Парадигма развития науки

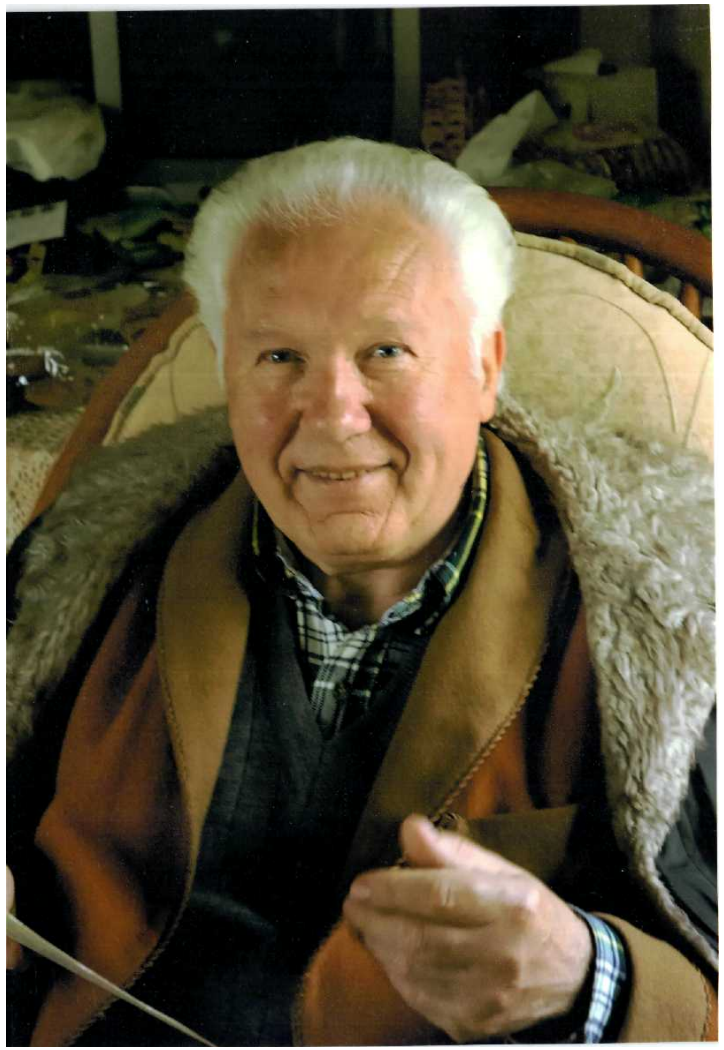
А. Е. Кононюк

Основы фундаментальной теории искусственного интеллекта

Книга 2

Эпистемология искусственного интеллекта

Киев
«Освіта України»
2017



Кононюк Анатолий Ефимович

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 теория искусственного интеллекта. В 20-и кн. Кн.2. — К.:Освіта України.2017.—660 с.

ISBN 978-966-373-693-8 (многотомное издание)
ISBN 978-966-373-694-6 (книга 2)

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 978-966-373-694-6 (книга 2)

Оглавление

1. Общие характеристик систем искусственного интеллекта

1.1. О системах искусственного интеллекта

Одной из важнейших особенностей развития науки является возникновение очень сложной иерархии специализированных дисциплин. На место древнего ученого-философа, такого как Аристотель, который мог охватить практически всю совокупность доступных в его время знаний, пришли поколения ученых, обладающих все большей глубиной знаний и все большей узостью интересов и компетенции. Вероятно, основной причиной, породившей тенденцию к раздроблению науки на узкие специальности, является ограниченность возможностей человеческого разума. Поскольку объем знаний стал больше того, который человек в состоянии воспринять, всякое увеличение знания необходимо приводит к тому, что человек может охватить все меньшую его часть. Чем глубже это знание, тем более специализированным оно должно быть.

Углубление специализации по дисциплинам присуще не только естественным наукам. В других областях человеческой деятельности, например в технике, медицине, гуманитарных науках, искусстве, наблюдается та же тенденция. Так, техника из одной дисциплины превратилась в спектр инженерных отраслей, таких, как механика, электротехника, химическое машиностроение или атомная техника, и каждая из них, в свою очередь, подразделяется на множество узких специальностей.

Одной из главных особенностей науки второй половины XX столетия является появление ряда родственных научных направлений, таких, как кибернетика, общесистемные исследования, теория информации, теория управления, математическая теория систем, теория принятия решений, исследование операций и искусственный интеллект. Все эти области, появление и развитие которых тесно связано с возникновением и прогрессом **компьютерной технологии**, обладают одним общим свойством—они имеют дело с такими **системными задачами, в которых главенствующими являются информационные, реляционные и структурные аспекты**, в то время как **тип сущностей, образующих систему, имеет значительно меньшее значение.** Становится все более очевидным, что полезно было бы посмотреть на эти взаимосвязанные интеллектуальные разработки как

на части более общего поля исследований, которую мы будем называть *наукой об искусственном интеллекте*.

В науке об искусственном интеллекте будем различать три основных компонента:

1) *область* исследования (функционирования);
2) совокупность *знаний* об этой области;
3) *методологию* (совокупность согласованных (взаимосвязанных) методов) накопления новых знаний об этой области и использования этих знаний для решения относящихся к ней интеллектуальных задач.

Осуществим охарактеристику этих трех компонента — **область, знания и методологию науки о системах ИИ**. Кроме того, постараемся привести доводы, что науку о системах ИИ нельзя непосредственно сравнивать с другими науками, а правильнее было бы рассматривать ее как **новое измерение в науке**. Можно сказать, что предметом любой научной дисциплины является определенный класс систем к которым относится и СИИ. В самом деле, термин *система* является одним из самых распространенных терминов, используемых при описании работ в самых разных научных дисциплинах.

Посмотрев в толковые словари, можно найти примерно такое толкование слова «система — *множество элементов, находящихся в отношениях или связях друг с другом, образующих целостность или органическое единство*».

Если следовать общепринятому определению, то термин «**система искусственого интеллекта (СИИ)**» означает, в общем, **множество элементов и отношений между ними**. Термин ***отношение*** используется здесь в самом широком смысле, включающем весь набор родственных понятий, таких, как **ограничение, структура, информация, организация, сцепление, связь, соединение, взаимосвязь, зависимость, корреляция, образец и т. д.** Скажем, система ИИ S представляет, таким образом, упорядоченную пару $S=(A, R)$, где A есть множество соответствующих элементов, а R — множество отношений между элементами множества A. Подобная концепция системы ИИ слишком обща и, следовательно, практическое значение ее невелико. Чтобы сделать это определение практически полезным, его нужно уточнить, ввести определенные классы упорядоченных пар (A, R), относящихся к выделенным интеллектуальным задачам. Эти классы можно ввести с помощью одного из **двух фундаментальных критериев различия**:

а) выделение систем ИИ, базирующихся на определенных типах элементов;

б) выделение систем ИИ, базирующихся на определенных типах отношений.

Классификационные критерии а) и б) можно рассматривать как ортогональные. Примером критерия а) служит традиционное подразделение науки и техники на дисциплины и специальности, причем каждая из них занимается определенным типом элементов (физических, химических, биологических, политических, экономических и т. д.). При этом никакой определенный тип отношений не фиксируется. **Поскольку элементы разных типов требуют разных экспериментальных (инструментальных) средств для сбора данных, эта классификация по существу имеет *экспериментальную основу.***

Критерий б) дает совершенно другую классификацию систем ИИ: **класс задается определенным типом отношений,** а тип элементов, на которых определены эти отношения, не фиксируется. **Такая классификация непосредственно связана с обработкой данных, а не сих сбором, и основа ее преимущественно теоретическая.**

Как будет подробно рассмотрено ниже, самыми большими классами систем по критерию б) являются **классы, описывающие различные *эпистемологические (познавательные) уровни,* т. е. уровни знания относительно рассматриваемых феноменов.** Далее они уточняются с помощью разных *методологических отличий.* Каждый класс систем ИИ, заданный определенным эпистемологическим уровнем и конкретными методологическими отличиями, **подразделяется далее на еще меньшие классы.** Каждый из этих классов состоит из систем ИИ, эквивалентных с точки зрения конкретных, практически существенных сторон определенных в них отношений. **Такую эквивалентность будем называть изоморфизмом, а определенные по ней классы эквивалентности — *изоморфными классами СИИ.***

В зависимости от характеристик отношений, относительно которых требуется изоморфность систем ИИ, одни изоморфные классы являются подмножествами других. Ясно, что наименьшими изоморфными классами являются такие классы, в которых входящие в них системы ИИ изоморфны относительно всех характеристик определенных на них отношений.

Поскольку системы ИИ в каждом конкретном изоморфном классе эквивалентны только с точки зрения некоторых характеристик их отношений, они могут базироваться на совершенно разных типах элементов. Если рассматривать только характеристики отношений в системах ИИ, то достаточно каждый класс изоморфных систем заменить одной системой ИИ, представляющей этот класс. Так как выбор этих представителей в принципе произволен, то важно, чтобы

для всех изоморфных классов использовался один и тот же критерий выбора. **Для наших целей будем выбирать в качестве представителей системы ИИ, в которых множествами элементов являются абстрактные (неинтерпретированные) множества одной природы, а отношения описаны в подходящей стандартной форме**.
Представителей изоморфных классов, удовлетворяющих этим требованиям, при определенной интерпретации термина «стандартный» будем называть общими системами ИИ. Таким образом, ***<u>общая система ИИ</u>— это стандартная и неинтерпретированная система, выбранная в качестве представителя класса систем, эквивалентных (изоморфных) относительно некоторых практически существенных характеристик отношений.***
В этом определении термин «стандартная» используется для ссылки на описание, удовлетворяющее определенным соглашениям, которые определяются в первую очередь применением данной системы ИИ. Например, соответствующая **форма представления системы ИИ в <u>компьютере</u> может быть принята в качестве ее стандартного описания.**
Ортогональность классификационных критериев а) и б) показана на рис. 1.

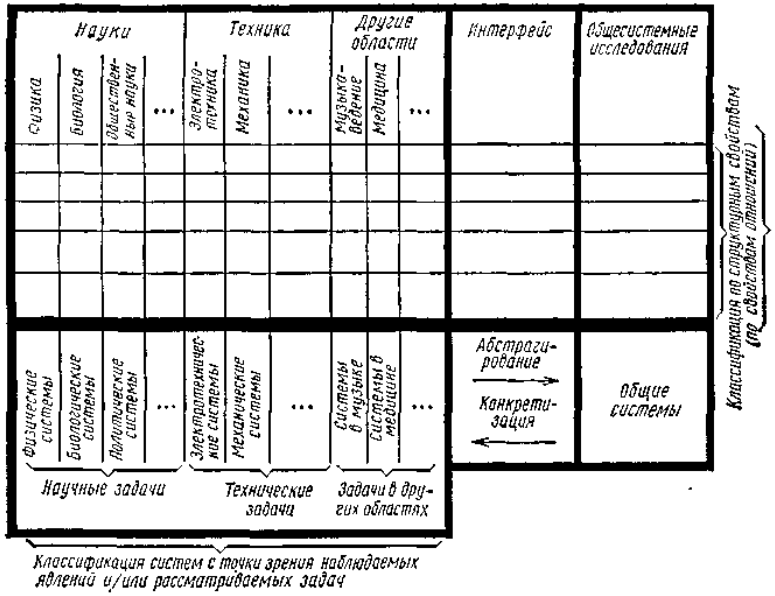


Рис. 1. Два способа классификации систем ИИ

Классы систем ИИ, содержащие различные типы элементов (множество *A),* изображаются горизонтальными полосами; классы систем ИИ, содержащие различные отношения (множество *R),* — вертикальными.
Хотя классификация по критерию б) не совсем приемлена в традиционной науке, ее важность значительна. **Все исследования свойств систем ИИ и связанные с этим интеллектуальные задачи, проистекающие из данной классификации будем называть «<u>наука о системах искусственного интеллекта</u>»**.
В *область науки о системах ИИ* будем включать все типы свойств отношений, существенные для отдельных классов систем ИИ или в очень редких случаях существенные для всех систем. Предложенная классифакация систем ИИ по отношениям определяет способ разбиения области исследований науки о системах на подобласти точно так же, как традиционная наука подразделяется на подобласти — различные дисциплины и специальности.
Знания в науке о системах ИИ, т. е. знания, относящиеся к различным классам свойств отношений в системах ИИ, можно получать **либо с помощью математики, либо с помощью экспериментов с моделями систем, на компьютерах.** Примерами знаний в науке о системах, полученных математическим путем, являются закон необходимого разнообразия Эшби, принципы максимума энтропии и минимума кросс-энтропии или различные законы об информации, управляющей системами. Если говорить о знаниях, полученных экспериментальным путем, то ***лабораторией для науки о системах ИИ*** **является компьютер**. Он позволяет экспериментировать ученому-системщику точно так же, как это делают другие ученые в своих лабораториях, хотя экспериментальные понятия, которыми он оперирует, представляют собой абстрактные структурные (**моделируемые на компьютере**), **а не конкретные <u>свойства реального мира</u>**. В данной работе описываются некоторые примеры системных знаний, полученные с помощью экспериментов на компьютере.
Третья компонента науки о системах ИИ — *системная методология*— это стройная совокупность методов изучения свойств различных классов систем ИИ и решения интеллектуальных задач, т. е. задач, касающихся отношений в системах ИИ. Хорошая классификация систем ИИ с точки зрения отношений — **ядро системной методологии**. Соответствующим образом разработанная классификация является **основой для исчерпывающего описания и таксономии интеллектуальных задач.** Главная задача системной методологии — предоставление в распоряжение потенциальных пользователей, представляющих разные

дисциплины и предметные области, методов решения всех определенных типов интеллектуальных задач.
Если проанализировать разделение традиционной науки на дисциплины, то станет очевидно, что наука о системах ИИ носит междисциплинарный характер. Этот факт имеет, по крайней мере, два следствия. Во-первых, системные знания и методология в принципе могут быть использованы практически во всех разделах традиционной науки. Во-вторых, наука о системах ИИ обладает гибкостью, позволяющей изучать свойства отношений в таких системах и, следовательно, в интеллектуальных задачах, где фигурируют характеристики, исследуемые обычно в самых разных областях традиционной науки. Это позволяет изучать подобные системы и решать такие интеллектуальные задачи в целом, а не рассматривать их как собрание несвязанных предметных подсистем и подзадач.
Как уже говорилось выше, наука о системах ИИ, как и другие науки, имеет определенную область исследования, обладает совокупностью знаний и методологией. И тем не менее это не наука в обычном смысле: традиционная наука ориентируется на исследование разных категорий явлений, а наука о системах ИИ изучает различные классы отношений. По существу, ее можно рассматривать как **новое измерение науки**, а не как новую науку, сопоставимую с другими.
Два измерения в науке, которые отражает двумерная классификация систем ИИ, показанная на рис. 1, являются взаимодополняющими. Их сочетание в научных исследованиях оказывается более мощным средством, чем использование каждого из направлений в отдельности. Традиционное измерение науки определяет смысл и место любого исследования. С другой стороны, системное измерение позволяет содержательно работать с любой наперед выбранной системой, независимо от того, ограничена ли она рамками одной традиционной научной дисциплины или нег.
Представляется, что с точки зрения свойств науки в истории человечества можно естественным образом выделить три основных периода.
1. *Донаучный период* (приблизительно до XVI в.). Характерными чертами периода являются здравый смысл, теоретизирование, метод проб и ошибок, ремесленные навыки, дедуктивные рассуждения и опора на традицию.
2. *Одномерная наука* (начало XVII — середина XX вв.). Характерные черты: объединение теорий, дедуктивные рассуждения, особое внимание к эксперименту, которое привело к возникновению базирующихся на эксперименте дисциплин и специальностей в науке. Они появились прежде всего из-за различий в экспериментальных

(инструментальных) средствах, а не из-за различий в свойствах отношений исследуемых систем.
3. *Двумерная наука* (развивается примерно с середины XX в.). Характерные черты: возникновение науки о системах, в том числе и науках о системах ИИ, занимающейся свойствами отношений, а не экспериментальными свойствами исследуемых систем, и ее интеграция с основанными на эксперименте традиционными научными дисциплинами.
Таким образом, можно сказать, что главное в развитии науки во второй половине нашего века — это переход от одномерной науки, в основном опирающейся на экспериментирование, к науке двумерной, в которую наука о системах, базирующаяся прежде всего на отношениях, постепенно входит в качестве второго измерения. Важность этой совершенно новой парадигмы науки, двумерность науки, еще не вполне осознана, но ее последствия для будущего представляются чрезвычайно глубокими.

1.2. Решение интеллектуальных задач

Как отмечал К. Норберг-Шульц: «*Только при полном понимании задач можно найти соответствующие способы их решения. Для результатов важнее поставить правильные вопросы, чем правильно ответить на ошибочные».*
С понятием решения интеллектуальных задач главной проблемой искусственного интеллекта является (связано) **три вопроса о его смысле и существенности**:
1. Можно ли выделить интеллектуальные задачи в особый и хорошо определенный класс задач?
2. Может ли класс интеллектуальных задач быть описан операционно, чтобы стало возможным создание доступной методологии решения задач этого класса?
3. Имеет ли класс интеллектуальных задач достаточное практическое значение, чтобы оправдать работы по развитию методологии решения интеллектуальных задач?
Видимо, на все эти вопросы можно ответить положительно.
Обоснование этого мнения может быть понятно только по прочтении данной работы, а сейчас приведем следующие краткие предварительные соображения.
Как уже говорилось в разд. 1.1, понятие общих систем ИИ как стандартных представителей практически важных классов эквивалентности систем естественным образом возникает из двумерной классификации систем ИИ, изображенной на рис. 1. Хотя и совер-

шенно ясно, что общие системы бесконечно разнообразны, **это разнообразие может быть адекватно охвачено *конечным числом типов общих систем ИИ,* каждый из которых характеризуется определенным эпистемологическим (познавательным) уровнем и конечным набором соответствующих и существенных методологических отличий.**

Поскольку существенные типы общих систем определены, постольку они образуют пространство, на котором можно определить типы интеллектуальных задач. Это пространство будем называть *пространством интеллектуальных задач.* Любой тип интеллектуальных задач определяется в терминах **упорядоченных связей между двумя типами систем ИИ —начальным и конечным, а также набором типов требований, совместимых с типами этих систем ИИ.**

Для конкретных интеллектуальных задач эти требования могут быть **целями или ограничениями**. Хотя разнообразие реальных требований, предъявляемых к непустому пространству интеллектуальных задач, бесконечно, они могут быть адекватно представлены конечным числом типов, как уже указывалось в предыдущем разделе. Таким образом, **любой тип интеллектуальных задач характеризуется типами двух рассматриваемых систем ИИ и конечным набором определенных типов требований.**

Интеллектуальная задача определенного типа станет конкретной, если заданы конкретные требования для всех типов и в зависимости от типов требований заданы конкретная **исходная система ИИ определенного типа или системы ИИ обоих определенных типов.** В первом случае **начальная система ИИ представляет собой начальное состояние интеллектуальной задачи**. **Решение интеллектуальной задачи (целевое состояние интеллектуальной задачи) представляет собой одну (или более) конкретную конечную систему ИИ требуемого типа. Во втором случае начальное состояние интеллектуальной задачи представляется двумя конкретными системами ИИ, а решением является некоторое отношение между ними.**

Из такой характеристики интеллектуальных задач следует, что они образуют особый хорошо определенный класс общих интеллектуальных задач. Тот факт, что бесконечное разнообразие этих интеллектуальных задач сводимо к конечному числу хорошо определенных типов интеллектуальных задач, **делает возможным создание методологии решения данного класса интеллектуальных задач.** Таким образом, **на первые два наших вопроса можно ответить положительно**. Третий вопрос требует дополнительного обсуждения.

