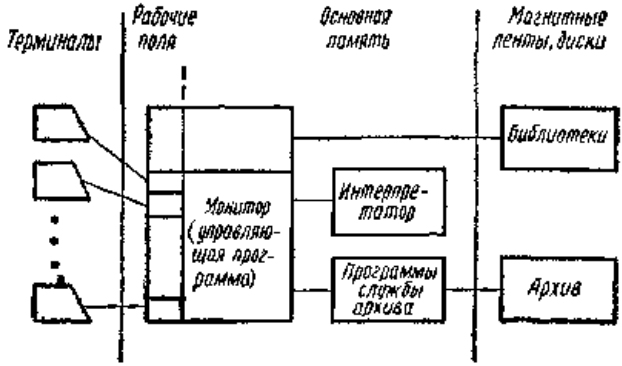


Рис. 2.

Такую систему можно разделить на четыре основные части: монитор, интерпретатор, архив и библиотеки — личные и общего пользования. Монитор обеспечивает информационный обмен с терминалами, выделяет участки (основной) памяти под рабочие поля пользователей, распределяет время интерпретатора. Интерпретатор состоит из набора программ, выполняющих директивы или системные команды, а также программ, реализующих конструкции языка программирования. В библиотеках размещаются написанные на входном языке пользователями (в том числе и разработчиками системы) программы, тексты, информационные массивы и т. п. Кроме индивидуальных библиотек, имеется библиотека общего пользования, доступная всем пользователям системы. В такой библиотеке хранятся тренирующие и обучающие программы.

13.4. Организация помощи пользователю в системах разделения времени

Возможность быстрого обмена сообщениями между пользователем и ЭВМ уже само по себе зачастую служит достаточно мощным «обучающим» фактором. Однако в СРВ применяются и специальные методы по усилению обучающих возможностей этих режимов. Приведем краткий анализ наиболее распространенных из этих методов.

В СРВ типа запрос — ответ или типа АИС взаимодействие с пользователем осуществляется либо на языке, близком к естественному, либо путем заполнения пользователем форматов, предъявляемых машиной, или выбора из «меню». Такая организация взаимодействия существенно облегчает обуче ние или, точнее, научение и работу с системой для случайных пользователей или пользователей, не обладающих специальными навыками работы с автоматизированными средствами обработки информации. Непосредственный и быстрый обмен сообщениями позволяет применить гибкие схемы отбора и итеративного поиска, а в.случае применения ограниченного естественного языка — снимать смысловые неоднозначности и неточности и т. п..

Развитие АИС и приближение их к пользователю-непрофессионалу идет в направлении некоторой формализации языка запросов с одновременной организацией помощи и обучения пользователя по правилам описания запроса во входном языке (с контролем правильности этой записи и развернутой диагностикой допущенных ошибок) и по степени и характеру влияния на конечный результат поиска изменения отдельных параметров и формулировки запроса.

Техника выбора из «меню», применяемая в АИС, также развивается в направлении инициируемого машиной информирования пользователя о

тематике базы данных, типах допустимых запросов и соотнесенных с этими типами средних показателях полноты, точности и времени поиска. Это позволяет освободить пользователя от предварительного изучения мнемоники языка запросов, а также добиваться тех же практических результатов с меньшими затратами ресурсов машины.

В СРВ типа специализированной системы программирования обучение пользователей и оказание им помощи обычно реализуются с помощью библиотеки общего доступа, хранящей справочную информацию, тренирующие и обучающие программы. Обращение к такой библиотеке выполняется с помощью директив типа ОПИШИ (объект), УЧИТЬ, ТРЕНИРОВАТЬ, ПОМОГИ и др. Эти директивы обычно применяются пользователем в тех случаях, когда он не знает формата или назначения какого-либо оператора входного языка программирования. По директивам ОПИШИ, ПОМОГИ вызывается описание соответствующего оператора, а по директивам типа УЧИТЬ, ТРЕНИРОВАТЬ — обучающая или тренирующая программа.

Целям обучения пользователей служит также так называемая встроенная документация прикладных программ.

13.5. Автоматизированные обучающие системы

Дальнейшее развитие средств оказания помощи пользователям СРВ, рассмотренных в 13.3, приводит к необходимости включения в нее автоматизированной обучающей системы (АОС):

1) предъявляющей пользователю учебный материал вместе с контрольными заданиями и вопросами;
2) осуществляющей контроль правильности выполнения заданий и ответов на вопросы;
3) выдающей необходимые разъяснения, подсказки и помощь на основе сбора и обработки статистических данных об индивидуальных особенностях пользователей-обучаемых.

Актуальность использования ЭВМ для автоматизированного обучения не только пользователей, но и студентов вузов и техникумов, учащихся средних школ и профессионально-технических училищ подтверждает как зарубежный, так и отечественный опыт.

Как показывают проведенные исследования, применение АОС позволяет повысить качество подготовки специалистов за счет индивидуализации обучения, возможности управления познавательным процессом по гибкой, адаптирующейся к характеристикам обучаемых программе, освобождения преподавателей от ряда трудоемких учебных операций нетворческого типа, унифицирования учебных курсов на уровне лучших

образцов, применения прогрессивных форм самостоятельной работы, наконец, предоставления различного рода услуг, присущих средствам вычислительной техники.

Внедрение АОС в систему образования происходит вместе с развитием новых методов в обучении и управлении учебным процессом: программированного обучения, стандартизированного контроля знаний, обучающих машин, научной организации труда в учебном заведении. Выпускается широкий ассортимент терминалов, в состав которых, кроме обычных средств связи с ЭВМ, могут быть включены аудио-визуальные средства обучения: слайд- и кинопроекторы, экраны для демонстрации движущихся изображений, звуковая аппаратура и др.

Разработаны и реализованы проблемно-ориентированные языки, позволяющие преподавателям, которые не имеют специальной подготовки в области программирования и вычислительной техники, написать на языке, достаточно простом и в то же время мощном, обучающие и контролирующие программы, реализующие основные идеи программированного обучения, стандартизированного и тестового контроля знаний.

На этих языках, а также на языках программирования написаны значительные библиотеки обучающих и контролирующих программ.

В качестве примера объема внедрения АОС в учебных заведениях США может служить опыт школ графства Монтгомери (штат Мериленд). В трех школах графства установлены терминалы, которые обслуживает региональная ЭВМ учебного назначения. В этой системе имеется около 40 учебных курсов по геометрии, арифметике, алгебре, физике, географии, основам вычислительной техники и др. Большинство курсов написано учителями этих же школ, которые для повышения квалификации сами обучались на АОС.

АОС, не предназначенными специально для обучения пользователей ЭВМ, являются, например, АОС для обучения дифференцированию и интегрированию.

Следует указать, что работа обучаемых с АОС может протекать в двух режимах:

1) режиме разделения времени; в иностранной литературе такие АОС относят к классу CAI (Computer-Assisted Instruction) или CAL(Computer-Assisted Learning), т.е. «обучение с помощью ЭВМ»;

2) режиме пакета — системы называются CMI (Computer-Managed Instruction), т. е. «обучение, направляемое ЭВМ».

Оценивая накопленный опыт использования АОС, отметим его положительные стороны, выражающиеся в некотором сокращении общего времени обучения и в повышении его качества за счет:

возможностей оперативного и документального контроля и

наблюдения за деятельностью обучаемых, что позволяет преподавателю или обучающей программе принимать более адекватные дидактические решения, вовремя оказывать помощь отстающим и т. п.:

создания условий для самостоятельной учебной работы обучаемого с возможностью выбора индивидуального темпа обучения и уровня помощи;

использования в учебных целях вычислительных и моделирующих возможностей ЭВМ.

Развитие АОС связывается с решением следующих трех проблем:

1) более полного использования ЭВМ как средства решения задач (в том числе учебных), стоящих перед обучаемым;
2) создания достаточно объемных учебных курсов, позволяющих возложить на машину более значительную часть учебной работы с обучаемыми;
3) дальнейшего углубления индивидуализации и помощи в обучении как по запросам обучаемого, так и автоматически — на основе анализа его модели.

Решение первой проблемы следует прежде всего привязывать к обучению с помощью ЭВМ пользователей, при котором машина выступает одновременно **как средство обучения и как объект изучения**, что позволяет, в частности, при обучении программированию осуществлять развитый синтаксический и семантический контроль ответов обучаемого с указанием необходимых действий по локализации и устранению ошибок и интерпретацию сообщений обучаемого (в том числе — ошибочных ответов с целью показать последствия допущенных ошибок).

Вторая проблема, связанная с реализацией на ЭВМ достаточно объемного учебного материала — целых курсов, больших разделов учебной дисциплины (можно сказать «больших обучающих программ»), позволяющих возложить на машину более значительную часть работы с обучаемыми, пока далека от своего решения. Это объясняется тем, что в настоящее время все еще формируется «инструментарий» для создания больших обучающих программ. Реализованные в ряде АОС языки написания обучающих курсов, из-за ограниченности средств не всегда позволяют создавать достаточно «интеллектуальные» большие обучающие программы.

По поводу решения третьей проблемы известный американский специалист в области АОС П. Суппес сказал следующее: «По существу принципиальные трудности, стоящие на пути обучения с помощью машин, имеют не технический, а педагогический характер: не известно, каковы должны быть методы индивидуального обучения и как должна быть составлена учебная программа, рассчитанная не на группу, а на

отдельного учащегося... И мы до сих пор не имеем никакой по-настоящему отчетливой научной идеи о том, до какой степени индивидуализации можно довести обучение. По-видимому, понадобится определенное время, прежде чем этой проблеме удастся найти достаточно глубокое обоснование». Как видно, корни решения этой проблемы находятся в области педагогики, психологии и зависят от создания средств формализованного описания и проектирования различных видов обучающих программ.

Большинство CAI- и CMI-систем, более или менее широко внедренных в практику учебной работы, либо обеспечивают полное управление обучаемым, либо дают ему возможность управлять ходом процесса обучения с помощью фиксированного языка директив (директивы ПОМОГИ, ПОВТОРИТЬ и т. п.).

В области АОС ведутся исследования, направленные на создание режимов обучения, приближающихся по своей форме к диалогу между учеником и учителем-репетитором. Примером такого режима является обучение с помощью АОС, у которой обучаемый может запросить на языке, близком к естественному, учебный материал, справку, помощь, в которых он, по его мнению, нуждается.

Для оценки перспективности создания развитых АОС представляет, интерес сравнение диалогового обучения с обучением, полностью управляемым вычислительной машиной. Такое сравнение было выполнено, например, в Лидском университете (Англия) на материале обучения планированию сбора и статистической обработки экспериментальных данных.

Контрольная группа студентов отвечала на вопросы и выполняла задания обучающей программы, имеющей три уровня трудности.

АОС автоматически переводила обучаемого с одного уровня трудности на другой в зависимости от его успехов.

Экспериментальной группе студентов, работающей в режиме диалогового обучения, сообщалось о типах контрольных вопросов, на которые они должны будут ответить, для того чтобы отстоять свой план эксперимента. Чтобы ответить на эти вопросы, студенты имели доступ к информации, размещенной в связанных между собой файлах. Студенту предлагалась «карта» («меню») из этих файлов, так чтобы он сам мог выбрать помощь и обучение по разделу знаний, который ему нужен. Например, он мог потребовать обучить его свойствам статистических тестов и затем самостоятельно проверить, какая группа тестов наилучше подходит к его случаю. В определенных пределах студент мог свободно двигаться между указанными файлами, но в конце он обязательно должен был пройти экзамен, где без всякой помощи он должен был показать, что его план эксперимента удовлетворяет определенным

требованиям.

Проведенные исследования показали более высокую эффективность обучения как первой, так и второй методики по сравнению с традиционным обучением, основанным на использовании обычного печатного руководства по планированию экспериментов. Для более сложных процедур планирования методика, управляемая машиной, оказалась более эффективной (хотя и требовала больше времени). Это связано, видимо, с тем, что в этих случаях студенты сами, без руководства не могли адекватно использовать систему предлагаемых им файлов.

Но по мере приобретения опыта, а также для более простых случаев диалоговая методика оказалась более эффективной. Кроме того, она позволяла сократить время обучения, и студенты предпочитали работать по ней, а не по методике, задаваемой вычислительной машиной.

Немногочисленные диалоговые АОС находятся на этапе экспериментального опробования и не получили пока широкого распространения. По нашему мнению, это прежде всего объясняется тем, что практически нет приемлемого решения проблемы сопряжения обучающей программы с банком данных или автоматизированной информационной системой учебного назначения. Кроме того, не разработаны методики формализованного описания человеко-машинных процедур или «сценариев» диалогового (в указанном смысле) обучения.

АОС, рассмотренные выше, реализуют, как правило, одну и ту же общую стратегию обучения: сначала у обучаемых формируются средства решения задач определенного класса, а затем они начинают применять эти средства для непосредственного решения своих задач.

Возрастающая потребность решения на ЭВМ задач пользователями, не имеющими профессиональной подготовки в области ЭВМ и являющимися специалистами в других областях, заставила по-новому взглянуть на процесс подготовки пользователей и выдвинула проблему разработки новых методов обучения пользователей, основанных на управляемом ЭВМ овладении пользователем (обучаемым) машинными средствами (например, незнакомым ему языком программирования) в самом процессе решения на машине задач из области его профессиональной деятельности. При этом и обучение, и программирование должны рассматриваться как процессы решения задач определенного типа, иначе трудно реализовать взаимодействие, сочетающее эти оба вида деятельности.

13.6. Анализ основных особенностей диалога человека и ЭВМ

В первом приближении диалоговые режимы будем рассматривать как такие режимы разделения времени, над которыми «надстроены» средства обучения и (или) управления пользователем «со стороны» машины. В этом плане режимы, описанные в п.13.2, обеспечивают зачатки диалога во всяком случае на тех участках протокола взаимодействия, где пользователь получает справку или вызывает обучающую программу. Включение в СРВ типа системы программирования программы, управляющей действиями пользователя при конструировании алгоритма решения задачи определенного класса, при уточнении ее формулировки и т. п.,— более развитый вариант системы, «способной к диалогу». Включение в состав такой системы некоего «переключателя», возвращающего пользователя, например, после попытки самостоятельного программирования в «нужное место» упомянутой выше управляющей программы, представляет собой следующий шаг в создании диалоговых систем на базе ЭВМ.

Такое динамическое распределение действий партнеров по выполнению одной и той же функции, а также постоянное «доучивание» партнера напоминают диалог между людьми при решении одной и той же задачи. **Здесь ЭВМ может «претендовать» на роль равноценного партнера, имеющего определенные знания о предмете диалога, а также умеющего обеспечивать выполнение в какой-то мере наравне с человеком такого действия, как, например, создание программы решения задачи на входном языке.**

Оказание машиной активной помощи пользователям при конструировании алгоритмов решений задач, при уточнении их формулировок, автоматизация обучения пользователей создают новые предпосылки к диалогу человека и ЭВМ, который приближается по своему содержанию к диалогу между двумя партнерами — людьми.

В литературе (прежде всего в психологической) под диалогом обычно понимается процесс непосредственного и достаточно быстрого обмена сообщениями между двумя субъектами, при котором существует постоянная смена ролей **информатора и реципиента (т. е. выдающего и принимающего сообщение соответственно).**

Это и подобные, определения явно или неявно применяют как при рассмотрении взаимодействия между людьми, так и между человеком и ЭВМ. Однако слова «субъекты», «информатор» и «реципиент» несут различную смысловую нагрузку в зависимости от того, рассматривается обмен сообщениями между людьми или между

человеком и искусственной системой.
Говоря о диалоге между людьми, обычно подразумевают наличие: целенаправленного обмена сообщениями; взаимопонимания партнеров; определенной равноценности всей их деятельности в процессе обмена сообщениями; расширения знаний, умений и навыков одного партнера за счет знаний, умений и навыков другого.
Ряд исследователей диалога человека и ЭВМ ставит своей задачей моделирование свободной «беседы», неограниченного речевого взаимодействия между двумя субъектами — людьми, для которого были бы присущи указанные свойства.
Другим «полюсом» при рассмотрении «диалогового» взаимодействия между человеком и ЭВМ является технический подход, при котором основное внимание обращают на быстроту и возможность прямого обмена сообщениями, а также на чисто языковое их оформление. При этом категории целенаправленности, взаимопонимания, равноценности деятельности и обучения партнеров либо не рассматриваются, либо затушевываются.
В отличие от первого подхода наша трактовка диалога человека и ЭВМ ограничивает это понятие рамками совместного решения задачи, в котором и человек, и машина выполняют динамически меняющиеся функции, обеспечивая тем самым повышение эффективности всего процесса решения задачи, начиная от ее формулировки и кончая выполнением отлаженной программы.
С другой стороны, в отличие от расширительной трактовки диалога человека и ЭВМ, характерного для второго подхода, наша трактовка этого понятия будет строиться на явном учете в процессе решения задачи категорий целенаправленности, взаимопонимания, равноценности деятельности и обучения партнеров. Рассмотрим более подробно каждую из этих категорий.
Диалог между двумя людьми предполагает наличие у партнеров цели, ради достижения которой инициируется обмен сообщениями.
При взаимодействии человека и ЭВМ в пакетном режиме или традиционном РРВ такой целью обычно располагает человек. Однако в СРВ, оснащенной средствами обучения и управления пользователем, промежуточные цели, ведущие к достижению основной, могут быть присущи и вычислительной машине. Например, обнаружив отсутствие у человека знаний или умений, необходимых для продолжения взаимодействия в направлении достижения основной цели, ЭВМ может переключиться в режим обучения, поставив таким образом перед пользователем частную цель**: овладеть неким разделом знаний.**
Взаимопонимание проявляется в знании каждым из партнеров системы языковых знаков или кодов, из которых строятся отдельные сообщения, а

также в наличии хотя бы частично совпадающего представления о предмете (теме) беседы. **Чем большее количество знаний и умений оказывается общим для обоих партнеров, тем проще достигается понимание ими друг друга.** Если же партнеры не располагают некоторым минимумом общих знаний и (или) умений, то возникает необходимость доучивания по крайней мере одного из них. Так, если партнеры не владеют общим языком для представления сообщений, то один из партнеров должен изучить язык другого партнера, а это приведет к тому, что усвоение языка общения становится в свою очередь целью взаимодействия.

Исследование вопроса о роли языка обмена сообщениями между человеком и ЭВМ и разработка систем с естественным для человека языком общения — весьма сложная проблема, которой посвящено немало работ. Однако нам представляется, что требование к абсолютной естественности языка диалога не является первостепенным; **во многих случаях гораздо важнее четкое и однозначное понимание некоторых фактов и (или) команд, чем форма их представления.** Поэтому в первом приближении, будем считать, что сообщения от машины к человеку должны поступать на естественном для человека и решаемой задачи языке, а от человека к машине — на некотором формализованном, близком к естественному языке, ускоряющем «понимание» сообщений машиной и в то же время таким, чтобы необходимость его изучения по привела к прекращению взаимодействия или к подмене его цели.

При такой постановке проблемы взаимопонимания становится ясней и особая роль обучения пользователя машинным средствам решения задач, осуществляемого вычислительной машиной. В результате такого обучения человек должен, так формулировать свои сообщения, чтобы машина могла выполнять именно те действия, которые он от нее ожидает. Если реакция машины соответствует тому, что ожидал человек, то можно считать, что машина поняла сообщение человека.

С другой стороны, в развитых диалоговых системах ЭВМ также должна обучаться как в процессе решения отдельной задачи, так и «между задачами», накапливая и обобщая опыт их решения с целью, например, сокращения участия пользователя в процессе решения, а также с целью оказания ему неназойливой помощи, соответствующей уровню его подготовки и отношения к решению определенного класса задач.

Равноценность деятельности партнеров по диалогу предполагает способность каждого из них совершать действия сходного характера, направленные на решение одной и той же задачи. При взаимодействии человека и ЭВМ такая равноценность может прежде всего проявляться при написании программы на входном языке, когда некоторые части такой программы строятся синтезатором или конструктором программ, а

оставшиеся части пишутся пользователем.
Другим примером равноценности деятельности может служить распознавание класса информационных объектов, выполняемое параллельно и оператором, и ЭВМ в некоторой системе управления. Здесь человек, для которого действие распознавания во многих случаях является рутинным, может зачастую опережать ЭВМ, сокращая тем самым общее время решения задачи.
В общем случае распределение сходных функций между человеком и машиной должно осуществляться динамически в процессе взаимодействия на основании некоторых критериев эффективности (время решения, стоимость и пр.). Это должно обеспечивать наилучшее сочетание возможностей как человека, так и ЭВМ.
Итак, в отличие от традиционной расширительной трактовки диалога человека и ЭВМ, охватывающей практически все режимы разделения времени, согласно нашей трактовке диалоговыми являются только те режимы разделения времени, в которых наблюдаются:
а) наличие цели взаимодействия у обоих партнеров;
б) определенная степень равноценности деятельности в процессе решения задачи;
в) расширение и усовершенствование знаний (умений) одного партнера за счет знаний (умений) другого партнера, в частности обучение одного партнера другим партнером;
г) обмен сообщениями, направленный на установление понимания одним партнером сообщений другого партнера.
К разделению режимов разделения времени на диалоговые и недиалоговые можно подходить по-разному: в первом случае диалоговым можно считать взаимодействие, в котором наблюдаются все четыре качества а)—г); во втором случае достаточно наличия хотя бы одного из четырех указанных особенностей. Рассматривая взаимодействие человека и ЭВМ, будем считать диалогом такой режим разделения времени, в котором наблюдается хотя бы одна из указанных четырех особенностей.
При современном уровне развития СРВ создание на их основе диалоговых режимов зависит прежде всего от создания единой теории и методов проектирования обучающих и обслуживающих «надстроек» над режимами разделения времени.

13.7. Методология создания диалоговых систем

Для создания «диалоговой» технологии решения задач с помощью ЭВМ, включающей создание автоматизированных обучающих систем

широкого назначения и обеспечивающей возможность эффективной работы пользователей-непрофессионалов вместе с облегчением работы пользователей других категорий, необходимо решить комплекс следующих задач.

1. Разработать теоретические основы «задачного» подхода к исследованию взаимодействия пользователя и ЭВМ, включая: обобщенные модели задач и средств их решения, пригодные для анализа с единых позиций естественных и искусственных решающих систем; математический аппарат (язык) для описания как формулировок задач, так и сообщений, которыми обмениваются пользователь и ЭВМ в процессе решения задач.
2. Предложить решение проблемы формализованного описания различных видов взаимодействия пользователя и ЭВМ в рамках «задачного» подхода и с помощью разработанных моделей и языка.
3. Разработать формализованные («задачные») модели взаимопонимания между партнерами, передачи задачи партнеру, совместного ее решения, учения и обучения. На основе этих моделей осуществить проектирование (синтез) комплекса типовых процедур: автоматизированного обучения человека; автоматизированного обслуживания пользователей при вводе данных в готовые программы, программировании и конструировании программ.
4. Разработать и исследовать диалоговое взаимодействие пользователя и ЭВМ с целью проектирования программных систем «способных к диалогу».

В плане перечисленных задач общая методология создания диалоговых систем состоит в следующем.

На основе «задачного» подхода, моделей и языка для представления задач и решающих систем выделить и дать формализованное описание базовых компонентов различных видов взаимодействия пользователя и ЭВМ.

С учетом этих компонентов осуществить синтез (проектирование) типовых режимов и «сценариев» взаимодействия, включая диалог пользователя и ЭВМ, а также провести экспериментальную проверку пригодности и эффективности как этих режимов и «сценариев», так и аппарата, применявшегося для их построения.

По результатам проведенных исследований — теоретических и экспериментальных — создать технологичные инструментальные программные системы вместе с методикой проектирования и ведения как обучающих и обслуживающих программ, так и диалоговых режимов, основанных нл этих программах. В заключение проверить пригодность и

эффективность инструментальных систем, а также наметить пути их дальнейшего развития и интеграции с системами разделения времени и стандартными операционными системами ЭВМ.

14. Основы теории «задачном» подхода к исследованию взаимодействия человека и компьютера

14.1. Требования «задачного» подхода к исследованию взаимодействия человека и ЭВМ

При теоретическом анализе задач и процессов их решения обычно ограничиваются характеристикой понятия задачи в рамках какой-либо из сложившихся наук. Между тем для проектирования и создания человеко-машинных систем большой методический интерес представляет нахождение того общего, что есть в подходах различных наук к описанию задач, и определение на этой основе общенаучного понятия задачи. Такое обобщение можно рассматривать как **специальную научную дисциплину, изучающую задачи и процессы их решения в значительной мере независимо от той предметной области, к которой они относятся, а также от того, какая материальная система — естественная, искусственная или комбинированная — их решает. (** В 1971 г. А.М. Довгяло был предложен термин для такой научной дисциплины—проблемология, т. е. наука о задачах и процессах их решения. Независимо в том же году этот термин был применен в тезисах румынских ученых Ш. Джоржеску и В. Тоною).

Первым шагом на пути оформления этой дисциплины должно быть уточнение самого понятия задачи.

В психологии мышления над задачей обычно понимают ситуацию для субъекта, которая характеризуется не просто незнанием, а осознанием человеком того, что в известном есть нечто неизвестное, существенно важное для него (человека) и в то же время такое, что его нельзя сразу выяснить». При этом решение задачи рассматривается как некоторый интеллектуальный, творческий акт. В противоположность этому в области вычислительной техники и программирования к процессам

решения задач относят выполнение полностью отлаженной программы над заранее подготовленными и введенными в машину данными. Поэтому с целью анализа с единых позиций различных видов взаимодействия человека и ЭВМ необходимо разработать некоторое обобщенное представление задачи, удовлетворяющее как первой, так и второй интерпретации этого понятия. Например, вариант взаимодействия пользователя и ЭВМ, описанный в п. 13.2, можно рассматривать как процесс совместного решения задачи, в котором и пользователь и ЭВМ выполняют динамически меняющиеся функции, причем вычислительная машина выполняет не только рутинные действия, но и действия, считающиеся интеллектуальными (конструирование алгоритма, обучение пользователя и т. п.).

«Задачный» подход к исследованию взаимодействия человека и ЭВМ будет выражаться в определении типов решаемых задач, в выделении абстрактных средств их решения, а также в «задачном» анализе различных видов взаимодействия человека и ЭВМ. Одной из целей задачного подхода является также построение структур различного рода систем искусственного интеллекта, в большей или меньшей степени «способных» к решению задач и к диалогу; другой целью служит определение содержания обучения пользователя, вступающего во взаимодействие с ЭВМ, и др.

«Задачный» подход к исследованию взаимодействия человека и ЭВМ разрабатывался и исследовался в течение ряда лет, начиная примерно с 1970 г. (Непосредственным толчком к этим исследованиям послужил цикл докладов, прочитанных в 1969 г. в Институте кибернетики АН УССР А. М. Парачевым).

Основные предпосылки к такому исследованию были изложены в В.М. Глушковым, А.М. Довгяло и др.

Приводимые ниже определения понятий носят в значительной мере интуитивный характер и не претендуют на удовлетворение математических критериев строгости. Мы лишь стремились к повышению четкости понятий как за счет уменьшения двусмысленностей и тавтологий, так и за счет введения по возможности однозначных связей между вводимыми понятиями, что, на наш взгляд, является определенным прогрессом по сравнению с положением, характерным для большинства исследований по искусственному интеллекту, человеко-машинным системам и «диалогу» в широком смысле этого слова, а также для примыкающих к нашему исследованию работ по психологии мышления и педагогике.

14.2. Базовые понятия и их определения

Вводя определения таких базовых проблемологических понятий, как «изменение», «операция», «модель», прежде всего будем иметь в виду существование и функционирование**:**

а) «пассивных» предметов или объектов; «активных» предметов, воздействующих на другие предметы;

б) исследователей или наблюдателей, которые тем или иным образом выделяют в окружающем мире «пассивные» и «активные» предметы и изучают их взаимодействия.

Определение 1. Под предметом понимаем все то, что может быть как-то воспринято, представлено, названо исследователем.

Предмет A может находиться в состояниях $A_1,...,A_m$, каждое из которых определяется множеством значений свойств предмета. В тех случаях, когда это удобно, о каждом из возможных состояний $A_1,...,A_m$ будем говорить так же, как об отдельном предмете $A_1,..., A_m$.

Предметы могут вступать между собой в различные отношения. Примерами отношений могут служить отношения между людьми — родителями и детьми, начальником и подчиненными, между деталями, составляющими определенный механизм, между словами и предложениями.

Если заданы множества $D_1, D_2,..., D_n$, то R является n-местным отношением на этих n множествах, если R представляет собой множество упорядоченных кортежей $<d_1, d_2..... d_n>$, таких, что d_1 принадлежит к D_1, и т д., d_n принадлежит к D_n. Множества $D_1, D_2,..., D_n$ называют областями определения или доменами отношения R. Величина n называется порядком отношения.

Приведем пример отношения, названного «ЧАСТЬ» (имя R —ЧАСТЬ):

ЧАСТЬ

Обозначение части	Название части	Целое	Единица измерения
a	Сторона верхнего основания	Усеченная правильная пирамида	дм
b	Сторона основания	Правильная пирамида	дм
h	Высота	Усеченная пирамида	см
x	Высота	Правильная пирамида	см

Области определения этого отношения D_1 = ОБОЗНАЧЕНИЕ ЧАСТИ; D_2 = НАЗВАНИЕ ЧАСТИ; D_3=ЦЕЛОЕ (т. е. тот объект, часть которого задается отношением *R); * D_4 = ЕДИНИЦА ИЗМЕРЕНИЯ.

Определение 2. Система *(S)* — это множество предметов *{A},* рассматриваемое исследователем вместе с отношениями *{R}* между этими предметами. Можно считать, что *S = {{A}, R}}*
Предметы, образующие множество {*A*} в S, являются компонентами системы. Если система S_1 является компонентом системы S_2, то S_1 будем также называть подсистемой системы S_2.
Определение 3. Изменение — это превращение одного предмета в другой или переход некоторого предмета из одного состояния в другое. Изменение, состоящее в том, что предмет A_1 превращается в предмет A_2, символически представим в виде $A_1 \rightarrow A_2$.
Возникновение или исчезновение предмета рассматривается как его изменение.
О воздействии предмета *B* на предмет *A* будем говорить в тех случаях, когда предмет *B* вызывает или предотвращает некоторое изменение предмета *A*. Предмет *B* назовем воздействующим или активным предметом (системой), а предмет *A* — объектом воздействия.
Описывая воздействия некоторой активной системы на те или иные объекты, часто полезно выделять компоненты или свойства этой системы, обеспечивающие осуществление воздействий определенных типов. Эти компоненты или свойства назовем операторами.
Активную систему, содержащую операторы, назовем оперирующей системой.
Оператор всегда применяется к некоторому предмету (предметам), который назовем операндом.
Оператор σ применим к операнду *A* (оператор σ и операнд *A* релевантны друг для друга), если применение σ к *A* может привести к изменению предмета *A*: $A \xrightarrow{\sigma} B$ или *B*= σ *(A).*

Определение 4. Применение оператора σ к релевантному для него операнду *A* назовем операцией σ *(A).* Операция — это то, что оперирующая система делала, могла делать или должна была делать, а изменение, осуществленное под ее воздействием,— это то, что она сделала, произвела (или могла сделать, или должна была сделать).
Если не требуется указывать тип операнда, условимся обозначать операцию σ*(A)* одним оператором σ.
Определение 5. Если тот факт, что система *B* несет информацию о системе *A,* позволяет некоторой оперирующей системе S использовать *B* в качестве операнда (операндов) вместо *A*, то B является моделью *A* для системы S.
В дальнейшем для моделей систем и предметов будем применять следующие обозначения; $A^*_{k,i}$ —модель системы (предмета) A_k для системы S_i, где «*» — символ модели.