Решение интеллектуальных задач, как мы утверждаем, сводится к решению интеллектуальных задач, состояния которых представлены общими системами ИИ хорошо определенного типа. Тем самым, рассматриваются только те аспекты интеллектуальных задач, которые свободно интерпретируются и не зависят от контекста. Таким образом, **применение методологии решения интеллектуальных задач основано на допущении, что из конкретных интеллектуальных задач могут быть выделены свободно интерпретируемые и контекстно независимые интеллектуальных задачи.**

Имеет ли смысл и нужно ли делить интеллектуальные задачи таким образом? Мы считаем, что это так. В самом деле, используем же мы это деление при решении простых повседневных задач, применяя, например, арифметику. Б. Зиглер очень хорошо формулирует:

«|Ни у кого не вызывает сомнения роль арифметики в науке, технике и управлении. Арифметика проникла повсюду, но при этом она является математической дисциплиной с собственными аксиомами и логической структурой. Ее содержание не принадлежит никакой другой дисциплине, но ее ко всем можно применить. Так, студентов-биологов и студентов-инженеров учат сложению одинаково, различие состоит в том, что, когда и затем складывать.

На практике моделирование и имитация также проникли во все области. Однако они имеют собственные подходы к описанию модели, ее упрощению, обоснованию, имитации, изучению, и эти подходы принадлежат только данной конкретной дисциплине. С этими утверждениями согласятся все. Никто, однако, не будет утверждать, что названные подходы можно выделить и абстрагировать в общепринятом виде.»

Хотя Зиглер говорит о моделировании и имитации, его наблюдения равным образом применимы и к другим классам задач, таким, как проектирование систем ИИ, их анализ, идентификация, реконструкция, управление, оценка функционирования, тестирование и т. д. Для многих подзадач этих задач могут быть созданы тонкие методы решения в терминах соответствующих общих систем ИИ, т. е. не связанные определенной интерпретацией или контекстом. Подобные методы значительно повышают эффективность и унифицируют процесс решения сложных интеллектуальных задач точно так же, как арифметика облегчает решение очень простых задач.

Назовем концептуальную схему, в которой типы интеллектуальных задач определены совместно с методами решения интеллектуальных задач этих типов, *фундаментальным решателем интеллектуальных задач* (ФРИЗ).

При решении интеллектуальных задач в различных контекстах, связанных с разными традиционными областями науки, техники, медицины и т. д., а также в междисциплинарных исследованиях **ФРИЗ** должен рассматриваться в первую очередь как **методологическое средство, использующее компьютерные средства (системы).** Располагая этими средствами, следует обращаться к ним всякий раз, когда в процессе решения какой-то проблемы возникает необходимость решения интеллектуальных задач.
На рис. 2 показана структура ФРИЗ как методологическое средство научного исследования СИИ в различных областях науки.

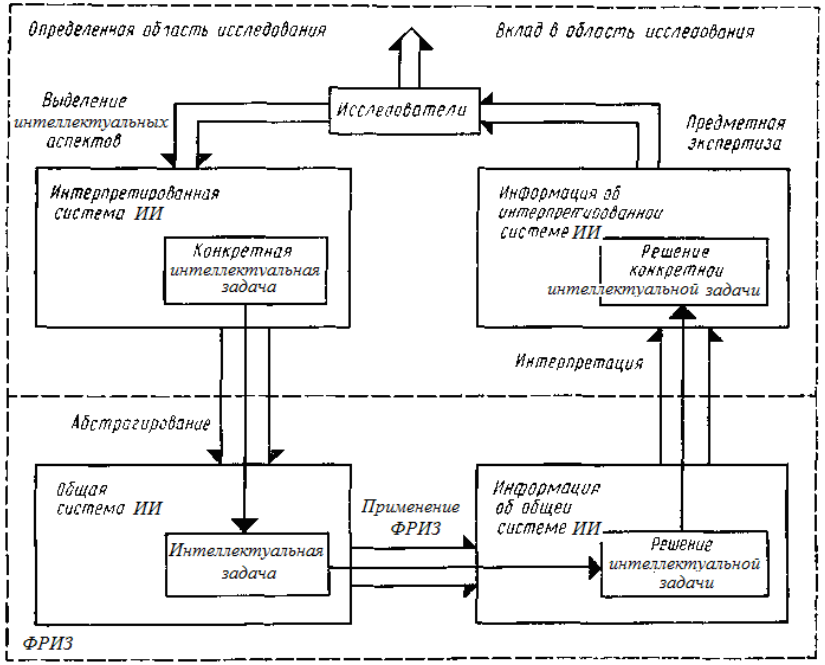


Рис. 2. Структура ФРИЗ как методологическое средство СИИ

В работе ФРИЗ можно выделить **два уровня операций**.
1. (Представлен внутренними прямоугольниками). Если исследователь достаточно знаком с базовым языком ФРИЗ, чтобы сформулировать интерпретацию своей задачи в виде интеллектуальной задачи, то в этом случае исследователь или пользователь сам определяет интерпретированную задачу в терминах ФРИЗ (как будет описано ниже), а ФРИЗ решает задачу и отображает решение в термины интепретированной системы ИИ. Подобная ситуация также

возникает, если необходимо разработать процедуры, имеющие вид простых вопросов, задаваемых пользователю. Отвечая на эти вопросы пользователь определяет интеллектуальную задачу, подходящую к данной ситуации.
2. (Внешние прямоугольники). Многие интеллектуальные исследования достаточно сложны, так что исследователь может содержательно использовать больший объем информации, чем это требуется для решения определенной интеллектуальной задачи. В этом случае также можно разработать процедуры, производящие преобразования из интерпретированной системы ИИ в общую. Основываясь на информации, сопровождающей это преобразование, ФРИЗ может перевести новую информацию относительно общей системы ИИ в термины интерпретированной системы ИИ. Таким образом, исследователь можем получить о ней новые сведения Использование ФРИЗ требует, следовательно, введения интерфейса между вовлеченными в исследования дисциплинами. Такой интерфейс состоит из двух альтернативных процессов — *абстрагирования* и *интерпретации.* В научных исследованиях использование этих процессов носит непрерывный и постоянный характер. Это важнейшее свойство науки хорошо выражено Дж. Сп. Брауном:
Наука — это непрерывный живой процесс, ее содержание — это деятельность, а не данные. Наука отличается от просто данных, как преподаватель отличается от библиотеки . Научное знание, словно отрицательная энтропия, постоянно стремится к уменьшению. Мешает же ей превратиться в короткий эпизод только стремление бесконечно повторять эксперименты и проверять результаты... Наука это игра в значения: один игрок старается свести значительное к незначительному, задавая все новые вопросы, а другой стремится противопоставить этому натиску все новые эксперименты. Ученый, словно шахматист-энтузиаст, часто сам играет за обоих противников. Повторения научных результатов преследуют две цели во-первых, они препятствуют возникновению новых вопросов, стремящихся приуменьшить значимость результатов, во вторых, каждое успешное повторение увеличивает значимость, которую такой вопрос мог сократить. Таким образом, происходит состязание между вопросами и результатами.»
Схема на рис. 2 подходит не только для науки. Она с успехом может быть применена в технике, медицине или управлении. Хотя задачи в этих областях (т. е. проектирование систем, их тестирование, диагностика, принятие решений и т. д.) отличаются от задач, возникающих при научном исследовании, роль ФРИЗ в оказании

помощи пользователю при решении интеллектуальных подзадач по существу остается той же.
Решение интеллектуальных задач, как оно представляется в ФРИЗ, возможно только в сочетании с традиционными методами науки или других областей, где общие задачи возникают в специфических контекстах. ФРИЗ, для того чтобы оказаться практически полезным, должен охватывать как можно более широкий класс интеллектуальных задач, в особенности интеллектуальные задачи, являющиеся общими для многих дисциплин. Следовательно, концептуальная схема ФРИЗ должна быть получена абстрагированием и организацией интеллектуальных понятий и задач из как можно большего числа дисциплин и приложением этой схемы, когда это нужно, к новым понятиям и задачам с целью образования единого целого. Структура ФРИЗ, как она описана в этой работе, и в самом деле создавалась именно таким образом.
Настоящую версию ФРИЗ, так же как и любую из его будущих версий, не следует рассматривать как окончательную. Если принята определенная структура, то всегда имеется вероятность, что в процессе использования ФРИЗ мы рано или поздно обнаружим новые интеллектуальные понятия и задачи, неукладывающиеся в эту структуру. И хотя некоторые из них окажутся слишком специализированными, другие могут иметь настолько широкое применение, что будет оправданным их включение в эту структуру и разработка соответствующих методов. Таким образом, ФРИЗ развивается во взаимодействии с пользователями, а сфера его применения в процессе развития непрерывно расширяется. С этой точки зрения ФРИЗ можно рассматривать и как действующую программу исследований.
Итак, можно заключить, что интеллектуальные задачи — это содержательные подзадачи общих задач, возникающих в традиционных дисциплинах науки и других областях человеческой деятельности, что эти подзадачи могут быть описаны операционально, и что методология их решения представляет большие возможности как для задач традиционных дисциплин, так и для междисциплинарных задач.

1.3. Иерархия систем ИИ

Напомним, что эпистемология или теория познания, раздел философии, в котором изучаются природа и сфера распространения знания, его предпосылки и основы, а также критерии истинности знания.
Иерархия эпистемологических уровней систем ИИ образует каркас таксономии систем ИИ в ФРИЗ. Представляется, что в той или иной

форме подобная иерархия совершенно необходима при создании любого пакета методов решения интеллектуальных задач. Хотя отдельные эпистемологические уровни систем ИИ подробно описаны в последующих разделах настоящей работы, в этом разделе дается упрощенное предварительное описание свойств этой иерархии в целом.
Данная иерархия опирается на несколько элементарных понятий: *исследователь* (наблюдатель) и его среда, исследуемый (наблюдаемый) *объект* и его среда и *взаимодействие* между исследователем и объектом.
Самый нижний уровень в этой иерархии, обозначаемый как уровень 0, это *система, различаемая исследователем как система.* То есть, исследователь выбирает способ, каким он хочет взаимодействовать с исследуемым объектом. В большинстве случаев этот выбор не вполне произволен. По крайней мере частично он определяется целью исследования, условиями исследования (доступностью измерительных инструментов, финансовыми возможностями, временными рамками, юридическими ограничениями и т. д.), а также имеющимися знаниями, относящимися к данному исследованию. В приведенной ниже выдержке из работы В. Гейнса обсуждается такое очень общее понимание термина «система»:
«Определение. Система — это то, что различается как система. На первый взгляд это никакое не утверждение. Системой является все, что мы хотим рассматривать как систему. Можно ли что-нибудь сказать по этому поводу? Имеется ли здесь какое-то основание для науки о системах? Я отвечу на оба эти вопроса утвердительно и покажу, что это определение полно смысла и имеет богатую интерпретацию. Позвольте мне прежде всего ответить на одно очевидное возражение и обернуть его в свою пользу. Можно спросить: «Что специфически системного в этом определении?» «Нельзя ли точно так же применить его ко всем другим объектам, которые я захочу определить», т. е кролик — это то, что различается как кролик. «Но, — отвечу я, — мое определение адекватно определяет систему, в то время как Ваше определяет кролика неадекватно» В этом суть теории систем: определение того, что некая сущность является системой, является необходимым и достаточным критерием того, что она является системой, и это верно только для систем. В то же время различение, что некая сущность является чем-то еще, необходимо для того, чтобы эта сущность была этим чем-то, но недостаточно.
Выражаясь образно, можно сказать, что **понятие системы стоит на самом верху иерархии понятий**. Это место выглядит очень важным Может быть, так оно и есть. Но когда мы понимаем, что это высокое

место достигнуто за счет довольно негативного достоинства отсутствия отличительных свойств, то такая характеристика оказывается не такой уж впечатляющей. Я полагаю, что это определение системы как уникального понятия сделано для того, чтобы объяснить многие достоинства и недостатки теории систем. Сила этого понятия в его абсолютной общности, и мы явно указываем на это полное отсутствие качественных характеристик в термине «общая теория систем» вместо того, чтобы затемнять суть, привести какой-нибудь почтенный прикрывающий термин вроде математической теории систем. Слабость и в то же время главное достоинство этого понятия в том, что его никак нельзя дополнительно охарактеризовать. Слабость потому, что мы не можем оценить значимость дополнительных характеристик изучаемого предмета Достоинство же в том, что эти дополнительные характеристики сами по себе для обсуждения не нужны и только затемняют суть дела, так как принимают во внимание крайности суждений специалистов.»
Способ взаимодействия с самим объектом можно описать не-сколькими альтернативными способами.
В структуре ФРИЗ (рис. 3) система эпистемологического уровня 0 **определена через множество переменных, множество потенциальных состояний (значений), выделяемых для каждой переменной, и некий операционный способ описания смысла этих состояний в терминах проявлений соответствующих атрибутов данного объекта.**

ВЗАИМОДЕИСТВИЕ С ВНЕШНИМ МИРОМ ПРОИСХОДИТ ЧЕРЕЗ ИСХОДНУЮ СИСТЕМУ ИИ ДЛИ ПОЛУЧЕНИЯ СИСТЕМЫ ДАННЫХ,КОТОРАЯ МОДЕЛИРУЕТСЯ СИСТЕМАМИ ИИ БОЛЕЕ ВЫСОКОГО УРОВНЕЙ

Рис. 3. Иерархия эпистемологических уровней систем ИИ

Для определенных на этом уровне систем ИИ используется термин ***исходная система ИИ***, указывающий на то, что **подобная система является, по крайней мере потенциально, источником эмпирических данных.**
В литературе по теории систем для этих систем используются также названия «примитивная система» и «система без данных», предполагающие, что система этого уровня **представляет простейшую стадию процесса исследования систем, не использующую данные о доступных переменных.**

Множество переменных обычно подразделяется на **два подмножества**, называемые ***основными переменными* и *параметрами***.
Совокупность состояний всех параметрических переменных образует *параметрическое множество,* при котором наблюдается изменение в состояниях отдельных основных переменных. Чаще всего в качестве параметров выступают время, пространство и различные совокупности объектов одного типа (социальные группы, группы стран, продукция одного типа и т. д.).
Исходные системы ИИ полезно классифицировать по различным критериям, по которым имеются методически существенные отличия в конкретных свойствах множеств переменных или множеств состояний. Согласно одному из таких критериев основные переменные могут быть разделены на *входные* и *выходные переменные*. При таком разделении состояния входных переменных рассматриваются как условия, влияющие на выходные переменные. **Входные переменные не являются предметом исследования**; считается, что они определяются неким фактором, не входящим в рассматриваемую систему. **Этот фактор называют *средой* системы ИИ, в которую часто включают и исследователя.** Важно, что понятие входных переменных не противоречит понятию независимых переменных.
Системы ИИ, в которых переменные разделены на входные и выходные, будем называть ***направленными****;* системы ИИ, в которых такое разделение не задано, будем называть ***нейтральными.***
Выделяют также ряд дополнительных отличий множеств состояний, связанных с введенными переменными (основными и параметрическими), что позволяет проводить более глубокую методологическую классификацию исходных систем ИИ. Это, например, отличия между четкими и нечеткими переменными, между дискретными и непрерывными переменными, между переменными с разными шкалами значений.
На других более высоких эпистемологических уровнях системы ИИ **отличаются друг от друга уровнем знаний относительно переменных соответствующей исходной системы.** В системах ИИ более высокого уровня используются все знания соответствующих систем ИИ более низких уровней и, кроме того, содержатся дополнительные знания, недоступные низшим уровням. Таким образом, **исходная система ИИ содержится во всех системах ИИ более высоких уровней**.
После того как исходная система ИИ дополнена данными, т. е. **действительными состояниями основных переменных при определенном наборе параметров,** мы рассматриваем новую систему ИИ (исходную систему с данными) как определенную на эпистемологи-

ческом уровне 1. Системы ИИ этого уровня называются ***системами данных***. В зависимости от задачи данные могут быть получены из наблюдений или с помощью измерений (как в задаче моделирования систем ИИ) или определены как желательные состояния (в задаче проектирования систем ИИ).
Более высокие эпистемологические уровни содержат знания о некоторых *инвариантных параметрам характеристиках отношений рассматриваемых переменных,* посредством которых можно генерировать данные при соответствующих начальных или граничных условиях. Генерируемые данные могут быть точными (детерминированными) или приблизительными в каком-то определенном смысле (стохастическими, нечеткими).
На уровне 2 инвариантность параметров представлена одной обобщенной характеристикой, задающей ограничение на множестве основных переменных при данном множестве параметров. В множество основных переменных входят переменные, определяемые соответствующей исходной системой ИИ и, возможно, некоторые дополнительные. **Каждая дополнительная переменная определяется конкретным *правилом преобразования* на множестве параметров,** применимом или к основной переменной исходной системы ИИ, или к гипотетической (ненаблюдаемой) переменной, введенной пользователем (составителем модели, проектировщиком). Эта переменная обычно называется *внутренней.* **Правило преобразования представляет собой взаимно однозначную функцию, присваивающую каждому элементу множества параметров другой (единственный) элемент того же множества.**
Поскольку задачей параметрически инвариантного ограничения является описание процесса, при котором состояния основных переменных могут порождаться по множеству параметров при любых начальных или граничных условиях, системы ИИ уровня 2 будем называть *порождающими системами ИИ (generative system).*
На эпистемологическом уровне 3 системы ИИ, определенные как порождающие системы ИИ (или иногда системы более низкого уровня), называются *подсистемами ИИ* общей системы ИИ. Эти подсистемы ИИ **могут соединяться в том смысле, что они могут иметь некоторые общие переменные или взаимодействовать как-то иначе.** Системы ИИ этого уровня будем называть *структурированными системами ИИ (structure system).*
На эпистемологическом уровне 4 системы ИИ состоят из совокупности систем ИИ, определенных на более низком уровне, и некоторой инвариантной параметрам *метахарактеристики* (**правила, отношения, процедуры**), описывающей изменения в системах ИИ более низкого

уровня. Требуется, чтобы системы ИИ более низкого уровня имели одну и ту же исходную систему ИИ и были определены на уровне 1, 2 или 3. Определенные таким образом системы ИИ будем называть *метасистемами ИИ.* На уровне 5 допускается, что метахарактеристика может изменять множество параметров согласно инвариантной параметрам характеристике более высокого уровня или мета-метахарактеристике. Такие системы ИИ будем называть *мета-метасистемами ИИ* или *метасистемами второго порядка ИИ.* Аналогично определяются метасистемы ИИ более высоких порядков.