Будем использовать и более сложные обозначения для моделей второго порядка или моделей моделей. Например: $(A^*_{k,1})^*_2 = A^{**}_{k,12} = A^*_{k,12}$ — модель для системы S_2 модели подсистемы A_k, имеющейся у системы S_1.

Различают следующие типы моделей:

1) материальные, т. е. такие, материальная форма которых существенно влияет на их функционирование;

2) материализованные, т. е. такие, которые, хотя и существуют в материальной форме, но функционирование которых мало зависит от ее особенностей; они делятся на **иконические**— элементы которых сами являются моделями соответствующих элементов моделируемого объекта; **знаковые**, элементы которых (знаки), вообще говоря, не являются моделями; **смешанные**, содержащие элементы знаковых и иконических моделей;

3) идеальные модели — абстрагированные от материальной формы. Всякой идеальной модели соответствует несущая ее материальная или материализованная модель, например, понятийной модели, существующей в науке, соответствует некоторая система текстов.

Обратимся к знаковым моделям или описаниям операций. Операции могут описываться в различных модальностях. Нас прежде всего будут интересовать модальности «факт» («что делала система»); «возможность» («что она могла летать»); «требование» («что должна была делать система»). Описание операции в модальности «требование» является императивом, или командой.

14.3. Процедуры и алгоритмы

Для последующего изложения представляет интерес рассмотрение различных систем операций.

Определение 6. Процедурой назовем систему операций $\sum = \{\{\sigma\}, \{R\}\}$, где $\{\sigma\}$ — некоторое множество операций, $\{R\}$ — множество отношений между операциями $\{\sigma\}$.

В зависимости от вида R будем различать:

1) разветвленные процедуры, т. е. такие, в которых R представляет собой ориентированный граф, содержащий не менее чем одну операцию, из которой исходит не менее двух ребер;

2) распараллеленные процедуры, т. е. такие, в которых не менее двух операций выполняется одновременно; (Вообще говоря, распараллеленная процедура должна задаваться двумя отношениями: отношением, определяющим переходы от одной операции к другой, и отношением «ВЫПОЛНЯЮТСЯ ОДНОВРЕМЕННО».)

3) линейные процедуры, т. е. процедуры, не содержащие ни разветвлений, ни распараллеливаний,
4) циклические процедуры, которые содержат отношение, являющееся ориентированным циклом или контуром.
Определение 7. Если операнд и (или) образ некоторой операции является предметом, принадлежащим к внешнему по отношению к оперирующей системе миру, то такая операция называется внешней. Множество внешних операторов, входящих в *{σ},* обозначим *{ν}*.
Определение 8. Если релевантные для некоторой операции операнды или образы являются компонентами только оперирующей системы, то такую операцию назовем внутренней. Множество внутренних операторов обозначим *{μ}.*
Если в состав некоторой процедуры входит хотя бы одна внешняя операция, то такую процедуру будем называть внешней. Внутренней назовем такую процедуру, в состав которой входят только внутренние операции.
Выбор разветвления внутри процедуры может быть **принудительным и свободным**. Выбор является принудительным, когда избираемая операция осуществляется в соответствии с некоторыми условиями, содержащими ссылки на тот или иной признак (признаки) какого-либо предмета (предметов). Принцип принудительного выбора не выдерживается, если упомянутые признаки определены недостаточно точно. В этом случае приходится говорить не о множествах (в строгом смысле) значений переменных, опознаваемых оперирующей системой при выборе пути, а о так называемых размытых множествах. А. Заде определяет «размытые множества» как «классы, в которых могут быть степени членства, промежуточные между полным членством и отсутствием членства».
Выбор является свободным, когда оперирующая система может выбрать любой из возможных путей, причем вероятности выбора не устанавливаются. Помимо «размытых алгоритмов», принцип свободного выбора реализуется, в частности, в процедурах, описанных В. Б. Борщевым и Ю. А. Шрейдером под названием «диспозиции», в так называемых недетерминистических алгоритмах, описанных Р. Флойдом**,** в «недетерминированных схемах алгоритмов», описанных Р. И. Подловченко.
Разветвленную процедуру, в которой реализуется принцип принудительного выбора, будем называть **детерминированной**, а процедуру, в которой осуществлен принцип свободного выбора — **недетерминированной.**
Определение 9. Процедура называется алгоритмической, если она не содержит недетерминированных разветвлений и состоит только из

эффективных операций (термин «эффективный» употребляется здесь в смысле, близком к тому, какой обычно придается ему в математике.), т. е. из операций, обеспечивающих совершенно определенные воздействия на рассматриваемые (интересующие исследователя) предметы.

Определение 10. Процедуру назовем квазиалгоритмической, если она не содержит недетерминированных разветвлений и наряду с эффективными операциями содержит по крайней мере одну квазиэффективную операцию, т. е. операцию, обеспечивающую с достаточно высокой вероятностью, определенные воздействия на рассматриваемые предметы

Эффективные операции характерны для идеализированных оперирующих систем. Бывают также случаи, когда, описывая функционирование реальных оперирующих систем, например вычислительных машин, можно пренебречь их отличием от идеализированных оперирующих систем (абстрактных цифровых автоматов), осуществляющих только эффективные операции. Однако при рассмотрении надежности ЭВМ абстракция безошибочности уже недопустима. При переходе от описаний функционирования технических систем к описанию человеческой деятельности абстракция безошибочности еще более сужается и понятие квазиэффективной операции и процедуры оказывается здесь весьма полезным.

Определение 11. Алгоритмом (квазиалгоритмом) назовем знаковую модель в модальности «требование» алгоритмической (квазиалгоритмической) процедуры, в которой предусматривается выполнение конечного числа операций, а также хотя бы одна из операций имеет родовой операнд, т. е. операнд, представляющий собой произвольный элемент множества индивидуальных предметов, заданного некоторым набором признаков.

Общая («генетическая») схема базовых проблемологических понятий, введенных нами 14.2 и 14.3, представлена на рис. 1.

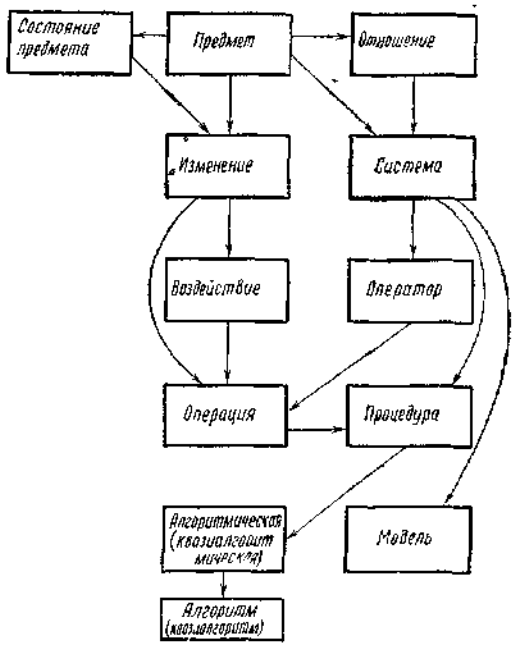


Рис 1.

14.4. Обобщенная модель задачи и решающей системы

В данном разделе предлагаются обобщенные модели задачи и решающей системы, применяемые для исследования различных видов взаимодействия пользователя и ЭВМ.

Определение 12. Предметная область — это тройка $K=(\{k\}, \{\sigma\}, \{R\})$, где (k) — множество предметов; $\{\sigma\}$ — множество возможных операций, заданных на $\{k\}$; $\{R\}$— множество предикатов (отношений, заданных на $\{k\}$, $\{\sigma\}$).

Операции, входящие в предметную область, также могут находиться между собой в различных отношениях, образуя процедуры.

Содержательно предметная область представляет собой множество допустимых (с точки зрения физических и логических законов) состояний предметов и отношений между ними.

Примерами предметной области могут служить априорная «модель мира» в работах по автоматизации решения задач, по искусственному интеллекту, «язык», учебный предмет или его раздел и т. п.

Определение 13. Задачей $З_k$ называется (рис. 2) тройка *{К, К*акт, Ктр*}, где *К* — предметная область; Ктр* — модель требуемого состояния предметной области, причем само требуемое состояние Ктр входит в *К* после решения задачи.

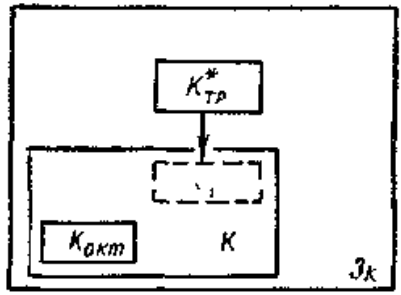


Рис.2

Определение 14. Формулировкой задачи (ФЗ) назовем ее знаковую модель, т. е. ФЗ= *(К, К*акт, Ктр*). *

Пример. Формулировка задачи: «Найти объем V_A правильной четырехугольной усеченной пирамиды, если даны ее высота *h,* сторона *a* верхнего основания и сторона *b* нижнего основания».

Здесь предполагается, что предметная область или предмет этой задачи *К** включает знания решающего (например, учащегося старшего класса) по стереометрии. Характерной особенностью актуального состояния этой задачи является наличие в его составе отношения $R_{ин}$.

$R_{ин}$	Наименование компонента $K^*_{акт}$	Обозначение	Полнота информации
	Правильная четырехугольная усеченная пирамида	*A*	известно
	Высота *A*	*h*	»
	Сторона верхнего основания *A*	*a*	»
	Сторона нижнего основания *A*	*b*	»
	Объем *A*	V_A	неизвестно

Требование задачи Ктр** состоит в нахождении неизвестного решающему функционального отношения $X(V_A, a, b, h)$.

Определение 15. Изменение актуального состояния в требуемое назовем процессом решения или просто решением задачи. Решение задачи может быть выражено любой системой изменений, ЭКВИВАЛЕНТНОЙ $К_{акт} \rightarrow К_{тр}$.

Требуемое состояние $К_{тр}$ назовем также результатом решения задачи. Введенное нами понятие задачи является простейшим из рассматриваемых в данной работе уточнений понятия задачи. Это

понятие хорошо отражает факт существования задач, в основном, в виде их формулировок, не связанных, не соотнесенных с системами, их решающими. Обычно рассматриваются решения задач, обеспечиваемые теми или иными системами.

Определение 16. Оперирующую систему, которая обеспечивает, может обеспечивать или должна обеспечивать решение задачи, назовем решающей системой (РС).

Для осуществления необходимых воздействий, обеспечивающих изменение $K_{акт}$ в $K_{тр}$, всякая решающая система прежде всего должна обладать некоторым, множеством операторов {σ} и (или) процедур {∑}.

В качестве решающих систем могут выступать люди, коллективы людей, вычислительные машины, человеко-машинные системы и т. д.

В соответствии с определением 13 задача может рассматриваться в двух планах: с абстрагированием от решающей системы (чтобы подчеркнуть этот факт, назовем такую задачу неотнесенной) и без такого абстрагирования — в этом случае будем говорить об отнесенной задаче. Важность изучения отнесенных задач, особенно в психологии, зачастую приводит к тому, что под задачей понимают не просто внешнюю ситуацию для субъекта, а ситуацию, которая характеризуется, как уже было сказано ранее, «не просто незнанием, а осознанием человеком того, что в известном есть нечто неизвестное, существенно важное для него (человека) и в то же время такое, что его нельзя сразу выяснить». В этой связи Д. Берлайн отмечает, что «часто говорят о задаче как о чем-то, что существует во внешнем мире. Она предъявляется субъекту на листе бумаги, или он обнаруживает ее где-то в природе. Однако то, что составляет задачу для одного индивидуума, может не быть задачей для другого».

Определение 17. Отнесенной задачей (ОЗ) назовем множество ОЗ= {З, *РС} = {К,* $K_{акт}$, $K_{тр}$*, РС}, где РС — решающая система, компонентами которой являются множества средств решения задачи.

Как отмечает Г. А. Балл, «решающая система РС, к которой отнесена задача, может быть реально существующей или потенциальной, предполагаемой; конкретной (единичной) или любой из некоторого класса систем. Она может быть описана с разной степенью полноты. Так, в качестве решающей системы может быть указан «инженер», «инженер-конструктор»..... «инженер-конструктор высокой квалификации И. И. Иванов, специально изучивший документацию по данному проекту и т. д.».

В любой задаче, как правило, можно выделить другие задачи — более мелкие, простые, частные,— решение которых зачастую является

необходимым предварительным условием решения первоначальной задачи.

Определение 18. Подзадачей задачи $З_k$ ($ПЗ_k$) назовем такую задачу, для которой справедливо хотя бы одно из следующих условий:

а) решение $ПЗ_k$, т. е. изменение $ПК_{акт} \rightarrow ПК_{тр}$, входит в изменение, представляющее собой решение $З_k$, т. е. в $К_{акт} \rightarrow К_{тр}$;

б) процедура решения $ПЗ_k$, т. е. $\sum_{пк}$, входит в процедуру решения $З_k$;

в) модель процедуры решения $ПЗ_k$, т. е. $\sum^*_{пк}$, входит в модель процедуры решений $З_k$ ($\sum_{пк}^* \subset \sum_к^*$).

Определение 18 справедливо как для неотнесенных, так и для отнесенных задач. Кроме того, приведенное определение подзадачи является более строгим в смысле использования базовых проблемологических понятий, чем аналогичные определения, предложенные Г. А. Баллом.

Пример. Для $З_k$ с формулировкой «найти объем V_A правильной четырехугольной усеченной пирамиды, если даны ее высота *h*, сторона *a* верхнего основания и сторона *b* нижнего основания», подзадачами могут быть:

1) задача по нахождению объема полной («большой») пирамиды (см. пример в конце темы 14).

2) задача по нахождению объема второй полной пирамиды, дополняющей усеченную пирамиду до «большой» полной пирамиды;

3) задача по нахождению высоты второй полной пирамиды.

Рассмотрим состав решающей системы. В отличие от оперирующей системы РС должна содержать не только операторы $\{ \sigma_j \}$ и процедуры $\{ \sum_j \}$, но и следующие специфические операнды:

1) по каждой решаемой ею задаче — $K^*_{тр,j}$ и $\{K^*_{акт,j}\}$;

2) множество U^*_j моделей предметных областей (предметов задач) $U^*_j \supseteq K^*_j$. В частном случае, у специализированной $РС_j$ $U^*_j = K^*_j$; множество $\{ U^*_j \}$ представляет собой основу знаний решающей системы $РС_j$;

3) модели $\{\sigma_j^*\}$ и $\{\sum^*_j\}$, описанные в модальности требования.

Модели $\sum^*$ у людей, решающих задачу, зачастую называют способами решения задач; применительное к ЭВМ модели $\sum^*_j$ — это программы на входных языках программирования, реализующие алгоритмы решения задач.

Для хранения указанных моделей решающая система должна располагать подсистемой памяти. Будем рассматривать подсистему памяти как имеющую двухуровневую структуру и состоящую из подсистемы кратковременной и подсистемы долговременной памяти. Такое представление согласуется как с данными об организации

памяти человека, хранящего текущие внутренние операнды и образы (у нас $K^*_{тр,\,j}$ и $\{K^*_{акт,\,j}\}$) в кратковременной памяти, а опыт и модели внешнего мира $(K^*_j,\ U^*_j \supset K^*_j, \{\sigma_j^*\},\ \{\sum^*_j\})$ —в долговременной, так и с организацией памяти систем разделения времени и систем искусственного интеллекта.

Интересно отметить соотношение между процедурами $\{\sum\}$ и знаниями РС, по этим процедурам, т. е. моделями $\{\sum^*_j\}$. Если представить себе, что операции $\{\sigma\}$, составляющие процедуры $\{\sum\}$, способны к преобразованию очень широкого спектра данных (как в смысле их обобщенности, так и физического представления), то «банк» процедур $РС_j$ может быть сокращен, так как каждая процедура будет строиться из операций $\{\sigma\}$ по моделям $\{\sum^*_j\}$ по мере надобности. При этом решение любой задачи может быть представлено как более или менее многоуровневый иерархический процесс, причем нижний уровень будут занимать операции σ, работающие с некоторыми «атомарными» данными.

Будем считать, что оперирующая или решающая система обладает некоторой процедурой $\sum_k$ если:

а) у РС имеется соответствующая система операций, готовая к выполнению данной системой;

б) в памяти данной системы хранится модель $\sum^*_k$, состоящая, например, из моделей таких операторов σ, которыми располагает данная система.

Применительно к ЭВМ, выступающей в качестве оперирующей или решающей системы, вариант а) соответствует случаю, когда в машине имеется, например, микропрограмма, реализующая $\sum_k$. Вариант б) соответствует случаю, когда ЭВМ содержит рабочую программу (абсолютную фазу), являющуюся моделью $\sum^*_k$, а также располагает всеми операциями, составляющими $\sum_k$.

Важным для нашего изложения является случай, когда моделью $\sum_k{}^*$ владеет одна решающая система, а выполняет операции, составляющие $\sum_k$, другая РС.

Представляет интерес рассмотрение в плане развиваемых здесь положений видов владения процедурой или способом решения задачи у человека. Ряд исследователей отмечают здесь следующие два основных процесса:

1) свертку (сокращение, сжатие) модели процедуры $\sum_k$ после ее успешных выполнений;

2) саморефлексию или «осознание» своего владения процедурой.

Применительно к рассматриваемой модели РС свертывание модели процедуры может привести в конечном счете к представлению ее в «знаниях» РС в виде описания некоторой специализированной

операции. Примером может служить навык поддержания тела на поверхности воды, который у достаточно опытного пловца является скорее отдельной операцией, чем их осознаваемой системой. Примером внутренней операции того же рода может служить навык быстрого счета в уме.

Промежуточный вариант свертки процедуры состоит в сокращении количества операций, ее составляющих, и, соответственно, в уменьшении подробности ее описания, хранящегося в знаниях РС. Пример свертывания процедуры, приводящего к сокращению количества операций, дан в примере, приведенном в конце темы 14.

При свертывании процедуры возможны различные варианты саморефлексии или «осознания» решающей системой своего владения этой процедурой:

1) владение, которое включает в себя ее полное знание, т. е. наличие у РС полного пооперационного описания процедуры;
2) владение, которое не включает ее полного знания (имеется в виду наличие только свернутой модели процедуры с сохранением ее расчленения на операции, т. е. промежуточный вариант свертки);
3) владение, когда человек даже уже не осознает выполняемых операций, т. е. процедура превратилась в одну специфическую операцию, зафиксированную в «знаниях» РС, например, в виде одной команды или императива.

Как мы уже говорили (см. определения 9 и 10), частными видами процедур являются, алгоритмическая и квазиалгоритмическая процедуры.

Знаковой моделью этих процедур в модальности «требование» являются соответственно алгоритм и квазиалгоритм (определение 11).

При решении задач человеком возможны следующие варианты владения квазиалгоритмом их решения:

1) квазиалгоритм дан человеку в виде внешней опоры, например в виде печатной инструкции, согласно которой человек должен выполнять соответствующие операции;
2) человек помнит квазиалгоритм (хранит его в своей памяти) и способен выполнить предусмотренные операции, осознавая факт выполнения каждой операции;
3) система операций, предусмотренных квазиалгоритмом, сформулирована на уровне навыка, т. е. человек не осознает факт выполнения каждой операции.

14.5. Типы задач

Изложенный выше формализованный понятийный аппарат позволяет уточнить основные типы задач. А.М. Довгяло были проведены исследования, направленные на формализованное описание основных типов задач с использованием категорий, которые были введены в п.14. 2—14.4. Они показали, что исчерпывающая классификация задач, даже при использовании относительно небольшого числа базовых понятий, представляет собой специальную проблему, далеко выходящую за рамки настоящей работы.

Поэтому мы выделим и опишем только те типы задач (а также подзадач, существенных для решения многих задач), решение которых будет более подробно изучаться применительно к различным видам взаимодействия человека и ЭВМ.

Определение 19. Отнесенную задачу $ОЗ_k$ назовем относительно разрешимой, если решающая система РС обладает средствами (по крайней мере процедурой $\sum_{k,\,i}$ или операторами $\{\sigma_i\}$ вместе с необходимыми знаниями, а также ресурсами — временем, памятью и т. п.), которые обеспечивают или могут обеспечить решение $З_k$.

Задача, относительно разрешимая решающей системой РС, может быть относительно неразрешимой для другой $РС_j$, если последняя не располагает средствами и (или) ресурсами для ее решения.

Например, учебная задача по какому-нибудь предмету, как правило, относительно разрешима для учителя, преподающего этот предмет, и может быть относительно неразрешимой для ученика, который не усвоил необходимый для решения данной задачи раздел учебного материала.

Среди относительно неразрешимых задач выделяют принципиально неразрешимые и принципиально разрешимые.

Определение 20. Назовем задачу (отнесенную или неотнесепную) принципиально неразрешимой, если в соответствии, с закономерностями той области действительности, к которой относится задача, невозможно требуемое состояние предмета задачи, либо невозможно изменение актуального состояния предмета, задачи в принципиально возможное $К_{тр}$. Принципиально неразрешима, например, задача построения вечного двигателя или задача оживления умершего человека после наступления необратимых изменений в нервной системе. Все задачи, не являющиеся принципиально неразрешимыми, являются принципиально разрешимыми.

Рассмотрим подразделение задач, неотнесенных и отнесенных, и зависимости от вида предмета задачи К и ее требовании $К_{тр}$.

Определение 21. Назовем $З_k$ материально-направленной задачей, если ее предмет $К$ включает предмет $К_{внешн}$ ($К_{внешн} \subseteq К$), который является какой-то частью материального мира и не выступает и качестве модели, а требование $К^*_{тр}$ является моделью требуемого состояния предмета $К_{внешн}$, т. е. $К^*_{тр} = К^*_{внешн}$.

Таким образом, множество, определяющее материально-направленную задачу, имеет вид $\{\{К, К_{внешн} \subseteq К\}, К_{акт}, К^*_{внешн}\}$.

Определение 22. Назовем $З_k$ идеально-направленной задачей, если ее предмет не включает $К_{внешн}$, т. е. является некоторой моделью, а требование есть модель требуемого состояния этой модели. Множество, определяющее идельно-направленную задачу, имеет вид $\{К^*, К_{акт}, К^*_{тр}\}$.

Обратим внимание на существование идеально-направленных отнесенных задач, являющихся подзадачами материально - направленных задач. Таковы, например, задачи по нахождению модели процедуры $\sum_k$ * изменения материальных объектов входящих в предмет $З_k$. С другой стороны, материально – направленные отнесенные задачи также могут являться подзадачами идеально-направленной задачи. Примером может служить разработка оборудования для решения задач теоретической физики.

Отнесенные идеалью-направленные задачи будем разделять на теоретические и практические.

Определение 23. Назовем $ОЗ_k$ теоретической задачей в том случае, если изменения $К^*$ и $К_{акт}^*$ возможны только в результате воздействия той РС, к которой отнесена $З_k$.

В практической идельно-направленной $ОЗ_k$ модели $К^*$ и $К_{акт}^*$ могут изменяться не только в результате воздействий РС, к которой отнесена$З_k$, но и самопроизвольно, спонтанно либо под воздействием внешней среды.

Всякая материально-направленная задача является практической (внешняя среда может воздействовать на $К_{внешн} \subseteq К$ хотя бы в силу обязательности выполнения для $К_{внешн}$ всеобщих физических законов).

Выделим различные типы отнесенных задач в зависимости от особенностей решающей системы.

Определение 24. Будем называть отнесенную задачу $ОЗ_k$ внутренней, если ее требование входит в состав РС, к которой отнесена $З_k$.

Например, множество, которое определяет материально-направленную $ОЗ_k$, внутреннюю для $РС_j$, может иметь вид $\{\{К, К_{внешн} \subseteq К\}, К_{акт}, К^*_{внешн,j} \subset РС_j, РС_j\}$.

Определение 25. Назовем отнесенную задачу внешней, если ее требование не входит в состав РС, к которой отнесена данная задача.

Определение 26. Относительно разрешимую задачу $ОЗ_k$ назовем рутинной (квазирутинной), если решающая система, к которой отнесена $З_k$, располагает алгоритмом или квазиалгоритмом решения $З_k$ (При том включается и операция или алгоритм (квазиалгоритм) установления принадлежности $З_k$ к классу задач, решаемых с помощью имеющегося у РС алгоритма (квазиалгоритма).

Определение 27. Отнесенную задачу $ОЗ_k$ назовем нерутинной (проблемной), если решающая система, к которой отнесена $З_k$, не располагает ни алгоритмом, ни квазиалгоритмом решения $З_k$.

Рассмотрим, какие можно выделить типы идеально-направленных задач в зависимости от особенностей актуального состояния предмета задачи. Выделим случай, когда $К_{акт}$ представляет собой не отдельный предмет или множество предметов (моделей), а систему моделей, т. е. $К^*_{акт} = \{\{ k^*_{акт} \}=\{R_k\}\}$, где $\{ k^*_{акт} \}$ — множество моделей компонентов $К^*_{акт}$, а $\{R_k\}$ — множество отношений, заданных над компонентами $\{k^*_{акт}\}$.

Определение 28. Назовем задачей усовершенствования знаний (ЗУ) такую идеально-направленную задачу, у которой $К^*_{акт} = \{\{k^*_{акт}\}, R_{ин}\}$, где $R_{ин}$ — отношение над $\{k^*_{акт}\}$, содержащее область определения, указывающую на известные и неизвестные компоненты $\{k^*_{акт}\}$.

Требование ЗУ состоит в переводе всех или некоторых из неизвестных компонентов $К^*_{акт}$ в известные, т. е. в переводе всех или некоторых из неизвестных моделей в такое состояние, когда полнота содержащейся в них информации оказывается достаточной.

Те неизвестные компоненты $К_{акт}$, к которым относится требование ЗУ, называют искомыми.

Примером ЗУ является задача, приведенная в примере п. 14.4.

Отношение $R_{ин}$ («известно-неизвестно») определяет актуальное состояние этой задачи.

Рассмотрим теперь отнесенные задачи усовершенствования знаний.

Определение 29. Если предмет отнесенной задачи усовершенствования знаний входит в состав решающей системы, к которой отнесена эта же задача, то такую задачу назовем познавательной.

Предметом познавательной задачи может быть любое подмножество модельных объектов, входящих в состав РС, в том числе множество моделей предметных областей U_j^*, множество моделей операций $\{\sigma^*\}$ и процедур $\{\sum^*\}$ и др.

Актуальное состояние предмета познавательной задачи, как и всякой задачи усовершенствования знаний, представляет собой систему указанных моделей, в которой отношение $R_{ин}$ указывает на известные и неизвестные модели. Такое $К_{акт}^*$ может быть результатом

саморефлексии, направленной на уяснение решающей системой собственных знаний в связи с постановкой познавательной задачи. Требование познавательной задачи отмечает, какие компоненты знаний должны быть переведены из неизвестных для данной РС в известные.

Если идеально-направленная нерутинная задача усовершенствования знаний отнесена к некоторому обществу, то такая ЗУ является задачей научного исследования (ЗНИ).

Результатом решения ЗНИ являются как собственно знания общества, так и технологии — модели системы процедур, обеспечивающих целенаправленное изменение материальных и идеальных объектов.

Появление технологии в результате решения соответствующей ЗУ (или ЗНИ) еще не значит, что она может быть немедленно реализована: необходима решающая система (или множество РС), обладающая набором операторов, составляющих технологию.

Формирование недостающих операторов может идти как по пути создания соответствующих РС, так и по пути усовершенствования операционных возможностей существующих РС. В последнем случае мы имеем дело со сцециальным видом познавательных задач, предметом которых являются операции и (или) процедуры.

Частным видом задач усовершенствования знаний являются ЗУ, у которых $\{k^*_{акт}\} = \{A^*_{нач}, \sum^*_A, A^*_{кон}\}$, где $A^*_{нач}$— модель начального состояния некоторого предмета; $A^*_{кон}$ — модель конечного состояния этого предмета; $\sum^*_A$ — модель процедуры, обеспечивающей изменение $A^*_{нач} \rightarrow A^*_{кон}$. В зависимости от вида отношения $R_{ин}$ («известно-неизвестно»), входящего в $K^*_{акт}$, среди таких задач будем различать:

а) задачи исполнения

$R_{ин}$	$\{k^*_{акт}\}$	полнота информации
	$A^*_{нач}$	известно
	Σ^*_A	известно
	$A^*_{кон}$	неизвестно

б) задачи диагноза

$R_{ин}$	$\{k^*_{акт}\}$	полнота информации
	$A^*_{нач}$	неизвестно
	Σ^*_A	известно
	$A^*_{кон}$	известно

в) задачи ввода данных или использования процедуры

$R_{ин}$	$\{k^*_{акт}\}$	полнота информации
	$A^*_{нач}$	неизвестно
	Σ^*_A	известно
	$A^*_{кон}$	неизвестно

г) задачи преобразования или доказательства

$R_{ин}$	$\{k^*_{акт}\}$	полнота информации
	$A^*_{нач}$	известно
	Σ^*_A	неизвестно
	$A^*_{кон}$	известно

д) задачи конструирования

$R_{ин}$	$\{k^*_{акт}\}$	полнота информации
	$A^*_{нач}$	неизвестно
	Σ^*_A	неизвестно
	$A^*_{кон}$	известно

е) задачи прогнозирования

$R_{ин}$	$\{k^*_{акт}\}$	полнота информации
	$A^*_{нач}$	известно
	Σ^*_A	неизвестно
	$A^*_{кон}$	неизвестно

Названия выделенным типам задач даны в предположении, что требование каждой ЗУ относится ко всем неизвестным отношения $R_{ин}$. Среди задач усовершенствования знаний будем выделять задачи с доступом и без доступа к внешней информации. Задачей без доступа к внешней информации назовем отнесенную задачу, если в процессе ее решения РС может использовать только модели, содержащиеся в самой этой системе и в решаемой задаче. Если это ограничение не выдерживается, то такую задачу будем называть задачей с доступом к внешней информации.

Дадим определения подзадачам, которые в принципе входят в процесс решения практически любой задачи. Речь идет о подзадачах контроля, разрешимости и нахождения модели процедуры решения; каждая из них может рассматриваться и как самостоятельная задача.

Определение 30. Подзадачей разрешимости для задачи $З_k$ назовем такую задачу ($ЗР_k$), требование которой состоит в выяснении того, к какому типу разрешимых или неразрешимых задач (см. определения 19 и 20) относится $З_k$.

Определение 31. Подзадачей нахождения способа решения задачи $З_k$ назовем такую задачу ($З\sum_k$), требование которой состоит в нахождении модели $З\sum_k$ процедуры решения $З_k$.

Важно отметить, что $\sum_k$ не всегда выполняется той РС, которая решила $З\sum_k$. Классический пример $З\sum_k$ и $З_k$ такого рода — написание пользователем программы решения $З_k$ и выполнение этой программы на ЭВМ в режиме пакета.

Определение 32. Задачей контроля (ЗК) назовем множество $\{П_{зк}, П_{зк,акт} = \{A, ЭТ, M\}, T_{ак}\}$, где $П_{зк}$ — предмет задачи контроля; A — контролируемая система (объект); ЭТ — эталонная система; M — некоторое отношение между A и ЭТ; в частности, M может быть отношением равенства, эквивалентности или толерантности между A и ЭТ; $T_{зк}$ — требование задачи контроля, выражаемое в общем виде вопросом «удовлетворяют ли A и ЭТ некоторому отношению M?» или императивом «доказать, что A и ЭТ удовлетворяют некоторому отношению M».

Если ЗК является подзадачей контроля правильности решения некоторой задачи $З_k$, то в этом случае $П_{зк} = K$ — предмету $З_k$; $A = K_{тр}$, а ЭТ может быть первоначальной, т. е. зафиксированной в формулировке $З_k$, моделью $K^*_{тр.0}$. В этом случае процедура решения $ЗК_k$ включает формулирование решающей системой, к которой отнесена $ЗК_k$, модели $K^*_{тр.кон}$ из полученного $K_{тр}$ (по возможности, без опоры на $K^*_{тр.0}$) и установление вида отношения между $A = K^*_{тр.кон}$ и ЭТ = $K^*_{тр.0}$.

Как и все прочие задачи, ЗК может быть идеально- и материально-направленной, внешней и внутренней, рутинной (квазирутинной) и нерутинной.

Определение 33. Если некоторая решающая система располагает информацией о том, решена или не решена задача $З_k$, либо для этой РС подзадача контроля является рутинной (квазирутинной), то назовем задачу $З_k$ определенной (квазиопределенной) ; в противном случае назовем $З_k$ неопределенной задачей.

РС, решающая нерутинную ЗК, должна сформулировать и осуществить процедуру доказательства того, что A и Э удовлетворяют модельному отношению M.

Типы задач, рассмотренных нами в данном параграфе, представлены на рис. 3.

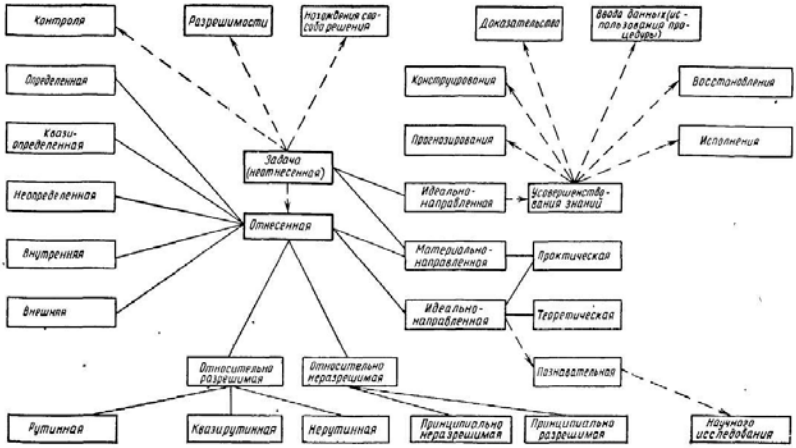


Рис.3

14.6. Язык описания формулировок задач

Поскольку в соответствии с определением 14 формулировка задачи — это ее знаковая модель, то и U^* — подсистему знаний решающей системы, включающую модель предметной области задачи — будем также рассматривать как некоторую знаковую модель. Для того чтобы РС, к которой отнесена $З_k$ могла эффективно оперировать с $Ф З_k$ и $K^* \subset U^*$, необходимо, чтобы эти модели были представлены на некотором внутреннем «срезе» РС на одном и том же языке.

Уточним теперь с помощью введенной терминологии определение взаимодействия человека и ЭВМ. Взаимодействие между пользователем и ЭВМ. представляет собой процесс обмена сообщениями, который обуславливается необходимостью осуществления процедуры решения $З_k$ этими двумя решающими системами. Для понимания смысла некоторого принятого сообщения каждая из РС в принципе должна строить его внутреннее представление на том же языке, на котором представлены в этой системе модели $Ф З_k$ и U^*.

Итак, для эффективного внутреннего представления моделей $Ф З_k$, U^* и смысла принятых сообщений как в искусственных, так и в естественных РС необходим некоторый единый язык их формализации (ЯФ). Анализ работ по (внутреннему) представлению знаний под углом развиваемого здесь задачного подхода и с учетом последующей реализации указанных моделей на современных ЭВМ приводит нас к выводу, что в основу ЯФ должен быть положен язык прикладного

исчисления предикатов. При построении ЯФ будем также руководствоваться следующими соображениями:
а) алфавит ЯФ должен содержать имена (обозначения) объектов и отношений, задающих знания РС *U**;
б) требование задачи ($K^*_{тр}$)*, представленное на ЯФ, должно синтаксически отличаться от других компонентов формулировки $З_k$;
в) ЯФ должен служить лишь целям представления и распознавания смысла сообщений в рамках задачного подхода и не предназначаться для прямого перевода на него фраз естественного языка. Однако, как и в случае информационно-логического языка (ИЛЯ), наиболее близкого к ЯФ по своей методической роли и выразительности, предложения на ЯФ должны соотноситься по своей структуре с предложениями естественного (русского) языка.
Наиболее критическим моментом при построении ЯФ является все же степень его приближения к естественному языку. Здесь надо учитывать следующие обстоятельства. Требования к языку обмена сообщениями зачастую противоречивы. Например, при взаимодействии с ЭВМ пользователи высокой квалификации на первое место ставят лаконичность и мощность входного языка машины в ущерб близости к естественному разговорному языку. Для неподготовленных пользователей первостепенным является обмен сообщениями на языке, близком к естественному. Однако в естественном языке по внешнему виду предложений зачастую бывает трудно: однозначно распознать утверждения, вопросы и императивы; определить, какой объект «поставлен под вопрос» или «под императив»; вывести утверждение, являющееся ответом на вопрос или результатом выполнения императива.
Для эффективного решения этих задач необходимо ввести соответствующий формализованный язык.
Представление знаний машины в виде отношений является хорошим компромиссом между естественностью и строгостью входного языка. Действительно, при взаимодействии с ЭВМ пользователь зачастую представляет себе знания машины как систему изменяющихся во времени нормализованных отношений различной степени. Изменения подразумевают вставки, замены и коррекции строк отношения; за такими изменениями легко следить как в «уме», так и на бумаге, представляя соответствующие отношения в виде всевозможных таблиц с разным заполнением.
В языке формализации, как и в естественном, будем выделять:
1) повествовательные предложения или утверждения, высказывания, информативы (П);
2) вопросительные предложения или вопросы (В);

3) повелительные предложения или императивы (И).

В настоящее время известны три основные группы языков формализации этих предложений: исчисление предикатов первого порядка и теории отношений для представления информатив; эротетическая логика для описания вопросов и деонтическая логика для описания императивов.

Мы предлагаем язык для формализации всех трех видов предложений. Впервые попытка построить такой язык была предпринята А.М.Довгяло совместно с Б. А. Платоновым в 1973 г., когда были сформулированы (раздельно!) три основные подмножества ЯФ — язык повествовательных предложений, язык вопросов и язык повелительных предложений.

Алфавит языка формализации можно представить в следующем виде.

1. $A, B, A_1, \ldots$ — названия или имена индивидуальных (константных) предметов или их состояний $A, B, A_{1i}, \ldots$ В качестве названий предметов будем также использовать слова и словосочетания русского языка, представленные строчными буквами (это облегчает перевод элементарных сообщений с естественного языка на ЯФ и наоборот).
2. $x_1, x_2, \ldots$ — предметные переменные. Значениями этих переменных являются имена индивидуальных предметов из некоторого множества предметов. Если $A_1, A_2, \ldots$ — предметы некоторой предметной области, то значение x — любой элемент множества A.
3. $R_1, R_2, \ldots$ — названия конкретных отношений, или свойств. В качестве названия отношений или свойств будем также использовать слова и словосочетания русского языка, представленные прописными буквами.
4. $X_1, X_2, \ldots$ — названия переменных отношений. Значением каждой из этих переменных может быть конкретное свойство или отношение из некоторого множества отношений или свойств.
5. $z_1, z_2, z_3 \ldots$ — названия переменных для высказываний, имеющих одно из двух значений: «истина» и «ложь».
6. $\neg, \&, \vee, \supset$ — логические операторы, имеющие соответственно значения «НЕ», «И», «ИЛИ», «ЕСЛИ ... , ТО».
7. $\forall$ — квантор общности, читаемый как слова «все», «всякий».
8. $\exists$ — квантор существования. Имеет то же значение, что и слово «существует» или слово «некоторый».
9. Символ «?» — оператор вопроса. Он «ставит под вопрос» перечисленные в п. 1—8 переменные или постоянные объекты. Обычно знак вопроса ставится в конце предложения, поэтому только лишь по смыслу предложения можно догадаться, к какому члену предложения он относится. При формализованном задании вопросов

на описываемом здесь языке объект под вопросом указывается непосредственно за символом «?».

10. Символ «!» — оператор императива. Он «ставит под императив» некоторое отношение (или предикат) действия. Содержательно оператор императива интерпретируется как словосочетание: «Проделать действие...», где следующий после «!» символ — имя операции или процедуры, которую надлежит выполнить.

Формулы в ЯФ определяются следующими правилами.

1. Символы z_1, z_2, z_3... являются формулами. Этими символами мы обозначаем повествовательные предложения, имеющие одно из двух значений: «истина» или «ложь».

2. Если f обозначает n-местное отношение (n-местный предикат), постоянное (R) или переменное (X), а $(k_1, k_2,...,k_n)$ есть подпредикатное выражение, являющееся кортежем предметных постоянных $A, B, C,...$ или переменных $x_1, x_2,$, то $f(k_1,.....k_n)$ — формула.

3 Если $f(x)$ — формула, содержащая предметную переменную x, то $\forall\, xf(x)$ и $\exists\, xf\,(x)$ — формулы.

4. Если f_1 и f_2 — формулы, то $\neg\, f_1$, $f_1 \& f_2$, $f_1 \vee f_2$, $f_1 \supset f_2$ — формулы.

5. Если $f(y)$ — формула, не содержащая символа «?», и y — имя объекта, имя отношения, переменная или квантор, то $(?y)f(y)$ — формула, называемая вопросом. Наличие в формуле $(?y)f(y)$ символа «?» рядом с символом y означает, что компонент у формулы $f(y)$ «поставлен под вопрос».

Пример.

1. «Какой город является столицей Литовской ССР?»	$(?x)\ R\ (x, A)$, где R — отношение «быть столицей»: R [ГОРОД | РЕСПУБЛИКА], x — переменный объекта из области значений ГОРОД; A — Литовская ССР.
2. «Все ли предметы из области значений D_1 обладают свойством R_1?»	$(?\forall)\ \forall\, x_1 R_1 (x_1)$

6. Если $f(y)$—формула, не содержащая символа «!», и y — имя объекта или отношения, переменного или постоянного, то $(!y)\ f\ (y)$—формула, называемая императивом. Наличие в императиве символа y рядом с символом «!» означает, что компонент y формулы $f\ (y)$ поставлен под императив.

Пример. Императив $И_1$: «Сложить A и В».

$И_1$на $ЯФ_1$ $(!x_1)$ СЛОЖИТЬ *(x1, A, B),* где x_1 — значение переменной из области определения СУММА отношения

СЛОЖИТЬ	СУММА	первое слагаемое	второе слагаемое

Этот пример иллюстрирует императив получения объекта.
Императив получения отношения X_1 может быть представлен формулой $(!X_1)f(X_1)$.

Пример. Императив $И_2$: «Получить формулу X_1 объема V_A правильной четырехугольной усеченной пирамиды *A,* если даны ее высота *h,* сторона *a* верхнего основания и сторона *b* нижнего основания».
$И_2$ на $ЯФ_1$ $(!X_1)X_1(V_A, A, h, a, b)$.

Рассмотрим основные типы вопросов.
Формулы $(?y)f(y)$ и $(?y)\neg f(y)$ — простые вопросы, если *f(y)* не содержат операторов &, $\vee$, $\supset$,?.
Если *y* в формуле *(?y)f(y)* — постоянный объект или отношение, то *(?y)f(y)* — прямой вопрос.
Если *y* в формуле *{?y)f(y)* — переменная, то *(?y)f(y)* — косвенный или непрямой вопрос.
Вопросы типа «сколько...», «почему...», «когда...», «как...» могут быть перефразированы в вопросы «какое количество...», «какая причина...», «в какой момент времени...», «каким образом...», и далее они легко представляются на $ЯФ_1$ как косвенные вопросы.
Если после приема вопроса $В_i$ реципиент становится информатором и выдает сообщение типа $П_j$ (информатива, высказывание), то сообщение $П_j$ является ответом на вопрос $В_i$.
Правильным ответом на прямой вопрос *(?y)f(y)* будем считать высказывание *f(y)* (ответ типа «да*) или высказывание $\neg$ *f(y)* (ответ типа «нет»).
Получение правильного ответа на вопрос — операция не чисто формальная, поскольку, давая ответ, решающая система должна сформировать высказывание *f(y)* таким, чтобы его логическое значение стало истинным, в частности, находящимся в некотором отношении со знаниями «отвечающей» решающей системы (например, *f(y)* входит в знания этой РС).
Если высказывания *f(y)* или $\neg$ *f(y)* не входят в знания отвечающего (либо не могут быть им получены) и отвечающий осознает этот факт, то он обычно сообщает об этом в ответе типа «не знаю».
Все другие ответы на прямые вопросы, не являющиеся правильными или ответами типа «не знаю», назовем неправильными.

Правильным ответом на косвенный вопрос (?*y*)*f*(*y*) будем считать либо высказывание *f(A)* или *f(R),* образовавшееся из *(?y) f(y)* после отбрасывания компоненты под вопросом и замены в оставшемся выражении переменных *y* на постоянные *A* и *R* из области значений *y,* либо высказывание $\neg \exists\, yf(y)$ - «не существует *y* такое, что...».

Как и и случае прямого вопроса, получение правильного ответа решающей системы на косвенный вопрос не является чисто формальным преобразованием, а основывается на знаниях РС области определения переменной *y.* Помимо указанных ответом на косвенный вопрос может быть неопределенный ответ типа «не знаю». Все другие ответы на косвенный вопрос, не являющиеся правильными или неопределенными, будем называть неправильными.

Введенный нами ЯФ позволяет уточнить и интуитивные представления о правильности вопросов.

Изложим правила, по которым можно определить, является ли данный вопрос правильным и, соответственно, может ли в принципе быть получен на него правильный ответ.

Правило 1. Простой вопрос *(?y) f(y)* является правильным тогда и только тогда, когда в выражении *f(y)* объекты, обозначаемые символом *y,* существуют, если *y* — постоянная, либо область значений *y* не пуста, если *y* — символ переменной.

Правило 2. Простой вопрос *(?y)f(y)* является правильным тогда и только тогда, когда в выражении *f(y)* не встречаются одновременно свободные предикатные переменные и свободные предметные переменные.

Правило 3. Простой вопрос *(?y) f(y)* является правильным тогда и только тогда, когда разница между количеством свободных переменных, содержащихся в *f(y),* и количеством переменных в компоненте под вопросом равна нулю.

Сложные вопросы образуются из простых вопросов или из вопросов и формул типа 1, 2 и 3 с помощью символов &, $\vee$, $\supset$. Правила определения правильности сложных вопросов на $ЯФ_1$ такие же, как и для сложных вопросов в языке L_q или в эротетическом языке .

Рассмотрим основные типы императивов.

Формулы $(!y)\ f(y)$ и $(!y)\neg f(y)$ — простые императивы, если *f()* не содержит операторов !, &, $\vee$, $\supset$.

Если простая формула *(!y) f(y)* не содержит переменных компонентов, то эту формулу назовем императивом подтверждения.

Если простая формула *(!y)f(y)* содержит переменные объекты или отношения, то такую формулу назовем императивом получения (нахождения).

Если после приема императива $И_i$ реципиент становится информатором и выдает сообщение типа $П_j$ (информатива, высказывания), то сообщение $П_j$ назовем отчетом о выполнении императива.
Для идеально-направленных задач, у которых предмет задачи является знаковой моделью, отчет о выполнении обычно включает и результат решения.
Правила 1—3 определения формальной правильности простых вопросов полностью применимы и для простых императивов.
Рассмотрим более подробно особенности императивов, основываясь на модели решающей системы.
Если реципиентом императива является решающая система, то компонентом под императивом есть элемент множества операторов *{σ}* (процедур {∑}) реципиента либо объекты, получаемые в результате выполнения *σ* или ∑.
Пусть (!∑)∑ *(A_1,..., A_n)* — простой императив подтверждения. Тогда правильным отчетом о выполнении императива подтверждения является высказывание $\Sigma(A_1, \ldots, A_n)$ или $\neg\Sigma(A_1, \ldots, A_n)$, получившееся из формулы (!∑)∑ *(A_1, ..., A_n)* после отбрасывания компоненты под императивом.
Пример. Императив «Включить устройство A_i» на $ЯФ_1$ представляется формулой: (!ВКЛЮЧЕНО) ВКЛЮЧЕНО *(A_1)*. Отчет о выполнении включения выражается высказыванием ВКЛЮЧЕНО *(A_1)* — «устройство включено» — или ¬ ВКЛЮЧЕНО *(A_1)*.
Истинность высказываний $\Sigma(A_1, \ldots, A_n)$ и $\neg\Sigma(A_1, \ldots, A_n)$ устанавливается по тому объекту или состоянию объекта, которые явились результатом выполнения операции или процедуры ∑.
Легко увидеть аналогию между правильными отчетами о выполнении императива подтверждения (∑ *(A_1,..., A_n)* — «СДЕЛАНО», $\neg\Sigma(A_1, \ldots, A_n)$ — «НЕ СДЕЛАНО») и правильными ответами на прямой вопрос («ДА», «НЕТ»). Однако в отличие от прямых вопросов императивы подтверждения не ограничиваются отчетом («СДЕЛАНО», «НЕ СДЕЛАНО») о состоянии знаний РС, но и подразумевают осуществление соответствующих операций или процедур. В этом смысле показательным является сопоставление прямого вопроса с императивом подтверждения, требующим доказать, что некоторые объекты находятся в определенном отношении.
Пример.

$И_1$: «Доказать, что A и B находятся в отношении» $И_1$ на ЯФ: $(!R)R(A, B)$

$В_1$: «Находятся ли A и B в отношении R?»

$П_1$ { а) СДЕЛАНО, $R(A, B)$; б) $A \xrightarrow{\Sigma} B$, причем $\Sigma(A, B)$ эквивалентно $R(A, B)$ }

$В_1$ на ЯФ: $(?R)R(A, B)$

$П_1$: $DA, R(A, B)$

В этом примере основным результатом выполнения императива $И_1$ является получение процедуры $\sum$ — доказательства того, что A и B находятся в отношении R. При этом операторы, задающие процедуру $\sum$, должны входить в состав системы реципиента императива $И_1$. Только в этом случае императив $И_1$может рассматриваться как правильно выполненный.

Таким образом, при прочих равных условиях обработка императива подтверждения для реципиента — более сложная задача, чем ответ на эквивалентный этому императиву прямой вопрос. Этот вывод подкрепляют также результаты анализа протоколов решения задач.

Пусть $(!x_1)\sigma(x_1, A_2,..,A_n)$ — простой императив получения. Тогда правильным отчетом о выполнении этого императива являются:

1) высказывание $\sigma(A_1, A_2,..,A_n)$, образованное из формулы $(!x_1)\ \sigma\ (x_1, A_2,..., A_n)$ путем отбрасывания компоненты под императивом и замены в оставшемся выражении переменной x_1 на постоянную A_1, являющуюся результатом выполнения операции σ;

2) высказывание $\neg \exists x_1 \sigma(x_1, A_2, \ldots, A_n)$ — «не существует такое x_1, которое нужно было получить в результате операции σ».

Истинность высказываний $\sigma(A_1, A_2,..., A_n)$ и $\neg \exists x_1 \sigma(x_1,..., A_n)$ устанавливается по наличию или отсутствию «нового» объекта A_1.

Несмотря на внешнюю аналогию между правильными отчетами о выполнении императива получения и правильными ответами на косвенный вопрос, в отличие от косвенных вопросов, подразумевающих, вообще говоря, выбор среди множества имеющихся объектов, императивы получения направлены на расширение этого множества за счет конструирования, формирования нового объекта. При прочих равных условиях вторая операция труднее первой, поэтому в протоколах решения задач часто можно усмотреть замену императива получения на эквивалентный косвенный вопрос. Так, например, шаг 3 протокола решения задачи, приведенный в примере в конце темы 14, по существу подразумевает замену императива «Получить формулу объема V_A усеченной пирамиды А, для которой известны высоты h, сторона верхнего основания a и сторона нижнего основания b» — $\{!X_1)X_1(V_A, A, a, b, h)$ — на косвенный вопрос «Какая формула из области значений ФОРМУЛЫ ОБЪЕМА является форму-

лой объема пирамиды А?» — *(?X_1)X_1(VA, Л, a, b, h).* Таким образом, потенциальная выполнимость императива получения устанавливается по наличию у РС-реципиента операторов (или процедур), необходимых для формирования объекта, имя которого находится под императивом.

14.7. Основные операторы решающей системы

Следующие шаги в детализации компонентов и подсистем решающей системы будут направлены на:
1) определение тезауруса РС;
2) описание основных операторов автономной решающей системы.
Как всякая целенаправленная система для решения $З_k$ $РС_j$ должна (при решении $З_k$ целью решающей системы является требование задачи $(K_{тр})$) :
1) формировать адекватный текущий образ внешней среды, т.е. $К^*_{акт.j}$;
2) обладать априорной информацией (знаниями) о внешней среде, в частности, располагать моделью предметной области $К^*_j$;
3) обладать информацией о самой себе, о своих свойствах и возможностях, т. е. располагать моделью $(PC_j)\ ^*_j$.
Определение 34. Знаниями PC_j назовем подсистему $U^*_j = \{\{u^*\}, \{R_u\}\}$, где $\{u^*\}$ — множество знаковых моделей (понятий, определений и т. п.), а $\{R_u\}$ — множество отношений над $\{u^*\}$.
Для решения $З_k$ знания PC_j должны включать некоторую модель $К^*_j$ предмета задачи (предметной области) K или K^*, т. е. $К^*_j \subset U^*_j$.
Определение 35. Тезаурусом решающей системы назовем такую ее подсистему PC_{jj}, компоненты которой являются моделями знаний и умений PC_j.
Компонентами тезауруса являются модели операций μ* и ν*, хранящиеся в долговременной памяти РС, отношения типа $R_{ин}$ (известно—неизвестно) и др.
Классификация операторов и процедур РС. Необходимость осуществлять целенаправленное изменение $K_{акт}$ в $K_{тр}$ на основе изменения сложных внутренних операндов типа $К^*_j$, $\{\sigma^*_j\}$, $\{K^*_{акт,j}\}$, введенных в п.14. 4, требует дальнейшей дифференциации $\{\sigma\}$ и $\{\sum\}$. В связи с этим будем выделять: операции ω и процедуры Ω, обеспечивающие непосредственное изменение $K_{акт}$ в $K_{тр}$ или $K^*_{акт}$ в $K^*_{тр}$ при решении идеально-направленных задач; операции υ и процедуры Θ, обеспечивающие формирование процедур Ω_k решения $З_k$ либо

применение определенных операторов первой группы ω, «в нужном месте» процесса решения $З_k$.

Примерами операторов первого рода являются:

1) операторы процедурных языков программирования;
2) арифметические и алгебраические операторы, которыми владеет человек;
3) внешние операторы манипулирования материальными объектами (операторы ДВИГАТЬСЯ, СХВАТИТЬ, ОТПУСТИТЬ, ПОСТАВИТЬ, ПОДНЯТЬ и т. п., выполняемые людьми и роботами);
4) внутренние операторы реляционной алгебры, операндами которых являются отношения; к таким операторам относятся ОБЪЕДИНЕНИЕ, РАЗДЕЛЕНИЕ, ПРОЕКЦИЯ (РЕДАКТИРОВАНИЕ), УПОРЯДОЧЕНИЕ, ДОБАВЛЕНИЕ, УДАЛЕНИЕ, ЗАМЕНА и др.

Процедуры, описывающие процессы решения задач, как правило, составляют из операторов первого рода. Только специальные методы анализа и наблюдения позволяют выявить в естественных решающих системах «над» операторами и процедурами первого рода операторы и процедуры второго рода, которые формируют первые и управляют ими.

Рассмотрим прежде всего множество внутренних операторов второго рода, характерных в первую очередь для решения теоретических идеально-направленных нерутинных задач.

Предлагаемый здесь набор операторов $\{\upsilon\}$ абстрагирован от какой-либо содержательной предметной области (в том числе и от типа операторов первого рода $\{\omega\}$) и является в значительной мере универсальным, т. е. свободным от «профессиональной ориентации» решающей системы. Это позволит анализировать с единых позиций процессы решения задач человеко-машинными системами, а также в определенной степени стандартизировать описание того материала по обобщенным описаниям процессов решения задач, который накоплен психологией мышления и педагогикой.

Набор операторов второго рода является результатом исследований, выполненных А.М.Довгяло в 1970—1977 гг. Эти исследования включали: анализ протоколов решения задач в работах психологического характера и др.; разработку и опробование человеко-машинного решения простых инженерных задач и задач обработки данных.

Выделим следующие внутренние операторы второго рода:

1) оператор α, обеспечивающий изменение, в том числе образование, первичное формирование в РС модели требуемого состояния предмета задачи;

2) оператор β, обеспечивающий изменение, в том числе образование в РС, модели актуального состояния предмета задачи;
3) оператор γ, обеспечивающий установление отношений или их моделей над находящимися в памяти *РС* компонентами $K^*_{тр,j}$, с одной стороны, и компонентами U^*_j, K^*_j и $K^*_{акт,\,j}$ — с другой (частным видом этих отношений могут быть связи «. .умственная работа решающего предстает перед нами как воскрешение относящихся к делу элементов его опыта, как связь этого опыта с решаемой задачей, как мобилизационная и организационная работа»);
4) оператор оценки и принятия решения к, обеспечивающий переход решающей системы к одному из имеющихся у нее операторов {σ}, к одной из процедур {∑} либо к одной из их моделей (термину «принятие решения» соответствуют также термины «выбор эвристики или эвристической процедуры», «выбор дальнейшего пути», «выбор стратегии решения» и др. Принятие решения человеком зачастую протекает в свернутом виде, так что интерпретация оператора к на материале протоколов решения задач человеком во многих случаях затруднена. В частности, Ю. Н. Кулюткин пишет по этому поводу. «Человек, ищущий решение, обычно не осознает, что он делает выбор из множества возможных вариантов. Главное для него не выбрать, а построить решение. Однако объективно такое множество возможных ходов — правильных и неправильных — всегда существует...»);
5) оператор планирования η, обеспечивающий формирование или извлечение из памяти РС плана решения P_k, который представляет собой обобщенную императивную модель процедуры решения задачи;
6) оператор программирования ζ, обеспечивающий извлечение из памяти РС моделей операции или процедуры первого рода, т. е. ω_k или Ω_k, выполняющей непосредственное изменение $K_{акт}$ в $K_{тр}$ или $K^*_{акт}$ в $K^*_{тр}$ и подстановку этих моделей вместо компонентов плана P_k. Языковым представлением программы Ω^*_k является только язык операторов и процедур первого рода (в отличие от плана P_k, который может описываться и в терминах {Θ}, {υ}, подзадач задачи $З_k$ и др.).

П р и м е р. Рассмотрим работу решающей системы, в роли которой выступает группа по тушению пожара на некотором объекте. Требуемым состоянием предметной области является потушенный объект. Лицо, ответственное за принятие решений по ликвидации пожара, именуемое далее $РС_1$, получает информацию о возникновении пожара на объекте и положении вещей там в это время, т. е. о $K_{акт}$, решает свою подзадачу $ПЗ_{k1}$, заключающуюся в выработке модели процедуры Ω^*_k тушения пожара на объекте и передает эту

информацию в виде предписания спасательным группам, обладающим операторами $\{\omega_1,..., \omega_n\}$ и средствами тушения пожаров.
Принятие задачи решающей системой $РС_1$ заключается в создании своих моделей $K^*_{акт,\,1}$ и $K^{**}_{тр,\,1}$ с помощью соответственно операций второго рода β_1 и α_1. Затем после «увязки» $K^{**}_{тр,\,1}$ и $K^*_{акт,\,1}$ с помощью операции второго рода γ_1 она пытается найти готовую модель процедуры Ω^*_k в своих знаниях либо в справочных материалах. Если это ей не удается сделать, то $РС_1$ пробует разыскать рекомендации или план P_k решения задачи $З_k$, чтобы облегчить себе построение модели процедуры Ω^*_k из моделей $\{\omega^*_1,..., \omega^*_n\}$.
Если готовый план не обнаружен, то $РС_1$ разрабатывает такой план с использованием операции η_1, причем вначале она пытается переформулировать аналогичный план, либо извлечь частный план из более общего. В случае неудачи «работают» операции: γ_1, подготавливающая формат «базового» отношения между» $K^{**}_{тр,\,1}$ и $K^*_{акт,\,1}$, и $к_1$, выбирающая общую стратегию планирования решения $З_k$ и т. д. Общая схема РС с тезаурусом и операторами второго рода дана на рис. 4.

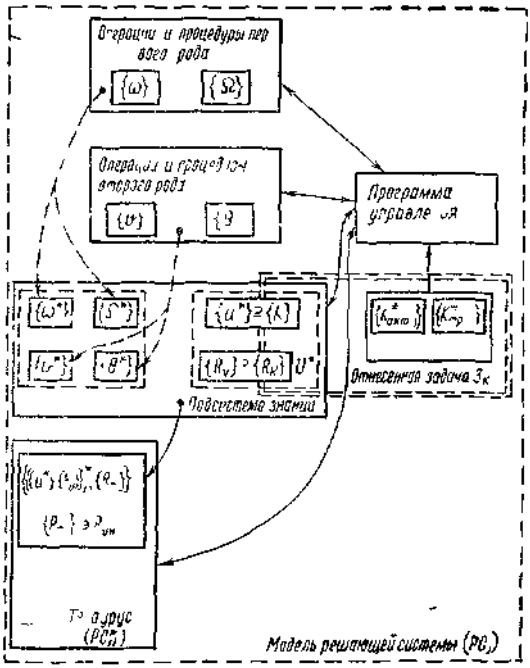


Рис. 4.

Развернутая иллюстрация того, как «работают» и как связаны между собой операторы {υ}, дана в примере, приведенном в конце раздела 14. Большая часть протокола решения стереометрической задачи, приненного в этом примере, показывает работу операторов второго рода. И только в заключительной части протокола (шаги 9—11) фигурируют операторы первого рода — сложение, вычитание, деление, возведение в степень и другие, составляющие процедуру первого рода решения данной задачи, т. е. процедуру вывода формулы объема правильной четырехугольной усеченной пирамиды из выражений, полученных с помощью операторов второго рода.

Дадим более подробное описание каждого из операторов второго рода.