1.4. Роль математики и компьютеров в формировании СИИ

Как уже говорилось в разд. 1.2, понятие решения интеллектуальных задач естественным образом возникло из двумерной классификации систем ИИ, изображенной на рис. 1. Они предназначены для работы со свойствами отношений систем ИИ, не зависящими от интерпретаций и контекста. Однако тем же занимается и математика. Какая же тогда между ними разница?

Очень приближенно математику можно разделить на фундаментальную и прикладную. Фундаментальная математика занимается в основном разработкой разных аксиоматических теорий независимо от того, имеют они практическое значение или нет. Фундаментальная математика особенно интересуется доказательством теорем, следующих из постулированных предположений (аксиом), и **не ее цель определять, существует ли некая интерпретация теории, для которой эти предположения истинны**. Эта позиция «искусства для искусства», очень влиятельная в математике еще с XIX в., даже подчеркивается некоторыми математиками, считающими ее принципиальной позицией для этой науки. Однако несмотря на такой подход **многие математические теории в разной степени, но все же имеют отношение к реальности.** Иногда обнаружение подобной связи является счастливой случайностью. Чаще, однако, оно представляется результатом бессознательного процесса в сознании математика (интуиции, озарения) или его осознанной попытки (часто скрытой или, по крайней мере, недекларируемой) **абстрагировать и формализовать некоторые аспекты реальности.**

Здесь необходимо упомянуть о том, что аксиоматическая формализация по своей природе имеет некоторые ограничения. В 1931 г. К. Гёдель показал, **что аксиоматические теории (например, аксиоматическая теория обычной арифметики) таковы, что нельзя доказать их внутреннюю непротиворечивость (т. е. то, что из аксиом нельзя вывести взаимоисключающие теоремы). Точнее, непротиворечивость аксиоматической теории не может быть доказана с помощью ее собственных правил вывода.** Доказательство непротиворечивости, опирающееся на более мощные правила вывода, может существовать, но тогда следует доказать непротиворечивость положений, используемых в новых правилах вывода. Для этого могут потребоваться еще более мощные правила. Повторение этого рассуждения показывает, **что на вопрос о полноте каких-то математических теорий окончательно ответить нельзя.** Что еще важнее, Гёдель также показал, что если некие аксиоматические теории, непротиворечивость которых недоказуема, являются непротиворечивыми, то они будут неполны (т. е. **некие истинные утверждения этих теорий нельзя вывести из их аксиом**). Следовательно, **существуют математические теории, которые или противоречивы, или неполны, и невозможно определить, к какой из двух категорий каждая из них принадлежит.**

Назначение прикладной математики — поиск практических интерпретаций математических теорий и после нахождения таких интерпретаций создание на основе теорий методических средств работы с интерпретированными системами и связанными с ними задачами. В этом смысле прикладная математика ориентирована на разработку методов, базирующихся на определенных математических теориях, и использование их в как можно большем числе конкретных приложений. Эти методы подчиняются фундаментальным ограничениям математических теорий, на которые указал Гёдель. Более того, **любая математическая теория выводится из некоторых определенных предположений (аксиом) и, следовательно, любая методика, опирающаяся на эту теорию, применима только к тем задачам, которые отвечают этим предположениям**. Если задача им не отвечает, а математик-прикладник, владеющий данной методикой, все-таки хочет ее применить, ему нужно переформулировать свою задачу так, чтобы она удовлетворяла этим ограничениям. Однако это означает, что теперь будет решена другая задача. Очень часто изменение задачи явно не констатируется, в результате чего создается впечатление, что была решена исходная задача, хотя на самом деле это не так.

Таким образом**, математики-прикладники предоставляют пользователям (ученым, инженерам и т. д.) набор методических средств, каждое из которых получено из какой-либо математической теории, которая, в свою очередь, опирается на определенный набор предположений.** Чаще всего математические теории разрабатываются в предположениях, представляющих интерес или подходящих с точки зрения математического аппарата. Как следствие порожденные ими методы покрывают лишь отдельные небольшие фрагменты всего спектра интеллектуальных задач. Идея решения интеллектуальных задач является в некотором смысле реакцией на это неудовлетворительное положение.

В противоположность прикладной математике решение интеллектуальных задач предназначено для исследования области интеллектуальных задач как единого целого. Оно, в частности, пытается определить практически ценные подзадачи, возникающие в как можно более широком классе реальных интеллектуальных задач. Эта **направленность решения интеллектуальных задач на полноту и практическую значимость проводимых исследований** отличает ее от математики, ориентированной на исследование методик, базирующихся на подходящих (и часто произвольных) математических свойствах.

Таким образом, *приоритет задач* в решении интеллектуальных задач резко контрастирует с *приоритетом методов* в прикладной математике. Наиболее важным **назначением науки о решении интеллектуальных задач является разработка методов решения интеллектуальных задач в их естественной формулировке, либо не использующих при решении упрощающих предположений, либо, если это невозможно, с упрощениями, позволяющими решить задачу, но в то же время как можно меньше искажающими ее.**

Методологические методы решения задач имеют подчиненное значение и выбираются так, чтобы как можно лучше соответствовать задачам, а не наоборот. **Более того, эти методы по природе своей не обязаны быть чисто математическими, а могут представлять собой сочетание математических, вычислительных, эвристических, экспериментальных и других методов.**

При управлении сложностью процесса решения редко удается обойтись без упрощающих предположений. Однако для любой задачи такие упрощающие предположения могут быть введены самыми разными способами. Всякий такой набор предположений определенным образом сокращает диапазон возможных решений и в то же время снижает сложность процесса решения.

Для конкретной интеллектуальной задачи **множество предположений относительно ее решений называется *методологической парадигмой***. Если задача решается при определенной методологической парадигме, то найденное решение не содержит особенностей, несовместимых с этой парадигмой.

Целесообразно рассматривать парадигму, являющуюся подмножеством предположений другой парадигмы, как обобщение последней. **При заданном множестве всех предположений, рассматриваемых для данного типа задач, отношение «парадигма А является более общей, чем парадигма В» (т. е. А содержит подмножество предположений, содержащихся в В) задает частичное упорядочение для всех содержательных парадигм, относящихся к данному типу задач. Термин «содержательная парадигма» может пониматься строго как характеристика множества предположений, гарантирующего возможность решения всех конкретных задач данного типа.** В то же время стороны его можно понимать и как более слабое требование, чтобы только некоторые частные задачи данного типа решались при данной парадигме.

Самая общая парадигма для задачи любого типа единственна - это парадигма без предположений. Однако обычно существует несколько менее общих, но плодотворных для решения задач данного типа парадигм.

Сейчас наблюдается тенденция к обобщению парадигм, стимулируемая достижениями в развитии вычислительной техники. Любое обобщение парадигмы расширяет множество возможных решений задачи и во многих случаях позволяет получить лучшее решение. Однако одновременно это требует усложнения процедуры решения. **Изучение связей между возможными методологическими парадигмами и классами интеллектуальных задач является предметом *метаметодологии систем ИИ***. Это важная новая область исследований, в которой пока еще мало сделано. Центральным вопросом метаметодологии систем ИИ является разработка таких парадинм, которые для различных классов задач и нынешнего состояния вычислительной техники обеспечивали бы наилучший **компромисс для двух противоречивых критериев — качества решения и сложности процедуры решения**. Основная трудность подобного исследования состоит в том, что **для данной задачи при одной и той же методологической парадигме может быть разработано множество альтернативных процедур решения.**

Другой задачей метаметодологии систем ИИ является определение и описание кластеров интеллектуальных парадигм, хорошо дополняющих друг друга, так что их можно эффективно использовать

параллельно при решении одной и той же задачи. Вместе они могут дать исследователю значительно больше, чем каждая из них в отдельности.

Всякая математическая теория, имеющая смысл с точки зрения схемы решения интеллектуальных задач (такой, как ФРИЗ), является по существу **методологической парадигмой**. Она связана с типом задачи и представляет собой локальную систему, в которой могут разрабатываться методы решения конкретных задач данного типа.

Одна из задач методологии систем — это **компиляция (составление)** математических теорий и определение их места в полном пространстве задач. Другая задача — предложение новых содержательных парадигм, при этом конечной целью является описвние и упорядочение всех возможных парадигм для каждого типа задач. Поскольку выявление новой парадигмы служит толчком для создания новой математической теории, всесторонние исследования в метаметодологии систем ИИ послужат мощным стимулом для фундаментальных математических исследований, имеющих большое практическое значение. Таким образом, математика вносит свой вклад в решение интеллектуальных задач и способствует ее развитию.

Дж. Кемени отметил, что «*новым партнером человека является высокоскоростной компьютер.»*

Наука о системах ИИ сильно зависит от компьютера, **представляющего собой одновременно и ее лабораторию, и важнейшее методологическое средство.** Поэтому не удивительно, что современный системный подход начал формироваться почти сразу после появления в конце 40-х — начале 50-х гг. полностью автоматизированных цифровых вычислительных машин. Все это время наука о системах и компьютерная технология развивались бок о бок и влияли друг на друга.

Прогресс компьютерной технологии совместно с достижениями в области искусственного интеллекта дали новые методологические возможности, помогли прояснить или уточнить формулировку некоторых фундаментальных философских проблем, сделать более конструктивными некоторые умозрительные идеи, а также **сделали возможным реализацию функций человеческого мозга на компьютере**. Однако цель решения интеллектуальных задач — не заменить мозг человека машиной, а симбиотически дополнить его компьютером, снабженным пакетом соответствующих методических средств. Такой подход основан на том, что при столкновении с очень сложными системами мозг проявляет способности, намного превосходящие самые сложные методы, реализованные на самых современных компьютерах. Современное понимание этих

способностей достаточно примитивно и, безусловно, неудовлетворительно. **Несмотря на успехи искусственного интеллекта, а также нейрофизиологии, психологии и других наук есть основания считать, что некоторые способности человеческого мозга никогда не будут поняты до конца.**

Возможно, самыми ценными свойствами человеческого мозга являются интуиция, озарение, способность к глобальному охвату, особенно если они хорошо развиты. Сложные системы, однако, часто обладают свойствами, не поддающимися интуитивному пониманию и глобальной оценке. Эти свойства являются ловушками для ума в том смысле, что подталкивают его к неправильным представлениям. Для обнаружения этих ловушек нужно, как правило, проделать утомительную работу по детальному анализу изучаемой системы. В этом отношении мозг не очень силен и ограничен в возможностях, а детальный анализ —это как раз та область, где компьютер его превосходит. Это свойство компьютера позволяет ему играть важную роль ***гаранта* и *усилителя интуиции***.

Симбиоз человека (ученого, лица, принимающего решение, конструктора и т. п.) и методологически вооруженного компьютера, такого, как ФРИЗ, **позволяет ввести и применить новые подходы к решению различных интеллектуальных задач, существенно более мощные, чем используемые человеком или машиной в отдельности. Сила человека в его знании предмета исследования, понимания и использовании контекста, в котором производится исследование, в интуиции, способности к глобальному охвату, в чувстве правильного решения, аудиовизуальных возможностях, творчестве и тому подобное. Сила компьютера — это его вычислительная мощность, легкость, с которой он производит огромное число операций, значительно превосходящая в этом отношении возможности человека. Правильно использованная вычислительная мощность компьютера существенно увеличивает интеллектуальную силу человека, осуществляя для него необходимый детальный анализ и, как уже отмечалось, помогая избежать многих интуитивно не обнаруживаемых ловушек, связанных со сложностью систем.**

Одной из таких ловушек является обычно принимаемое без доказательств предположение, что свойства систем в целом могут быть восстановлены по знаниям о соответствующих свойствах, связанных с их подсистемами. Например, в междисциплинарных социологических проектах обычно предполагается, что мы понимаем систему в целом, если мы понимаем ее экономическую, правовую, политическую, экологическую и другие рассматриваемые

подсистемы. Подобное предположение, к сожалению, подтверждается очень редко, и, даже если подтверждается, его обоснованность зависит от выбранных подсистем. Нет оснований считать, что «естественные» подсистемы (экономические, политические и т. д.) являются адекватными в том смысле, что они содержат достаточно информации, чтобы можно было достаточно точно реконструировать (понять) систему в целом. Если же предположение о возможности реконструкции всей системы по определенным ее подсистемам не подтвердилось, то всевозможные выводы относительно всей системы, полученные из подсистем, могут оказаться некорректными и вводящими в заблуждение. **Хотя информация о возможностях реконструкции неявно содержится в данных о системе в целом, явное ее определение требует детального анализа этих данных.** Методы проведения такого анализа, называемого **анализом реконструируемости**, описываются в последующих разделах данной работы. **Человек, если не считать весьма небольших систем, с анализом реконструируемости не справляется, в то время как у компьютера есть огромные возможности по проведению такого анализа для систем, имеющих практическое значение. Анализ реконструируемости — это просто один из примеров важной методологической области, практическая значимость которой определяется применением сложной компьютерной технологии**. При решении интеллектуальных задач такие примеры отнюдь не редки, а скорее типичны.

Использование компьютера как гаранта и усилителя интуиции при решении интеллектуальных задач — это одно из двух важнейших его применений в науке о системах ИИ.

Другим является его использование в качестве лаборатории науки о системах ИИ. В этом случае он используется для проведения экспериментов со смоделированными на нем системами ИИ. Можно выделить по крайней мере три цели проведения таких экспериментов.

1. *Традиционное использование моделирования.* Система ИИ, воспроизводящая соответствующие свойства объекта исследования, моделируется на компьютере для порождения сценариев при различных предположениях относительно среды системы ИИ, а также при различных параметрах самой системы ИИ.
2. *Открытие или проверка законов науки о системах ИИ.* Эксперименты такого рода проводятся на компьютере с большим числом разных систем ИИ одного и того же класса. Цель таких экспериментов — открытие полезнчх свойств, описывающих класс исследуемых систем ИИ, или, наоборот, проверка выдвинутых относительно этого класса предположений.

Один из наиболее характерных экспериментов такого рода был проведен Гарднером и Эшби. **Целью эксперимента было определение влияния размера системы (числа переменных) и ее связанности (числа зависимостей между переменными) на вероятность устойчивости в определенном классе систем**. Гарднер и Эшби ограничили свое исследование весьма конкретным классом систем (линейными динамическими системами, описываемыми системой линейных дифференциальных уравнений первого порядка с постоянными коэффициентами). Среди других результатов их исследование привело к открытию **критической связности** и дало следующий статистически достоверный закон для изучаемого класса систем: **если линейная динамическая система (как она описана выше) достаточно велика (состоит из 10 или более переменных) и ее связанность (процент ненулевых недиагональных элементов в матрице, описывающей эту систему) меньше 13% (критическая связанность), тогда данная система почти наверняка устойчива.** Если ее связанность больше 13%, она почти наверняка неустойчива; 2%-го отклонения от критической связанности оказывается достаточно для того, чтобы ответ на вопрос об устойчивости из «почти наверняка устойчива» превратился в «почти наверняка неустойчива» (рис. 4).

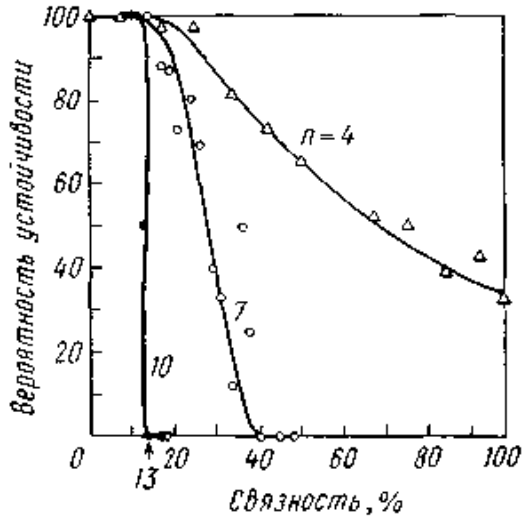


Рис. 4 Зависимость устойчивости системы от связности (экспериментальные результаты получены Гарднером и Эшби)

3. *Экспериментальные характеристики методов.* Постановка задачи, решение которой известно, моделируется на компьютере. Для решения этой задачи используется исследуемый метод (обычно это метод решения задач, имеющих недедуктивную природу). Полученный результат сравнивается с известным решением. Эта процедура повторяется достаточное число раз для различных постановок задач исследуемого класса, что позволяет определить полученные характеристики введенного метода. Такие характеристики очень важны для пользователей, применяющих разные методы, так как дают возможность правильно интерпретировать полученные результаты и принять соответствующие решения. Подобные эксперименты для одной из задач анализа реконструируемости описаны в последующих разделах.

Таким образом, связи между решением интеллектуальных задач и вычислительной техникой весьма важны. Можно с уверенностью сказать, что решение интеллектуальных задач, как оно понимается и описывается в данной работе, не имело бы в действительности никакой практической ценности, если бы не использование мощной вычислительной техники.