Оператор α изменения модели требуемого состояния предмета задачи. Первая реализация оператора α обеспечивает ОБРАЗОВАНИЕ модели $K^*_{тр,\,1}$ из целей и тезауруса (в частности, образование $K^*_{тр}$ из тезауруса решающих систем основывается на выделении из отношений типа $R_{ин}$ (ИЗВЕСТНО — НЕИЗВЕСТНО) строк или строки, имеющих в области определения ПОЛНОТА ИНФОРМАЦИИ величину НЕИЗВЕСТНО) решающей системы, а также из формулировки задачи, переданной решающей системе извне. Образование $K^*_{тр,\,1}$ из такой ФЗ представляет собой также изменение внешней задачи во внутреннюю (см. определения 24, 25).

Образование модели $K^*_{тр,\,1}$ из $ФЗ_k$ с учетом необходимости кодирования внешней $ФЗ_k$ (оператор ν) можно представить следующим образом: $K^*_{тр,\,1} = \alpha\,(\nu\,(ФЗ_k))$.

Примеры применения α при образовании $K^*_{тр,\,1}$ из формулировки задачи $ФЗ_k$ и изменения $K^*_{тр,\,1}$ в $K^*_{тр,\,2}$ даны в примере, приведенном в конце темы 14 (шаги 1 и 6 протокола решения задачи).

Если $K^*_{тр}$, K^* и $K^*_{акт}$ хранятся в решающей системе в форме множеств или отношений, очередная операция α_t обеспечивает СУЖЕНИЕ, РАСШИРЕНИЕ и ПЕРЕОПРЕДЕЛЕНИЕ $K^*_{тр,t-1}$. В последнем случае первоначальная задача заменяется родственной ей задачей, причем такой, для которой у $РС_j$ предположительно имеется больше средств решения, чем для первоначальной задачи.

Оператор β изменения модели актуального состояния предмета задачи. Первая реализация оператора β обеспечивает ОБРАЗОВАНИЕ модели $K^*_{акт,1}$ из: формулировки задачи; знаний U^*_j; генерируемой информации; объектов внешнего мира (В этом случае в $K^*_{акт}$ включаются объективные знания K_0, частью которых являются знания $РС_j$, $K^*_j \subset K_0$. Вторым вариантом применения β является построение $K^*_{акт}$ из оригинала (моделируемого с помощью β).

Пример ОБРАЗОВАНИЯ $K^*_{акт.1}$ из формулировки задачи дан в примере, приведенном в конце раздела 14.

Последующие реализации β обеспечивают СУЖЕНИЕ, РАСШИРЕНИЕ (расширение $K^*_{акт}$ зачастую происходит за счет моделей, извлекаемых из знаний $U^*_{j\ j}$-*й* системы, из ее тезауруса, из предметной области (предмета задачи) K и др.) или ПЕРЕОПРЕДЕЛЕНИЕ $K^*_{акт}$. В последнем случае один и тот же компонент $K^*_{акт}$ выступает в разных отношениях (связях) с другими объектами, привлекаемыми РС для решения задачи.

Заключительная реализация оператора β обычно имеет место в случае успешного решения подзадачи контроля (см. определение 32), когда установлено, что результат решения $З_k$, $K_{тр}$ соответствует его первоначальной модели $K^*_{тр,\ 1}$. При этом $K_{тр}$ входит в заключительное актуальное состояние предмета $З_k$, являющееся операндом операции β.

Оператор формирования отношений *(γ).* Как мы уже говорили, оператор γ обеспечивает формирование отношений R или моделей этих отношений L* над компонентами $K^*_{тр,\ j}$, с одной стороны, и компонентами $K_{акт,j}$, $U^*_j \supset K^*_j$, а также компонентами тезауруса $РС_j$ — с другой стороны. Частным видом отношений L могут быть связи.

Отметим основные особенности отношения типа «связь». Будем считать, что «два или более различных предмета связаны, если по наличию или отсутствию некоторых свойств (унарных отношений) у одних из них можно судить о наличии или отсутствии тех или иных свойств у других из них (возникновение и исчезновение предметов можно рассматривать как частный случай)».

Примерами связей могут служить функциональные и стохастические зависимости, механические сопряжения между телами, причинно-следственные зависимости, логические следования, вывод одних знаний из других и т. д.

Выявление связей позволяет познавать (выводить, находить) предметы не непосредственно, а косвенно, через другие предметы, находящиеся с ними в той или иной связи.

В отношениях L компоненты $K^*_{тр}$ являются ключевыми. Зачастую результатом операции γ является формирование только «шапки», т. е. модели L^* отношения L. Примером может служить формирование «шапки» неизвестного решающей системе отношения X в протоколе решения задачи, приведенном в примере, который описан в конце раздела 14 (шаг 2).

В том же протоколе (см. шаг 4) представлено формирование отношения ЧАСТЬ.

Таким образом, операторы α и β формируют множества предметов, над которыми оператор γ задает отношение L или модель этого отношения L^*. В первом случае определяется предполагаемый вид зависимости

между искомым и данными, во втором — указывается на факт существования этой, зависимости.

Определение 36. Задачной системой (ЗС) назовем такую задачу, которая рассматривается исследователем вместе с отношением (отношениями) L (или L^*), заданным над компонентами K, $К_{акт}$ и $К_{тр}$* и (или) моделями этих компонентов.

Примеры задачных систем даны в примере, который описан в конце раздела 14 (шаг 2, шаг 7).

Частным видом ЗС является функция различия между $К_{тр}$*и $К_{акт}$*, нередко используемая в работах по автоматическому решению задач. В общем виде эту функцию можно представить следующим отношением:

Примеры функций различия даны в примере, который описан в конце раздела 14. Там, как и во многих других аналогичных случаях, функция различия представлена в виде множества факторов, препятствующих определению характера отношения между $К_{тр}$*и $К_{акт}$*.

Оператор к оценки и принятия решения. На основе оценки сложившейся на t-ом шаге ситуации, которая описывается множествами $\{ЗС_t\}$, $К_{тр,t}$*и $К_{акт,t}$*, t= 1,..., оператор к производит выбор среди возможных альтернатив и формирует императив $И_{t,k}$, наиболее обобщенную модель процедуры решения $З_k$ («идею» решения задачи).

Примеры императивов $И_{t,k}$ приведены в примере, который описан в конце раздела 14. Шаг 3: «Найти в отношении ОБЪЕМ формулу объема правильной усеченной прямоугольной пирамиды». Шаг 4: «Сформулировать задачу, близкую или аналошчную $З_k$, и притом такую, для которой у решаемой системы больше знаний». Шаг 5 данного протокола иллюстрирует, как происходит оценка продвижения в решении задачи с помощью оператора $к$.

Приведем примеры императивов И в обобщенном виде: $И_{поиск}$ — найти (отыскать) в памяти готовый результат или алгоритм (квазиалгоритм) получения результата; $И_{пз}$ — свести (сузить) задачу к подзадаче; $И_{расш}$ — расширить задачу З до задачи З' так, чтобы З явилась подзадачей З'; $И_{рф}$ — переформулировать задачу З в аналогичную, «родственную» задаче З, притом такую, для которой у решающей системы имеется (во всяком случае, предположительно) больше средств решения, чем у задачи З. («Нахождение пути к решению задачи, кажущейся недоступной, при помощи специально для этого придуманной, а затем

решенной, вспомогательной задачи — это одно из наиболее характерных проявлений умственной деятельности»).
Поскольку принятие решения зачастую трудно отделить от получения оценки λ_t ситуации $(K^*_{акт,\,t},\ K^*_{тр,t},\ L_t)$, то будем считать, что оператор к вырабатывает абсолютные и относительные оценки, на основе которых производится управление порядком выполнения операторов $\{\upsilon\}$, «включение» процедуры Ω_k решения $З_k$, прекращение решения $З_k$ в результате успешного решения подзадачи контроля $ЗК_k$ и др.
Получение оценок может рассматриваться также как отдельная подзадача, прежде всего, если: недостаточно данных в самой РС; РС не располагает планом и (или) программой оценки; для формирования оценки необходимо совершить «пробные действия», т. е. реализовать частичный план и частичную программу решения $З_k$, чтобы «посмотреть, что из этого получается».
В связи с тем, что поиск информации в памяти РС или во внешнем мире является существенной подзадачей многих задач, рассмотрим более подробно отработку решающей системой императива поиска $И_{поиск}$. Здесь возможны следующие варианты:
а) РС располагает процедурой поиска;
б) РС не располагает процедурой поиска, но имеет программу или план поиска;
в) у РС нет ни плана, ни программы поиска.
Если РС — это ЭВМ, то первый вариант поиска соответствует случаю, когда машина располагает соответствующей рабочей программой поиска (абсолютной фазой), готовой к выполнению. Если же речь идет о человеке, вспоминающем необходимые данные, то в первом случае у него отработка императива $И_{поиск}$ также зачастую протекает в свернутом виде: операции первого рода, составляющие процедуру поиска, выполняются слитно, подсознательно.
В случае варианта в) целесообразно специально рассматривать подзадачу поиска. В рамках нашей модели РС формулировкой подзадачи поиска является косвенный вопрос, т. е. вопрос типа «Какой объект...», «Какое отношение...».
Решение этой подзадачи в свою очередь начинается с операторов α, β, γ и т.д. Подзадача поиска приобретает еще более ярко выраженную самостоятельность в случае необходимости произвести поиск не в памяти РС, а во внешней среде.

Оператор планирования η. Оператор η производит изменение декларативного представления знаний РС по задаче $З_k$ $(ЗС_k,\ К^*_{тр},\ К^*_{акт})$ в императивное (процедурное) представление, т. е. в план P_k решения $З_k$. План P_k представляет собой обобщенную модель процедуры решения задачи. Компонентами плана могут быть императивные

модели разных по своей природе объектов — операций первого рода, операций второго рода, формулировки подзадач и др. Так, в плане, приведенном в примере, который описан в конце раздела 14 (шаг 4), первым компонентом является модель операции первого рода (вычитания), выражающей основную идею решения $З_k$ —определение объема усеченной пирамиды как разности объемов полной пирамиды, включающей усеченную, и «малой» полной пирамиды, дополняющей усеченную до первой полной пирамиды. Вторым и третьим компонентами плана P_k являются в этом примере формулировки двух подзадач, на которые решающий разделил $З_k$.

План развертывается, т. е. «накапливается» и уточняется решающей системой пошагово, так что компонент плана P_t произведенный на t-м шаге, добавляется к P_{t-1} либо заменяет в P_{t-1} некоторый компонент, образуя P_t. Если РС не располагает готовым планом решения $З_k$, то, как правило, среди операндов первой по счету операции η_1 нет плана или его компонентов.

Оператор программирования ζ. Программирование плана P_k будем рассматривать как операцию, осуществляющую подстановку моделей операций или процедур первого рода (из $\{ \omega^* \}$ и $\{\Omega^*\}$) вместо тех компонентов плана, которые не принадлежат к множеству $\{\omega^*\}$ или $\{ \Omega^* \}$.

Обычно план содержит меньшее число компонентов, чем соответствующая ему программа. Например, в примере, который описан в конце раздела 14 (шаг 9) последний компонент плана ω_4^* для получения x может быть расписан в следующую программу:

ω^*_{41}: ИЗБАВИТЬСЯ ОТ ЗНАМЕНАТЕЛЯ.

ω^*_{42} : ПЕРЕНЕСТИ НЕИЗВЕСТНЫЕ В ЛЕВУЮ ЧАСТЬ УРАВНЕНИЯ, А ИЗВЕСТНЫЕ — В ПРАВУЮ.

ω^*_{43} : ВЫНЕСТИ НЕИЗВЕСТНЫЕ ЗА СКОБКИ... и т.д.

Рассмотрим более подробно соотношение операции планирования и операции программирования.

Если план P_k рассматривать как компонент $K_{акт}$, та его языковым носителем будет язык всей той предметной области K, которая входит в $З_k$. Языком описания программы Ω_k^* является только язык моделей операторов (и процедур) первого рода в модальности «требование». Особенно наглядно языковое различие между P_k и Ω_k^* проявляется при разработке программ для ЭВМ, где обычно план P_k (метод решения задачи $З_k$) описывают на языке физики, экономики и т. п., а программу Ω_k^* — на алгоритмических языках программирования.

Однако языковое различие между P_k и Ω_k^* не должно затемнять их принципиальной общности: и тот, и другой предмет являются

различными моделями одного и того же объекта — процедуры Ω_k решения $З_k$.

При рассмотрении решения задач человеком и ЭВМ полезно также отметить важность локального контроля правильности изменения (программирования) плана P_k в программу Ω_k*, при этом контроль может быть оформлен как подзадача синтаксической проверки и корректировки программы Ω_k* .

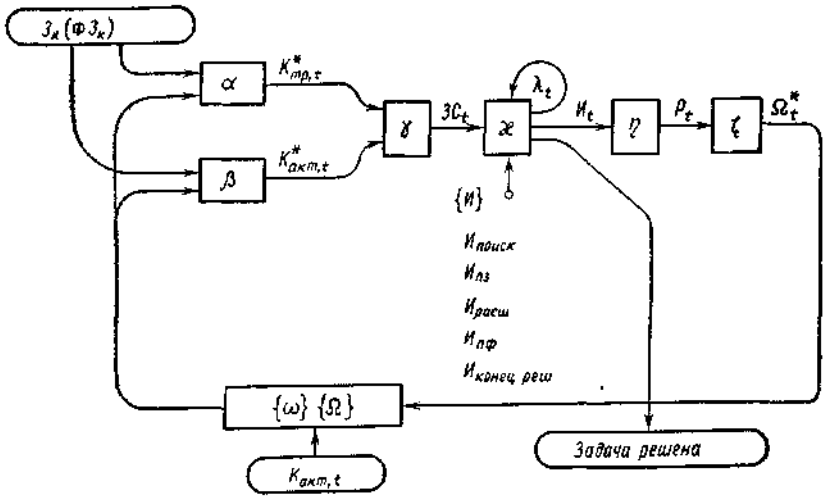


Рис. 5.

Типовая схема функционирования рассматриваемой модели

РС представлена на рис. 5.

14.8. Количественные характеристики задач и процессов их решения

Количественными характеристиками отнесенных задач и процессов их решения являются:

1) время T_k решения задачи $З_k$, т. е. период времени, потраченный решающей системой на изменение $K_{акт}$ в $K_{тр}$ ($K^*_{акт}$ в $K^*_{тр}$ для идеально-направленных задач);

2) стоимость C_k решения задачи $З_k$ (учитывающая расход ресурсов, например, памяти РС);

3) показатель Λ_k качества решения $З_k$ (точность $K_{тр}$, надежность получения $K_{тр}$ и др.).

В общем случае эффективность $Э_k$ решения $З_k$ может быть представлена с помощью функции $Э_k = F_э (T_k, C_k, \Lambda_k)$, где $F_э$ — непрерывная функция, монотонно убывающая при увеличении T_k и C_k и монотонно возрастающая при увеличении Λ_k.
Если рассматривать время T_k решения задачи как ресурс РС, то C_k должна включать стоимость времени решения задачи.
В ряде случаев при определении $Э_k$ следует учитывать также расходы $С_{раз}$ и время $T_{раз}$, необходимые для создания решающей системы, к которой должна быть отнесена $З_k$.

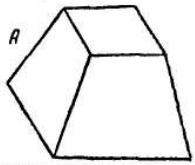
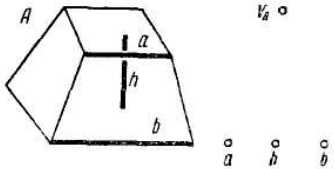
14.9. Примеры применения проблемологических категорий к описанию протоколов решения задач

А. Формулировка задачи: «Найти объем V_A правильной четырехугольной усеченной пирамиды, если даны ее высота h, сторона a верхнего основания и сторона b нижнего основания».
Здесь предполагается, что предмет этой задачи, K^*, включает знания решающего, например, из стереометрии, в том числе: модели объектов $\{k\}$ —«пирамида», «сторона», «высота», «треугольник» и др.; свойства и отношения $\{R_k\}$ — «правильная», «четырехугольная», «усеченная», «сторона нижнего основания (пирамиды)» и др.
Характерной особенностью актуального состояния этой задачи является наличие в его составе отношения $R_{ин}$ («ИЗВЕСТНО — НЕИЗВЕСТНО»);

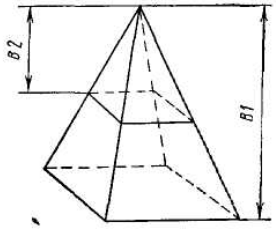
$R_{ин}$	Наименование компонента	Обозначение	Полнота информации
	Правильная четырехугольная усеченная пирамида	A	известно
	Высота	h	известно
	Сторона верхнего основания	a	известно
	Сторона нижнего основания	b	известно
	Объем A	V_A	неизвестно

Требование задачи $К_{тр}$** состоит в нахождении неизвестного решающему функционального отношения $X(A, V_A, a, b, h)$.
Данная задача относится к идеально-направленным (K^* и $К_{тр}$**) познавательным задачам.

Задача отнесена к решающей системе, обладающей необходимыми знаниями для ее решения без доступа к внешней информации. Кроме того, задача является внешней, теоретической и нерутинной, так как предполагается, что решающий не располагает ни готовым результатом (т. е. формулой $X\{A, V_A, a, b, h\}$), ни алгоритмом или квазиалгоритмом для вывода формулы объема усеченной пирамиды. С другой стороны, решающий располагает необходимыми для решения этой задачи: операторами первого рода $\{\omega\}$ — сложением, вычитанием, подстановкой и др.; полным набором операций второго рода, т. е. операторами $\{\vartheta\} = \{\alpha, \beta, \gamma, \varkappa, \eta, \xi\}$, введенными в п.14.7. Анализ протокола решения задачи разбит на шаги, каждый из которых охватывает родственную или логически завершенную группу операций. Для упрощения обозначений будем избегать символов моделей второго (**) и выше порядка, обычных при решении идеально-направленных задач. Остальные обозначения соответствуют обозначениям, принятым в разделе 14.

Номер шага	Протокол Д. Пойя	Описание протокола
1	«Что требуется? Мы задаем себе этот вопрос и стараемся как можно яснее вообразить форму тела, объем которого хотим найти (посмотрите на рисунок). Наш мысленный образ естественно интерпретировать как точку,— обозначим ее через V_A. 	Здесь происходит образование в РС первой по порядку модели требуемого состояния предмета задачи из ее формулировки $K^*_{\text{тр},1}=\alpha_1(\Phi З_k)=$ $=(!\,X)X(A, V_A, a, b, h)$.
	Но невозможно найти неизвестное, если мы про него ничего не знаем. Что дано? или *что у нас имеется?* — спрашиваем мы себя и останавливаем внимание на линиях фигуры, длина которых указана, т. е. на отрезках a, h, b. Наша мысленная картина изменилась, и отражением этого являются три точки, появившиеся на рисунке справа. Эти точки изображают данные» 	Далее происходит образование в памяти первой по порядку модели актуального состояния предмета задачи из ее формулировки $K^*_{\text{акт},1}=\beta_1(\Phi З_k)=\{A, V_A,$ $a, h, b, R_{\text{ин}\,1}\}$ $R_{\text{ин}\,1}$ Обозначение компонента / Полнота информации: b — известно; A — известно; a — известно; h — известно; X — неизвестно
2	«... мы ставим своей целью связать неизвестное V_A с данными a, h, и b — нам нужно ликвидировать разрыв между ними».	На этом шаге формируется с помощью оператора γ_1 **модель** L_1^* неизвестного пока вычисляемого функционального отношения над компонентом V_A требования задачи и компонентами a, h, и b ее актуального состояния: $X(A, V_A, a, h, b)$. Если рассматривать L_1^* вместе с предметами a, h и b, то здесь также можно говорить о задачной системе $ЗС_1$. Отношение L_1^* связывает V_A с одной стороны, и A, a, b и h,— с другой, а также указывает на «разрыв» между V_A и $\{a, h, b\}$, так как связывающая функция X пока нам неизвестна.

Номер шага	Протокол Д. Пойа	Описание протокола
3	«Но каким путем двигаться дальше, какой избрать курс? Если вы не в состоянии решить предложенную задачу, то...»	Мы считаем, что здесь Д. Пойа под «решить предложенную задачу» подразумевает попытку решающего найти (вспомнить) готовую формулу вида $X(A, V_A, a, b, h)$, т. е. здесь имели место операция $\varkappa_1$, результатом которой явилась выработка императива поиска $И_{поиска}$. Выделение подзадачи поиска—результат операции планирования η_1. Соответствующие знания решающего могут быть представлены отношением вида $R_{оф}$ (ОБЪЕКТ, ФОРМУЛА ОБЪЕМА). Поиск не увенчался успехом: в $R_{оф}$ нет строки с формулой объема пирамиды A.
4	«...попробуйте найти близкую ей более легкую задачу. В нашем случае далеко ходить не надо. В самом деле, что представляет собой неизвестное? — Объем усеченной пирамиды. А что это за геометрическое тело? Как оно ОПРЕДЕЛЯЕТСЯ? — Как часть полной пирамиды».	Здесь решающий в результате выполнения $\varkappa_2$ вырабатывает следующий императив $И_{пф}$: «Сформулировать задачу $З_k'$, близкую или аналогичную $З_k$, и притом такую, для которой у меня имеется больше знаний» Наличие дополнительных знаний, по новой задаче, например, знание формулы объема полной пирамиды, делает $З_k'$ более легкой для решающего.
5	...«Какая часть? — Часть, заключенная... Далее не будем продолжать, этого уже достаточно; сформулируем определение иначе. Усеченной пирамидой называется часть полной пирамиды, которая остается после отбрасывания малой пирамиды, отсекаемой плоскостью, параллельно основанию». «Если бы мы знали объемы этих двух пирамид,— обозначим их соответственно через V_{B1} и V_{B2}— то можно было бы найти объем V_A усеченной пирамиды: $V_A = V_{B1} - V_{B2}$. Попытаемся найти объемы V_{B1} и V_{B2}— в этом состоит наша идея!»	Здесь происходит переформулирование задачи — получение $K^*_{акт}(2)$ с помощью операции $\beta_2(K^*_{акт,\,1})$; $K^*_{акт}(2) = \{A, B1, B2, V_A, V_{B1}, V_{B2}, R_{ив2}\}$. $R_{ив2}$: Обозначение компонента — Полнота информации A — известно V_A — неизвестно V_{B1} — неизвестно V_{B2} — неизвестно

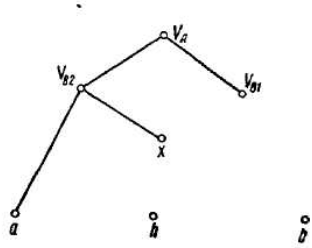
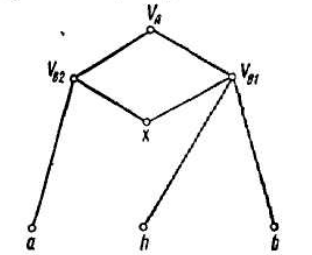
Номер шага	Протокол Д. Пойя	Описание протокола
		Далее операция γ_2 определяет отношение между объемами V_A, V_{B1} и V_{B2} как отношение целого и частей: ЧАСТЬ Целое: V_{B1}; Часть 1: V_{B2}; Часть 2: V_A
	«Итак, мы свели первоначальную задачу о нахождении объема V_A к двум вспомогательным родственным ей задачам, а именно, к нахождению V_{B1} и V_{B2}».	Операция планирования η_2 образует первый вариант плана P_2 решения $З_k$ $ФПЗ_{k1}$: ОПРЕДЕЛИТЬ ФОРМУЛУ X_{B2} ОБЪЕМА ПИРАМИДЫ $B2$. $ФПЗ_{k2}$: ОПРЕДЕЛИТЬ ФОРМУЛУ X_{B1} ОБЪЕМА ПИРАМИДЫ $B1$. $ФПЗ_{k3}$: ОПРЕДЕЛИТЬ РАЗНОСТЬ ОБЪЕМОВ V_{B1} и V_{B2}, ПОЛУЧАЯ ОБЪЕМ V_A. Операция программирования ξ_1 осуществляет подстановку модели операции ВЫЧИТАНИЕ (V_{B1}, V_{B2}, V_A) или $V_A = V_{B1} - V_{B2}$ в план вместо $ФПЗ_{k3}$, а в отношение $R_{ин2}$ вместо строки $R_{ин2}$(V_A, НЕИЗВЕСТНО) заносится строка $R_{ин2}$ (V_A, ИЗВЕСТНО).
6	«Наша работа далеко еще не закончена; нам нужно найти два неизвестных V_{B1} и V_{B2}. Однако положение не кажется безнадежным; полная пирамида, как геометрическая фигура, нам лучше знакома, чем усеченная пирамида, и хотя вместо одного неизвестного V_A появились два неизвестных V_{B1} и V_{B2}, оба они одной и той же природы и находятся в одинаковом взаимоотношении с данными величинами, соответственно с b и a. Мы приступили к ликвидации разрыва между неизвестными и данными; оставшаяся часть бреши уже первоначальной» «... решение задачи о нахождении V_A сводится к решению двух задач о нахождении V_{B1} и V_{B2}»	Здесь описывается выработка оценок $\{K^*_{тр}, K^*_{акт}\}$ до и после нахождения плана P_2. Согласно нашей модели процесс решения задачи оценки входит в операцию $ж_3$. Для выработки оценок λ_1 и λ_2 в протоколе Д Пойя учитывается: *полнота* знаний о пирамиде $A \in K^*_{акт,1}$ и пирамидах $B1$ и $B2$. Для $B1$ и $B2$ решающий располагает формулой вычисления объема, входящей в отношение $R_{оф}$ (см. шаг 3 и 4); количество несвязанных неизвестных в сравниваемых множествах $\{K^*_{тр}$ (1), $K^*_{акт}$ (1)$\}$ и $\{K^*_{тр}$ (1), $K^*_{акт}$ (2)$\}$, (два — V_{B1} и V_{B2} — во втором множестве вместо одного V_A в первом множестве); тип связи между неизвестными и данными. V_{B1} и V_{B2} находятся в

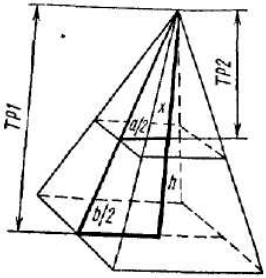
Номер шага	Протокол Д. Пойя	Описание протокола
		одинаковом отношении с b и a— сторонами оснований этих пирамид, т. е в следующем отношении (R_B):

$V_B = 1/3yz^2$

Объект B	Объем V_B	Высота y	Сторона основания z
B_1	V_{B1}	y_1	b
B_2	V_{B2}	y_2	a

		Для $\{K^*_{тр}(1),\ K^*_{акт}(2)\}$ благоприятных моментов больше, чем для первого множества, поэтому полученный план P_2 «утверждается» и РС приступает к решению подзадач, входящих в P_2
7	«Что представляет собой неизвестное V_{B2}?— Объем пирамиды. Как можно получить такой объект? Какие нужны данные, чтобы получить такое неизвестное?...— Объем пирамиды можно вычислить, если известны две величины: площадь основания и высота пирамиды... высота пирамиды не дана, но ее можно попытаться найти. Обозначим ее через x. Тогда $V_{B2}=1/3a^2x$».	Происходит образование новой модели $K_{тр,2}{}^*=(IX_{B2})X_{B2}(B_2, V_{B2}, y_2a)$, которая замещает после выполнения операции α_2 прежнюю модель $K^*_{тр,1}=$ (см. шаг 1). Формируется также (после выполнения операции β_3) из $K_{акт,2}$ и знаний решающего, представленных отношением $R_{фо}$, R_B, актуальное состояние предметной области подзадачи $ПЗ_{k1}$ $K_{акт,3}=\{B2, V_{B2}, a, x, V_B=1/3yz^2, R_{инв}\}$, где в $R_{инв}$ V_{B2} и x отмечены, как неизвестные. После выполнения операции γ_3, отмечающей, что объем V_{B2} можно вычислить с помощью отношения R_B или $V_B=1/3yz^2$, и операции ζ_2, осуществляющей подстановку величин x и a вместо y и z, V_{B2} переводится из категории «неизвестно» в категорию «известно».
8	Таким образом, к V_{B2} «можно прийти, отправляясь от x и a, т. е. V_{B2} может быть выражено через x и a Хотя все еще остаются два неизвестных (в правой части рисунка все еще нависают свободные концы), некоторый прогресс достигнут. Нам удалось связать неизвестное V_A по крайней мере с одной из данных величин, а именно с a»	Формула $V_{B2}=1/3a^2x$ — результат решения или требуемое состояние предмета подзадачи $ПЗ_{k1}$. Прежде чем перейти к решению подзадачи $ПЗ_{k2}$ РС оценивает (операция $\varkappa_4$), насколько получение решения ПЗ $_{k1}$ способствует решению первоначальной задачи $З_k$. Для этого сравнивается множество $ЗС_2=\{K^*_{тр,1},\ K^*_{акт,2},\ \omega^*_B(V_{B1}, V_{B2}, V_A)\}$

Номер шага	Протокол Д Пойя	Описание протокола
		с множеством $ЗС_2=\{K^*_{тр\ 1},\ K^*_{акт,\ 3},\ \omega^*_B\ (V_{B1}, V_{B2}, V_A),\ V_{B2}=1/3a^2x\}$ и вырабатываются сравнительные оценки λ_3 и λ_4 положения после 7-го и после 8-го шага. Сравнение в пользу ситуации после 8-го шага, так как количество неизвестных не возросло, но удалось ввести в полученную промежуточную формулу одно известное *(a)*.
9	«Следующий шаг теперь, конечно, очевиден. Неизвестные V_{B2} и V_{B1} имеют одинаковую природу, ...мы уже нашли выражение для объема V_{B2} через основание и высоту, аналогично можно выразить и объем V_{B1} $V_{B1}=1/3b^2\ (x+h)$». «На правой половине рисунка появились три новые наклонные линии, соединяющие V_{B1} с *b*, *h* и *x* Эти линии указывают, что к V_{B1} можно прийти, отправляясь от *b*, *h* и *x*, т. е. что V_{B1} может быть выражено через *b*, *h* и *x*» 	Процедура решения подзадачи $ПЗ_{k2}$ полностью аналогична процедуре решения $ПЗ_{k1}$, так как неизвестные V_{B2} и V_{B1} входят в одно и то же отношение R_B (см: шаг 6). Процедура $\Sigma_{пк2}$ включает операции α_3, β_4, γ_4, ζ_4 и $\varkappa_4$, выполняемые над операндами того же вида, что и при решении $ПЗ_{k1}$ с простой заменой *B2* на *B1*, *a* на *b* и *x* на *(x+h)*. Получение результата — формулы $V_{B1}=1/3b^2\times(x+h)$ — сопровождается операцией $\varkappa_5$, положительно оценивающей продвижение в решении $З_k$. Вместо плана P_2 у нас появился фрагмент программы решения $З_k$. ω^*_1 :ВЫЧИСЛИТЬ $V_{B2}=1/3\ a^2x$ ω^*_2 ВЫЧИСЛИТЬ $V_{B1}=1/3\ b^2(x+h)$ ω^*_3 :ВЫЧИСЛИТЬ $V_A=V_{B1}-V_{B2}$.
10	«Таким образом, остается только одна нависающая точка, не связанная с данными—точка *x* Свободное пространство еще более сузилось: теперь такое пространство имеется лишь между *x* и данными величинами». «Как можно *найти такое неизвестное*? Как можно получить подобный объект? Длина отрезка проще всего вычисляется с помощью треугольного, если это возможно) или на основании подобия двух треугольников. На нашей фигуре под-	Выделяется, с помощью операции планирования η_3 очередная (и последняя) подзадача $ПЗ_{k4}$ в решении $З_k$. «Найти высоту *x* пирамиды B2». Решение этой подзадачи выполняется с помощью тех же операторов α, β, γ аналогично *предыдущим подзадачам*. Из отношения подобия треугольников $ТР_1$ и $ТР_2$ (см рисунок) строится с помощью операции ζ_4 программа решения $ПЗ_{k4}$, представляющая собой очередной компонент программы решения $З_k$.