1.5. Архитектура решения интеллектуальных задач

Хотя совершенно ясно, что решение интеллектуальных задач бесконечно разнообразно, становится все понятнее, что **это бесконечное разнообразие может быть достаточно полно представлено конечным множеством интеллектуальных задач.** Это множество взаимосвязанных интеллектуальных задач получается в результате применения нескольких фундаментальных принципов, согласно которым **классифицируются и организуются системы ИИ**.

Решение интеллектуальных задач может изучаться и развиваться на разных уровнях обобщения и детализации. На самом высоком уровне общности основное внимание уделяется разработке практически важных принципов организации систем ИИ и выработке общего взгляда на процессы решения интеллектуальных задач. **Такие общие аспекты решения интеллектуальных задач будем называть *архитектурой решения интеллектуальных задач.***

Архитектура — это одна из самых древних профессий. В самом деле, уже в Древней Греции 2000 лет назад она была прекрасно развита и считалась самостоятельной профессией. Одним из лучших описаний архитектуры остается знаменитая книга «Об архитектуре» древнеримского архитектора и инженера I в. до нашей эры Марка

Витрувия Поллиона. Приведенный ниже отрывок из этой книги хорошо выражает его взгляды:

«Архитектор должен обладать знанием многих областей науки и разного рода ученостью, так как его суждением проверяются все создания других искусств. Это знание — дитя практики и теории.»

В своей долгой истории архитектура ассоциировалась почти исключительно с проектированием зданий. Только совсем недавно было показано, что некоторые общие принципы архитектуры относятся не только к проектированию зданий, но равным образом существенны и в других областях проектирования.

Необходимость использования приемов архитектуры при проектировании вычислительных машин была осознана в начале 60-х гг XXв. Сам термин «компьютерная архитектура» был, по всей видимости, введен Ф. Бруксом в связи с разработкой компьютера STRETCH фирмы IBM. В статье «Архитектурная философия» он вводит следующее определение:

«Компьютерная архитектура, как и всякая другая архитектура, это искусство **определения требований пользователя к структуре**, а затем проектирование таким образом, чтобы она как можно полнее соответствовала этим требованиям при заданных экономических и технологических ограничениях».

Этот подход к компьютерной архитектуре, возникший из опыта проектирования ЭВМ STRETCH был последовательно выдержан при создании IBM System/360 — семейства одинаково спроектированных взаимно совместимых ЭВМ, отражавших как требования пользователей, так и экономические и технологические возможности. Результат этого подхода к проектированию компьютеров излагается в статье «Архитектура IBM System/360», написанной архитекторами этой серии машин Амдалем, Блаау и Бруксом.

Вскоре после этого компьютерная архитектура стала считаться важной частью компьютерного проекта.

Признание компьютерной архитектуры было первым шагом в расширении понятия архитектуры за традиционные рамки ее понимания как архитектуры зданий. Мы попытаемся сделать еще один шаг, распространяющий понятие архитектуры на все системы ИИ. Подобное обобщение предлагается не впервые. Так, например, еще в 1962 г. Г. Саймон рассматривал подобную идею, называя ее «архитектурой сложности». Г. Земанек постоянно доказывал важность обобщения понятия архитектуры на все системы и ввел такие названия, как обобщенная архитектура и абстрактная архитектура. Подобного же мнения придерживается Дж. Таунер в своей книге «Архитектура знания». Однако ни одно из этих предложений не относится к решению

интеллектуальных задач. В этом смысле данная работа является единственной в своем роде.

Настоящая тенденция к расширению сферы архитектуры за рамки ее традиционной области — строительства зданий (и, возможно, других родственных сооружений вроде мостов и кораблей) не противоречит общепринятому и приведенному в словарях определению архитектуры. Например, в *Оксфордском словаре английского языка* архитектура определяется как «искусство или наука построения или конструирования любых сооружений для нужд людей», как «действие или процесс построения» или как «здание или сооружение». Таким образом, можно видеть, что, во-первых, термин «архитектура» имеет три разных толкования: **определенная дисциплина, определенный тип человеческой деятельности и определенный результат этой деятельности**, и, во-вторых, во всех трех толкованиях архитектура понимается не только как архитектура зданий.

Из определения, приведенного в толковом словаре, можно легко выделить две ключевые характеристики архитектуры: 1) архитектура связана с проектированием, конструированием, построением и т. п., т. е. **с процессами создания искусственных объектов**; 2) она имеет дело с **использованием созданных объектов человеком**.

Рассмотрим характеристики подробнее.

Хотя архитектура и ориентирована на проектирование, конструирование и построение, полностью все эти виды деятельности она не охватывает. Таким образом, архитектор является проектировщиком, работа которого завершается другими людьми. Его роль состоит в наблюдении за проектом на глобальном уровне; он занимается аспектами проекта, **включающими любые интерфейсы с пользователем**. В хорошем архитектурном проекте другие вопросы, не интересные с точки зрения пользователя, должны оставаться открытыми. Однако архитектор обязан учитывать возможности технологии и экономические ограничения, чтобы быть уверенным, что его архитектурный проект может быть реализован без значительных затруднений.

Архитектурное проектирование предназначено для подготовки общих спецификаций, определяемых нуждами и пожеланиями пользователя и используемых на этапах проектирования и конструирования. Таким образом, первая задача архитектора при создании проекта — это определение реальных потребностей и пожеланий пользователя. Эта задача сформулирована французским архитектором Ж- Годе: «Архитектор прежде всего должен определять содержание, из которого он может затем извлечь форму».

В хорошем архитектурном проекте многие детали будущей конструкции остаются непроработанными, оставляя достаточно свободы для дальнейшего проектирования и конструирования, но в нем имеются все спецификации существенных для пользователя характеристик. В этом смысле он представляет собой общее описание будущей конструкции, сделанное с определенного расстояния. Хорошего архитектора более чем что-либо другое отличает именно эта способность выбирать нужное отдаление, с которого хорошо различимы все существенные для пользователя свойства, но в то же время не видны остальные.

Архитектурное проектирование, как пишет Г. Земанека, — это проектирование сверху вниз, определяющее каждую деталь как функцию целого. С этой точки зрения архитектурное проектирование дополняет формальное определение: определить детали по общей структуре можно только в том случае, если метод описания позволяет совершенно свободно опускать детали и говорить о желаемых свойствах системы в целом до начала любой работы по объединению частей сооружения.

По Г. Блаау**,** одному из архитекторов IBM System/360, в проекте любой системы можно выделить три характерных уровня: **архитектура, исполнение и реализация**. *Архитектура* системы — это функциональное проявление системы с точки зрения пользователя; под *исполнением* понимается логическое описание внутренней структуры, делающей возможным осуществление этих функций; *реализация* — это физическое воплощение исполнения.

В архитектуре выдерживаются некоторые общепринятые принципы. Они четко изложены Блаау. Приведем перечень этих принципов:

1. *Согласованность.* Хорошая архитектура согласована, т. е. частично знание системы позволяет предсказать остальное.
2. *Ортогональность.* Этот принцип требует, чтобы функции были независимы друг от друга и специфицированы по отдельности.
3. *Соответственность.* Согласно этому принципу следует включать в архитектуру только те функции, которые соответствуют существенным требованиям к системе, другими словами, в хорошей архитектуре нет ненужных функций.
4. *Экономность.* Никакая функция в описании архитектуры не должна в том или ином виде дублировать другую.
5. *Прозрачность.* Функции, найденные в процессе исполнения, должны быть известны пользователю.
6. *Общность.* Если функция должна быть введена, ее следует вводить в таком виде, чтобы она отвечала как можно большему числу назначений.

7 *Открытость*. Пользователю должно быть позволено использовать функцию иначе, чем это предполагалось при проектировании.
8. *Полнота.* Введенные функции должны с учетом экономических и технологических ограничений как можно полнее соответствовать требованиям и пожеланиям пользователя.
Архитектура решения интеллектуальных задач должна соответствовать основным целям и принципам любой архитектуры, как это описано в данном разделе. Прежде всего, эта архитектура должна быть ориентирована на пользователя, т. е. должна охватывать все типы интеллектуальных задач, с которыми будет работать предполагаемый пользователь. Хотя представление об этом пользователе должно быть как можно более широким, в первую очередь она ориентирована на ученых, инженеров и других специалистов. Различные типы интеллектуальных задач определены, исходя преимущественно из тех интеллектуальных задач, которые возникают в разных областях науки и техники, а также в таких областях, как медицина, управление или право.
На архитектурном уровне решение интеллектуальных задач должно рассматриваться и описываться с определенной дистанции, позволяющей, не отвлекаясь на детали, распознать общую структуру. Реальный архитектурный проект для решения интеллектуальных задач, вроде упомянутого в разд. 1.2 ФРИЗ, должен разрабатываться сверху вниз и отражать различные принципы хорошей архитектуры.

2. Исходные системы ИИ и системы данных

2.1. О понятиях объект и система объектов ИИ

Человек разумный как представитель животного мира отличается тем, что обладает свойством отличать себя в окружающем мире посредством мышления (умственнного творчества).
Разум – средство (инструмент) анализа ситуции (проблемы) в окужающем мире.
Мышление – средство виртуального синтеза процесса решения задач.

Мышление (творчество) – неалгоритмизуемо. Результат мышления (творчества) - алгоритм решения интеллектуальной задачи.
Человек обладает средством мышления – мозгом. Мозг представляет собой источник интеллекта. Посредством интеллекта человек формирует (представляет, разрабатывает) в своем сознании эвристики и алгоритмы познания окружающего мира и алгоритмы созидания в нем. Человет мыслит образами, представляемые в его сознании в виде понятий и сцен. Результат объективного мышления (мысли) есть истина, которая представляет собой свершившееся событие. В свою очередь, прогноз - это событие в будущем.
Человек познает окружающий мир посредством знаний. Знания – это совокупность предметов находящихся в причинно-следственных отношениях.
Знания об окружающем мире есть результат человеческих ощущений.
Знания представляются фреймами и сценами (интеллектуальными модулями)
Человек обладает способностью запоминать, обрабатывать и использовать информацию, полученную из окружающего мира, и таким образом усиливать свою способность к восприятию. Эти **способности человека к восприятию** будем называть его **интеллектом**. Интеллект позволяет анализировоть состояние окружающего мира, способствует человеку (посредством мышления) принимать решения и реализовывать их. **Решение интеллектуальных задач** – это способность целенаправленно перерабатывть имеющуюся информацию (знания), путем (посредством) обучения на опыте
Люди взаимодействуют с различными объектами из окружающего мира, причем это взаимодействие для любого объекта обычно ограничивается рядом отличительных признаков (представительными свойствами). Хотя подобное взаимодействие может становиться все разнообразнее по мере того, как все больше накапливается знаний об объекте, оно всегда существенно ограничивается пределами восприятия человека, его способностью к действию и другими возможностями. В таких действиях человека, как научное исследование, инженерное проектирование, медицинская диагностика, расследование преступления или художественное творчество, взаимодействие с объектами бывает более заметно выражено и часто выходит за пределы обычных человеческих возможностей.

Людей, активно работающих в любой из традиционных отраслей науки, техники или в других областях (медицине, праве и т. д.), интересует в их профессиональной деятельности определенные типы объектов. Так, например, экологи занимаются такими объектами, как озера, реки и леса; музыковеды — музыкальными сочинениями и композиторами; психологи изучают отдельных людей и малые социальные группы; инженеры занимаются всевозможными объектами, созданными человеком, например электростанциями, автомобилями, самолетами, компьютерами и тому подобным; врачи имеют дело с пациентами, а ветеринары — с больными животными; криминалисты расследуют преступления, а биологи изучают различные явления в живой природе.

Под объектом будем понимать часть окружающего мира, выделяемую как единое целое в течение длительного (ощутимого) отрезка времени. Согласно этому определению объекты могут быть как **материальными**, так и **абстрактными**. ***Материальные объекты*** будем разделять на ***естественные*** (такие, как кусок скалы, клетка организма, солнце или группа животных) и ***искусственные (созданные человеком***) (такие, как аэропорт, вычислительный центр, город Киев или больница). Будем различать также и ***абстрактные объекты*** (такие, как музыкальное сочинение, стихотворение или конституция Украины), которые обычно создаются человеком. **Объекты обладают практически бесконечным числом признаков и свойств**, **любые из которых можно изучать** и, как следствие, никакой **объект невозможно изучить полностью**. Это означает, что для **изучения объекта**, **необходимо отобрать ограниченное (достаточное) число характеристик, наилучшим образом описывающих данный объект как явление.** После того как такой отбор сделан, необходимо определить **процедуру измерения (наблюдения) каждого отобранного признака или свойства**, которые, в свою очередь, задают **абстрактную переменную**, **представляющую образ (отображение предмета в сознании человека) соответствующего признака или свойства.**

Мы говорим, что в интересующем нас объекте **система задается набором соответствующих признаков и свойств объекта и назначением каждому из них определенной переменной (с помощью процедуры измерения).** Таким образом, **система всегда рассматривается не как реальная вещь, а как абстрагирование или отображение некоторых признаков и свойств объекта. *Это важное различие между понятием объекта и понятием системы*** .

Рассмотрим вопрос, как следует определить (распознать) систему. Например, можно указать на автомобиль и сказать, что «система — это

такая-то вещь, находящаяся там-то». Однако такой подход имеет существенный недостаток: **всякий материальный объект включает бесконечное число переменных и, следовательно, бесконечное число возможных систем.** Автомобиль, например, характеризуется не только длиной и положением, но и массой, температурой, электропроводностью, кристаллической структурой, химическими примесями, радиоактивностью, скоростью, коэффициентом отражения, пределом прочности на растяжение, бактериальной зараженностью, поглощением света, упругостью, формой, весом и т. д. и т. п. **параметрами**. Нереально было бы исследовать все эти **параметры**. Нужно **выделить и изучить параметры, относящиеся к проблеме, которая определена.** Таким образом, можно сказать, что **система — это не предмет, а список переменных**.

Термин «**переменная**» используется нами для обозначения **абстрактного образа признака или свойства**. Поэтому, чтобы можно было определить их точно, нужно сначала разобраться, **что же такое признак и свойство.**

Заметим, что с каждым признаком и свойством связано множество их **появлений (проявлений)**. Так, например, если свойством является относительная влажность в определенном месте Земли, то **множество проявлений — это всевозможные значения относительной влажности** (определяемые некоторым определенным способом), в диапазоне 0—100%; **если свойством является количество гормона эстрогена в 1 см^3 женской крови, то определенное количество этого гормона является проявлением этого свойства**; если свойство — это цвет светофора на перекрестке, регулирующего движение транспорта в определенном направлении, то проявлениями этого свойства обычно бывают красный, желтый и зеленый цвета.

При единичном наблюдении признак или свойство имеет одно конкретное проявление. Для **определения возможных изменений его проявлений требуется множество наблюдений этого признака или свойства.** Для этого, однако, необходимо, чтобы отдельные наблюдения признака или свойства, осуществляемые с помощью **одной и той же процедуры наблюдения**, каким-то образом отличались одно от другого. Любой **существенный признак или свойство, *используемые* для определения различий в наблюдениях одного и того же признака или свойства, будем называть *базой (backdrop).*** Выбор этого (такого) термина объясняется тем, что **всякая различающая особенность, какой бы она ни была, является своего рода базой, с которой наблюдается признак или свойство.**

Типичной базой, пригодной для практического использования для любого признака или свойства, является ***время.*** **В этом случае разные**

наблюдения одного признака или свойства отличаются друг от друга тем, что они сделаны в разные моменты (промежутки) времени. Например, относительную влажность в определенном месте можно измерять в разные моменты времени, скажем каждый час. Аналогично множество замеров объема эстрогена в 1 $см^3$ крови одного пациента можно получить, делая анализ в разное время, например в 8^{00} и 20^{00} ежедневно в течение всего периода исследования.

В некоторых случаях разные наблюдения одного и того же **признака по времени неразличимы** (т. е. либо сделаны одновременно, либо время вообще не имеет значения), зато **отличаются положения в *пространстве,*** в которых сделаны наблюдения. Например, различные свойства, характеризующие качество акустики, можно наблюдать в один и тот же момент времени в разных точках концертного зала. В некоторых дисциплинах **пространство как база** играет особенно важную роль, например в кристаллографии, строительстве, оптике, изобразительном искусстве и анатомии. Понятие пространства не будем ограничивать одно-, двух- или трехмерным евклидовым пространством. Например, пространство, образуемое точками сферы с римановой геометрией на ней, может оказаться требуемой базой для некоторых свойств (таких как, геологических, климатических и географических характеристик Земли). **Последовательность слов в тексте можно рассматривать как одномерное (абстрактное) пространство**; такие свойства, как позиция и функция слова в любом предложении, число букв в отдельном слове и т. д. могут наблюдаться в каждой точке (в любом слове) этого текста.

Время и пространство не единственно возможные базы. Множественные наблюдения одного и того же свойства могут различаться друг от друга по индивидам некой *группы (population),* на которой определено данное свойство. Это может быть социальная группа, набор производимых товаров определенного типа, множество слов в каком-то стихотворении или рассказе, множество стран, популяция лабораторных мышей и т. д. Например, при любой переписи населения такие **признаки, как возраст, пол, доход, занятость, научные знания и т. д**., наблюдаются одновременно у всех людей, составляющих население страны.

Базы трех основных типов — время, пространство, группа — можно комбинировать. Хотя возможны любые комбинации, наиболее важными комбинациями являются: время — пространство и время — группа. Приведем некоторые примеры.

Время — пространство. Примером данной комбинации является кинофильм, особенно если он используется в исследованиях (изучение микробиологических процессов, роста детей или дорожных ситуаций

на загруженных перекрестках и т. п.); большая часть признаков в метеорологии (относительная влажность, температура, скорость и направление ветра, типы облаков и т. д.) наблюдаются во многих местах Земли и в разное время; другим примером такой комбинации баз является последовательность позиций на шахматной доске.

Время — группа. Свойства, характеризующие положение в экономике, политике и обществе разных стран, наблюдаются различными организациями, например ООН; ежедневно проводятся наблюдения за популяцией лабораторных мышей, за их физиологическим состоянием и поведением, а также за признаками, находящимися под контролем исследователя (стимуляция, медикаментозное и хирургическое лечение); за такими характеристиками, как число опубликованных книг и журналов определенной категории, средняя цена книги и среднее число подписчиков журнала, суммарный доход и за множеством других ежегодно следит группа издателей.

Помимо использования времени, пространства и групп в качестве баз, они могут выступать и как свойства. Например, при ежедневном наблюдении времени восхода и захода Солнца в разных местах Земли, **свойством является время, а его базами — время и пространство**; рекордные значения времени в каком-либо спорте, например, в плавании на 400 м вольным стилем, являются наблюдениями свойства, различающегося во времени (например, датами установления рекордов); **время является базой для наблюдения за таким свойством, как местоположение судна**; можно наблюдать за значениями времени, необходимого для решения задачи пакетом программ, работающих на одном и том же компьютере.

Приведенные примеры показывают, что **выбор требуемых (необходимых) баз довольно гибок, но не произволен**. Ограничения при этом выборе довольно однозначно выражены в описанных ниже требованиях, которым должны удовлетворять правильно выбранные базы; эти ограничения могут служить руководящими принципами в процессе определения системы на интересующем нас объекте.

Во-первых, **базы должны быть применимы ко всем признакам и свойствам системы*,* для которой они определены**. Например, ни время, ни пространство не могут быть использованы для отличия испытаний продукции определенного типа (не имеет значения, когда и где были проведены эти испытания); ни пространство, ни группа не применимы в качестве баз для свойств, характеризующих музыкальное сочинение; ни время, ни группу нельзя использовать при описании мозаики.

Во-вторых, ***базы системы должны отвечать назначению, для которого определяется данная система.*** Так, например, **при**

наблюдении за выздоровлением пациента после хирургической операции наблюдают за соответствующими признаками. Единственной подходящей для этого базой является время. Но если целью является создание медицинской базы данных, то те же самые характеристики будут различаться не только по времени, но и по фамилиям и другим характеристикам пациентов, находящихся на одной стадии выздоровления, данные о которых и будут заноситься в базу данных. Таким образом, в данном случае применимыми в качестве баз оказываются и время и группа.

В-третьих, ***наблюдения всех признаков или свойств системы должны однозначно определяться базами системы,*** т. е. **каждый элемент базового множества** (значение определенного момента времени, точка пространства, элемент группы или соответствующая комбинация элементов) **определяет одно и только одно проявление любого из признаков или свойств**. Например, при **исследовании признаков слов текста** (позиция и функция слова в предложении, число букв в слове и т. д.) **базой является группа слов, входящих в этот текст**. Очевидно, что такая база применима к этим признакам (свойствам) и соответствует цели исследования. Однако она не удовлетворяет требованию однозначного различения наблюдений. В самом деле, одно и то же слово может находиться в одной и той же позиции и иметь ту же функцию в нескольких предложениях в данном тексте, но в нем будет одно и то же число букв. Для того чтобы отличить любое наблюдение, нам нужно обратиться в данном случае к **одномерному абстрактному пространству, точкой которого является положение слова в тексте.**

Если смысл признаков или свойств и баз и их взаимоотношения поняты правильно, то ясно, как **формально определить систему**, заданную на объекте, иначе говоря, **систему объекта. Она представляет собой множество свойств, с каждым из которых связано множество его проявлений, и множество баз, с каждой из которых связано множество ее элементов**

Формально система объекта — это

$$\mathbf{O}=(\{(a_i, A_i)|i\in N_n\}, \{b_j, B_j|j\in N_m\}), \quad (1)$$

где $N_n=\{1, 2, ..., n\}$, а $N_m=\{1, 2, .. , m\}$ (буквой N с положительным целым индексом всегда будем обозначать множество значений целых положительных чисел от 1 до значения этого индекса);

через a_i и A_i обозначены соответственно **свойство и множество его проявлений**; b_j и B_j — **база и множество ее элементов**, а **O — система объекта**.

Для некоторых **признаков (свойств) и баз** множества A_i и B_j из уравнения (1) определяются достаточно хорошо. Например, если в качестве базы используется группа (социальная группа или группа больных, продукция определенного типа или группа стран), множество B_j обычно четко определено. Аналогично хорошо определено множество проявлений A_i для таких **признаков** a_i, как ежемесячный доход человека, цвета светофора, число букв в слове или больных в палате. **В науке, однако, во многих случаях эти множества неизвестны и могут быть определены только с помощью общенаучных рассуждений**. Тем не менее независимо от обстоятельств **их можно связать с хорошо определенными множествами с помощью конкретных процедур наблюдения или измерения**. Эти хорошо **определенные множества представляют собой образы множеств** A_i и B_j, **в терминах этих множеств и формулируются знания о признаках и свойствах.**

Само существование признаков, свойств, баз и, соответственно, множеств A_i и B_j, как атрибутов объектов является предметом научных дискусий. На этот счет имеются разные точки зрения — от наивного реализма, безоговорочно принимающего их существование, до крайних форм операционализма (или инструментализма), отрицающего их существование и утверждающего, что **содержание любого научного понятия целиком и полностью определяется спецификацией процедуры измерения**

Данная философская проблема не имеет отношения к задачам искусственного интелекта. Однако мы должны понимать, что множества A_i и B_j часто бывают неизвестны, независимо от отношения к проблеме существования. В подобных случаях система объекта смысла не имеет и может приобрести содержание только благодаря **заданию конкретных процедур наблюдения или измерения, с помощью которых создаются образы признаков и свойств.** Таким образом, **система объекта должна рассматриваться как компонента большей системы**; рассматривать систему объекта саму по себе практически бесполезно.

2.1.1. Понтия и правила

Одна из наиболее широких областей соприкосновения между теориями интеллектуальных процессов и другими подходами состоит в изучении понятий, причем этот термин обозначает либо множество элементов, удовлетворяющих некоторому критерию, либо сам этот критерий. Сначала мы рассмотрим пример применения теории интеллектуальных процессов к понятиям и формированию понятий, а затем обсудим некоторые функциональные роли, которые понятия выполняют в познании и мышлении

В исследовании Саймона и Котовского, посвященном выработке человеком понятий для последовательных образов, специально рассматривались последовательные образы, которые содержатся в особой системе тестов «Thurstone Letter Series Completion Test», т. е. последовательности типа *atbataatbat*— или *urtustuttu*—. Их точка зрения на понятия, обучение понятиям и роль понятии в мышлении, однако, обладает большой общностью. Их намерение, как они сами говорят, состояло в том, чтобы «объяснить, в какой форме человек помнит или «хранит» образ последовательности; как он образует образ последовательности из понятия или правила, которое он держит в памяти, и как он вырабатывает понятие или правило, выводя его из некоего примера. Теория имеет форму программы для компьютера, которая имитирует процессы образования последовательности и вырабатывания правита и создает в памяти компьютера символические структуры, которые призваны представлять хранимое в памяти понятие».

Представление понятий в памяти достигается за счет использования очень простого языка, имеющегося в распоряжении того, кто вырабатывает и использует понятия. Предполагается, что испытуемый знает английский алфавит и способен осуществлять операции с символами алфавита, пользуясь такими элементарными представлениями, как «следующая», что означает следующую по порядку букву алфавита. Саймону и Котовскому удается показать, как испытуемые, обладающие указанными способностями, могут *описать* последовательные образы на этом языке. Например, образ *aibataatbat* можно рассматривать как состоящий из трех-буквенных периодов: *atb ata atb at*—. Внутри каждого периода первая буква *a,* вторая — *t,* а буквы *b* и *a* чередуются в качестве последней буквы в периоде. Другими словами, первая и вторая позиции заняты постоянными *a* и *t* соответственно, в то время как третья позиция принимает значения из двухбуквенного цикла, содержащего *a* и *b.*

Описания для образов состоят из двух частей, причем первая инициирует вторую. Первая часть описания указывает циклы, по которым отличаются переменные, а также место в каждом цикле, с которого начинается период. В данном случае мы должны указать в инициирующей части описания, что цикл состоит из букв *a* и *b* и что ряд начинается с того, что на место переменной позиции ставится буква *b.* Вторая часть описания характеризует ряд по отношению к непосредственно предшествующему периоду (или, для первого периода, по отношению к началу). Выражение (2) описывает такой ряд в принятой нами системе обозначений.

Описание —: период длиной, три; инициирование,
подсписок 1; повторение последовательности,
подсписок 4.
Подсписок 1—: первая переменная, подсписок 2,
Подсписок 2—: алфавит, подсписок 3; начальный
элемент, *b* .
Подсписок 3: *a, b.*
Подсписок 4—: поместить, *a;* поместить, *b ;* поместить,
первая переменная;
следующая, первая переменная.

(2)

Это громоздкий, но зато точный способ указания того факта, что имеется одна переменная для инициирования ряда и что эта переменная использует цикл (*a*, *b)* и начинается с *b* . Третья строка связывает наименование элемента с подсписком, описывающим повторение последовательности, что указывается в строках 8—10. Мы провели описание, как если бы оно было *предписанием* для действия: «Помести *a*; помести *t* ; помести текущее значение переменной; замени текущее значением переменной следующим (чередуя тем самым значения *a* и *b)».* С тем же успехом мы, однако, можем рассматривать (2) как описание, указывающее системе, как создать ряд *aibaiaat*—. Именно в этом смысле познавательные объекты, такие, как представленный записью (2), могут рассматриваться **и как правила, и как описания**. Поэтому вполне понятно утверждение Саймона и Котовского, что «**всякий, кто выучил описание образов, выучил также понятие, воплощенное в соответствующей последовательности**». Отметим, что если в строках 5—6 выражения (2) пара (начальный элемент, *b)* изменяется на пару (начальный элемент, *a* или *b)*, то правило становится *обобщением* первоначального выражения (2) и будет верным как для ряда *albataat*—, так и для ряда *ataatbat*—.

Используя этот программированный пример информационного подхода к формированию понятий, **рассмотрим функциональную роль, которую играют эти понятия в познании и мышлении.** Возьмем понятие «две синие фигуры» в эксперименте Брунера. Представим себе теперь правило для понятия и некоторые положительные примеры, удовлетворяющие этому правилу, как познавательные элементы некоторого индивидуума. Заметим, что **в общем случае понятие включает меньше информации и подробностей, чем любые элементы, служащие его примерами.** Кроме того, как мы увидим на этом разборе, два познавательных

элемента, служащие примерами некоторого понятия, совсем не обязательно должны быть эквивалентными в других отношениях. Поэтому легко видеть, что отношение между понятием и некоторым частным примером — (это отношение *абстрагирования* или *обобщения.* Более того, процесс абстрагирования или обобщения в мышлении может трактоваться как процесс, приводящий к познавательным элементам, находящимся именно в этом отношении с первоначальными примерами.

Можно выразиться еще точнее. Большое количество различных понятий может быть определено даже по отношению к столь ограниченному множеству примеров, как карточки Брунера. Понятие типа «две синие фигуры»— это абстракция или обобщение *по отношению* к ограниченному подмножеству характеристик множества примеров. Вопрос этот имеет определенное значение, потому что **мы полагаем, что индивидуумы образуют абстракции, обобщения и понятия в связи с определенными целями.** В результате мы можем ожидать, что между некоторыми характеристиками и некоторыми функциями должны образоваться сцепления. Действительно, одним из наиболее важных аспектов в **эвристиках человека**, одной из главных **абстракций его индивидуального опыта** является **его представление о существенных переменных в некотором частном классе ситуаций.**

Отметим также, что поскольку индивидуум оценивает познавательные или перцептуальные элементы по отношению к некоторому правилу, он может быть с разумным основанием описан как «имеющий *установку* на», «направляющий *внимание* на» или *«ожидающий»* определенные характеристики существенных переменных, на основе которых и сформулировано правило. В результате интеллектуальная модель использования понятий обеспечивает подробную интерпретацию таких явлений.

Такой взгляд на интеллектуальные процессы в человеческом познании и восприятии четко **выделяет функциональную роль понятий, как познавательных фильтров.** Чтобы увидеть, как наличие понятий дает индивидууму возможность фокусировать свое внимание и использовать отдельные элементы из нагромождения деталей, которые имеются в его памяти и в окружающей его среде, нам нужно только предположить, что некая обобщенная система интеллектуальных процессов способна использовать содержащуюся в правиле информацию, чтобы направлять операции по обработке информации (как процесс проверки использовал правила в опыте Брунера или при подборе приемлемого комплекта одежды). Программа Саймона и Котовского дает пример модели, которая и обеспечивает это,

сосредоточиваясь в один момент на одном аспекте ряда букв, а в другой момент — уже на каком-то другом.

Интересно выяснить, какого рода познавательные элементы мы берем, когда в явном виде включаем временные отношения в наши представления. Рассмотрим например, правило (3). Многоточие, как и раньше, означает продолжение.

```
Правило  — :   имеет, подсписок 1.
Подсписок 1:      подсписок 2, подсписок 3.
 Подсписок 2 —: цвет,  черный;  размер,   малый;
                   ведет к,  подсписок 3; ...
Подсписок 3 —: цвет, синий; размер, малый; ..
```

(3)

За исключением признака *ведет к,* который устанавливает временное отношение между подэлементами, отдельные пары признаков и значений выбраны произвольно в чисто иллюстративных целях. Таким образом, признак *ведет к* устанавливает связь между двумя подэлементами, в которых отношение *ведет к* состоит из всех пар *(a, b),* таких, что появлению *b* предшествует *a* Следовательно, правило (3) в целом определяет отношение, состоящее из пар элементов *(a, b),* каждый из которых имеет указанные характеристики или свойства, причем один элемент приводит к другому. **Такие познавательные элементы имеют характер представлений синдромов и следствий, причем каждый синдром состоит из совокупности симптомов, свойств или знаков**; например, совокупность (черный, малый, ...) могла бы наблюдаться диагностиком, психиатром, астрономом и т. д. в его работе.

Дальнейшая оценка некоторых функций правил, как познавательных элементов, может быть получена путем рассмотрения временного отношения между выводом правила и наблюдением или открытием примеров. Предположим, например, что индивидуум формирует конкретное правило до того, как он открыл какие-либо элементы, удовлетворяющие правилу. В таком случае мы можем рассматривать **правило как пример того, что мы обычно называем догадкой, предчувствием, моделью, гипотезой или теорией.** Какие из выражений будут более подходящими, зависит от таких факторов, как точность, с которой формулируется правило, степень, в какой оно вытекает из более общего правила и совместимо с ним, и т. д. Наоборот, если правило определяет, что некоторое множество *следует* из *предшествующего* наблюдения отдельных элементов, то мы можем рассматривать его как индуктивное обобщение.

Мы сделали кое-какие предположения о значении понятий в отношении их *функциональной роли* в умственной деятельности индивидуума.

2.2. Признаки объектов

2.2.1. Характеристика признаков

Существует ряд признаков, по которым определяют, во-первых, относится ли рассматриваемый признак к объекту (элементу) или к самой системе, во-вторых, — к какому именно объекту относится рассматриваемый признак и, в-третьих, помогают ли эти признаки составить правильно **формулу определения понятия — словесную характеристику сущности элемента в виде совокупности признаков, необходимых и достаточных для идетификации элемента**. Чтобы правильно составить формулу определения понятия, удовлетворить требованию единства элемента, **объект формулируется только из признаков того объекта , к которому он относится.** Поэтому знание признаков объекта очень важно.

Признаком объекта **будем называть всякое внесение в формулу определения указание на использование в объекте элемента**; **на особую форму любого упомянутого в формуле объекта; на взаимное расположение объектов; на наличие или форму связей между объектами, на соотношение размеров объектов, всякое указание на параметры, характеризующие температурные, временные, пневматические и другие режимы (состаяния) и т. д.**

Все признаки объектов можно разделить **по степени важности** (существенности) признака для характеристики объекта и **по группам, характеризующим сущность объектов** .

2.2.2. Классификация признаков по степени их важности для характеристики объекта.

По этому основанию признаки объектов делятся на **существенные (главные), дополнительные и случайные (излишние**).

Каждый объект, так же как и любой другой предмет, отображается в нашем сознании в виде множества признаков.

Признаки характеризуют как весь объект в целом, так и его части. Например, они могут характеризовать конструкцию, элементы объекта, принципы его работы, материал, из которого он сделан, его качество,

окраску и т. п. ***Совокупность всех признаков отличает данный объект от любого другого***. **Нет двух объектов, абсолютно похожих друг на друга**. В совокупности признаков, присущих объекту, всегда найдутся признаки (например, точность изображения отдельных элементов, точность дозировки, положение объекта в пространстве относительно других тел и т. п.), отличающие его от другого, даже очень похожего объекта.

Однако среди многообразия признаков, характеризующих объект, есть признаки, выражающие сущность, природу объекта, его **коренные свойства**. ***Такие признаки называют существенными (главными).*** **Каждый из существенных признаков необходим, и все вместе достаточны для характеристики объекта.**

Другие признаки лишь дополняют полностью охарактеризованный существенными признаками объект или конкретизируют его существенные признаки. Если отсутствие в характеристике объекта существенного признака приводит к неопределенности, к невозможности создания или описания (понимания) этого объекта, то отсутствие дополнительного признака лишь уменьшает степень конкретизации объекта, не вызывая затруднений в его реализации (понимании).

При этом в одном объекте в зависимости от решаемой задачи существенные и дополнительные признаки могут меняться местами; дополнительный признак может стать существенным, а существенный — дополнительным. Например, при создании автомобилей скорой помощи для стран с тропическим климатом, дополнительный признак— антикоррозийное покрытие нижней части автомобиля— становится существенным. При описании объекта обычно часть дополнительных признаков в зависимости от цели решаемой задачи становится существенной, а часть существенных признаков становится дополнительной или вообще отпадает. Например, на некоторых марках современных автомобилей скорой помощи отсутствует заводная ручка — существенный признак старых конструкций. При создании на основе одних объектов других такой переход существенных признаков на место дополнительных и наоборот еще заметнее.

В формальной двузначной логике обычно обходятся двумя типами признаков: **существенными и дополнительными**. В трехзначной логике при анализе признаков объектов такая классификация недостаточна, и поэтому приходится вводить третий тип, а именно **излишние или случайные признаки.**

Если существенные признаки характеризуют, а дополнительные конкретизируют объект, то случайные признаки не требуются ни для характеристики, ни для конкретизации объекта. Дело в том, что **любой конкретный объект наряду с существенными и дополнительными**

признаками содержит массу случайных признаков, характерных для местных конкретных условий и не имеющих к объекту никакого отношения. Эти случайные признаки не характеризуют объект, поэтому следует фиксировать только существенные и дополнительные признаки, имеющие отношение не только к донному объекту, но и к другим похожим объектам этого же класса.