Номер шага	Протокол Д. Пойя	Описание протокола
	ходящего треугольника нет; кроме того, нам еще нужно, чтобы отрезок x был одной из его сторон Такой треугольник мог бы лежать, например, в плоскости, проходящей через высоту малой пирамиды с объемом V_{B2}; эта плоскость проходила бы тогда также через высоту большой пирамиды с объемом V_{B1}, которая подобна малой пирамиде». Да, нам нужны именно эти подобные треугольники, лежащие в плоскости, проведенной через высоту и параллельной стороне основания одной из наших пирамид»  	ω_4^*: ВЫЧИСЛИТЬ x из $\frac{x}{x+h}=\frac{a/2}{b/2}=\frac{a}{b}$.
11	«Мы начинаем вторую часть нашей работы там, где была закончена первая. Прежде всего мы принимаемся за введенное нами ранее неизвестное x; из последнего равенства $\frac{x}{x+h}=\frac{a}{b}$ получаем $x=\frac{ah}{b-a}$. Затем мы подставляем это значение x в два предыдущих равенства $V_{B2}=1/3\,a^2x$ и $V_{B1}=1/3\,b^2(x+h)$ и находим $V_{B2}=\frac{a^3h}{3(b-a)}$; $V_{B1}=\frac{b^3h}{3(b-a)}$. Наконец, мы используем равенство, впервые выписанное в п. 5: $V_A=V_{B1}-V_{B2}=\frac{a^2+ab+b^2}{3}h$. Это и есть искомое выражение».	Здесь программа $<\omega_1^*, \omega_2^*, \omega_3^*, \omega_4^*>$ решение $З_k$ осуществляется решающей системой; выполняется каждая из указанных ВЫЧИСЛИТЕЛЬНЫХ операций над выражениями $D_1: \frac{x}{x+h}=\frac{a}{b}$; $D_2: V_{B2}=\frac{a^2x}{3}$; $D_3: V_{B1}=\frac{b^2(x+h)}{3}$ $D_4: V_A=V_{B1}-V_{B2}$ $\omega_1(D_1)=\left(x=\frac{ah}{b-a}\right)$; $\omega_2(D_2, D_1)=\left(V_{B2}=\frac{a^3h}{3(b-a)}\right)$; $\omega_3(D_3, D_1)=\left(V_{B1}=\frac{b^3h}{3(b-a)}\right)$; $\omega_4(D_2, D_3, D_4)=\left(V_A=\frac{a^2+ab+b^2}{3}h\right)$. Результат ω_4 является искомой формулой или требуемым состоянием предметной области нашей задачи

Б. Теоретико-множественное представление свертывания процедуры решения математических задач (по С. И Шапиро)
Свертывание процедуры решения задачи представляет собой не механическое уменьшение числа операций, необходимых для получения результата,— не исключение, а скорее совмещение, включение одних операций в состав других, новых или уже имеющихся, когда несколько операций выступают как одна.
Так, при начальном уровне усвоения понятия высоты треугольника, процедура $\sum_1$ распознавания высоты включает, наряду с другими, следующие операции.

Обозначение операции	Описание операции
σ_1	Проверить, является ли вершина треугольника одним из концов рассматриваемого отрезка
σ_2	Найти противолежащую сторону треугольника
σ_3	Проверить принадлежность второго конца отрезка противолежащей стороне
σ_4	Продолжить противолежащую сторону
σ_5	Проверить принадлежность второго конца отрезка продолжению противолежащей стороны

По мере увеличения числа *n* выполнений этой процедуры, как показывает эксперимент, происходит ее свертывание за счет объединения операций σ_2 и σ_3, а также σ_4 и σ_5, т. е. вместо множества операций $\{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5\}$, входящих в $\sum_1$ возникает множество $\{\sigma_1, \sigma_{23}, \sigma_{45}\}$. Здесь σ_{23} соответствует объединению операций σ_2 и σ_3, σ_{45} — операций σ_4 и σ_5.
После *n*-го выполнения процедуры $\sum_1$ возникает множество $\{\sigma_1, \sigma_{23}, \sigma_{45}\}$, в котором вместо элементов $\sigma_2, \sigma_3, \sigma_4, \sigma_5$ образовались новые элементы $\{\sigma_2, \sigma_3\}$ и $\{\sigma_4, \sigma_5\}$, оформившиеся впоследствии как специальные операции σ_{23} и σ_{45}. Число элементов в процедуре $\sum_1$ уменьшилось на два.

15.Методы визуального отображения сцен

15.1.Структура и задачи визуального отображения сцен

Для формирования визуального отображения сцен необходимо иметь информацию об окружающей среде (о расположении предметов, об их

форме, размерах и т. д.). Несмотря на богатство естественного языка, сообщать в нем эту информацию сложно, а скорее всего и невозможно. Стремление представить в удобной форме сведения об окружении приводят к введению в структуру ИСФ средств восприятия визуальной информации - визуально отображенных сцен .

В данном разделе (теме) мы опишем подсистему визуального отображения сцен (ПВОС). Назначение подсистемы заключается в том, чтобы на основании двумерного изображения и информации о дальности получить в терминах знаний трехмерное описание предъявленной структуры сцен. Процесс отображения (восприятия) изображения сцен (в дальнейшем: изображения) может быть условно разделен на следующие этапы.

1. Предварительная обработка изображения. В этот этап включают кодирование и аппроксимацию изображения, применение к изображению пространственно-инвариантных преобразований (фильтрации, коррекции, сглаживания или обострения изображения) с целью повышения его качества.
2. Выделение на изображении областей и контуров и получение их формального описания. Задача данного этапа состоит в выделении на изображении, представленном в виде двумерной матрицы, яркостных точек, контуров и областей. Затем производится разбиение контура на сегменты, определение типа линии, соответствующей сегменту, **нахождение топологических и метрических свойств изображения.**
3. Выделение объектов, представленных на изображении. Этап заключается в семантической интерпретации линий и областей, выделенных на предыдущем этапе, с целью объединения их в связные объекты.
4. Определение трехмерной структуры сцены. Задача этапа состоит в определении пространственных отношений, связывающих объекты, представленные на сцене. Выполнение задачи осуществляется путем установления соответствия между изображением и сценой благодаря калибровке камеры, информации о дальности и т. п.
5. Опознавание выделенных в изображении объектов и составление описания сцены в соответствующих терминах.

В данном разделе (теме) не рассматриваются задачи первого и второго этапов, так как они лежат в стороне от основного содержания работы и являются темой самостоятельных исследований.

Рассматриваемые в этом разделе (теме) этапы касаются выделения из изображения семантической информации и связи ее с ИСФ. При изложении материала будем исходить из предположения, что определенная информация уже извлечена из изображения и

представлена в символическом виде (а не в виде яркостных точек изображения). Например, будем предполагать, что **из изображения извлечена и описана (представлена в символической форме) информация о линиях и областях изображения.** Кроме того, как правило, будем предполагать существование некоторой априорной информации о классе анализируемых сцен, например, что все сцены состоят только из многогранных объектов.

Задача рассматриваемых в разделе методов — отобразить символическую информацию о вводимой сцене на накопленные знания системы.

Приведенное выше деление на этапы весьма условно, а последовательность их выполнения может отличаться от указанной выше. При организации подсистем визуального отображения сцен в ИСФ не используют фиксированную последовательность вызова подпрограмм, реализующих определенные функции, вместо этого используют так называемый *гетерархичный принцип организации*. Это понятие не имеет еще четкою определения (термин «гетерархичный» можно трактовать как «с переменной иерархичностью»), но оно включает в себя организацию программ, обладающую приводимыми ниже свойствами.

1. *Система* должна быть *целенаправленной.* Процедуры на всех уровнях должны связываться с некоторыми определенными целями и должны быть довольно краткими. Цели, как правило, должны либо непосредственно вызывать несколько примитивных процедур, либо сводиться к небольшому числу подцелей. Как следствие система должна работать методом «сверху—вниз».
2. *Управление* является не централизованным, а *распределенным по всей системе.* Программные модули взаимо действуют как равноправные единицы.
3. От разработчика системы требуются минимально возможные знания о состоянии системы в момент вызова процедуры. Процедура сама должна знать и создавать условия, которые требуются для ее работы. Это свойство позволяет наращивать систему различным пользователям без детального знания всей системы (что является принципиальным требованием при разработке сложных программ).
4. Система должна обладать некоторыми знаниями о себе. Должны существовать программные модули, выражающие критику при обнаружении подозрительных ситуаций, и модули, предсказывающие неудачу примитивных подпрограмм. Средства связи между этими модулями, кроме обычных средств (передача данных), должны включать сообщения, аналогичные совету, поддержке, замечанию, недовольству, критике, вопросу, ответу и т п

5. Система должна обладать способностью делать пробные заключения и обнаруживать свои ошибки. Если обнаруживается, что предположение ошибочно, то система должна определять, какие факты в базовых данных являются наиболее проблематичными и какие изменения наиболее вероятны.

Графически такая система подобна сети, а не неизменной последовательности процедур. **Каждая процедура связывается с другой через множество возможных связок**, передающих управление. Какие из этих связок используются в конкретном случае, зависит как от контекста, определяемого решаемой задачей, так и от процедур, имеющихся в системе

Заметим, что при гетерархичной организации теряет смысл понятие подпрограмм верхнего и нижнего уровня, указывающего порядок вызова процедур. В данном случае целесообразно говорить об уровне процедур в смысле рода их работы Например, программа поиска линии, работающая с изображением, представленным в виде яркостных точек, может в определенных случаях вызывать программу анализа модели предполагаемого класса объектов, являющуюся программой более высокого уровня.

В системах, использующих гетерархичную организацию, управление может осуществляться под влиянием окружения. Как отметил Саймон, это позволяет при довольно простой ведущей программе, но сложном окружении, демонстрировать интеллектуальное поведение.

Нас будут интересовать довольно сложные изображения, при обработке которых не удается ограничиться отнесением предъявленного изображения к одному из известных классов, а требуется получить описание изображения, причем число возможных описаний столь велико, что бессмысленно считать каждое из них определением отдельного класса. **Под *описанием изображения* мы будем понимать перечень объектов, их свойств и взаимосвязей.** Другими словами, мы хотим подчеркнуть, что изображения, с которыми нам приходится иметь дело, являются сложными и не могут быть полностью обработаны методами распознавания образов.

15.2.2. Формальное описание структуры понятия «сцена»

Рассмотрим примеры формализмов, которые могут быть использованы как основа для описания произвольных сцен.

15.2.2.1. Синтаксические описания.

Рассмотрим простейшую сцену, изображенною на рис. 1.

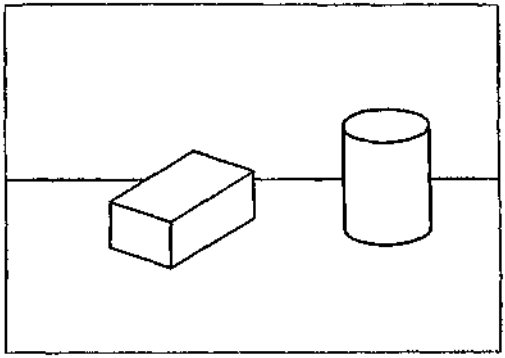


Рис. 1. Пример простой сцены.

При кратком описании понятия «сцена» может быть охарактеризована как «параллелепипед и цилиндр». Более детально данная сцена может быть описана как «параллелепипед, расположенный слева от цилиндра». Можно продолжить детализацию описания, охарактеризовав, например, параллелепипед как совокупность трех граней и т. д. до любого желаемого уровня конкретности. *Указанный способ последовательного уточнения описания изображения будем называть* ***лингвистическим,*** так как он подобен процессу анализа предложения естественного языка. Для анализа изображения в этом подходе, так же как в случае анализа предложений, аналогичным образом вводится понятие грамматики.

Однако, в отличие от грамматик, используемых в естественных языках, ***грамматики для изображений являются не одномерными, а двумерными.*** В одномерных строках естественной операцией соединения символов является *операция конкатенации — размещения символов друг за другом, в двумерных строках такой естественной операции не существует.* Поясним эту мысль на примере рис. 2. Одно из возможных деревьев разбора для данной сцены приведено на рис. 3.

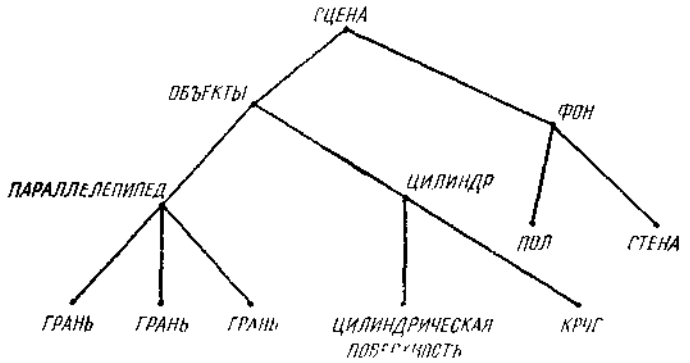


Рис.2. Дерево разбора для сцены, приведенной на рис.1.

Даже если мы точно определим терминальные (понятийные) вершины, дерево разбора будет только приблизительно описывать сцену. Например, три грани могут быть соединены множеством способов, из которых только некоторые дадут параллелепипед.

Существуют различные подходы к определению способа соединения символов в двухмерной строке (плоскости).

Наиболее прямолинейным является способ, полагающийся исключительно на описание границы некоторой фигуры, что дает возможность воспользоваться преимуществами естественного упорядочения точек в одномерном множестве. В качестве примера этого подхода приведем описание четырехугольника (грани).

ЧЕТЫРЕХУГОЛЬНИК — ОТРЕЗОК + ОТРЕЗОК + ОТРЕЗОК + ОТРЕЗОК, где «+» обозначает конкатенацию, и предполагается, что *результирующая строка должна замыкаться на себя*. Выбор терминального (понятийного) символа (отрезка) в этом простом примере является очевидным. Однако для фигур, состоящих из гладких кривых, этот выбор является менее очевидным, и, кроме того, часто трудно определить, где заканчивается один терминальный (понятийный) символ и начинается другой. Проблема идентификации терминальных (понятийных) символов в изображении свойственна не только подходу, основанному на описании границ, но и любому синтаксическому методу.

Предложенный метод не решает всех проблем. Действительно для описания параллелепипеда, приведенного на рис. 1, в терминах (понятиях) определенных выше четырехугольников необходимо конкретизировать операцию соединения четырехугольников. Одним из часто используемых способов является определение точек, в которых происходит соединение четырехугольников (рис.3).

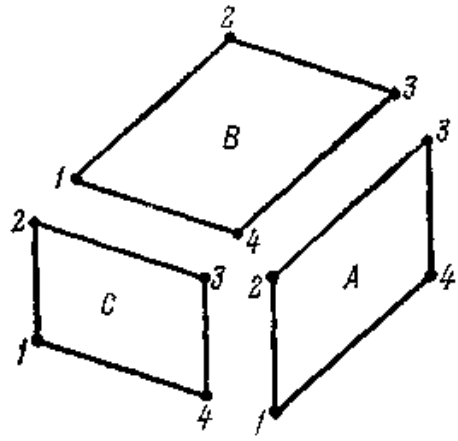


Рис.3. Разложение параллелепипеда на грани

В этом случае синтаксическое описание параллелепипеда может иметь вид: «соединить точку 1 четырехугольника *A* с точкой 4 четырехугольника С» и т. д. для всех вершин.

Аналогичный подход к определению отношений между символами основывается на стандартизации точек соединения. Условимся, например, что каждый терминальный (понятийный) символ имеет две такие точки, называемые «головой» и «хвостом». Будем считать, что операция конкатенации состоит в присоединении «головы» первого символа к «хвосту» второго путем их перемещения (без вращения) в плоскости изображения. «Хвостом» получившегося нетерминального (непонятийного) символа будем называть «хвост» первого терминального (понятийного) символа, а «головой» — «голову» второго терминального (понятийного) символа. Так, например, если *b, c, d*— терминальные (понятийные) символы, то нетерминальный (непонятийный) символ $A=b+c+d$ будет иметь в качестве «хвоста» «хвост» *b*, а в качестве «головы» — «голову» *d.* В качестве примера такого способа задания грамматики опишем цилиндр, изображенный на рис.1. При этом будем использовать терминальные (понятийные) символы, представленные на рис.4.

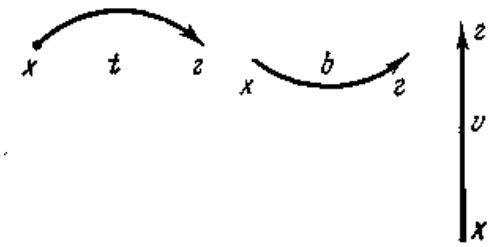


Рис. 4. Множество терминальных символов для сцены, изображенной на рис.1

Операцию конкатенации будем обозначать символом «+», а символом «~» — операцию переобозначения у терминального (понятийного) символа «головы» и «хвоста». Определим дополнительно операцию *, состоящую в присоединении «головы» символа *p* к «голове» символа *q* и «хвоста» *p* к «хвосту» *q*. «Головой» образованного символа будем считать точку соединения «голов» *p* и *q*, а «хвостом» — точку соединения «хвостов» *p* и *q*. В принятых обозначениях цилиндр можно записать следующим образом:

1. ЦИЛИНДР:: = СТОРОНА * КРУГ.
2. СТОРОНА:: = $\tilde{v}+b+v$.
3. КРУГ:: = $t*b$.

Рассмотрим на этом примере, как осуществляется синтаксический анализ изображения.

Нас в основном будет интересовать процесс распознавания изображения, а не порождения. Любая распознающая грамматика предполагает умение распознавать терминальные символы. Указанная операция в общем случае является довольно сложной, и поэтому при обработке двумерных строк (в отличие от одномерных) разбор «сверху—вниз» более предпочтителен, чем разбор «снизу—вверх», так как он определяет вид и возможное местоположение распознаваемого терминального символа.

В приводимом нами примере необходимо уметь распознавать вертикальные линии и два типа кривых линий.

Предположим, что нам предъявлено изображение и требуется определить, содержится ли в нем цилиндр. Применим процедуру разбора «сверху — вниз». Правила 1 и 2 говорят о том, что цилиндр должен включать в себя символ СТОРОНА и что этот символ должен содержать терминальный символ вертикального отрезка. Поэтому процедура будет осуществлять просмотр изображения с целью поиска вертикального отрезка. (Отметим, что при разборе символов

одномерной строки требуется просто выбрать первый элемент строки.) Найдя вертикальный отрезок, мы будем рассматривать его нижний конец как «голову», а верхний как «хвост», так как в соответствии с правилом 2 мы ищем $\tilde{v}$. Из правила 2 видно, что терминальный символ *b* должен быть присоединен к «голове» вертикального отрезка, поэтому необходимо исследовать область изображения в районе нижнего конца отрезка $\tilde{v}$ с целью поиска кривой вида *b*. Если кривая *b* не найдена, то необходимо искать другой вертикальный отрезок. Если кривая *b* найдена, то в соответствии с правилом 2 необходимо искать в районе конца кривой *b* вертикальный отрезок. Если вертикальный отрезок найден, то в изображении опознан нетерминальный символ СТОРОНА. Затем в соответствии с правилом 1 осуществляется поиск символа КРУГ. Если символ КРУГ найден и его расположение на изображении соответствует операции *, то операция распознавания заканчивается успешно.

Как уже было указано ранее, использование метода разбора «сверху—вниз» позволяет направить процедуру распознавания терминальный символов. В нашем примере, за исключением распознавания первого терминальный символа и, указанная процедура применялась не ко всему изображению, а к его некоторой области. Этот целенаправленный аспект алгоритма разбора «сверху—вниз» не только значительно уменьшает количество вычислений, но и уменьшает вероятность обнаружения ошибочного терминального символа, т. е. относящегося не к исследуемой фигуре, а к некоторому другому объекту анализируемого изображения.

Заканчивая рассмотрение синтаксических методов, отметим два основных аспекта, возникающих при использовании некоторой грамматики. **Первый состоит в уже упомянутой ранее проблеме распознавания в изображении терминальных символов**. Природа процесса разбора такова, что ошибка в распознавании одного терминального символа может привести к существенно отличному результату. **Второй аспект касается целесообразности применения синтаксического анализа к тем или иным классам изображений.** Одной из основных существенных сторон использования грамматики является **рекурсивность, т. е. возможность в компактной форме представлять некоторые характеристики (признаки) входной строки**. Таким образом, преимущества от использования грамматики могут проявиться на изображениях, состоящих из небольшого множества терминальных символов, образующих нетерминальные компоненты небольшим количеством рекурсивных правил. В изображениях, не удовлетворяющих указанному требованию, использование грамматики становится формальным способом

исчерпывающего описания «в словах» полного содержания изображения, не дающего преимуществ в компактности.

15.2.2.2. Семантические сети

При анализе многих изображений **структуру изображения целесообразно описывать «в словах», т. е. символически**. Кроме синтаксических методов, приведенных выше, для этих целей удобно использовать *семантические сети, т. е. представлять структуру сцены в виде графа*. При этом узлы графа помечаются наименованиями частей сцены, а дуги — наименованиями семантических отношений, в которых находятся связываемые ими узлы. В качестве примера опишем в виде семантической сети сцену, изображенную на рис. 1. Структура графа будет подобна дереву разбора, представленному на рис.2. Для описания сцены введем, например, следующие отношения: «часть», «тип», «смежный», «слева», «справа», «выше», «ниже». Вид одной из возможных семантических сетей, описывающих указанную сцену, приведен на рис.5.

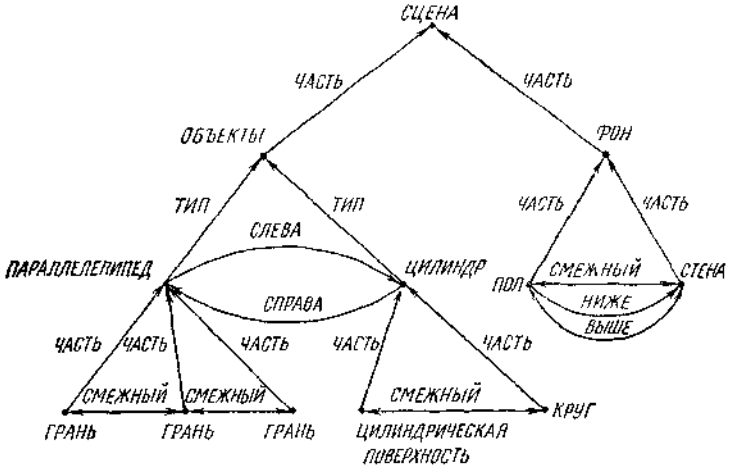


Рис.5. Семантическая сеть для сцены, приведенной на рис. 1.

Дуги на рисунке направлены таким образом, что при чтении наименования узла у «хвоста» дуги, наименования дуги и наименования узла у «головы» дуги получается фраза естественного языка. Например, ЦИЛИНДР ТИП ОБЪЕКТОВ.

Как будет показано ниже, в виде семантических сетей удобно представлять не только описание анализируемой сцены, но и знание

системы о понятиях (модели понятий), которые должны быть обнаружены на сцене.

15.2.3. Трехмерными модели объектов

Описание сцены, состоящей из трехмерных объектов, можно осуществить одним из **двух способов**. Первый способ состоит в игнорировании трехмерной природы реальных объектов и описании сцены в понятиях двумерных конструкций. Второй способ описывает собственно трехмерные объекты, а не их изображения. **Будем называть эти способы соответственно двумерным и трехмерным описанием сцены.** Проиллюстрируем различие между способами на примере сцены, представленной на рис.1. Двумерное описание может быть выражено, например, так: «три смежных четырехугольника, эллипс с примыкающей к нему криволинейной фигурой и три коллинеарных прямых отрезка». Трехмерное описание может быть определено, например, такими словами: «параллелепипед и цилиндр, стоящие на полу перед стеной». И тот, и другой способы используются при распознавании изображений. Мы отдадим предпочтение трехмерному способу описания сцены, так как он является более общим, дает реальное представление об окружении.

В связи с тем, что отображение двумерного изображения на трехмерную сцену не является однозначным (существует бесконечное число трехмерных объектов, соответствующих одному изображению), трехмерное описание может быть извлечено из двумерного изображения только на основания какой-либо дополнительной информации об объектах, присутствующих на сцене. Эта информация задается в виде трехмерных моделей объектов окружения. **Трехмерное описание сцены вырабатывается как результат работы некоторой процедуры, интерпретирующей предъявленное изображение в понятиях трехмерной модели**. Если окружение является достаточно простым, то модели могут выражать только небольшое количество деталей сцены при довольно простой процедуре, сопоставляющей изображение модели. Так, например, для сцены, изображенной на рис. 1 модель могла бы быть описана так: «все ребра параллелепипеда являются прямолинейными отрезками, а у цилиндра некоторые ребра образованы криволинейными отрезками». Даже такой грубой модели было бы достаточно для интерпретации рисунка 1 в понятиях трехмерных объектов.

Для более сложного окружения будет требоваться более точная и полная модель. Часто, однако, возможно представлять сложные

признаки, характеризующие изображения в виде простых признаков о моделях. В общем случае будет требоваться достаточно полное сравнение между трехмерными моделями и изображениями.

В частном случае, когда модель определяется конечным числом точек в трехмерном пространстве, а количество моделей конечно, для сопоставления изображения с моделью может быть использована следующая процедура. Предъявленное изображение сопоставляется с каждой моделью. При этом модель рассматривается со всех углов зрения. Для каждого угла зрения вычисляется проекция точек модели и производится сопоставление их с точками изображения. Модель, проекции которой в некотором положении наилучшим образом сопоставляются с изображением, считается моделью анализируемого изображения.

Проблема вычисления лучшего сопоставления между моделью и изображением может быть разделена на две подпроблемы:

1) идентификация точек модели и изображения, по которым производится сопоставление;
2) вычисления степени соответствия модели и изображения.

Первая подпроблема в общей постановке является очень сложной, и каких-либо глубоких результатов пока не получено. Однако в конкретных случаях, например, для сцен, состоящих только из многогранников, существуют элементарные геометрические свойства, достаточные для установления соответствия между изображением и моделью (или во всяком случае для существенного уменьшения неоднозначности соответствия). Для указанных ограничений (многогранники) вторая подпроблема может быть решена аналитически при полном описании трехмерной структуры каждой модели. Мы не будем останавливаться на описании этого метода, так как он требует полного описания моделей объектов (точнее, трехмерных координат всех вершин, выбранных для сопоставления) и применим только к простым сценам, что редко бывает в реальных ситуациях.

Ниже мы опишем методы, позволяющие анализировать сцены без полного знания моделей объектов, однако основанные на некоторых априорных знаниях о классе объектов, которые могут появляться на сцене. Мы ограничимся рассмотрением объектов, образованных многогранниками. Выбор довольно простого, но распространенного класса объектов вызван двумя причинами.

1. Указанное ограничение на класс рассматриваемых объектов практически не влияет на решение основной задачи — составление описания предъявленной сцены в терминах (понятиях) знаний субъекта.

2. Только для указанных ограничений разработаны процедуры анализа изображений, практически не зависящие от вида конкретных объектов.

Для методичности изложения будем описывать этапы обработки изображения последовательно, но не следует забывать, что обработка осуществляется по гетерархическому принципу. Даже при восприятии простых реальных объектов возникают различного рода помехи (блики, тени и т. п.), не позволяющие выполнять этапы последовательно. В общих чертах процесс обработки изображения происходит следующим образом. Сначала выявляется из изображения наиболее достоверная и легко извлекаемая информацию, такая как размер и форма внешнего контура. Затем из множества всех моделей, имеющихся в системе, отбираются модели — кандидаты, не противоречащие первичной информации. Среди них выбирается наиболее подходящая модель, которая используется для того, чтобы направлять дальнейший процесс обработки изображения. Анализируя выбранную модель, предсказываются свойства, которые должны быть исследованы на изображении в первую очередь, и указывается место их расположения. При обнаружении предсказанных свойств на изображении процесс предсказания продолжается либо до полного опознавания объекта, либо до выявления различия между предсказанием и изображением. При обнаружении различий следует определить, не вызваны ли они поворотом объекта в пространстве или загороженностью одного тела другим и т. п. Если различие относится к такому классу, то оно устраняется. Так, например, можно предположить, что эллипс на изображении соответствует окружности в модели. Приняв данное предположение, следует определить угол наклона наблюдаемой поверхности и произвести пересчет остальных видимых точек. Если различие является неустранимым, то это значит, что выбранная модель не соответствует анализируемой сцене. В этом случае следует выбрать из списка моделей-кандидатов очередную модель, наиболее соответствующую собранной к данному моменту информации.Минский разработал теорию, на основании которой осуществляется не выбор новой модели, а получение ее из предыдущей путем трансформаций.

15.2.4. Разбиение сцены на отдельные объекты

Задача разбиения сцены на отдельные объекты является необходимой для сцен реальной сложности. Общий механизм, осуществляющий

указанную операцию, пока не предложен. Однако при принятых нами ограничениях эта задача имеет эффективные решения.

15.2.4.1. Семантика линий.

Нас будет интересовать семантика, выражаемая линиями (отрезками), из которых состоит изображение. Будем предполагать, что сцена содержит только многогранники степени три, т. е. точно три плоскости пересекаются в каждой вершине многогранника. Заметим, что при этом предположении линия на изображении может иметь только одно из трех значений:

1) обозначать ребро, являющееся пересечением граней, образующих впадину (ребро—«впадина»);

2) обозначать ребро, являющееся пересечением граней, образующих выступ («выпуклое» ребро), у которого на изображении видны его грани;

3) изображать выпуклое ребро, у которого на изображении скрыта одна из граней.

Будем называть линии первого типа *впадинами* и помечать знаком «—», линии второго типа — *выпуклыми ребрами* («+»), а линии третьего типа — *скрытыми ребрами* (→). Направление стрелки → выбирается таким образом, что если смотреть в направлении стрелки, то видимая грань ребра находится справа. Все перечисленные типы ребер представлены на сцене, изображенной на рис.6.