Каждая группа объектов характеризуется своими специфическими признаками, которые, однако, подчинены общей закономерности по их значимости. В эту общую закономерность включается также взаимодействие признаков объектов. **Для характеристики объекта важно не просто перечислить его главные, обязательные признаки, но и выявить определенную, присущую данному конкретному объекту, взаимосвязь этих признаков.** Указание на взаимосвязь главных, обязательных признаков имеет принципиальное значение, так как свидетельствует об их единстве и, следовательно, об одном решении. Если же между группами главных, обязательных признаков нет структурной взаимосвязи, если группы признаков не зависят одна от другой и характеризуемая ими часть объекта может применяться и других сочетаниях, в других комплексных объектах , то это свидетельствует об отсутствии единства

Иногда считают, что объект характеризуется только существенными признаками, а остальные значения не имеют, поскольку для установления факта производится в основном по принятой логической системе. **В формуле определения понятия объекта , в котором перечисляются все главные, обязательные, взаимосвязанные признаки объекта, характеризует объект лишь в самом общем принципиальном виде, он отражает как бы основное содержание (иногда говорят — идею) объекта, но не отражает конкретных форм и решений частных задач.** Поэтому необходимо подробно описать объект, указав все его признаки: как главные, так и дополнительные.

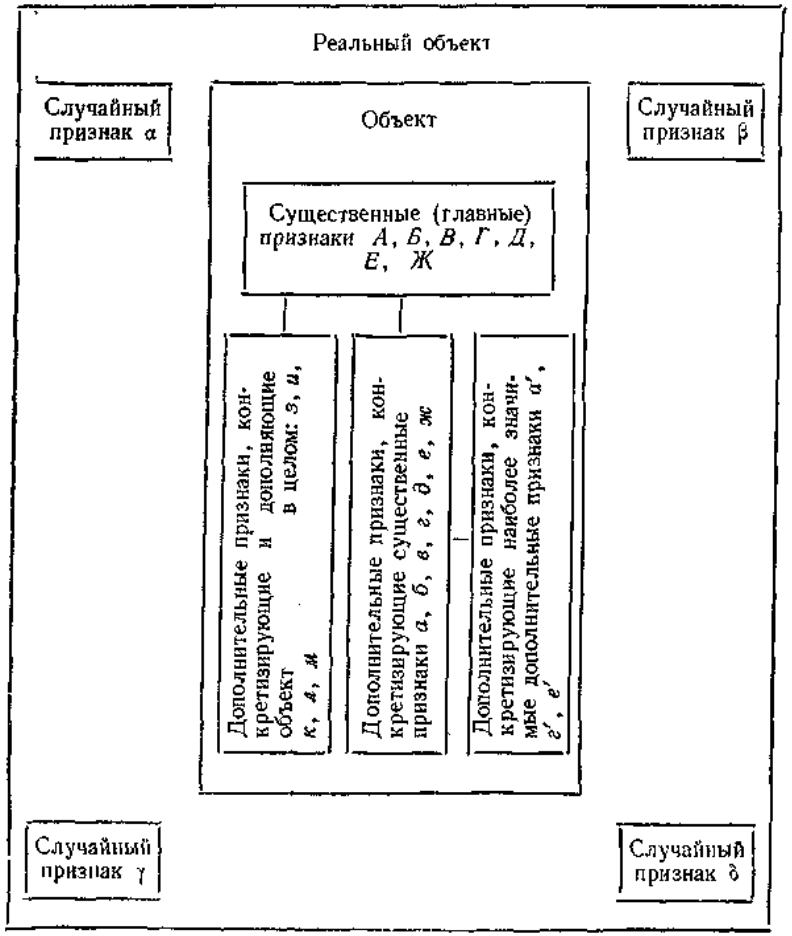
Дополнительные признаки есть у большинства объектов . Однако они не всегда отражаются в определениях, а поэтому их описания оказываются неполными, схематичными и часто состоят только из одного пункта.

Если объект охарактеризован только существенными признаками, может сложится впечатление, что объект в целом относится к одной категории понятий. Однако если в определении описаны как главные, так и дополнительные признаки, то оснований для возникновения таких суждений нет, ибо объект охарактеризован подробно и для его

понимания (использования) не требуется решать дополнительные задачи.

Дополнительные признаки объекта можно разграничить на признаки, дополняющие и конкретизирующие объект в целом; признаки, конкретизирующие существенные признаки объекта и признаки, конкретизирующие более значимые дополнительные признаки. Подобное деление носит чисто условный характер: признаки приведенных трех групп по существу ничем не отличаются друг от друга.

Структурная схема расчленения и взаимоподчинения признаков реального объекта, выглядит следующим образом.



Исследователь в анализируемом объекте выявляет и фиксирует минимум главных признаков, необходимых и достаточных для суждения о сущности объекта. Нельзя, например, характеризуя вещество, указать, что оно в основном состоит из двух конкретных ингредиентов, не упомянув остальных, если без них вещество не может быть получено. Но не обязательно указывать какой-либо конкретный краситель для вещества, если получение разной окраски вещества достигается с помощью различных красителей либо вообще без них. Этот признак необязательный, так как он не взаимосвязан с главными признаками.

Комплексный объект следует делить на отдельные части, если каждая из них несет самостоятельную функцию, структурно не взаимосвязана с другими, может выполнять свою функцию и в другом сочетании . У каждого самостоятельного объекта в этом случае выявляются и фиксируются главные признаки, хотя каждый отдельный объект представляет собой только часть . Однако **если отдельные части объекта структурно взаимосвязаны, т. е. способны выполнять свои функции только в данном виде, только в сочетании с другими конкретными частями объекта, то в этом случае делить комплексный объект нельзя, так как взаимосвязь подобных частей является одним из главных и обязательных признаков, характеризующих комплексный объект.** Часто затруднения возникают и вследствие большого количества обязательных признаков (это характерно для больших объектов) и сложности их отграничения от дополнительных признаков, так как для формирования определения объекта нужны не только главные, но и многие другие признаки.

От правильности анализа признаков зависит правильность формирования определения понятия объекта .

2.2.3. Классификация признаков по группам, характеризующим объекты.

Множество признаков, характеризующих объекты , можно классифицировать по нескольким **формам или группам признаков**. На эти группы разграничиваются все признаки независимо от их важности для объекта, т. е. и **существенные, и дополнительные, и случайные признаки одинаково принадлежат к одной из групп признаков.** При этом каждый вид объектов имеет свои присущие только ему группы признаков. Однако между группами признаков разных видов объектов можно провести параллель, которая выливается в классификацию признаков, общую для всех видов

объектов . Для каждого вида объекта будем различать пять групп признаков, которые представлены в табл. 1.

Таблица 1

Классификация признаков по группам, характеризующим объекты

Номер группы	Группа признаков, характеризующие объекты со стороны	Признаки, присущие		
		Объекту	способу	веществу
I	Структуры	Элементы	Приемы, процедуры	Ингредиенты
II	Взаимоположения, взаимосвязи	Взаимоположение и взаимосвязь элементов	Последовательность операций	Взаимоположение ингредиентов
III	Формы	Геометрическая форма элементов, форма взаимосвязи элементов	Режимы, конкретные технологические параметры	Форма отдельных ингредиентов
IV	Соотношения	Соотношение размеров элементов	Соотношение материалов, используемых в процессах	Соотношение ингредиентов
V	Материала	Материал, из которого выполнены элементы	Элементы и материал, используемые в процессах	Характеристика ингредиентов

Следует иметь в виду, что кроме приведенных в таблице групп признаков, общих для всех видов объектов , существуют и другие группы признаков, характерные для каждого вида объекта.

2.3. Методы формирования научно-образовываемых понятий и их определений

2.3.1. О понятии

В процессе научного творчества человек может сделать открытие, создать новую теорию, построить научную гипотизу и пр. Истины эти строго разграничены.

Формирование определения понятия объекта осуществляется как установление объективно существующих закономерностей, свойств и явлений материального мира, вносящие определенные изменения в уровень (степень) познатия данного объекта.

Закономерности, свойства и явления объектов существуют в природе независимо от воли людей. Часть этих свойств и явлений уже установлена, они нам известны и формируют наше представление об объектах. Другие закономерности и явления хотя и существуют, но пока неизвестны и будут устанавливаться и процессе творческой деятельности человека, направленной на изучение объектов. Мы знаем, что между телами действуют силы гравитации, что тела при нагревании расширяются, что в проводнике при пересечении им магнитных силовых линий возникает электрический ток и т. д. Эти наши знания основаны на когда-то сделанных открытиях.

Неважно, каким образом происходило познание: наблюдением, теоретическим исследованием, экспериментально с применением сложных приборов и т. д. Важно, что **образование нового понятия и формирование его определения подняло уровень наших знаний об окружающем мире еще на одну ступеньку.**

Мы можем что-то узнать о явлении или закономерностисти, использовать их, устранить их вредное влияние, но не можем изменить явление или закономерность как таковые.

Понятие - это мозговой продукт человеческого творчества, оно создается человеком как ответ на запросы его повседневной деятельности.

С появлением и развитием цивилизаций начинается научная и техническая деятельность человека и для **фиксации (описания) объектов, процессов и результатов познания появляются понятия**. Колесо, копье, очаг, лук и стрелы, соха суть понятий.

Если открытие новых явлений связано с областью познания, то формирование понятий относится к области жизнедеятельности человека. **Без понятий жизнь человека невозможна**. **Человек общается с окружающим миром посредством понятий. Отбери у человека эту возможность и человек погибнет.**

Используя вновь открытые свойства и явления материального мира, человек формирует новые понятия, характеризующие объекты, процессы и овеществленные объекты (вещества), совершенствует науку и технику для удовлетворения своих жизненных потребностей. **Большинство свойств, явлений и закономерностей существующих истин не могут быть использованы человеком непосредственно без описания их новыми понятиями, что в конечном итоге порождает необходимость создания системы понятий.** Открытое в свое время свойство проводника нагреваться при прохождении по нему электрического тока постепенно привело к появлению множества новых понятий для описания открытых новых явлений в этой области. Открытие М. Фарадеем явления возникновения в проводнике э. д. с. при пересечении им магнитных силовых линий дало возможность создать генератор электрического тока и электродвигатель, и одновременно потебовало для описания этого явления **образовать новые понятия и дать им определения**.

Понятия имеют важное методологическое значение. Продуктивность, работоспособность понятия зависит от его **научности, от того, насколько полно оно отражает определяемый предмет в свете имеющихся и перспективных потребностей практики и уровня развития науки. Понятия не неподвижны, а вечно движутся, переходят друг в друга, переливаются одно в другое, без этого они не отражают живой жизни. Многие из научных понятий истин подвергаются критическому пересмотру.**

Слово *понятие* многозначно. Понятием называют как процесс создания чего-то, так и результат этого процесса. Сам результат может быть различным по характеру: говорят, например, о таких понятиях , как машины и рифмы, технологические приемы и новые обряды, шахматы, письменности, азбуки, нумерации, системы счислений.

В науке термин *понятие* имеет научнообоснованное определенное значение. По большому счету, **понятие это результат научной деятельности** (если понимать **науку в широком смысле слова как отрасль человеческой деятельности, производящую знания**).

Отсутствие четкого определения понятия может породить неясности, недоразумения по поводу точного значения термина и, как следствие, неоднозначность его толкования.

Понятия выражаются различными способами.

1. Понятие выражается только термином, который может и не поясняться, так как либо он достаточно известен и понятен, либо его содержание должно быть определено путем формулирования научного определения или толкование его как аксиому.
2. Понятие поясняется характеризующими его признаками.
3. Введя термин, обозначающий понятие, **необходимо пояснить термин перечислением предметов, охватываемых понятием**.
4. Может быть дано прямое определение понятия, построенное по правилам логики, т. е. в виде так называемого **формально-логического определения.**
5. Определение понятия может быть дано с использованием более сложного приема, когда общее определение дополняется, конкретизируется, либо указываются исключения из общего определения, причем эти дополнения или исключения могут находиться в определениях других понятиях.

Например, построенное по правилам формально-логических определений, понятие «наука» содержит в качестве родового понятия *решение научной задачи,* и в качестве видовых отличий— *существенную научную новизну* и *нучный результат.*

Итак, **формируемое понятие должно обладать определенными признаками или, иными словами, отвечать следующим критериям**: **являться результатом нучной деятельности человека; понятие должно быть, как правило, эксклюзивным; оно должно обладать существенными отличиями в сравнении с известными понятиями.** Эти критерии будут рассмотрены в следующем параграфе.

2.3.2. Критерии определения понятия

Применительно к родовому признаку образования понятия возникает вопрос о его связи с понятием *процедуры образования.* **Процедура образования понятия определяется в функциональном плане, т. е. характеризуется как совокупность средств образования понятий в системе понятий, используемых в науке и общественной практике для увеличения (расширения) степени научного и практического познания предмета знания.** Основываясь на этом, можно считать, что родовой признак образования понятия следует понимать как требование, чтобы **понятие относилось к определенной области науки**.

Установим в качестве родового признака понятия именно *процедуру образования понятия,* что указывает на **строго определенное**

содержание понятия, а именно — связанное с объектом, процессом и однородным материальным объектом (веществом).

В связи с этим важное значение приобретает вопрос о предметах образования понятий. **К предметам образования понятий будем относить: объект, процесс и однородный материальный объект (вещество)**.

Таким образом, **семантически значимое понятие характеризуют не только общие критерии, но и предметы составляющие сущность понятия.**

Итак, как мы уже говорили, будем так регламентировать понятия истины: **объект, процесс и однородный материальный объект (вещество)**.

Объект как предмет понятия характеризуется конструктивными (компоновочными) средствами — определенными формами элементов, их взаимным расположением, средствами связи и взаимодействием, соотношением размеров и т. п.

Процесс как предмет понятия характеризуется технологическими средствами — различного рода процессами (обработки, наблюдения, контроля и пр.), содержанием которых являются процедуры (операции), их последовательность, сочетание, режимы (температурные, временные и пр.) и т. д.

Однородный материальный объект (вещество) как предмет понятия характеризуется качественными средствами — ингредиентами, их новым соотношением (долевым, весовым, процентным), новым сочетанием и пр.

В принципе этот предмет понятия характеризуется структурными отличиями (вновь синтезированного соединения). В описании понятия однородного материального объекта (вещества), полученного химическим путем, должны быть приведены также данные о его химическом строении, физико-химических свойствах, а также раскрыт процесс (процессы) получения этого вещества.

Эксклюзивность понятия. Эксклюзивность понятия имеет существенное значение при его семантическом толковании, **так как позволяет избежать повторяемости при образовании нового понятия**. Мы постараемся в предельно отчетливой форме выявить те допущения и абстракции, которые необходимы для формулирования эксклюзивности понятия и которые можно успешно использовать для установления формальной эксклюзивности понятия. Речь будет идти о **критерии эксклюзивности**, суть которого сводится к следующему. Научное познание есть процесс, имеющий какое-то *условное* начало во времени**.** Если в принципе можно утверждать, что наука имеет некоторое временное начало, и если принять, что будут рассматриваться

знания, зафиксированные в письменных текстах в виде понятий, то можно сделать следующие допущения:

1) все эти знания об истине выражены понятиями на языке **современной науки**;

2) **каждая единица знания представима понятием и это понятие может быть индексировано**;

3) **единицы знания можно представить в виде упорядоченного определенным образом списка или серии списков понятий.**

С учетом вышеизложенного критерий эксклюзивности понятия сформулируем следующим образом: **та или иная единица научного знания (которым является понятие) считается эксклюзивной, если она к моменту ее создания отсутствует в списке ранее установленных научных знаний (понятий).** Указанный критерий эксклюзивности полностью применим к понятиям: **понятие считается эксклюзивным, если оно неизвестно специалистам.**

Существенные отличия. Не всякое, даже весьма оригинальное, ранее неизвестное понятие может быть признано научно-образованным понятием: **научно-образованное понятие, которое хотят признать таковым, должно не просто логически вытекать из существующего уровня знаний, а представлять качественное развитие знания, превышать уровень существующего**. Качественная оценка научно-образованного понятия и заключается в установлении его соответствия (или несоответствия) **критерию существенных отличий**.

Установим, что **научно-образованное понятие считается обладающим существенными отличиями, если по сравнению с понятиями, известными в науке и технике оно характеризуется новой совокупностью признаков.** Следовательно, если при оценке эксклюзивности научно-образованного понятия соотносящимся предметом служит извлеченное из совокупности существующих понятий «отдельное знание» об объекте, ближайшем к определяемому по своим элементам (признакам) и функции, то при качественной оценке научно-образованного понятия соотносящимся предметом служит уже совокупность знаний (уровень науки).

Следует сказать, что дать исчерпывающее определение критерия существенных отличий — дело весьма трудное, однако смысл этого критерия, а также формы его выражения в конкретных предложениях в процессе накапливания опыта образования понятий непрерывно конкретизируются.

Как отмечалось, понятие не может быть охарактеризовано лишь общими критериями, содержащимися в общем определении: в понятие входит и **характеристика его предметов**. Причем, если общие

критерии достаточно стабильны, круг предметов, воплощающих научно-образованное понятие, непрерывно расширяется под влиянием научно-технического прогресса, различных экономических факторов.

Как мы уже говорили, будем различать следующие предметы: *объект, процесс, однородный материальный объект (вещество).* Следовательно, **научно-технический результат, который будет претендовать на признание его понятием, должен четко подпадать под один из установленных предметов.**

2.3.3. Объект как предмет понятия

Объект как предмет понятия — это обладающий существенными отличиями элемент или совокупностью элементов, находящихся в функционально-структурном единстве.

Объект характеризуется только ему присущими признаками. Если в формуле определения понятия объекта появляется признак, нехарактерный для объекта, значит формула определения понятия составлена неправильно или неправильно определен объект.

Признаками, характеризующими объект, являются:

I. Элементы представляют собой законченные материальные единицы, которые входят в объект.

II, а. Взаиморасположение элементов, т. е. положение элементов, которые описаны в первом пункте, относительно друг друга.

II. б. Взаимосвязь элементов, т. е. виды связующих органов или действий, при помощи которых элементы объекта воздействуют друг на друга, обеспечивая функционирование объекта.

III. Форма элемента, всего объекта или его части, а также форма взаимосвязи между элементами.

IV. Соотношение размеров элементов в объекте.

V. Материал, из которого выполнен элемент, группа элементов или весь объект.

I группа признаков, характеризующих любой объект, — элементы — является наиболее важной. **Без наличия элементов нельзя представить и описать ни один объект. Если нет признаков первой группы — нет объекта, так как сочетание признаков остальных групп без наличия элементов представляет собой бессмыслицу.** Наличие в формуле определения понятия новых элементов обычно говорит о том, что данное понятие обладает существенными отличиями. Как правило, новые элементы относятся к новым существенным признакам и отражаются в отличительной части первого пункта формулы определения понятия. Термины, при помощи которых признаки группы вводятся в формулу определения понятия,

— **это общеупотребительные, наиболее устоявшиеся названия элемента.** Если такого названия нет, то необходимо употребить ближайшее родовое, общеупотребительное наименование с видовым отличием, характеризующим название элемента. **Совершенно недопустимо использование специальных или местных, жаргонных наименований.** Обычно признаки I группы (элементы) следует вводить в формулу определения понятия **при помощи существительных в единственном числе и характеризовать причастиями совершенного вида.**

Чаще всего признаки I группы следует вводить в формулу определения понятия при помощи следующих слов: **снабжен, имеет, несет, размещен, установлен, встроен, содержит, включает, оснащен и т.п.**

Признаки II группы, т. е. признаки взаимоположения и взаимосвязи элементов в объекте, — следующая по значимости группа для характеристики образовующегося понятия. **Простой механический набор признаков объектов — еще не понятие**. **Оно станет понятием, когда будет раскрыто взаимоположение, взаимодействие и взаимосвязь элементов, составляющих объект**, что обеспечивается наличием признаков II группы. **Если элементы можно сравнить с кирпичами, из которых возводится здание, то признаки II группы являются тем связующим раствором, который соединяет все кирпичи в единое здание.**

Особенно важно указывать признаки взаимоположения и взаимосвязи при введении новых элементов (признаков I группы), поскольку неизвестно, где новый элемент должен быть расположен.

Новые признаки II группы желательно фиксировать в отличительной части формулы определения понятия.

Признаки взаимоположения и взаимосвязи следует вводить в формулу определения понятия обычно при помощи причастий совершенного вида, но иногда можно использовать и глаголы совершенного вида в прошедшем времени, например: прикреплен, соединен, расположен, размещен и т. д. Реже следует использовать глаголы несовершенного вида настоящего времени, например: воздействует, контактирует. Необходимо заменять такие глаголы на причастия, например: воздействующий, контактирующий и т. п.

Наиболее часто для выражения признаков взаимоположения и взаимосвязи в формуле определения понятия следует использовать слова-термины: воздействующий, связанный, контактирующий, встроенный, совмещенный, зафиксированный, взаимодействующий, соединенный, посредством, размещенный, с возможностью перемещения, сопряженный, скользящий.

Признаки III, IV и V групп, т.е. **признаки, характеризующие форму (обычно геометрическую) элемента или форму взаимосвязи между элементами**, **соотношение размеров элементов и материал, из которого они сделаны,** относят к **вспомогательным признакам**, т. е. к признакам, которые дополняют, конкретизируют понятие, выраженное при помощи первых двух групп признаков. Чаще всего сущность понятия достаточно полно выражается совокупностью элементов и признаков взаимоположения и взаимосвязи. Признаки, выражающие форму элемента, а еще реже — соотношение элементов и материал, привлекаются лишь для конкретизации определения понятия и поэтому носят вспомогательный характер. В формулировке определения понятия новые признаки последних трех групп используются обычно в отличительных частях зависимых пунктов формулы определения понятия.

При введении в формулу определения понятия признаков последних трех групп рекомендуется использовать слова: выполненный в виде, представляющий собой, в отношении (1 :2), П-образная.

2.3.4. Корректные формы образования понятий

Выше были подробно рассмотрены виды предметов понятий и их признаки. Эти признаки для каждого конкретного понятия могут быть известными или новыми, могут быть существенными, дополнительными или излишними, но все вместе, за исключением излишних признаков, они характеризуют одно понятие. **Новые признаки занимают в понятии особое место, ибо именно они делают его непохожим на остальные понятия, придают ему новые качества.**

Новые признаки проявляются в понятии в виде вполне определенных **корректных форм**, которые выражаются в определенной взаимосвязи новых и известных признаков, в определенных сочетаниях новых признаков и в различном введении в формулу определения понятия новых признаков, относящихся к разным группам. **Эти корректные формы полностью отражаются в формуле определения понятия**. Значит новый признак вводится в формулу определения понятия таким образом, чтобы он соответствовал (совместно с другими новыми признаками, если они имеются) одной из корректных форм понятия. **Введение нового признака в формулу определения понятия помимо разработанных корректных форм нежелательно. Соблюдение корректных форм при составлении формулы определения понятия помогает с определенной степенью достоверности**

определить наличие в образовываемом понятии существенных отличий, а также разграничить существенные признаки от дополнительных. На соблюдении в формуле определения понятия определенных корректных форм основана методика определения формального понятия. Однако здесь необходимо делать следующую оговорку.

Введение новых признаков в формулу определения понятия в виде определенных корректных форм не только необходимо для контроля наличия в понятии существенных отличий, но и обеспечивает правильность составления форулы определения понятия и точное распределение новых признаков по пунктам формулы определения понятия, т. е. их разграничение на существенные и дополнительные.

Приведем корректные формы понятий, при помощи которых новые признаки предметов вводятся в формулу определения понятия:

1. **Полностью новое сочетание признаков**. Указывает на наличие существенных отличий, приводится в отличительной части формулы определения понятия. Полностью новое сочетание признаков имеют так называемые **корневые понятия**.

Схема введения в формулу определения понятия: «Предмет (объект, процесс, однородный материальный объект)..., характеризующийся тем, что он состоит из *A + B + C + D...».*

2. **Частично новое сочетание признаков**. Указывает на наличие существенных отличий, приводится в отличительной части формулы определения понятия.

Схема введения в формулу определения понятия: «Предмет, содержащий *A+B,* отличающийся тем, что он содержит *C+D...»,* где *A+B* — известное сочетание признаков, C+D — новое сочетание признаков.

3. **Введение нового признака**. Указывает, как правило, на наличие существенных отличий.

Схема введения в формулу определения понятия: «Предмет, содержащий *A + B + C,* отличающийся тем, что он снабжен D...», где *A+B + C* — известные признаки, *D* — новый признак.

4. **Замена части признаков новыми**. Указывает, как правило, на наличие существенных отличий.

Схема введения в формулу определения понятия: «Предмет, содержащий *A + B,* отличающийся тем, что он содержит C...», где *A + B* — известные признаки, *C* — новый признак, который заменил имеющийся в прототипе признак *C.*

5. **Использование более конкретного признака в качестве общепринятого**. Нередко указывает на наличие существенных отличий.

Схема введения в формулу определения понятия: «Предмет, содержащий *A+B+C+D,* отличающийся тем, что *D* выполнен в виде *D′...»,* где *A+B+C+D* — известные признаки; *D′* — конкретный признак, используемый в качестве *D.*

Как видим, в первых пяти корректных формах понятия используются только новые структурные признаки.

В следующих трех корректных формах понятия используются новые **признаки связи и взаимоположения**.

6. **Новое взаимоположение признаков**. Часто указывает на наличие существенных отличий, хотя редко эта корректная форма понятия используется самостоятельно. Гораздо чаще она должна использоваться совместно с одной из предыдущих корректных форм понятия.

Схема введения в формулу определения понятия: «Предмет, *A + B + C+D,* отличающийся тем, что *A* расположено над *B,* а *C* встроено в *D»* или «Предмет, содержащий *A + B + C + D,* отличающийся тем, что *A* выполняют перед *B,* а *C* одновременно с *D...»,* где *A, B, C, D* — известные признаки.

7. **Новый тип связи и взаимодействия между признаками**. Часто указывает на наличие существенных отличий, особенно для таких понятий, как электрические схемы, для которых эта корректная форма понятия используется самостоятельно, для других понятий чаще используется совместно с корректными формами понятия, где присутствуют структурные признаки.

Схема введения в формулу определения понятия: «Предмет, содержащий *A + B + C+D,* отличающийся тем, что *A* контактирует с *B,* а *C* одновременно соединено с *A* и *D...*», где *A, B, C, D* — известные признаки.

8. **Совместное использование применявшихся ранее порознь признаков в виде нового сочетания.** Указывает на наличие существенных отличий только, если появляется новый эффект, которого не наблюдается при использовании порознь примененных признаков.

Схема введения в формулу определения понятия: «Предмет, содержащий *A + B,* отличающийся тем, что они используются совместно с *C + D...-»,* где *A+B* и *C+D* — известные признаки, применяемые по своему прямому назначению.

9. **Новая корректная форма (режим, структура) признака**. Для объектов не указывает на наличие существенных отличий.

Используется в зависимых пунктах формулы определения понятия. Нередко включается в первый пункт формулы определения понятия истины.

Схема введения в формулу определения понятия: «Предмет, содержащий *A + B + C+D,* отличается тем, что С выполнен цилиндрическим, а *D* в виде конуса...» или «Предмет, содержащий *X + B + C + D,* отличающийся тем, что С проводят при температуре до 50° С...», где *A + B + C + D* — известные признаки.

10. **Новое количественное соотношение признаков**. Для овеществленного объекта нередко указывает на наличие, а для остальных предметов понятий, как правило, на отсутствие существенных отличий (если нет других новых признаков).

Схема введения в формулу определения понятия: «Предмет, содержащий *A + B + C+D,* отличающийся тем, что он выполнен на основе *A, B* взято от *a%* до *b%, C* — от *c%* до *d%* и *D* — от *m%* до *n%*...», где *A+B + C + D* — известные признаки.

11. Сочетание двух и более одинаковых признаков. Указывает, как правило, на отсутствие существенных отличий. Используется, но не часто, в зависимых пунктах формулы определения понятия. Схема введения в формулу определения понятия: «Предмет, содержащий *A+ B + C+D,* отличающийся тем, что *D* образует цепочку из трех элементов...», где *A + B + C + +D* — известные признаки.

12. **Использование нового материала.** Иногда указывает на наличие существенных отличий в процессе, реже в объекте и практически никогда в материальном объекте.

Схема введения в формулу определения понятия: «Предмет, содержащий *A + B + C + D,* отличающийся тем, что *D* выполнен из фторопласта...», где *A+B + C+D* — известные признаки.

В этой форме под использованием нового материала для материального объекта понимают характеристику ингредиента, например, предварительно обработанный сплав.

К этой форме относится также иногда использование в процессах нового количественного сочетания материалов.

13. **Использование известных (технических) средств, ранее не применяемых для этих целей**. Характерно только для процесса. Как правило, указывает на отсутствие существенных отличий. Существенные отличия появляются лишь при использовании совместно с новой операцией.

Схема введения в формулу определения понятия: «Процесс, состоящий в *A+B + C+D,* отличающийся тем, что *D* осуществляют в вакуум-фильтре...», где *A +B + C+D* — известные признаки.

2.3.5. Последовательность выявления отличительных признаков

Разберем методику выявления отличительных признаков в научно-обоснованой формулировке понятий как последовательность логических приемов и операций, направленных на выделение идеи по образованию нового понятия на основании изучения результатов конкретной научно-технической разработки, сравнение ее с известными понятиями той же задачи и **формулировку вновь образовываемого понятия как единной взаимосвязанной (взаимозависимой) совокупности признаков**.

Рассмотрим **этапы выявления отличительных признаков** в научно-технической разработке.

Этап первый. Выявление отличительного признака начинается с определения, является ли предложение по образованию понятия решением задачи образования нового понятия. Устанавливается, имеет ли научно-техническое решение конкретные отличительные признаки, относится ли научно-техническая разработка к объектам, в которых можно определить такие отличительные признаки, которые позволят образовать новое понятие.

Этап второй. На этом этапе производится выделение в научно-технической разработке отличительных признаков, которые моглы бы быть обобщены на другие понятия этого класса или группы. Утилитарный, узко направленный, полезный только для данного конкретного случая признак практически никогда не может быть использован при образовании нового понятия, обычно он может быть использован для уточнения (конкретизации) существующих понятий. **Новое понятие должно отражать что-то новое в научно-техническом прогрессе, поэтому не может локализоваться, замкнуться только на одном научно-техническом решении**. **Новое понятие должно сделать шаг, пусть небольшой, по пути совершенствования науки и техники и оказать, пусть даже незначительное, влияние на развитие науки и техники.** В конкретной научно-технической разработке очень важно выделить общие признаки, которые были бы полезны при определенных условиях для целей группы образовывемых понятий и таким образом могли оказать влияние на научно-технический прогресс. Пусть даже это будет небольшая часть разработки или один из ее вариантов.

Этап третий. На этом этапе необходимо установить полноту информации по выявленному и обобщенному для целой группы понятий научно-техническому решению, очищенному от локальных

наносных условий. **Естественно, что перед началом разработки проводится ознакомление с научно-технической литературой. На базе знаний, почерпнутых из этой литературы, и рождается проект определения образовываемого понятия.**

Этап четвертый. Здесь рассматриваются признаки научно-технического решения и делается заключение, **к какому предмету (объекту, процессу, материальну веществу) относятся выявленные признаки.** Остановимся на следующих **двух моментах**. **Во-первых, вид предмета**, для которго разрабатывается понятие, **не обязательно должен соответствовать виду научно-технического решения. Во-вторых, если при выявлении признаков предмета разработки понятия будут выявлены признаки двух или более видов предметов, относящихся к решению одной задачи разработки понятия, то следует рассмотреть вопрос о целесообразности разработки прототипа понятия на каждый вид предмета отдельно или совместно, а может быть, только на один выбранный вид предмета.**

Иногда весьма трудно решить вопрос, на какой вид предмета составлять описание определения на предполагаемое понятие. В этом случае приходится параллельно формулировать определение понятия на несколько видов предметов, определять существенные отличия и полноту охвата содержания понятия по каждой формулировке определения и выбирать тот вид предмета понятия, где существенные отличия и полнота охвата содержания определения понятия значительнее.

Этап пятый. На этом этапе формулируется цель разработки понятия, причем желательно сформулировать все цели, т. е. **начальную, промежуточную, конечную** и пр., чтобы лучше понять назначение предложения по формированию определения понятия и затем при формулировке определения понятия легко выбрать цель, наиболее полно отражающую назначение предложения.

Естественно, что формулируется не цель конкретной разработки, из которой выявлены отличительные признаки, а цель разработки понятия, не содержащего недостатки, присущие аналогичным понятиям.

Этап шестой. На основе найденной научно-технической информации выбираются аналоги разрабатываемого понятия, т. е. предметы того же назначения, что и рассматриваемый, сходные по научно-технической сущности и по достигаемому результату при их использовании и близкие по совокупности признаков, которые послужили или могли бы послужить базой для разработки определения нового понятия.

Поиск аналогов ведется по всем доступным информационным источникам.

При поиске аналогов следует возможно более полно использовать преимущества справочно-поискового аппарата, а также разработанные поисковые системы, методы и средства автоматизированного поиска. Поиск аналогов заканчивается, если просмотрены все информационные источники, которые исследователь мог использовать, или найден аналог, полностью совпадающий по совокупности существенных признаков с предполагаемым понятием, или же аналог, часть признаков которого полностью совпадает с совокупностью существенных признаков разрабатываемого определения понятия.

Библиографические данные об аналогах заносятся в таблицу «Карта результатов исследования по источникам научно-технической информации».

Из найденных аналогов выбирается прототип разрабатываемого понятия — это наиболее близкий аналог по научно-технической сущности и по достигаемому результату при его использовании.

Для определения прототипа понятия проводится анализ отобранных при происке аналогов и выявляется для каждого из них признаки, сходные с признаками предполагаемого понятия.

Выбор прототипа определяется максимальным количеством сходных существенных признаков предполагаемого понятия и аналогов, одним-двумя существенными признаками, которые в большей степени по сравнению с другими влияют на научную эффективность понятия и могут быть выделены из числа сходных с признаками аналога, максимально близким научно-техническим решением по предполагаемому эффекту.

Если в формуле определения понятия характеризуются предметы разных видов, то прототип определяется для каждого из них. Если в формуле определения понятия предмет понятия охарактеризован совокупностью его существенных признаков как главных частей целого и дополнительно в последующих пунктах определения охарактеризовано содержание этих частей как форм выполнения существенных признаков, то в качестве прототипа может быть принят аналог, содержащий признаки, сходные с существенными признаками предполагаемого понятия или с признаками предмета, характеризующими содержание этих частей, т. е. содержание существенных признаков.

Если выявленный предмет разрабатываемого понятия представляет собой соединение независимых частей, не связанных между собой, т. е. каждая из которых не влияет на функции другой части, то в качестве

прототипа для такого понятия могли бы быть использованы одновременно несколько аналогов.

При выборе прототипа из аналогов учитываются и другие соображения. Разработчику нового понятия **в качестве прототипа понятия следует выбирать такой аналог, который помог бы полнее раскрыть сущность понятия, выявить его отличительные стороны**. Выбранный в качестве прототипа аналог понятия должен помочь выявить существенные отличия понятия. Может сложиться впечатление, что среди аналогов обычно есть одно понятие, которое по совокупности признаков максимально приближается к разрабатываемому новому понятию, и, следовательно, только оно является прототипом этой разработки. Действительно, если осуществляется разработка конкретного понятия, то так и будет. Но, как правило, в конкретной разработке ищут общее для целой группы понятий, которое и служит основой понятия. При этом степень обобщения может быть разной. И при каждой степени обобщения будет свой аналог, максимально совпадающий по совокупности признаков с этим обобщенным понятием, т. е. **свой прототип, прототип для данной степени обобщения разрабатываемого понятия.** Трудность же состоит в том, чтобы найти ту степень обобщения и соответствующий ей прототип, которые бы максимально проявляли содержание понятия, вскрывали его существенные отличия.

Как этого добиться? Общих рекомендаций здесь нет. **Необходимо каждую степень обобщения решения проверить на существенность отличий и выбрать наиболее перспективную с точки зрения проявления сущности понятия степень обобщения.** Кроме того, надо стараться так выбрать степень обобщения решения по разработке понятия, чтобы соответствующий прототип являлся объектом широко известным.

Этап седьмой. Далее определяются существенные отличия выявленных признаков. Для этого используется наиболее подходящий способ оценки существенных отличий.