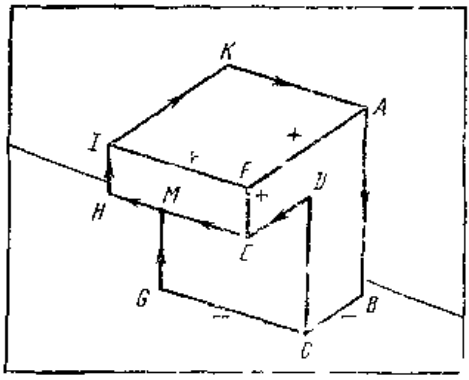


Рис.6. Изображение объекта с помеченными ребрами

Например, линии *EF* и *AF* являются выпуклыми ребрами, линии *BC* и *CG* — впадинами, линии *EH, AB*— скрытыми ребрами. При сделанных ранее предположениях значение линии остается неизменным по всей

ее длине. Вторым важным наблюдением является тот факт, что все вершины делятся на четыре типа. Действительно, так как вершины образованы пересечением трех плоскостей, которые делят пространство на восемь частей (октантов), то вершины можно разделить на типы в зависимости от того, сколько из восьми октантов, смежных с данной вершиной, принадлежит объекту. Существуют вершины типа *1, 3, 5* и *7*. На рис. 7 приведены возможные типы вершин.

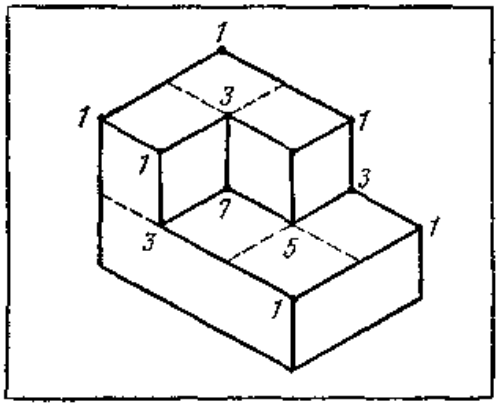


Рис.7. Изображение объекта с помеченными вершинами.

Необходимо иметь в виду, что вершины одного и того же типа могут иметь различное представление на изображении (см. рис.7). Это подчеркивает тот факт, что тип вершины является свойством самого объекта, а не его изображения. Изобразим все способы, в которых могут быть представлены вершины четырех типов. Следует иметь в виду, что вершина каждого типа может рассматриваться только из свободных октантов, т. е. октантов, не занятых объектом, которому принадлежит вершина. Заметим, что рассматривание объекта из различных точек одного октанта не приводит к существенному изменению изображения. Учитывая вышеизложенное, надо рассмотреть вершины типа *1, 3, 5, 7* соответственно с семи, пяти, трех и одной точек зрения. На рис. 8 приведен каталог возможных способов изображения четырех типов вершин.

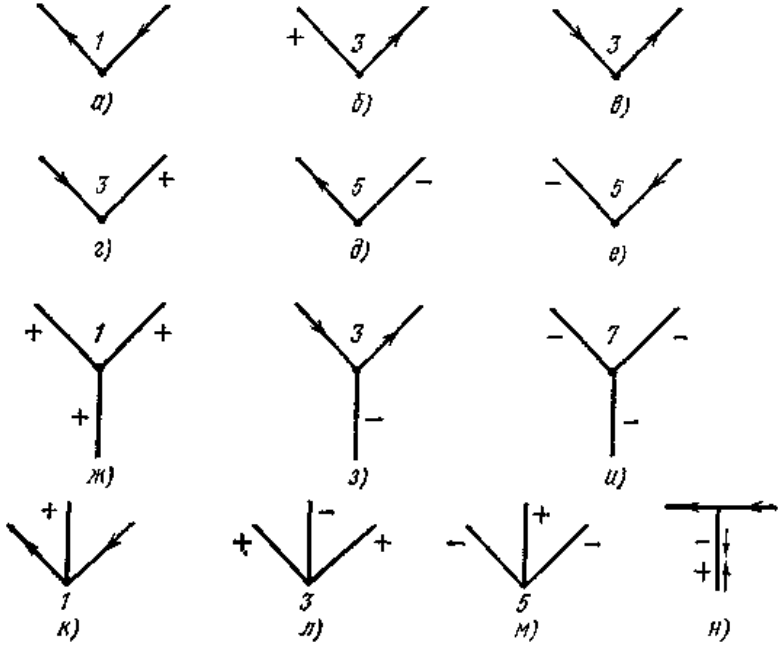


Рис.8. Каталог возможных способов изображения четырех типов вершин.

Вершина типа *1* с семи возможных точек зрения дает только три различные изображения. Многие из способов представления вершин, приведенных в каталоге, могут быть найдены на рис.6. Например, вершина *F* изображена на рис.8, *ж,* вершина *К*— на рис.8, *а,* вершина *А* —на рис.8, *к* и т. д. На рис.8, *н* показана так называемая *Т*-конфигурация, не соответствующая реальной вершине; она возникает, когда некоторая грань закрывает более удаленную часть объекта (см. точку *М* на рис.6). Четыре возможные способа разметки ребер *Т*-конфигурации указаны на рис.8, *н.* Будем ребро *Т*-конфигурации, которое может быть помечено разными символами, называть *основой (основанием),* а второе ребро — *перекладиной.*

Будем вершины, изображенные на рис. 8 *а—е* называть *V-вершинами,* вершины на рис.8, *ж—и — вершинами типа Y,* а вершины на рис.8, *к—м — вершинами типа W.*

Анализ каталога (рис. 8) показывает, что тип вершины и линий не может быть определен исследованием только локальной информации,

необходимо привлечение некоторой контекстной информации. Как это может быть сделано, мы покажем на примере рис.9.

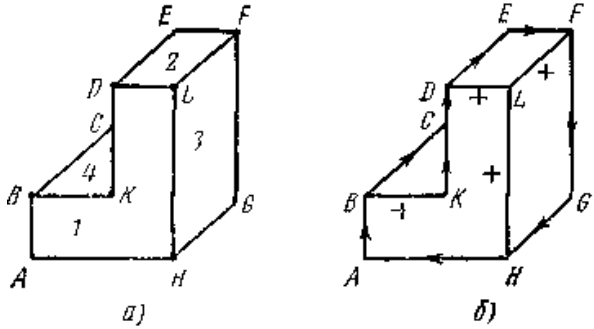


Рис.9. Пример разметки вершин.

Заметим, что ребра внешнего контура некоторого изолированного тела могут быть размечены меткой → при обходе контура по часовой стрелке (см. рис.9, б), так как при этом видимая грань тела всегда остается справа. Теперь займемся анализом вершин *B, D, F, H*. Каждая из этих вершин имеет тип *W* со стрелками на внешних ребрах. Анализ каталога показывает, что существует только одна вершина такого типа (см. рис.8, *к)*.

У этой вершины среднее ребро должно быть помечено знаком «+». Следовательно, ребра *BK, DL, LF, LH* должны быть помечены знаком «+». Точке *C* на рис. 9 соответствует вершина типа *T*. В соответствии с каталогом поперечина вершины типа *T* должна помечаться стрелкой, т. е. линия *KC* помечается стрелкой. В данном примере мы смогли формально и однозначно назначить метки каждой линии. Однако это не всегда возможно. Так, например, для изображения на рис. 6 вершины *B, C, G* не определить однозначно с помощью формального приема. Это отражает объективную причину: по изображению нельзя сказать, стоит ли объект на полу или висит над ним.

Формализуем описанную выше процедуру разметки следующим образом. Выберем некоторую вершину объекта и присвоим ей какую-либо разметку ребер из каталога. Разметка данной вершины наложит ограничения на вершины, смежные с ней. Будем повторять этот процесс до тех пор, пока не получим вершину, не удовлетворяющую каталогу, или не закончим разметку всего объекта. Представим данный процесс в виде дерева поиска. Узлам дерева поиска поставим в соответствие вершины изображения, а дугам, исходящим из данного узла, варианты разметки данной вершины, выбираемые из каталога.

Проиллюстрируем работу процедуры на примере объекта с отверстием, представленного на рис.10.

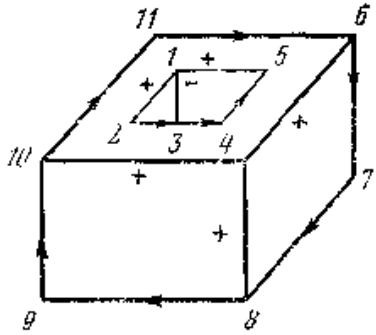


Рис.10. Объект с квадратным отверстием

Выберем произвольный порядок обработки вершин, например, совпадающий с присвоенными им номерами. Вершине *1,* являющейся частью отверстия, на основании каталога может быть присвоено три способа разметки (рис.8, *к, л, м).* Следовательно, из вершины *1* дерева поиска (см. рис.11) будет исходить три дуги.

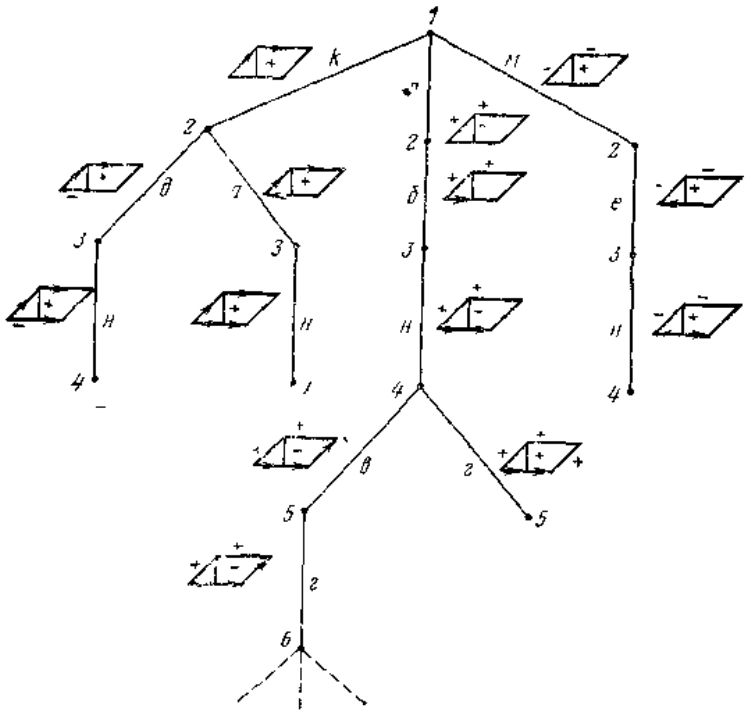


Рис.11. Дерево поиска для объекта с квадратным отверстием.

Будем помечать дуги дерева поиска буквами, соответствующими выбранному из каталога варианту разметки. Для удобства рядом с дугой будем изображать отверстие с принятой на данном шаге разметкой. После рассмотрения вершины *1* выбираем вершину *2*. Способы ее разметки будут находиться в зависимости от разметки, присвоенной вершине *1*. В варианте *1к* (см. рис.11) для разметки вершины *2* имеются две возможности (см. рис.8, *д* и рис.8, *а)*. Аналогичным образом раскрываются вершины дерева в вариантах *1л* и *1м*. Затем выбирается вершина *3*, имеющая тип *Т*. В варианте разметки *1к, 2д* для вершины *3* возможен только один способ разметки. Действительно, направление перекладины у вершины типа *Т* определяется каталогом однозначно (если смотреть в направлении стрелки, то ребро — основа должно находиться слева). Но тогда вершине *4* в каталоге не соответствует никакого способа разметки. Это означает, что данный вариант разметки является неприемлемым, и соответствующая ему ветвь дерева обрывается. Продолжая этот

процесс, можно получить единственную разметку линий всей фигуры, изображенной на рис.10. Напомним, что ранее мы установили возможность размечать линии внешнего контура фигуры по часовой стрелке. Это правило сократит перебор вариантов при разметке оставшихся вершин. Описанный алгоритм сводится к поиску пути в дереве**.**

15.2.4.2. Объединение областей в объекты.

Одна из основных проблем при анализе изображений многогранников состоит в выделении объектов. Существуют различные эвристические алгоритмы, направленные на решение этой задачи, однако пока нет ее полного теоретического анализа. Перед тем как перейти к изложению конкретных методов, сделаем некоторые предварительные замечания.

Предположим, что мы хотим на изображении, состоящем из линий, образующих многогранники (возможно, закрывающие друг друга), выделить отдельные объекты. Другими словами, нас интересует, какие из областей, выделенных на изображении, образуют некоторый многогранник. Если больше не наложено никаких ограничений, то задача не имеет однозначного решения. Действительно, даже для такого простого изображения, какое представлено на рис.6, существует два варианта: либо многогранник стоит на плите (или висит над ней), либо он образует одно целое с плитой. В связи с тем, что однозначного решения не существует, мы можем надеяться в лучшем случае на метод, дающий на большинстве изображений решения, аналогичные решениям человека.

Для обоснования эвристик рассмотрим простой многогранник, изображенный на рис.12.

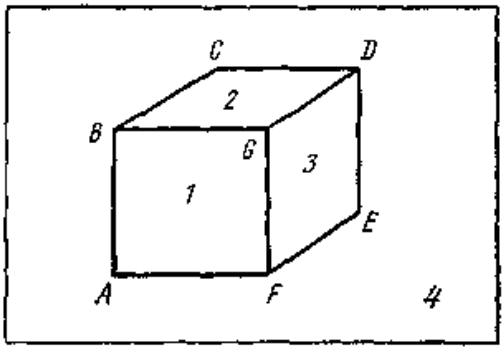


Рис.12. Пример простой сцены

Предположим, что любой разумный метод выделения частей из данного рисунка должен определить, что области *1, 2, 3* объединяются в одну группу, а область *4* (фон) в другую. Если мы рассмотрим 7 видимых вершин многогранника, то здесь присутствуют: три *V*-вершины, три *W*-вершины и одна *Y*-вершина. Это вместе с фактом, что области *1, 2* и *3* должны объединяться вместе, предполагает введения следующих эвристических правил:

1) *Y*-вершина дает основания предполагать, что представленные в ней области должны быть объединены;

2) *W*-вершина дает основание предполагать, что две области, ограниченные ребрами, образующими между собой острые углы, должны быть объединены.

На рис. 13 изображены два расположенных друг на друге многогранника и находящаяся за ними треугольная призма.

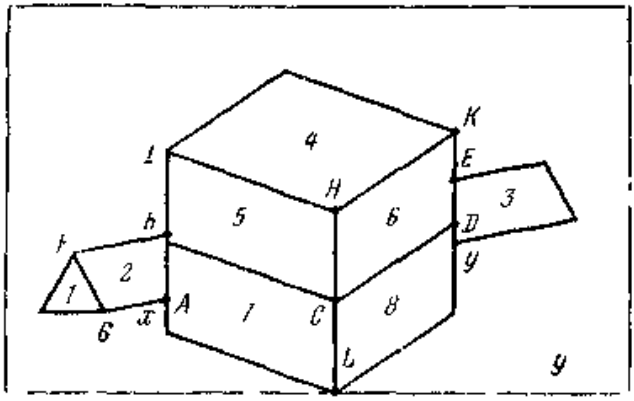


Рис.13. Сцена, изображающая частично скрытый объект.

Вершина *C* называется вершиной типа ψ. Она наводит на мысль ввести следующее эвристическое правило:

3) *ψ* -вершина дает основания предполагать, что две верхние области должны образовать одну группу, а две нижних области — другую. (На рис. 13 ψ-вершина C дает основание для объединения областей *5* и *6* и областей 7 и *8.)*

Рассмотрим теперь на рис. 13 *T*-вершины *A* и *D.* Ранее мы отмечали, что перекладина *T*-вершины закрывает более удаленную часть сцены (это справедливо не только для многогранников степени три). Кроме того, факт, что основы *T*-вершин *A* и *D* параллельны, дает основание предположить, что области *2* и *3* принадлежат одной группе, а области

x и *y* в окрестности точек *C* и *D* принадлежат другой. Можно сформулировать следующее эвристическое правило:

4) параллельность оснований двух *T*-вершин дает возможность предполагать, что области, находящиеся по одну сторону от оснований, образуют одну группу, а области, расположенные по другую сторону, образуют другую группу.

На рис. 14 дана иллюстрация четырех перечисленных выше эвристических правил. Пунктирами обозначены связки, указывающие на объединяемые области.

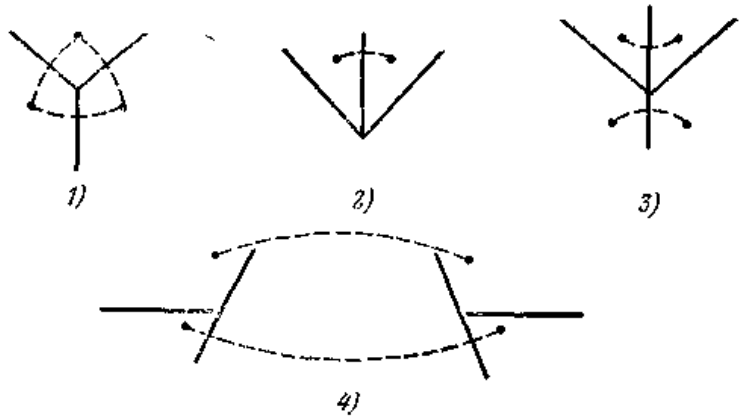


Рис.14. Иллюстрация четырех эвристических правил объединения областей.

Приведенные правила не являются достаточными, так как были сформулированы только на основании совсем простых примеров. Кажется очевидным, что на основании только одной связки не следует объединять области. Целесообразно проанализировать отношения между всеми связками и только на основании этой информации объединять области.

Проиллюстрируем этот подход на примере сцены, содержащей усеченную пирамиду, лежащую на плите (рис.15, *а).*

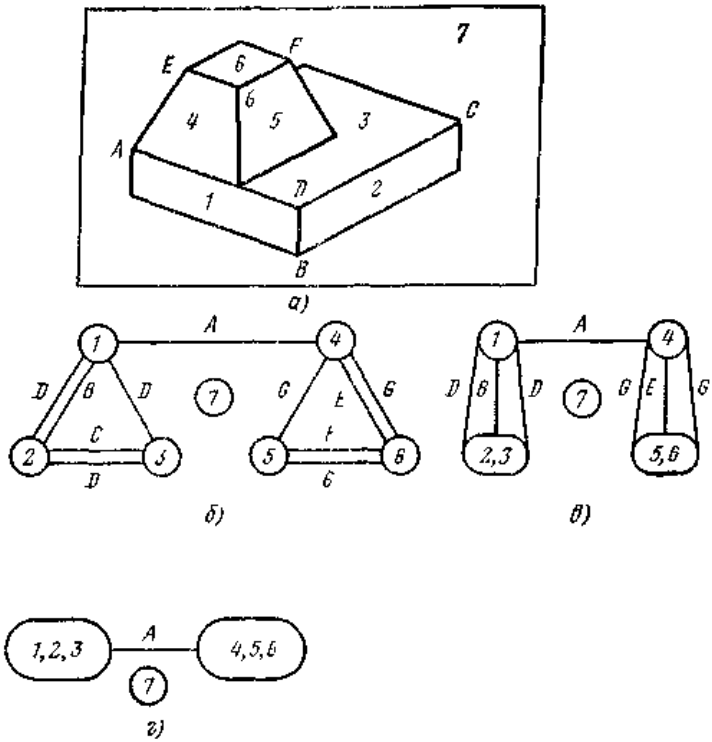


Рис.15. Пример работы алгоритма объединения областей.

На рис. 15, *б* в виде графа представлена информация о связи областей сцены, изображенной на рис.15, *а*. Узлы графа соответствуют областям изображения, а дуги — связкам, полученным с помощью применения приведенных ранее эвристических правил к некоторым вершинам (дугам присвоены имена этих вершин). Например, применив к вершине *D* эвристическое правило 1), получим связку между вершиной *1* и *2* (помеченную буквой *D)* и связки между вершинами *2* и *3, 1* и *3* (помеченные буквами *D).* Анализ рис.15, *б* подтверждает подозрение, что одной связки не достаточно для объединения областей. Действительно, области *1* и *4* являются различными, но связываются связкой, полученной при обработке вершины *A.* Для того чтобы устранить указанное несоответствие, добавляется следующее эвристическое правило:

5) два узла объединяются, если между ними существует по крайней мере две связки. Любые связки из этих двух узлов к другим узлам остаются во вновь полученном графе.

Рис.15, *в* изображает граф, полученный применением данного правила к узлам *2* и *3,* а затем к узлам 5 и *6* графа, представленного на рис.15, *б.* Применив это же правило к узлам *1* и *(2, 3)* и узлам *4* и (5, *6)* графа, изображенного на рис.15, *в*, получим окончательный вид графа рис.15, *г.* К этому графу правило применить нельзя, так как никакие узлы не связаны двумя дугами. Таким образом, на рис.15, *а* представлено три объекта *(1, 2, 3), (4, 5, 6)* и 7. Изложенный метод работает довольно успешно при распознавании многих сцен. Однако, можно привести примеры изображений, где этот метод дает неправильный результат. Например, фигура, изображенная на рис.9, *а,* не будет опознана как одно тело, так как области *1* и *4* объединяются только одной связкой. Это несоответствие можно устранить, введя очередное эвристическое правило:

6) если в исходном графе некоторая вершина (область) связана только с одной вершиной, то эти вершины объединяются.

Однако и после введения этого правила можно привести пример (рис.16), когда изображение будет неверно разделяться на отдельные объекты.

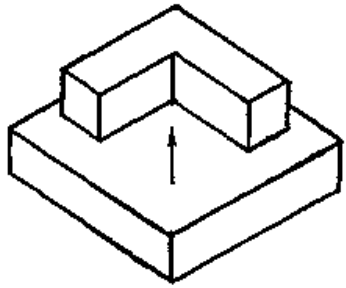


Рис.16. Пример неверного объединения двух тел в одно на основе правил 1—6.

Рассмотренные выше методы не учитывали некоторых свойств физических тел и в первую очередь наличие теней. Обобщение изложенного выше способа разметки линий позволяет применить его для описания сцен, содержащих тени. Уолтц ввел одиннадцать типов линий (вместо трех, рассмотренных нами) и построил каталог, содержащий более тысячи возможных способов сочетания линий в вершинах (аналогичный каталогу, представленному на рис.8). Процедура выделения тел, разработанная Уолтцом на основе данного

каталога, работает очень быстро, несмотря на большое количество потенциальных способов разметки вершин. Дело в том, что одновременно с увеличением числа способов разметки вершин увеличивается и количество ограничений, накладываемых на связанные вершины, что резко сокращает поиск. На рис. 17 приведен пример сцены, с которой программа Уолтца справляется довольно легко.

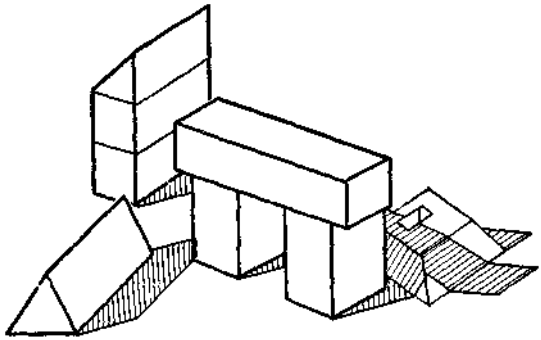


Рис.17. Сцена с тенями, обрабатываемая программой Уолтца.

Подводя итоги перечисленным приемам, следует отметить, что они являются весьма общими в классе многогранников. Они не требуют ни того, чтобы многогранники ограничивались степенью три, ни какой-либо другой дополнительной информации о специфике объектов сцены. Однако эти методы не являются полными, т. е. они совершают ошибки при анализе некоторых сцен. Вообще говоря, довольно трудно характеризовать класс сцен, на которых эти методы работают безошибочно.

Цель применения описанных методов состоит в том, что после их работы сцена разлагается на тела, и это упрощает задачу опознающей программы. Действительно, теперь ей вместо осмотра всей сцены и поиска на ней участков, совпадающих с моделями, необходимо осуществлять сравнение только с выделенными областями, представляющими отдельные тела. Прежде чем перейти к описанию методов опознавания, мы рассмотрим в следующем параграфе способы получения пространственных характеристик выделенных объектов, также дающих дополнительную информацию для методов опознавания.

15.2.5. Монокулярное определение трехмерной структуры сцены

Определение трехмерной структуры видимой части объектов является важной частью анализа сцены. Оно необходимо как для опознавания объектов, если их модель является трехмерной, так и для планирования перемещений ИИО. **Эта проблема в живой природе решается с помощью бинокулярного зрения**. Однако при наличии двух изображений возникает задача отыскания на них идентичных пар точек.

В общем случае эта проблема не может быть решена по одному двумерному изображению. Однако при принятом предположении (сцена состоит только из многогранников) и благодаря некоторым дополнительным фактам можно получить информацию о третьем измерении.

Изобразим преобразование, выполняемое камерой в виде схемы, приведенной на рис.18.

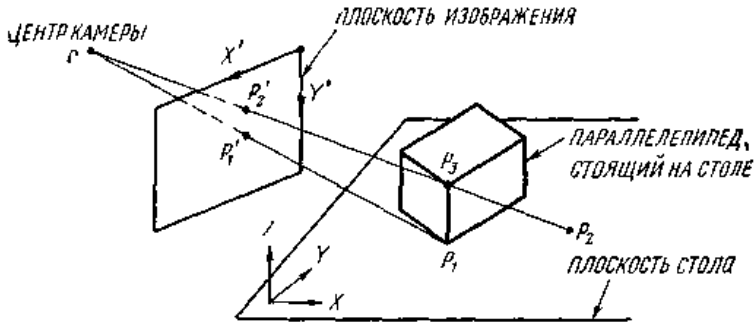


Рис. 18. Преобразование, выполняемое камерой.

Для любой точки $P_j=(X_j, Y_j, Z_j)$ реального мира существует единственная соответствующая ему точка изображения $P'_j = (X'_j, Y'_j)$. С другой стороны, каждой точке P'_j соответствует луч, проходящий через эту точку и центр камеры. Если камера соответствующим образом откалибрована, то каждой точке P'_j можно поставить во взаимнооднозначное соответствие некоторую точку P_j на поверхности стола. Она определяется как точка пересечения луча, исходящего из точки C, с плоскостью стола. Если мы примем естественную гипотезу, что объект, изображенный на рис.18, опирается на плоскость стола, а не висит в воздухе, то мы можем по точке P'_1 однозначно определить координаты точки P_1.

Отметим, что если известна некоторая информация об объекте, например, такая, как размер ребра P_1P_3 или «ребро P_1P_3 вертикально поверхности стола» или «одна из образующих его граней вертикальна», то можно определить координаты точки P_3. Предположим, что об объекте, изображенном на рис.12, нам известны координаты точек *A, F, E* и *G*. Так как три точки определяют положение плоскости, мы можем по известным координатам *A, F, G* определить положение плоскости *1* в трехмерном пространстве. Теперь становится возможным определить положение точки *B* как пересечение плоскости *1* с лучом, исходящим из центра камеры и проходящим через точку *B*. Подобным образом можно определить координаты точки *D* на основании известных координат точек *G, F, E*. Теперь нам известны координаты точек *B, G, D* и, следовательно, можно определить положение в пространстве плоскости 2, а по ней и координаты точки *C*. Итак, для изображенного на рис. 12 многогранника по координатам четырех точек *A, F, G* и *E* можно определить координаты оставшихся вершин, а следовательно, определить трехмерную структуру видимой части многогранника.

Формализуем приведенные выше рассуждения. Для простоты рассуждений будем полагать, что начало координат расположено в центре камеры. Тот факт, что точка *(x, y, z)* лежит в плоскости *v*, может быть выражен уравнением $V \cdot P = 1$, где *V* является вектором, перпендикулярным *v*, исходящим из начала координат и имеющим длину, обратную расстоянию от начала координат до плоскости, а *P* — вектор, исходящий из начала координат в точку *(x, y, z)*. Вектор *V* характеризует плоскость *v*. Если известно расположение луча, на котором лежит точка *P*, то это эквивалентно знанию положения единичного вектора *U* в направлении *P* и расстоянию α от начала координат до *P*. Учитывая вышеизложенное, можно записать, что

$$V \cdot \alpha\, U = 1 \quad \text{или} \quad V \cdot U = 1/\alpha = \lambda.$$

Применим этот анализ к изображению, представленному на рис. 12. Будем обозначать через V_i векторы, характеризующие плоскости граней *1, 2* и *3*, а через U_k — единичные векторы, направленные в вершину *K*. Тогда для плоскостей *1, 2* и *3* можно записать следующие уравнения:

$V_1 \cdot U_A = \lambda_A,\ V_2 \cdot U_B = \lambda_B,\ V_3 \cdot U_G = \lambda_G,$
$V_1 \cdot U_F = \lambda_F,\ V_2 \cdot U_C = \lambda_C,\ V_5 \cdot U_F = \lambda_F,$
$V_1 \cdot U_G = \lambda_G,\ V_2 \cdot U_G = \lambda_G,\ V_3 \cdot U_E = \lambda_E,$
$V_1 \cdot U_B = \lambda_B,\ V_2 \cdot U_D = \lambda_D,\ V_3 \cdot U_D = \lambda_D.$

Итак, мы имеем 12 линейных уравнений и 16 неизвестных *(λ_A, λ_B, λ_C, λ_D, λ_E, λ_F, λ_G* и 9 неизвестных, определяющих плоскости *1,2* и *3)*. Если уравнения линейно независимы, то однозначное решение может быть

получено, если будут зафиксированы любые четыре независимые переменные. В общем случае можно сказать, что мы имеем:

K = (число неизвестных) = 3× (число плоскостей)+
+ (число различных точек изображения).

M = (число уравнений) = $\sum_{\text{по всем плоскостям}}$ (число различных точек плоскости).

Таким образом, $L=K—M$ равно наименьшему числу точек изображения, информация о которых должна быть известна для того, чтобы однозначно определить трехмерную структуру многогранника.

Если на объекты сцены наложить ограничения, что они являются только многогранниками степени три, то можно получить интерпретацию описанной выше процедуры. Интерпретация позволяет последовательно определять местоположение в пространстве видимых граней объекта, что по сравнению с решением системы линейных уравнений дает возможность использовать информацию, получаемую в процессе вычислений, для направления вычисления. Интерпретация основывается на определенного рода дуальном графе, описывающем изображение многогранника. Узел дуального графа соответствует видимой грани многогранника, а дуга между двумя вершинами (гранями) соответствует ребру многогранника, разделяющему смежные грани (т. е. ребру, являющемуся общим для двух граней). Если две грани разделяются более чем одним ребром (все ребра толжны быть коллинеарны), то на графе изображается только одна дуга. Заметим, что из приведенного ранее метода разметки можно определить, является ли ребро общим для двух смежных, видимых граней. Действительно, если ребро помечено знаками «+» или «—», то оно является общим, если же оно помечено знаком → или ←, то оно не является общим для двух граней (одна грань закрывает другую).

На рис.19, *а* и 19, *б* представлен многогранник и соответствующий ему дуальный граф.

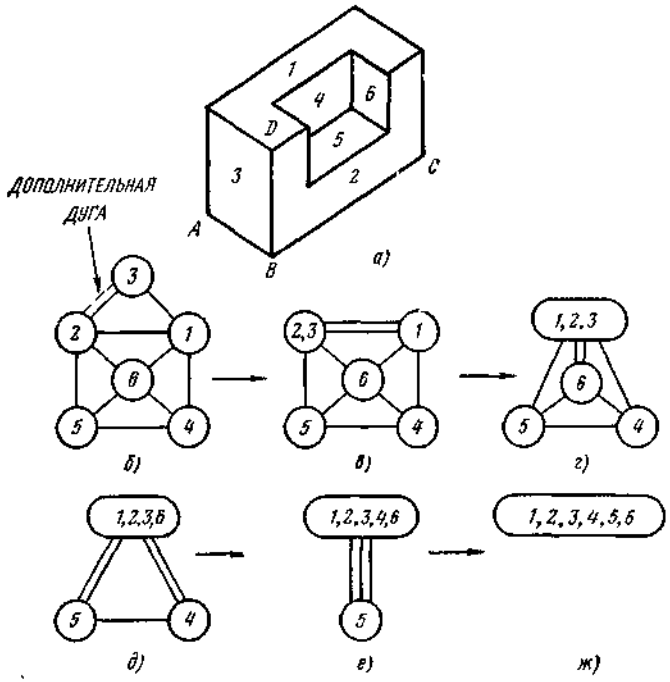


Рис. 19. Объединение дуг дуального графа с введением одной дополнительной дуги.

Покажем, как дуальный граф используется для определения последовательности перехода от одной грани многогранника к другой для определения его пространственной структуры. Предположим, что известны координаты вершин *A, B, C, D* (рис.19, *а),* т. е. известно расположение в пространстве плоскостей 2 и *3,* Поверхности, имеющие известное местоположение в пространстве, будем объединять в одну вершину (см. рис.19, *в),* связывая с ней все дуги, которые входили в объединенные вершины. Для того, чтобы инициировать процесс, введем дополнительное ребро между гранями, положение которых в пространстве известно (см. рис.19, б). Из рис.19, *в* видно, что грани 2 и 3 с известным местоположением связаны двумя дугами с гранью *1.* Следовательно, грань *1* граничит с гранями *2* и *3* по двум неколлинеарным ребрам, которые однозначно определяют положение грани *1*. Подобные рассуждения обосновывают следующее *правило преобразования графа*: *объединение в одну двух вершин, связанных более, чем одной дугой.* Применяя это правило многократно

к дуальному графу, изображенному на рис.19**,** б, получим единственную вершину. Это обозначает, что для рассматриваемой фигуры четырех точек достаточно, чтобы определить пространственную структуру всей фигуры. В общем случае четырех точек может быть недостаточно для определения пространственной структуры тела. С точки зрения приведенного выше преобразования это будет означать, что к дуальному графу надо будет присоединить более чем одну дополнительную дугу, чтобы преобразовать граф к одной вершине. Другими словами, *для однозначного описания видимой пространственной структуры объекта необходимо определить (например с помощью дальномера) трехмерные координаты еще одной точки объекта.* Нетрудно показать, что пространственная структура видимой части многогранника может быть определена с помощью *К*+3 независимых известных точек, где *К* — количество добавляемых ребер. На рис. 20приведен пример фигуры и соответствующего ей дуального графа, требующей для определения пространственной структуры пяти точек.

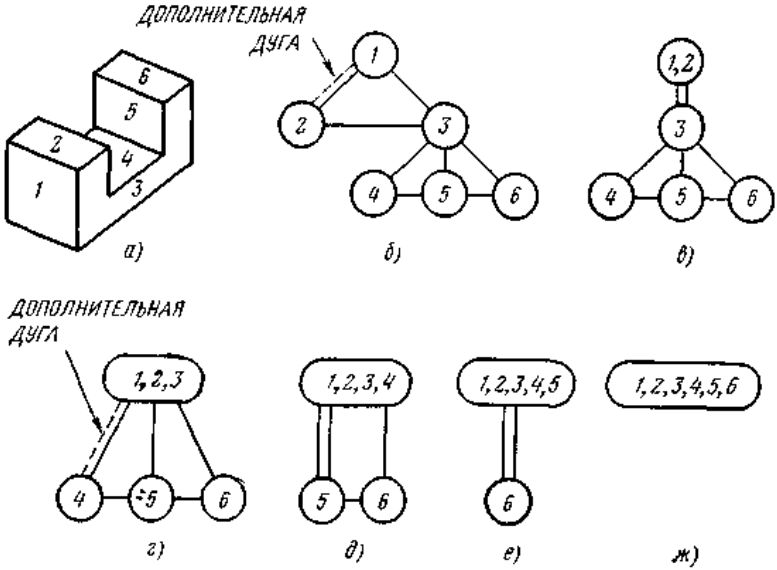


Рис.20. Объединение дуг дуального графа с введением двух дополнительных дуг.

В п.15.2.1 предъявлены некоторые требования к структуре комплекса программ, осуществляющего восприятие визуальной информации. Мы отмечали, что последовательность вызова программ комплекса не должна быть фиксированной и заранее предопределенной, она должна определяться анализируемой сценой. В качестве примера рассмотрим сцену, представленную на рис.21.

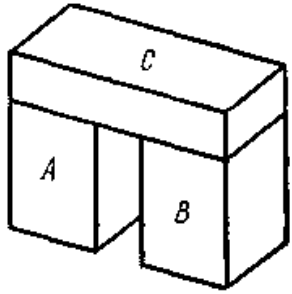


Рис.21. Определение глубины многогранника *A* на основании глубины многогранника *B*

Здесь ребро, определяющее глубину многогранника *A*, не видно, и если полная модель многогранника *A* неизвестна программе, то на основании описанных выше методов длину ребра не определить. Человек при анализе этой сцены сделает естественное предположение, что многогранники *A* и *B* одинаковы и на основании этого определит размер невидимого ребра.

Финин предложил подход к решению подобных задач. Предложенную им процедуру удобно определить в терминах циклического выполнения последовательности: **«группирование—гипотеза—проверка».** Рассмотрим работу процедуры анализа на примере сцены, изображенной на рис.21. Процедура пытается определить размеры объекта *A* и выясняет, что глубина его не видна из изображения. Тогда она относит объект *A* к какой-либо группе объектов. В рассматриваемом примере *A* и *B* могут быть отнесены к группе объектов, поддерживающих объект *C*. Затем процедура проверяет, существуют ли причины, мешающие отнести объекты *A* и *B* к одной группе? При этом процедура рассматривает примерно такие вопросы:

1. Относятся ли *A* и *B* к одному сорту объектов?
   Да, оба являются брусками.
2. Одинаковым ли образом ориентированы объекты *A* и B?
   Да.
3. Соответствуют ли видимые размеры объектов?
   Да.

4. Существуют ли причины сомневаться в том, что невидимое ребро объекта *A* отличается от аналогичного ребра объекта *B*?

Нет.

Если ответы аналогичны приведенным, то гипотеза применяется и анализ сцены продолжается.

Отметим, что группирование объектов является одним из способов использования контекста при анализе сцены.

Основанием для высказывания гипотезы о принадлежности объектов к одной группе может быть, например, следующая информация.

1. Объекты образуют стеки и ряды, связанные отношением «поддерживать» или «находиться перед».
2. Объекты выполняют общую функцию, например, колонны у арки или ножки у стола.
3. Объекты находятся в непосредственной близости.
4. Объекты относятся к одному типу.

Проверка гипотез осуществляется не только на основе измерения различий между объектами, но также на основе измерения, какие отклонения являются типичными и какие приемлемыми.

Описанная процедура требует дальнейшего уточнения и развития, однако общий подход заслуживает внимания.

15.2.6. Формирование моделей, опознавание объектов и описание сцены

На основании методов, аналогичных описанным в предыдущих параграфах, и некоторой априорной информации об объектах можно составить описание сцены.

Рассмотрим на примере сцены, изображенной на рис.22, *а,* процесс получения ее описания.

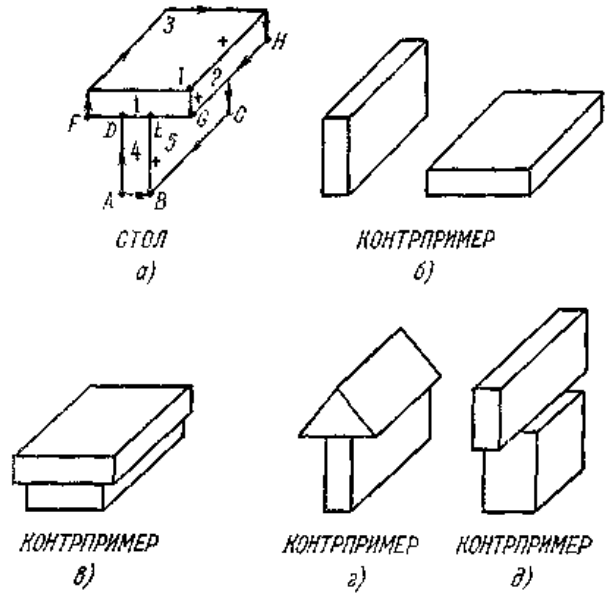


Рис.22. Последовательность примеров, определяющих понятие СТОЛ.

Будем полагать, что предьявляемые сцены содержат только прямоугольные параллелепипеды (для краткости будем называть плитами). Столь жесткое ограничение на вид используемых объектов не отражает реальных возможностей машинных методов обработки изображений и введено исключительно с целью привести компактный пример. Будем также полагать, что программе, обрабатывающей изображение сцены, известны следующие понятия: СТОЯТЬ, ЛЕЖАТЬ, ПОДДЕРЖИВАТЬ. Например, **программа может считать, что объект стоит, если его высота больше одного из горизонтальных размеров и объект лежит, если указанное условие не выполняется**. **Программа может определять понятие ПОДДЕРЖИВАТЬ как отношение одного объекта находиться под другим (при отсутствии других объектов между ними).**

Как и прежде, будем основываться на гипотезе об опорной плоскости, т. е. будем считать, что любой объект поддерживается либо горизонтальной плоскостью стола, либо горизонтальной плоскостью другого объекта. В результате работы 1 и 2 этапов (см. п. 15.2.1) подсистемы зрительного восприятия изображение предъявленной сце-

ны будет описано в терминах линий и областей. Применение к полученному описанию алгоритма разметки линий даст результат, изображенный на рис.22, *а*. На основании проведенной разметки линий, эвристические правила разбиения сцены на объекты обнаружат присутствие на сцене двух объектов: *(1, 2, 3)* и *(4, 5)*.

На анализируемом изображении часть объекта, образованного гранями *4* и 5, закрыта объектом, составленным из граней *1, 2* и *3*. Поэтому при разбиении сцены на объекты объект *(4, 5)* будет представлен в виде неполного контурного рисунка. Для упрощения процесса опознавания можно применять **эвристические процедуры**, достраивающие неполные контурные рисунки на основании некоторых знаний о свойствах объектов. В рассматриваемом примере в связи с его простотой (все объекты являются плитами) в этом нет необходимости.

После этапа достраивания неполных контурных рисунков следует этап опознавания объектов и определения взаимоотношений между объектами. Для сопоставления трехмерной модели объекта с его двумерным изображением необходимо получить по контурному рисунку заключение о трехмерных свойствах сцены. В п. 15.2.5 рассмотрена возможность использования гипотезы об опорной плоскости для определения этой информации.

В рассматриваемом примере знание, что точки *A, B* и *C* лежат на известной опорной плоскости, дает возможность определить их трехмерные координаты.

В связи с тем, что плита является прямоугольным параллелепипедом, плоскость, в которой лежит грань *4*, перпендикулярна опорной плоскости, и, следовательно, ее положение в пространстве может быть определено. Если положение грани *4* известно, то можно определить координаты точек *D* и *E* как пересечение луча, исходящего из центра камеры, с плоскостью, в которой лежит грань *4*. На основании подобных рассуждений могут быть определены координаты точек *F, G, H, I*. Знание координат точек *A, B, C, F, G, H, I* дает возможность вычислить длины сторон параллелепипедов и сделать заключение, что плита *(4, 5)* находится в состоянии СТОЯТЬ (так как *AD>AB)*, плита *(1, 2, 3)* находится в положении ЛЕЖАТЬ (так как *GI<FG* и *GI<GH)*, и плита *(4, 5)* находится с плитой *(1, 2, 3)* в отношении ПОДДЕРЖИВАТЬ (так как объект *(4, 5)* расположен под объектом *(1, 2, 3))*. Итак, на основании собранной информации описание сцены может быть представлено в виде семантической сети, изображенной на рис.23.

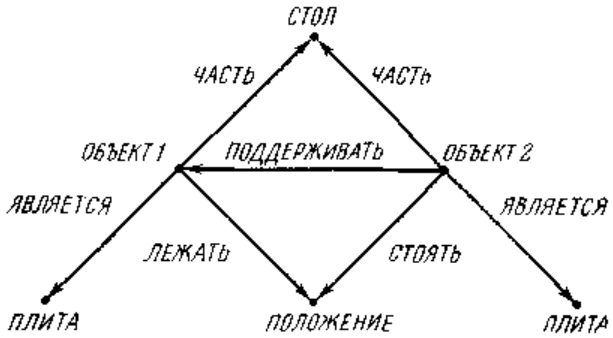


Рис.23. Описание понятия СТОЛ.

Покажем теперь, как понятия (модели), существующие в системе, могут быть использованы для образования новых, более сложных понятий.

Процесс образования понятия заключается в предъявлении машине последовательности сцен, характеризующих как собственно понятие, так и понятия, подобные образовываему, но отличающиеся от него в одной или нескольких принципиальных деталях (контрпримеры).

Мы будем рассматривать образованное понятие как модель, включающую описание обязательных признаков и описание признаков, которые не должны присутствовать в примерах, удовлетворяющих данному понятию.

На рис. 22 приведена последовательность сцен, определяющих образовываемое понятие СТОЛ. Программа анализа изображения при предъявлении первой из сцен, определяющей собственно понятие, составляет описание сцены в терминах известных ей понятий и отношений.

Примерный вид полученного описания приведен на рис. 23.

В определении стола существенным отношением является отношение поддержки, так как не существует столов без этого отношения между составляющими его объектами. Следовательно, **цель примеров — показать машине существенные признаки понятия**. (Позже мы увидим, что некоторые отношения становятся менее существенными, а другие запрещенными.)

Второй пример на рис. 22 является контрпримером. Его единственное отличие от первого примера заключается в том, что одна плита не поддерживает другую. Описание этого изображения отличается от описания первого примера отсутствием отношения ПОДДЕРЖИВАТЬ между плитами. Программа, сравнивающая описания, обнаружит это

единственное различие и сделает вывод, что именно отсутствие этого отношения привело к невозможности отнести второй пример к понятию СТОЛ. Таким образом, программа помечает отношение ПОДДЕРЖИВАТЬ как существенное заменой ею в описании понятия на отношение ДОЛЖЕН ПОДДЕРЖИВАТЬ. Заметим, что мы сформировали новое отношение на основании всего одного примера, а не на основании статистических выводов из серии опытов

Остальные контрпримеры (рис.22) выполняют аналогичную роль. Под их воздействием будет получена модель понятия СТОЛ (см. рис.24).

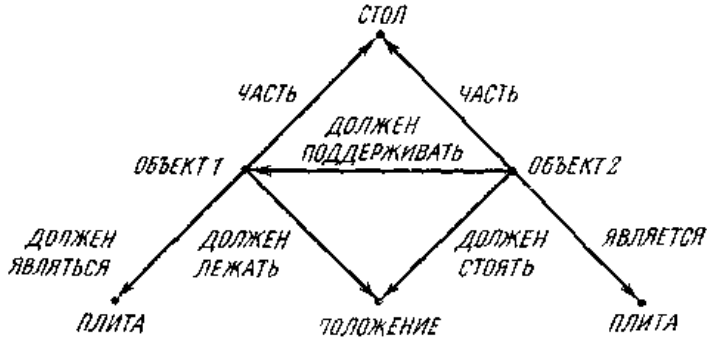


Рис.24. Модель понятия СТОЛ.

Теперь, когда основная идея ясна, рассмотрим более сложную последовательность примеров (рис.25), определяющую понятие АРКА.

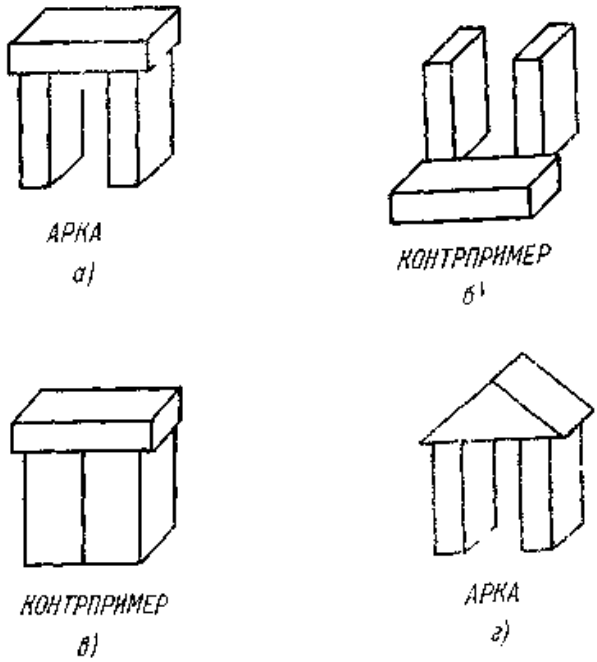


Рис.25. Последовательность примеров, определяющих понятие АРКА.

Первый пример последовательности дает, как и ранее, образец образовываемого понятия, на основании которого машина составляет начальное описание понятия. Следующий пример демонстрирует необходимость отношения ПОДДЕРЖИВАТЬ для понятия АРКА. В отличие от примера на рис.22, *б,* различие касается отсутствия не одного, а двух отношений ПОДДЕРЖИВАТЬ (для каждой из опор арки). Этот пример поднимает сложный вопрос: что делать, если контрпример отличается от описания образованного понятия более, чем в одном отношении или вершине. **Для решения этой проблемы необходимо разработать теорию, которая рассортировывает наблюдаемые различия по важности и использует наиболее интересные для формирования гипотез о модификации описания образовываемого понятия.** В связи с отсутствием в настоящее время подобной теории перечислим некоторые правила, пригодные для упорядочения различий.

1. Важность различий убывает по мере удаления от начального (верхнего) узла описания, соответствующего нулевому уровню сети Так, различия в отношении между объектами более важны, чем различия в форме объектов, которые в свою очередь более важны, чем различия в затемнении вершин
2. Различия в пределах данного уровня упорядочиваются в соответствии с их типом. Например, различие, состоящее в отсутствии отношения, связывающего две вершины, можно рассматривать как более важное, чем различие в появлении дополнительного отношения.
3. Различия, имеющие одинаковый уровень и тип, могут быть упорядочены на основе некоторых вторичных эвристик. Например, отношение ПОДДЕРЖИВАТЬ зачастую более важно, чем отношение КАСАТЬСЯ или СПРАВА, СЛЕВА.
4. Если два различия имеют одинаковое описание и сущность, то целесообразно считать, что они оба являются причинами отнесения образца к контрпримеру.

Последнее правило ориентировано на обработку образцов, аналогичных контрпримеру, приведенному на рис. 25, *б.*

Вернемся к рассмотрению последовательности, определяющей понятие АРКА (рис.25). Очередной контрпример (рис.25, *в)* несет косвенную информацию о необходимости расстояния между двумя опорами арки. Описание данного контрпримера имеет наиболее важное различие с текущим описанием понятия АРКА в наличии дополнительных отношений КАСАТЬСЯ (для каждой из опор). Машина в состоянии сделать заключение, что именно поэтому контрпример не соответствует понятию АРКА, и выработать в качестве обязательного условия для модели наличие между опорами отношения ДОЛЖНЫ НЕ КАСАТЬСЯ. Это не даст возможность программе идентификации объектов отнести к понятию АРКА объект, имеющий указанное запрещенное отношение.

Итак, мы показали способ, с помощью которого в модель может быть введена как информация, указывающая на необходимость некоторого признака, так и информация, отрицающая наличие некоторого признака.

Рассмотрим теперь, каким образом в модель может быть включена некоторая информация об обобщениях. Поясним суть подхода на примере рис.25, *г,* который указывает на возможность в определении понятия АРКА использовать вместо объекта ПЛИТА объект КЛИН. Простейший способ введения в модель указанного обобщения состоит в добавлении к сети нового понятия, используемого вместо вершины ПЛИТА, соединив вершины ПЛИТА и КЛИН с новым понятием отношением ПОДМНОЖЕСТВО (SUB).

Рассмотрим другой подход к решению этой задачи. На рис. 26 приведено описание понятий ПЛИТА и КЛИН и часть общих знаний, имеющихся у робота и связывающих данные понятия.

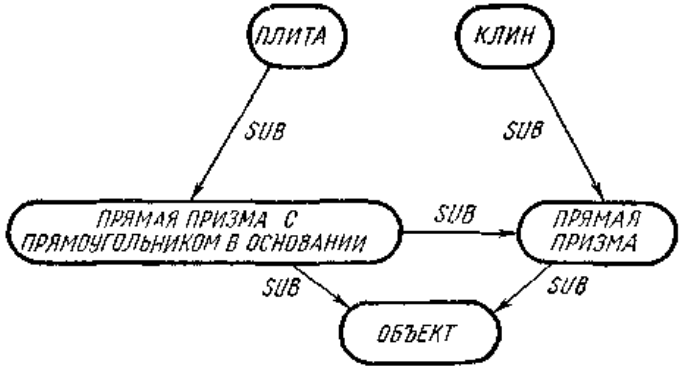


Рис.26. Связь понятий ПЛИТА, КЛИН и ОБЪЕКТ. Все вершины, связаны отношением ПОДМНОЖЕСТВО (SUB).

Из рисунка видно, что первым понятием, включающим множество плит и клиньев, является понятие ПРЯМАЯ ПРИЗМА. На основании этого мы можем заменить в модели вершину ПЛИТА на вершину ПРЯМАЯ ПРИЗМА.

Так как перечисленные выше соображения по образованию модели являются гипотезами, предлагаемыми машиной, то необходим механизм проверки этих гипотез и их корректировки. Ошибки могут быть обнаружены, когда последующие примеры, определяющие понятие, будут противоречить сделанным ранее предположениям. Например, если образец на рис.27, *а* определяет понятие *X,* а рис.27, *б* определяет контрпример, то будет сформировано предположение, что *X* ДОЛЖЕН СТОЯТЬ.

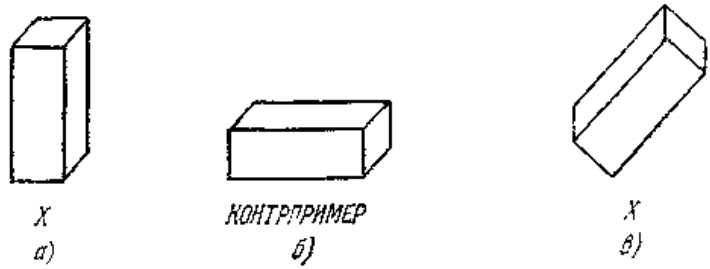


Рис. 27. Последовательность примеров, определяющих понятие *X*.

Отметим, что в принципе на основании, например, вторичных эвристик (см. п. 3 правил упорядочения различий) возможно сформировать альтернативное предположение: X НЕ ДОЛЖЕН ЛЕЖАТЬ. Если мы хотим навязать вторую интерпретацию, то для этого необходимо дать в качестве примера наклонный брусок (рис.27, *в)*. Описание, составленное машиной для наклонного бруска, не имеет отношения СТОЯТЬ и, следовательно, противоречит исходному описанию (рис.27, *а)*. При этих условиях система должна вернуться к предшествующему описанию и рассмотреть альтернативные варианты.

Следует отметить, что в описанном процессе образования модели успех во многом зависит от выбранной последовательности примеров, определяющих понятие. Это не является недостатком предлагаемого процесса, так как и обучение человека в значительной степени зависит от разумного подбора примеров. Более того, именно пренебрежение этим свойством приводило к неудаче многие методы обучения.

Описанный способ образования модели не является универсальным, поэтому при работе со сложными объектами используется интерактивный способ формирования новой модели (понятия). Суть его состоит в следующем. Человек-оператор предъявляет пример нового объекта на экране дисплея и указывает системе его отличительные признаки. Система выделяет из предъявленного примера в первую очередь легко извлекаемые признаки и образовывает на их основе модель нового объекта. Затем система выводит на дисплей изображения, соответствующие новой модели и существовавшим ранее моделям, похожим на нее. Оператор, рассматривая эти изображения, указывает системе на признаки наиболее выжные для различения объектов. **Он обучает систему извлекать эти признаки, пометив на экране их местоположение и указав на существующие (или разработав новые) процедуры выделения признаков.** После того как указанные признаки извлечены, а модель модифицирована, система сравнивает модифицированную модель с существующими, выделяет похожие, оператор указывает на дальнейшие детали и так до тех пор, пока новая модель понятия не будет отличаться от существующих.

Отметим, что включение в репертуар системы нового объекта может потребовать уточнения некоторых существовавших ранее моделей.

Рассмотрим теперь использование вновь образованной модели в процессе опознавания. Задача опознавания может быть определена довольно разнообразно. Рассмотрим следующие разновидности процесса опознавания:

1) опознавание предъявленной сцены;

2) опознавание всех понятий, представленных на сцене;
3) определение присутствия на сцене примера некоторого понятия.

Первая из указанных задач, заключающаяся в опознавании всей сцены, является простейшей.

Модель понятия соответствует анализируемой сцене, если

а) все отношения, помеченные в модели указателем ДОЛЖЕН, присутствуют в описании сцены;

б) все отношения, помеченные в модели указателем НЕ ДОЛЖЕН, не присутствуют в составленном описании;

в) различия, отличающиеся от указанных в пунктах а) и б), являются устранимыми (см. п.15.2.3).

Если сцена содержит много идентифицируемых рядов, стеков или других групп объектов, то необходимо модифицировать программу опознавания в связи с тем, что существенные отношения могут отсутствовать, так как одни объекты могут закрывать другие. Здесь возможно приписывание объекту некоторых признаков на основании признаков групп, в которую он входит.

Последняя задача, состоящая в поиске на сцене некоторого понятия, является наиболее трудной. Прямолинейный подход состоит в сопоставлении полученного описания сцены и модели опознаваемого понятия. Если при этом некоторые объекты сцены сопоставляются с соответствующими частями модели, то отношения, связывающие эти объекты с посторонними объектами сцены, устраняются и выполняется обычная процедура опознавания. В указанном процессе предстоит решить много задач, связанных с тем, как в процессе сопоставления выбирать на основе контекста правильный участок сцены. После того, как понятия опознаны, мы можем получить описание предъявленной сцены в терминах новых более сложных понятий.

15.7. Распознавание образов

Способность узнавать знакомое лицо среди тысяч других, способность найти в лице старика черты друга юности, с которым расстался десятки лет назад, умение отличить круг от эллипса, распознать формы неисчислимого количества разнообразных предметов всегда глубоко интересовали ученых. Еще в эпоху возрождения знаменитый архитектор Леон Баттиста Альбарти (1404—1472 гг.) спрашивал себя: как человек определяет сходство и различие лиц и предметов, сходство и различие человеческих голосов. Ведь лица людей, особенно людей одной и той же национальности, даже лица мужчин и женщин имеют ничтожные различия. И только человеческому глазу и мозгу нетрудно улавливать эти различия.

Человек обычно не может объяснить, как он распознает образ. Пока физиологам известны лишь некоторые качественные стороны этого процесса, но не сами методы, которыми пользуется человек.
Здесь интересно одно обстоятельство. Человек, не умея объяснить, как он сам распознает образ, сравнительно легко может обучить другого человека узнаванию этого образа. Существуют два способа обучения. В первом «учитель» выдает отработанный список правил, или, что то же самое, сообщает алгоритм, по которому нужно действовать. Но возможна и такая ситуация, когда он умеет правильно действовать, но не может объяснить, как это делать. В таком случае обучение может происходить путем наглядного примера. Если «ученик» в конце концов обучается действовать правильно, то он сознательно нашел требуемый алгоритм.
Можно привести простой пример из практики. Неопытному сталевару почти невозможно объяснить, как по цвету пламени определить момент окончания плавки. Но, наблюдая за работой мастера, он через некоторое время познает это искусство, и тогда говорят, что у него появился опыт, интуиция. Суть такого обучения заключается в том, что на основании наблюдений организуется поиск соответствующего алгоритма.
Проблема обнаружения и распознавания сигналов различной формы к 50-м годам XX столетия стала особенно актуальной. Достижения технической кибернетики и электронной вычислительной техники позволили решить широкий круг задач, позволили приступить к созданию машин для перевода с одного языка на другой, машин для управления автоматизированными системами на заводах, на транспорте, автоматов для продажи товаров и т. д. Сложилоеь положение, при котором самые разные отрасли народного хозяйства, промышленности, транспорта, ряда наук все настоятельнее требовали решить проблему обнаружения и опознания широкого класса сложных сигналов различной физической природы. Это повысило гибкость общения человека с различными машинами, которые могут воспринимать команды управления непосредственно с голоса или считывать текст. Во многих случаях решение машинного опознания сигналов позволило существенно повысить степень автоматизации.
С возникновения бионики как нового научного направления получило всеобщее признание мнение о том, что проблема опознания образов является одной из центральных проблем бионики. Во-первых, такому признанию способствовала сложившаяся обстановка, в которой, как говорилось выше, запросы практики всемерно способствовали развитию методов опознания. Во-вторых, эта конкретная задача является действительно бионической по своему существу; ведь с

самого начала и по сей день прообразом и эталоном почти любой опознающей системы является соответственный анализатор человека — зрительный, слуховой, тактильный.

Несмотря на большое количество проведенных в этом направлении исследований, положенно дел таково, что опознание образов продолжает составлять проблему большой важности и огромной трудности, которая далека еще от окончательных решений. Казалось бы в век автоматики, атомной энергии и ракет не так уж трудно создать машину, способную выучить тридцать две буквы русского алфавита, десять цифр и несколько знаков пунктуации. И действительно, есть машины, которые могут распознавать буквы и цифры, знаки препинания, знаки математических действий, но при одном условии: буквы и знаки должны иметь всегда совершенно одинаковую форму, постоянные размеры и наклон, они должны быть обязательно напечатаны и не иметь типографских изъянов. Только при соблюдении всех этих требований машина сможет безошибочно распознавать их.

Чтобы машина, умеющая читать, могла решить ту или иную задачу, условие и правила ее решения должны быть переведены с человеческого на машинный язык. Задание и инструкция, переведенные на такой язык, записываются носителе информации. Носитель информации размещают в устройстве, умеющее читать «буквы» такого машинного языка и преобразующее их в сигналы, которые и вводят в электронную вычислительную машину.

Весьма важно создание автоматов, распознающих не только зрительные образы, но и слуховые, осязательные и пр.

Однако вернемся к простейшему распознающему автомату, к читающему автомату. От него требуется лишь одно: преобразовывать буквы алфавита, цифры, знаки пунктуации и знаки математических действий в электрические сигналы. От самого читающего автомата не требуется, чтобы он понимал прочитанное. Он должен выполнять лишь роль глаза, датчика, воспринимающего изображение и сообщающего об этих изображениях мозгу — электронной вычислительной машине, он всего лишь ее глаза.

На смену первым малоэффективным программам опознания пришли новые, более эффективные; наступил этап макетирования новых опознающих устройств — перцептронов (от слова «перцепция»— восприятие).

В результате разработки технических моделей биологических анализаторов было создано несколько экспериментальных образцов перцептронов, предназначенных для автоматического восприятия и опознания зрительных образов. В принципе возможно создание

перцептронов, моделирующих органы слуха, обоняния, осязания и других чувств.

Зрительный перцептрон более всего напоминает сетчатку глаза. В перцептроне имеется несколько слоев «клеток», перерабатывающих сигналы; как и в сетчатке, эти слои соединены между собой сложными множественными связями; первичные сигналы перерабатываются таким образом, что на выходе перцептрона требуется значительно меньше элементов, чем на его входе. Так и на входе сетчатки глаза человека имеется 137 миллионов светочувствительных клеток, а на выходе — всего лишь миллион нервных клеток.

Первый слой в зрительном перцептроне, как и в сетчатке, представляет собой мозаику из светочувствительных клеток — фотоэлементов (рис.28).

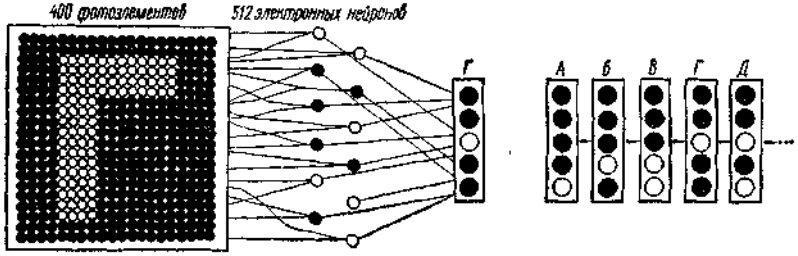


Рис. 28. Зрительный перцептрон.

Их может быть сколько угодно. Для проведения исследований был создан сравнительно простой перцептрон, в квадратной мозаиие которого 400 фотоэлементов — двадцать рядов по двадцать фотоэлементов в каждом. Во втором слое этого иерцонтропа имеется 512 одинаковых электронных нейронов, работающих по описанному выше принципу. Это означает, что если сигнал на входе нейрона меньше некоторой величины, ниже порога, нейрон не проводит импульса; когда же сигнал, хотя бы на самую малость превысит порог, нейрон передаст импульс дальше, в третий слой.

В третьем слое всего пять электронных нейронов. Они и сигнализируют о том, что видит перцептрон. Эти пять нейронов могут проводить импульсы на выход либо все одновременно, либо каждый в отдельности, либо во всех других возможных сочетаниях. Срабатывание каждого из них зависит от первичных сигналов, поступающих от нейронов второго слоя, а срабатывание каждого нейрона второго слоя зависит от сигналов, поступающих от фотоэлементов. Возможных сочетаний срабатываний нейронов

третьего слоя имеется всего 32 (по числу букв в русском алфавите). Это значит, что перцептрон может выдавать 32 различных сигнала и, следовательно, сигнализировать о 32 возможных изменениях образа на его входе.

Выходы 512 нейронов второго слоя соединяются проводами с фотоэлементами. Но соединения эти очень необычны. Как бы вопреки всякой логике, они намеренно соединяются вразброс, совершенно произвольно. Но потом уже эти соединения остаются неизменными. Выходы электронных нейронов второго слоя соединяются со входами нейронов третьего слоя. Здесь соединения сделаны по правилам: к каждому нейрону третьего слоя подходит 512 проводов — по одному проводу от каждого нейрона второго слоя.

В таком устройстве нельзя заранее знать, как и в каких случаях будут работать те или иные клетки. Но перцептрон можно учить началам грамоты, чтобы он умел распознавать 32 буквы алфавита. Требуется, чтобы каждой из букв соответствовал свой сигнал, свой код на выходе перцептрона. Такой сигнал определяется сочетанием включений пяти нейронов последнего слоя. Тогда, например, букве *А* будет соответствовать срабатывание первого нейрона третьего слоя при всех остальных несработавших. При показе буквы *Б* должен сработать только второй нейрон и т. д.

Если теперь на пути от нейронов второго слоя к нейронам третьего слоя установить специальные регуляторы величины сигнала, поступающего от каждого нейрона второго слоя к нейронам третьего слоя (всего 2560 регуляторов), то любой из них можно будет настраивать вручную. Меняя настройку, можно регулировать величину сигналов, поскольку положенно ручек регуляторов после настройки остается неизменным. При этом каждый такой регулятор является еще и запоминающей ячейкой. Поворачивая ручки регуляторов, можно вводить в память перцептрона все необходимые изменения.

Теперь можно производить «обучение» перцептрона. Оно проводится просто: на мозаику из фотоэлементов проектируют поочередно изображение каждой из букв алфавита и при каждом показе в определенном порядке и по определенным правилам поворачивают ручки регуляторов. Увидев букву впервые, перцептрон не может распознать ее и, конечно, не даст на выходе нужного сигнала; в третьем слое сработают совсем не те нейроны.

Но продолжая показывать перцептрону, например, букву *А,* поворачивают ручки регуляторов, добиваясь, чтобы в третьем слое выключились все нейроны, за исключением первого. Таким же способом обучают перцептрон и по остальным буквам. Обучение производится в несколько приемов, так как после первого этапа

перцептрон будет делать очень много ошибок, вернее сказать, он почти не будет работать правильно: в лучшем случае он угадает одну-две буквы. На втором этапе обучения снова приходится поворачивать ручки регуляторов, и так несколько раз. После каждого этапа перцедтрон будет становиться все грамотнее и, наконец, после 15—20 уроков научится совершенно безошибочно распознавать все буквы алфавита.

Опознавание какого-либо объекта может быть определено как формирование некоторого сигнала или кода, однозначно соответствующего опознаваемому объекту, независимо от изоморфных преобразований этого объекта. Под изоморфными преобразованиями понимаются такие видоизменения объекта, которые не приводят к уничтожению его основной сути, т. е. при изоморфных преобразованиях не должны быть уничтожены те основные признаки, по которым опознается объет. Простейшие изоморфные преобразования заключаются в изменении масштаба изображения, его яркости, ориентации и места расположения относительно других объектов или рецептора. Подобные преобразования заглавной буквы *А* иллюстрируются верхним рядом на рис.29.

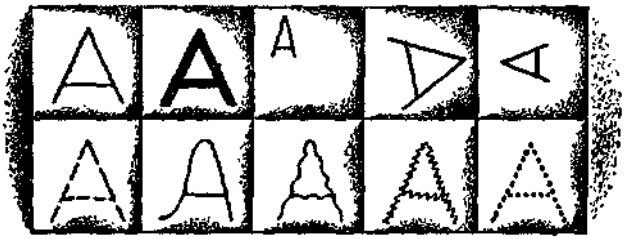


Рис. 29. Различные изменения начертания буквы *А*.

Более сложные преобразования того же изображения показаны в нижнем ряду. Здесь нарушения непрерывности линий буквы или некоторые искажения ее деталей могут сделать невозможным опознание ее автоматом, хотя человек по-прежнему воспринимает все эти начертания как букву *А*. Таким образом, с «точки зрения» автомата некоторые из этих преобразований могут сказаться не изоморфными, тогда как человек воспринимает их как изоморфные. В таких случаях для опознания могут быть использованы усовершенствованные перцептроны, которые как бы обучаются, изменяя в процессе работы свою программу и устанавливая и запоминая при этом основные признаки объекта, по которым он может наиболее эффективно распознаваться.

Многие конструкции распознающих машин сначала преобразуют изображение в последовательность электрических сигналов и подают их для анализа на электроннолучевую трубку. Такие устройства дают в среднем не болев 5 ошибок на миллион прочитанных знаков.
Один из методов машинного распознавания, носящий наименование квазитопологического, может быть упрощенно описан следующим образом. Представим себе устройство, позволяющее совершай, послодовательный обход всех элементов буквы *А* и фиксировать на ней точки обрыва линий кодом *1* и точки, где сходятся три и более линий,— кодом *0.*
Примем далее следующий метод обхода. Начав его, например, с нижней левой точки *а* (рис.30), будем двигаться до точки разветвления *б*, от которой нужно продолжать обход по ближайшему справа пути до точки *г* и далее до точки *д.* При попадании в точку, где линия обрывается (в данном случае точка *д),* нужно повернуть обратно и по тем же правилам продолжать движение до возвращения в исходную точку *а.*

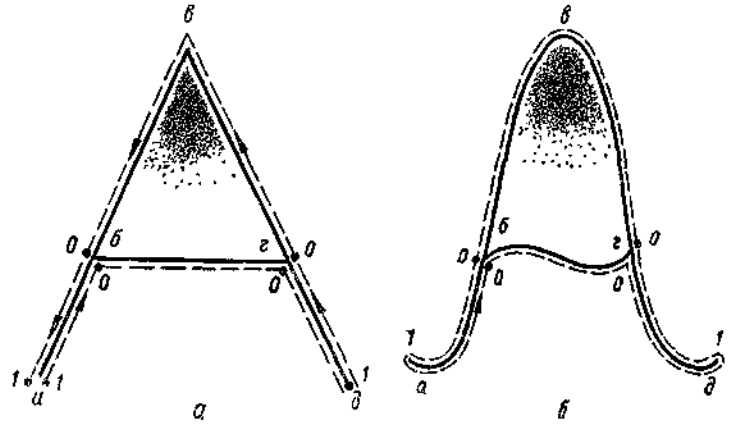


Рис. 30. Квазитопологический метод опознания.

Таким образом, при обходе буквы *А,* изображенной на рис.30, *а,* будут пройдены точки *а, б, г, д, г, в, б, а.* Это в соответствии с принятыми нами обозначениями характерных точек даст код: 1001001 (точка *в,* в которой нет ни обрыва, ни разветвления линий, не рассматривается как характерная).
Тот же самый код будет получен и при ином начертании буквы *А,* изображенном на рис.30, *б.* Очевидно, что при поворотах или изменении масштаба изображения буквы описанный способ обхода

будет давать один и тот же код, что значительно облегчает процесс опознавания. Тот же код может быть получен и при некоторых изменениях начертания буквы, как, например, закруглении острых углов, замене прямых линий кривыми и пр.
Сейчас производятся более совершенные и сложных перцептроны. У них уже не три слоя, а больше; есть четырохслойные и пятислойные перцептропы. Обучение перцептрона поручают разпознающему автомату.

ЗАКЛЮЧЕНИЕ

Мы показали, что основная проблематика искусственного интеллекта концентрируется вокруг разработки универсальных методов представления и эффективных методов решения задач в этих представлениях. К настоящему времени основные методы представления могут считаться достаточно сформулированными. Теоретико-графические модели, формальные логические исчисления, процедуральные и семантические представления допускают зпистемологически и эвристически адекватное представление широкого круга задач, связанных с созданием систем искусственного интеллекта и интеллектуальных искусственных объектов. Однако, необходимость направленного перебора альтернативных вариантов и особенно поиска оптимальных или близких к оптимальным решений, органически присущая системам искусственного интеллекта, особенно остро ставит вопрос об **эвристической эффективности**. Одним из основных направлений повышения эвристической эффективности является **вложение семантической информации в формализмы представления**. Задание оценочных функций, включающих эвристическую компоненту, непосредственное включение семантических отношений в графические представления, задание рекомендаций о наиболее перспективных путях поиска решений представляют собой различные конкретные и, заметим, неполные решения одного и того же ключевого вопроса — **как накапливать и использовать семантические знания в системах искусственного интеллекта.**
Мы надеемся, что идея накопления и использования семантических знаний, пронизывающая всю работу, не осталась незамеченной для читателя. К сожалению, мы не смогли везде предложить законченные методы воплощения этой идеи в конкретные модели и алгоритмы, но таково современное состояние этой, по нашему мнению, центральной

проблемы искусственного интеллекта. Другие пути повышения эвристической эффективности современных систем искусственного интеллекта, практически не нашедшие в силу своей неразработанности отражения в настоящей работе, сводятся, в основном, к **качественному улучшению систем восприятия и в особенности касаются организации оперативного взаимодействия накопленного знания, восприятия и процессов планирования в ходе решения задачи.**

В плане повышения эвристической эффективности систем следует отметить и разработку технических средств и программных систем, обеспечивающих децентрализованное и интерактивное решение различных фрагментов общей задачи как в локальном масштабе (например, параллельное исследование альтернативных путей решения задачи), так и в глобальном (обеспечение многоцелевого планирования, параллельное построение моделей нескольких независимых источников действия).

Необходимо упомянуть еще об одной важнейшей проблеме, имеющей отношение к искусственному интеллекту.

Методы представления и решения, описанные в этой работе, составляют лишь базисный формализм представления, накопления и использования знания. Открытыми вопросами при этом являются:

— как организовывать текущее знание системы в самостоятельные фрагменты, достаточно общие для использования во множестве ситуаций, и достаточно специализированные, чтобы все эти ситуации были практически полезными?

— каким образом широкий класс актов поведения может быть представлен в виде результата взаимодействия этих фрагментов?

— каким образом осуществлять *структурирование знания* — *статическое,* т. е. по его видам (и каковы эти виды знания?), и *динамическое,* т. е. по типам предъявленных системе задач (и каковы эти типы?)?

— как осуществлять эффективным образом решение задач в таких, основанных на знании системах, т. е как планировать поведение, отвечать на широкий круг вопросов и, наконец, «просто» понимать?

Поставленные вопросы являются традиционными для искусственного интеллекта, но сформулированы на более высоком концептуальном уровне. Тем не менее наивно полагать, что без ответа на них человечество сможет построить действительно «разумные машины». Решение указанных вопросов — дело будущего (хотя, по нашему мнению, не столь отдаленного), однако исследования, развернутые в самое последнее время вокруг построения систем взаимодействующих

сценариев (скелетов, акторов), позволяют смотреть в это будущее вполне оптимистически.

Можно утверждать, что в настоящее время созданы предпосылки, необходимые для перехода к созданию *систем* ИИ*, способных автономно функционировать в открытом динамическом мире, решал практически полезные задачи.* В свою очередь создание таких систем без сомнения приведет к появлению новых плодотворных постановок задач и методов их решения.

Литература

1. В.Ф.Асмус. Проблема интуиции в философии и математике. «Мысль», М.1965.

2. Дж. Барвайс. Введение в логику первого порядка. Справочная книга по математической логике. Ч.1. «Наука», М.1982. Пер. с англ..: Handbook of mathematical logic. J. Barwise (Ed). North-Holland P.C. 1977.

3. Дж.Булос, Р.Джеффри. Вычислимость и логика. М. «Мир» 1994. Пер. с английского: George S. Boolos, Richard C. Jeffrey. Computability and logic. Cambridge University press, 1989.

4. Е.А.Беляев, В.Я.Перминов. Философские и методологические проблемы математики. Изд-во Московского университета, 1981.

5. М.Бунге. Интуиция и наука. «Прогресс», М. 1967. Пер. с английского: M.Bunge. Intuition and Science. New York, 1962.

6. Н.Бурбаки. Начала математики. Ч.1, кн.1. Теория множеств. Мир. М. 1965. Пер. с французского.: Elements de Mathematique par N.Bourbaki. Livre 1. Theorie des ensembles. Troisieme edition, 1958.

7. Е.Вигнер. Непостижимая эффективность математики в естественных науках. УФН, т.94, вып.3, 1968, 535 – 546. Пер. с англ.: E.Wigner. The Unreasonable Effectiveness of Mathematics in the Natural Sciences, Comm. Pure and Appl. Math. 131, 1 (1960).

8. Р.Декарт. Правила для руководства ума. Избранные произведения. М.1950. Пер. с французского: Descartes R. Oeuvres, t. X. Paris, 1908.

9. М.Клайн. Математика. Утрата определённости. М.”Мир”, 1984. Пер. с англ.: Morris Kline. MATHEMATICS. The Loss of Certainty. N-Y, Oxford University Press, 1980.

10. С.К.Клини. Введение в метаматематику. ИЛ М. 1957. Пер.с англ. : Introduction tu metamathematics by Stephen Cole Kleene. 1952. D.van Nostrand Company, inc. New York , Toronto.

11. М.Кац, С.Улам. Математика и логика. Ретроспектива и перспективы. «Мир», М. 1971. Пер. с англ.: Mathematics and Logic. Retrospect and Prospects. Mark Kac and Stanislaw M. Ulam. N.-Y. Washington. London. 1968.

12. А.Е. Кононюк. Общая теория познания и созидания. Кн.1. Киев: «Освіта України», 2013. 648 с. *ecat.diit.edu.ua:81/ft/index_ru.html*
13. А.Е. Кононюк. Общая теория познания и созидания. Кн.2, ч.1. Киев: «Освіта України», 2013. 544 с.
ecat.diit.edu.ua:81/ft/index_ru.html
14. А.Е. Кононюк. Общая теория познания и созидания. Кн.2, ч.2. Киев: «Освіта України», 2013. 644 с.
ecat.diit.edu.ua:81/ft/index_ru.html
15. А.Е. Кононюк. Информациология. Общая теория информации. Кн.1. Киев: «Освіта України», 2011. 476 с.
ecat.diit.edu.ua:81/ft/index_ru.html
16. А.Е. Кононюк. Информациология. Общая теория информации. Кн.2. Киев: «Освіта України», 2011. 476 с.
ecat.diit.edu.ua:81/ft/index_ru.html
17. А.Е. Кононюк. Информациология. Общая теория информации. Кн.3. Киев: «Освіта України», 2011. 412 с.
ecat.diit.edu.ua:81/ft/index_ru.html
18. А.Е. Кононюк. Информациология. Общая теория информации. Кн.4. Киев: «Освіта України», 2011. 488 с.
ecat.diit.edu.ua:81/ft/index_ru.html

19. А.Е. Кононюк. Общая теория понятий. Кн.1. Киев: «Освіта України», 2014. 514с.

20. А.Е. Кононюк. Общая теория понятий. Кн.2. Киев: «Освіта України», 2014. 544с.

21. А.Е. Кононюк. Общая теория понятий. Кн.3. Киев: «Освіта України», 2014. 614с.

22. А.Е. Кононюк. Системология. Общая теория систем. Кн.1. Киев: «Освіта України», 2012. 564с. *ecat.diit.edu.ua:81/ft/index_ru.html*

23. А.Е. Кононюк. Системология. Общая теория систем. Кн.2. Ч.1. Киев: «Освіта України», 2014. 558с.
ecat.diit.edu.ua:81/ft/index_ru.html
24. А.Е. Кононюк. Системология. Общая теория систем. Кн.2. Ч.2. Киев: «Освіта України», 2014. 658с.
ecat.diit.edu.ua:81/ft/index_ru.html
25. А.Е. Кононюк. Системология. Общая теория систем. Кн.2. Ч.1. Киев: «Освіта України», 2014. 558с.
ecat.diit.edu.ua:81/ft/index_ru.html
26. А.Е. Кононюк. Общая теория распознавания. Кн.1. Киев: «Освіта України», 2012. 584 с. *ecat.diit.edu.ua:81/ft/index_ru.html*
27. А.Е. Кононюк. Общая теория распознавания. Кн.2. Киев: «Освіта України», 2012. 588 с. *ecat.diit.edu.ua:81/ft/index_ru.html*
28. А.Е. Кононюк. Консалтология. Общая теория консалтинга. Кн.1. Киев: «Освіта України», 2013. 448 с.
ecat.diit.edu.ua:81/ft/index_ru.html
29. А.Е. Кононюк. Консалтология. Общая теория консалтинга. Кн.2. Киев: «Освіта України», 2013. 412 с.
ecat.diit.edu.ua:81/ft/index_ru.html
30. А.Е. Кононюк. Консалтология. Общая теория консалтинга. Кн.3. Киев: «Освіта України», 2013. 520 с.
ecat.diit.edu.ua:81/ft/index_ru.html
31. А.Е. Кононюк. Консалтология. Общая теория консалтинга. Кн.4. Киев: «Освіта України», 2013. 508 с.
ecat.diit.edu.ua:81/ft/index_ru.html
32. А.Е. Кононюк. Дискретно-непрерывная математика. Начала. Кн.1. Киев: «Освіта України», 2012. 652с.
ecat.diit.edu.ua:81/ft/index_ru.html
33. А.Е. Кононюк. Дискретно-непрерывная математика. Множества. Кн.2. Ч.1. Киев: «Освіта України», 2012. 452с.
ecat.diit.edu.ua:81/ft/index_ru.html
34. А.Е. Кононюк. Дискретно-непрерывная математика. Множества. Кн.2. Ч.2. Киев: «Освіта України», 2013. 536 с.
ecat.diit.edu.ua:81/ft/index_ru.html
35. А.Е. Кононюк. Дискретно-непрерывная математика. Отношения. Кн.3. Ч. 1. Киев: «Освіта України», 2013. 552с.
ecat.diit.edu.ua:81/ft/index_ru.html

36. А.Е. Кононюк. Дискретно-непрерывная математика. Отношения. Кн.3. Ч. 2. Киев: «Освіта України», 2013. 548 с.
ecat.diit.edu.ua:81/ft/index_ru.html
37. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.1. Киев: «Освіта України», 2011. 452с.
ecat.diit.edu.ua:81/ft/index_ru.html
38. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.2. Киев: «Освіта України», 2011. 668 с.
ecat.diit.edu.ua:81/ft/index_ru.html

39. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.3. Киев: «Освіта України», 2015. 488 с.
http://lib.sumdu.edu.ua/library/DocDescription?doc_id=640902

40. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.4. Киев: «Освіта України», 2015. 548 с.

41. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.5. Киев: «Освіта України», 2015. 528 с.

42. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.6. Киев: «Освіта України», 2015. 608 с.

43. А.Е. Кононюк. Дискретно-непрерывная математика. Матрицы. Кн.5. Ч.1. Киев: «Освіта України», 2013. 612 с.
ecat.diit.edu.ua:81/ft/index_ru.html

44. А.Е. Кононюк. Дискретно-непрерывная математика. Матрицы. Кн.5. Ч.2. Киев: «Освіта України», 2013. 500 с.
ecat.diit.edu.ua:81/ft/index_ru.html

45. А.Е. Кононюк. Дискретно-непрерывная математика. Матрицы. Кн.5. Ч.3. Киев: «Освіта України», 2013. 520 с.
ecat.diit.edu.ua:81/ft/index_ru.html
46. А.Е. Кононюк. Дискретно-непрерывная математика. Матрицы. Кн.5. Ч.4. Киев: «Освіта України», 2013. 508 с.
ecat.diit.edu.ua:81/ft/index_ru.html

47. А.Е. Кононюк. Дискретно-непрерывная математика. Матрицы. Кн.5. Ч.5. Киев: «Освіта України», 2013. 672 с.
ecat.diit.edu.ua:81/ft/index_ru.html
48. А.Е. Кононюк. Дискретно-непрерывная математика. Поверхности. Кн.6. Ч.1. Киев: «Освіта України», 2012. 652с.
ecat.diit.edu.ua:81/ft/index_ru.html
49. А.Е. Кононюк. Дискретно-непрерывная математика. Графы. Кн.7. Ч.1 Киев: «Освіта України», 2014. 652с.
ecat.diit.edu.ua:81/ft/index_ru.html
50. А.Е. Кононюк. Дискретно-непрерывная математика. Графы. Кн.7. Ч.2 Киев: «Освіта України», 2014. 552с.
ecat.diit.edu.ua:81/ft/index_ru.html
51. А.Е. Кононюк. Дискретно-непрерывная математика. Графы. Кн.7. Ч.3 Киев: «Освіта України», 2015. 512с.
ecat.diit.edu.ua:81/ft/index_ru.html
52. А.Е. Кононюк. Дискретно-непрерывная математика. Графы. Кн.7. Ч.4 Киев: «Освіта України», 2015. 552с.
ecat.diit.edu.ua:81/ft/index_ru.html

53. А.Е. Кононюк. Дискретно-непрерывная математика. Графы. Кн.7. Ч.5 Киев: «Освіта України», 2015. 660с.

54. А.Е. Кононюк. Обобщенная теория моделирования. Кн.1. Ч.1 Киев: «Освіта України», 2012. 602с.
ecat.diit.edu.ua:81/ft/index_ru.html
55. А.Е. Кононюк. Обобщенная теория моделирования. Кн.1. Ч.2 Киев: «Освіта України», 2012. 708с. *ecat.diit.edu.ua:81/ft/index_ru.html*
56. А.Е. Кононюк. Обобщенная теория моделирования. Кн.1. Ч.3 Киев: «Освіта України», 2012. 568с. *ecat.diit.edu.ua:81/ft/index_ru.html*
57. А.Е. Кононюк. Обобщенная теория моделирования. Кн.2. Киев: «Освіта України», 2012. 548с. *ecat.diit.edu.ua:81/ft/index_ru.html*
58. А.Е. Кононюк. Обобщенная теория моделирования. Кн.3. Ч.1 Киев: «Освіта України», 2012. 636с. *ecat.diit.edu.ua:81/ft/index_ru.html*
59. А.Е. Кононюк. Обобщенная теория моделирования. Кн.3. Ч.2 Киев: «Освіта України», 2012. 448с. *ecat.diit.edu.ua:81/ft/index_ru.html*
60. А.Е. Кононюк. Обобщенная теория моделирования. Кн.3. Ч.3 Киев: «Освіта України», 2013. 588с. *ecat.diit.edu.ua:81/ft/index_ru.html*
61. А.Е. Кононюк. Основы теории оптимизации. Кн.1. Киев: «Освіта України», 2011. 602с. *ecat.diit.edu.ua:81/ft/index_ru.html*

62. А.Е. Кононюк. Основы теории оптимизации. Кн.2. Ч.1. Киев: «Освіта України», 2011. 552с. *ecat.diit.edu.ua:81/ft/index_ru.html*
63. А.Е. Кононюк. Основы теории оптимизации. Кн.2. Ч.2. Киев: «Освіта України», 2011. 616с. *ecat.diit.edu.ua:81/ft/index_ru.html*
64. А.Е. Кононюк. Основы теории оптимизации. Кн.2. Ч.3. Киев: «Освіта України», 2012. 456с. *ecat.diit.edu.ua:81/ft/index_ru.html*
65. А.Е. Кононюк. Основы теории оптимизации. Кн.2. Ч.4. Киев: «Освіта України», 2012. 512с. *ecat.diit.edu.ua:81/ft/index_ru.html*
66. А.Е. Кононюк. Основы научных исследований. Кн.1. Киев: «Освіта України», 2011. 508с. *ecat.diit.edu.ua:81/ft/index_ru.html*
67. А.Е. Кононюк. Основы научных исследований. Кн.2. Киев: «Освіта України», 2011. 452с. *ecat.diit.edu.ua:81/ft/index_ru.html*
68. А.Е. Кононюк. Основы научных исследований. Кн.3. Киев: «Освіта України», 2011. 456с. *ecat.diit.edu.ua:81/ft/index_ru.html*
69. А.Е. Кононюк. Основы научных исследований. Кн.4. Киев: «Освіта України», 2011. 456с. *ecat.diit.edu.ua:81/ft/index_ru.html*

70. А.Е. Кононюк. Общая теория коммуникаций. Кн.1. Киев: «Освіта України», 2014. 488с.

71. А.Е. Кононюк. Нейроні мережі і генетичні алгоритми. Киев: «Корнійчук», 2010. 448с. *ecat.diit.edu.ua:81/ft/index_ru.html*

72. Кононюк А. Е. Обобщенная теория познания и созидания. [В 2 кн.] Кн. 1 : Начала / А. Е. Кононюк. — Киев : Освіта України, 2013. *ecat.diit.edu.ua:81/ft/index_ru.html*

73. Кононюк А. Е. Обобщенная теория познания и созидания. [В 2 кн.] Кн. 2 : Теория познания. Ч. 1 / А. Е. Кононюк. — Киев : Освіта України, 2013 *ecat.diit.edu.ua:81/ft/index_ru.html*

74. Кононюк А. Ю. Вища математика. (Модульна технологія навчання) : навчальний посібник : в 2 кн. / А. Ю. Кононюк. — Київ : КНТ, 2009 — Кн. 1. — 2009. — 702 с. *ecat.diit.edu.ua:81/ft/index_ru.html*

75. Кононюк А. Ю. Вища математика. (Модульна технологія навчання) : навчальний посібник : в 2 кн. / А. Ю. Кононюк. — Київ :

КНТ, 2009 Кн. 2. — 2009. — 790 с. *ecat.diit.edu.ua:81/ft/index_ru.html*

76. А.Е. Кононюк. Дискретно-непрерывная математика. Поверхности. Кн.6. Ч.2. Киев: «Освіта України», 2012. 652с. http://www.dut.edu.ua/ua/lib/127/category/96/view/1297

77. А.Е. Кононюк. Дискретно-непрерывная математика. Пространства. Кн.8. Ч.1. Киев: «Освіта України», 2016. 748 с. http://www.dut.edu.ua/ua/lib/1/category/96/view/1439

78. А.Е. Кононюк. Дискретно-непрерывная математика. Пространства. Кн.8. Ч.2. Киев: «Освіта України», 2016. 480с. http://lib.sumdu.edu.ua/library/DocDescription?doc_id=640775

79. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.1. Киев: «Освіта України», 2016. 568с.

80. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.2. Киев: «Освіта України», 2016. 558с.

81. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.3. Киев: «Освіта України», 2016. 588с.

82. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.4. Киев: «Освіта України», 2016. 552с

83. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.5. Киев: «Освіта України», 2016. 836 с

84. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.6. Киев: «Освіта України», 2016. 576 с

85. Алберг Дж., Нильсон Э., Уолш Дж. Теория сплайнов и ее приложения.—М.: Мир, 1972 (Ahlberg J.H., Nilson E.N., Walsh J.L. The Theory of Splines and their Applications.—New York: Academic Press, 1967).

86. Березин И.С., Жидков Н.П. Методы вычислений. Т. 1,2.—М.: Наука, 1962, 1966.

87. Гардан П., Люка М. Машинная графика и автоматизация конструирования.— М.: Мир, 1987 (Techniques Graphiques Interactives et C.A.O./par Michel Lucas et Yvon Gar dan.—Prance: Hermes Publishing, 1983).

88. Гильберт Д., Кон-Фоссен С. Наглядная геометрия.—М.: Наука, 1981 (Hilbert D., Cohn-Vossen S. Anschauliche Geometrie.—Berlin: 1932).

89. Де Бор К. Практическое руководство по сплайнам.—М.: Радио и связь, 1985 (De Boor C. A Practical Guide to Splines.—Berlin: Springer, 1879).

90. Демидович Б.П., Марон И.А. Основы вычислительной математики.—М.: Наука, 1970.

91. Дубровин Б.А., Новиков С.П., Фоменко А.Т. Современная геометрия.— М.: Наука, 1986.

92. Ильин В. А., Позняк Э.Г. Аналитическая геометрия.—М.: Наука, 1981.

93. Ильин В. А., Позняк Э.Г. Линейная алгебра.—М.: Наука, 1984.

94. Калиткин Н.Н. Численные методы.—М.: Наука, 1978.

95. Ласло М. Вычислительная геометрия и компьютерная графика на С++ К—М.: Бином, 1997 (Laszlo M. J. Computational Geometry and Computer Graphics in C++.—Prentice Hall, 1996).

96. Препарата Ф., Шеймос М. Вычислительная геометрия.—М.: Мир, 1989 (Preparata F. P., Sham os M. Computational Geometry: An Introduction.— New York, Berlin, Tokyo: Springer-Verlag, 1985).

97. Роджерс Д., Адамс Дж. Математические основы машинной графики.—М.: Машиностроение, 1980 (Rogers D.F., Adams J. A. Mathematical Elements for Computer Graphics.—McGrow-Hill, 1976).

412 с□