В тех случаях, когда выявленное решение при разной степени обобщения имеет различные аналоги, все они должны быть проверены на существенные отличия с помощью различных способов. В этом случае определяется наиболее перспективный с точки зрения существенных отличий вариант обобщения содержания понятия. Аналог этого варианта становится прототипом разрабатываемого понятия.

2.3.6. Методика и последовательность выявления признаков, используемых при разработке понятий

Прежде чем приступить к разработке формулировки определения понятия, **необходимо выявить признаки предметов, которые будут использоваться при разработке понятия.** Эти признаки, как правило, сосредоточены в новых научных исследованиях и технических разработках. Научные исследования и технические разрабртки оформляются в виде научно-технических решений. **Именно в научно-технических решениях сосредоточены отличительные признаки предметов исследования, которые необходимо выявить, прежде чем приступать к разработке нового понятия.**

Выявление отличительных признаков предметов преследует следующие цели.

1. Проверить, удовлетворяют ли выявляемые отличительные признаки требованиям, предъявляемым к разработке понятий, правилен ли выбор прототипа понятия.
2. Закончить выявление отличительных признаков четкой формулировкой предложения по разработке понятия в виде **перечня взаимосвязанных признаков**.
3. Выявить **существенные и несущественные признаки** вновь разрабатываемого понятия.
4. Сформулировать **предмет разрабатываемого понятия**, точно определить **объем понятия**.

Разберем этапы поиска отличительных признаков в научно-технических разработках, которые будут положены в основу разработки нового понятия.

Этап первый — выделение признаков, характеризующих научно-техническое решение. Для этого производят **анализ — мысленное расчленение решения на составляющие его части**. Если анализируется объект, необходимо перечислить все основные составные части объекта, из которых он состоит, указать их взаимосвязь, форму выполнения и при необходимости — важнейшие соотношения размеров. Если анализируется процесс, то перечисляются операции, из которых он состоит, их последовательность в технологическом процессе и режим его выполнения, материалы и вещества, участвующие в процессе, технические средства для выполнения операций и т. п.

Не допускается описание данного вида объекта признаками, присущими объекту другого вида. Если сразу трудно определить вид объекта, так как научно-техническое решение характеризуется признаками нескольких (чаще двух) объектов, то **при анализе**

необходимо выписывать все признаки независимо от того, к какому объекту они принадлежат.

Анализ — это ответственный этап выявления отличительных признаков предметов. Не всегда легко разбить целое научно-техническое решение на его составляющие части. Это далеко не механическая работа, так как **требует детального знания сущности научно-технического решения**. В противном случае анализ будет произведен неправильно, что может привести к неправильному выявлению отличительных признаков в научно-техническом решении.

Этап второй — разграничение признаков, выявленных на предыдущем этапе, **на существенные и несущественные**.

Разграничением признаков на существенные и несущественные получают совокупность существенных признаков, описывающих вновь разрабатываемое понятие, и совокупность несущественных признаков, конкретизирующих вновь разрабатываемое понятие в один из его вариантов. **Из совокупности существенных признаков в дальнейшем может быть построен первый пункт формулы определения понятия,** а из совокупности несущественных признаков — дополнительные пункты формулы определения понятия.

Этап третий — формулировка цели разработки понятия.

Этап четвертый — определение вида понятия и проверка единства понятия. На этом этапе определяют, все ли выделенные существенные признаки предмета относятся к решению одной задачи, т. е. не нарушено ли при составлении формулы определения понятия единство понятия (**совокупность всех признаков должна относиться к решению только одной задачи, признаки должны быть объединены одной общей идеей, воплощенной с их помощью**). Если какой-либо признак выпадает из общей идеи решения задачи, не подчинен ей, а служит решению другой задачи, он нарушает единство понятия и должен быть удален из совокупности расчлененных признаков.

Этап пятый — выделение признаков, характеризующих аналоги понятий. Эта работа проводится точно так же, как выделение признаков в рассматриваемом научно-техническом решении (см. этап первый).

Этап шестой — разграничение признаков, характеризующих аналоги, на существенные и несущественные и разделение существенных признаков по типам, характерным для данного вида понятия. Работа проводится аналогично разграничению признаков предложенного научно-технического решения (см. этап второй).

Этап седьмой — проведение сопоставительного анализа существенных признаков предложенного решения с существенными

признаками аналогов. Сопоставительный анализ ведут обычно по **типам признаков**. Например, для объекта сначала выясняют, все ли элементы, на которые было разбито предложенное научно-техническое решение, присутствуют в аналогах; затем проводят сравнительный анализ признаков взаимосвязи и взаиморасположения элементов и т. п. При этом выясняются новые существенные признаки предложенного технического решения, которые отсутствуют в аналогах.

При проведении сравнительного анализа следует определить признаки разрабатываемого понятия, сходные с признаками аналогов понятий. Сходными называются признаки идентичные или эквивалентные.

Идентичные признаки — это признаки, совпадающие по выполняемой функции и форме, т. е. по конструкции, материалу, технологии и т. п.

Эквивалентными называются признаки, совпадающие по выполняемой функции и достигаемому результату.

При определении эквивалентности признаков принимается во внимание их взаимозаменяемость, т. е. выполнение ими одинаковой функции при отличии по форме выполнения (по конструкции, технологии или материалу).

Эквивалентность признака подтверждается также тем, что использование признака аналога вместо эквивалентного в разрабатываемом понятии не придает последнему дополнительных полезных качеств или существенных преимуществ перед аналогом понятия. **Признак разрабатываемого понятия считается известным, если он идентичен или эквивалентен признаку прототипа понятия**. **Признак разрабатываемого понятия считается новым, если в прототипе понятия отсутствует идентичный или эквивалентный ему признак**.

Этап восьмой — проверка правильности выбора прототипа разрабатываемого понятия. Когда разрабатываемое понятие и понятия-аналоги были расчленены на составляющие их признаки и был проведен сопоставительный анализ, выявивший новые признаки разрабатываемого понятия по сравнению с каждым аналогом, появляется возможность проверить правильность выбора прототипа. Прототип выбирается с учетом рекомендаций, изложенных ранее.

Этап девятый — составление формулы определения понятия. Проведенная на предыдущих этапах работа подготавливает материал для составления формулы определения понятия. И если эта работа проведена достаточно тщательно, то формулировка определения понятия обычно не вызывает затруднений.

2.3.7. Описания понятий

Составление описания понятия — это первый этап процесса разработки понятия.

Описание понятия должно включать:

1) описание понятия с формулой определения понятия

2) чертежи, схемы и другие материалы, иллюстрирующие разрабатываемое понятие, если они необходимы для наиболее полного раскрытия смысловой сущности и значимости понятия.

Описание определения понятия является важнейшей частью процесса разработки понятия. Без преувеличения можно сказать, что от тщательности, полноты и правильности описания разрабатываемого понятия зависит дальнейшее использование понятия в науке и практике. Между тем, именно эта часть работы по формированию определения понятия вызывает наибольшие трудности: **качество описаний определений понятий в ряде случаев чрезвычайно низкое, они составляются настолько неясно и неточно, что практически не могут быть поняты пользователями, особенно, если пользователями являются компьютеры.**

Нужно отметить, что **язык, используемый при составлении описания, и в особенности формулы определения понятия, специфичен, отличается от языка обычной научной публикации. Использование именно такого языка позволяет разобраться в семантической сущности понятия**, в силу чего в ряде случаев совершенство описания понятия, и в особенности формулы определения понятия истины, имеет решающее значение.

Описание с формулой определения понятия должно:

— полностью раскрывать семантическую сущность понятия и содержать достаточную информацию для дальнейшей разработки понятия;

— давать точное и ясное представление о новизне, существенных отличиях разрабатываемого понятия.

Описание определения понятия должно иметь следующую обязательную структуру: — имя понятия и класс принятой классификации понятий, к которому оно по мнению разработчика относится;

— область, к которой относится понятие, и преимущественная область использования понятия;

— характеристика аналогов понятия;

— характеристика прототипа, выбранного разработчиком;

— **критика прототипа**;

— цель разработки нового понятия;

— семантическая сущность понятия и его отличительные признаки;

— примеры конкретного выполнения;

— формула определения понятия.

Отклонения от указанной структуры описания может допускаться в исключительных случаях, когда из-за характера разрабатываемого понятия необходимо применить порядок изложения, способствующий лучшему пониманию понятия.

При изложении описания определения понятия необходимо:

— использовать термины, общепринятые в данной области;

— соблюдать единство терминологии;

— **использовать одну систему единиц измерения**.

Не следует вводить в текст описания каких-либо чертежей и схем; в случае необходимости допускаются только химические, математические и другие формулы, причем все буквенные значения, входящие в математические формулы, должны быть расшифрованы.

Выбор единиц измерений должен согласовываться с действующими стандартами Международной системы единиц СИ.

Допускается применение в тексте описания лишь общепринятых сокращений: т. д., т. е., т. п. и др.

Имя понятия должно быть кратким и точным, содержать не более 8—10 значимых слов и соответствовать сущности понятия; оно должно характеризовать назначение предмета понятия или указывать на принадлежность к определенной отрасли науки и техники. Сочетать краткость с двумя последними требованиями в ряде случаев нелегко. Например, имя «Ключ» кратко, но не конкретно; имя «Устройство для окраски крыльев самолетов и тому подобных деталей» хотя и необоснованно длинно, но не однозначно; имя «Процесс охлаждения объектов» представляется неопределенным.

Можно заключить, что идеальным при соблюдении вышеуказанных требований будет имя, вписывающееся в одну из рубрик классификатора понятий, что значительно облегчает классификацию рассматриваемых предложений. ***Имя понятия совпадает с начальными словами формулы определения понятия***.

Необходимо отметить, что **правильная формулировка имени имеет большое семантическое значение**, поскольку каталоги носителей понятийной информации (библиотеки понятий, альбомы понятий, модули понятий) содержат лишь имена понятий.

Если в разрабатываемом понятии содержатся два и более разных предмета, например процесс и объект, которые служат единой цели и могут применяться лишь совместно, имя понятия должно включать имена этих предметов (например «Процесс извлечения урана из урановых руд и устройство для его осуществления»).

Составление описания следует начинать с указания области, к которой относится понятие, и преимущественной области его использования. Эту часть описания следует начинать словами: «Понятие относится к...».

Описание не должно расширенно толковать понятие. Например, если формула определения понятия характеризует «Процесс изготовления свинцовой оболочки электрического кабеля», то в описании не следует писать, что «понятие относится к процессам изготовления металлических, например свинцовых, оболочек кабеля» или что «понятие относится к процессам изготовления свинцовых оболочек кабеля или труб».

Не следует в описании сужать объем понятия по сравнению с тем, как изложено в пунктах формулы определения понятия. Например, если формула определения понятия характеризует «Процесс изготовления металлических защитных оболочек электрических кабелей», то нельзя указывать в описании, что «понятие относится к процессам изготовления защитных оболочек из черных металлов», так как в формуле определения понятия также подразумевается изготовление этих оболочек и из алюминия, цинка и других цветных металлов.

В разделе «Характеристика аналогов понятия» следует описать известные аналогичные понятия той же задачи (аналога), т. е. предметы понятий того же назначения, что и разрабатываемое понятие, сходные с ним по сущности.

Аналоги приводятся из числа наиболее близких к разрабатываемому понятию.

В краткой характеристике аналога (или аналогов), т. е. в описании сущности известных понятий, должны быть раскрыты его (их) существенные признаки и обязательно указаны все те из них, которые имеют сходство с признаками разрабатываемого понятия. **Должны быть отмечены и недостатки аналогов, которые частично или полностью устраняются в разрабатываемом понятии.**

При поиске аналогичных разрабатываемому понятию, наиболее существенных решений необходимо исследовать описания понятий по соответствующему классу (классам).

В разделе «Характеристика выбранного прототипа» необходимо описать конкретное известное понятие объекта, процесса или вещества, наиболее близкого по сущности к разрабатываемому понятию, т. е. наиболее близкий аналог из ранее приведенных. При этом необходимо отметить все существенные признаки прототипа, общие для него и разрабатываемого понятия. В этом разделе должна быть приведена библиографическая ссылка на источник, в котором описан выбранный прототип.

В разделе «Критика прототипа» **описываются только те его недостатки, которые устраняются разрабатываемым понятием**.

В разделе «Цель разработки понятия» цель излагается объективно и обоснованно, без утверждений рекламного характера. Объективность цели обосновывается необходимостью удовлетворения научной потребности, вызвавшей к жизни данное решение научно-технической задачи, или необходимостью совершенствования уже известного понятия.

Цель разработки понятия, указанная в п. 1 формулы определения понятия, должна быть причинно связана с признаками предмета понятия, которые перечислены в формуле определения понятия и обеспечивают достижение этой цели.

В описании разрабатываемого понятия допускается изложение и других целей (не упомянутых в формуле определения понятия), которые имелись в виду при разработке понятия.

В разделе «**Сущность понятия**» должно быть приведено краткое изложение содержания понятия в виде совокупности всех существенных признаков с выделением признаков, которые характеризуют новизну разрабатываемого понятия. Для составления этого раздела используется формула определения понятия, но имеющиеся в ней ***признаки должны быть не просто перечислены, а подробно разъяснены***. При этом должна быть показана существенность отличий предмета понятия, т. е. ***раскрыта связь между новой совокупностью признаков.***

Если формула определения понятия многозвенная, в этом разделе описания в виде отдельных абзацев необходимо привести характеристику не только первого пункта, но и всех дополнительных пунктов формулы определения понятия.

Следует указать оптимальный вариант разрабатываемого понятия и дать его характеристику.

Если понятие несложно и однозначность толкования понятия очевидна без описания, то допускается в полном объеме такое понятие не описывать.

В описании понятия должно быть четко показано, почему два или более объектов могут быть использованы лишь совместно.

Ни одна из частей описания определения понятия не может быть заменена ссылкой на описание этой части в другом документе, например, в каком-либо литературном источнике. Однако в описании допускаются ссылки на источник (источники), в котором описаны уже известные признаки понятия, содержащиеся в описании.

В разделе описания «**Характеристика выбранного прототипа**» приводится характеристика этого понятия.

Цель разработки нового понятия должна быть подкреплена убедительными доказательствами ее достижения. **Необходимо провести объективный анализ преимуществ разрабатываемого понятия по сравнению с известными.** Следует привести расчетные данные или детально объяснить, каким образом может быть достигнута цель понятия.

Описание заканчивается **формулой определения понятия, т. е. кратким изложением признаков понятия, сделанным по рекомендованным правилам и характеризующим объем понятия, его новизну и цель.**

Можно сказать, что **успех составления формулы определения понятия зависит от правильности и глубины анализа описания понятия.** Если анализ полон и правилен, если верно определен прототип, если точно выявлены действительно принципиальные признаки понятия, то не представит значительного труда изложение формулы определения понятия в соответствии с вышеуказанными требованиями, так как в этом случае **формула определения понятия** будет как бы **краткой аннотацией проведенного по правилам анализа.**

Вышеизложенное показывает, что **формулу определения понятия необходимо составлять в строгом соответствии с принятыми правилами**. **Всякая ошибка в формуле определения понятия — логическая, грамматическая, формальная — может привести к непониманию смысла понятия, его неправильному толкованию, искажению его смысла, может сузить или необоснованно расширить предмет понятия, лишить его необходимой ясности и определенности.** Поэтому описание понятия и формула определения понятия, составленные с нарушением вышеуказанных правил, не обеспечивают реализацию цели, поставленной для разработки нового понятия.

2.3.8. Формула определенияе понятия

2.3.8.1. Назначение формулы определения понятия

Формула определения понятия — это краткая, составленная в виде аннотации по определенным правилам и форме словесная характеристика сущности понятия как единой совокупности признаков, взятых в их единстве, характеризующих понятие, которые необходимы и достаточны для однозначного семантического толкования данного понятия.

Описание понятия должно давать полное представление о понятии, раскрывать его смысловое значение. В описании обычно содержатся подробные сведения о понятии с главными и дополнительными признаками. При отсутствии формулы определения понятия из-за таких подробностей возникает необходимость в многочисленных оговорках в тексте описания, указывающих на необязательность тех или иных элементов описываемого предмета, на возможность замены их другими и т. д. Кроме того, определение объема понятия по подробному описанию, требует затраты большого количества времени. В результате возникла потребность в необходимости краткой формулировки определения понятия, т. е. в необходимости **формулы определения понятия**.

Формула определения понятия имеет научно-техническое и информационное значение.

Научно-техническое значение формулы определения понятия.

Формула определения понятия не только определяет границы понятия и фиксирует вклад ученых в научный прогресс**,** но и характеризует ту новую ступень, на которую поднимает науку данное понятие. **Формула определения понятия кратко и четко выражает научную сущность понятия**, т. е. **отображает в логическом определении предмет понятия совокупностью его существенных признаков**. Ученый в своей творческой работе обычно отправляется от какого-то существующего уровня развития науки и привносит новый научный результат, имеющий преимущества по сравнению с существующими. Максимально обобщенным предмет понятия описывается только в формуле определения понятия, где он, очищенный от всех конкретных, случайных признаков, предельно раскрывает смысел понятия.

Информационное значение формулы определения понятия.

В формуле определения понятия описывается совокупность признаков, необходимых и достаточных для разработки понятия. Специалист по данной отрасли, познакомившись с формулой определения понятия, может понять его и использовать понятие в своих исследованиях. Это значит, что **формула определения понятия приобретает важное информационное значение.**

2.3.8.2. Основные требования, предъявляемые к формуле определения понятия и его разработке

Для того чтобы формула определения понятия в полной мере отвечала своему назначению, она должна обладать следующими основными качествами: **лаконичностью, общностью, полнотой и определенностью, а также отвечать требованию единства понятия.**

Лаконичность формулы определения понятия требует определения предмета понятия без лишних слов. Для достижения лаконичности и единообразия в толковании формулы определения понятия следует следовать определенным правила формирования формулы определения понятия, в известной мере условным.

Общность (широта) формулы определения понятия заключается в том, что она определяет смысл понятия в возможно более широких границах. Достаточно общая формула определения понятия охватывает своим смыслом понятие во всех обобщенных видах, а не только в частном изложении. Для достижения общности предмет понятия должен быть охарактеризован общими признаками, соответствующими его смыслу.

Полнота формулы определения понятия определяется включением в нее всех существенных признаков, составляющих понятие, причем не только общих, но и частных.

Определенность формулы означает, что записанные в ней признаки, характеризующие предмет, не допускают произвольного истолкования.

Единство понятия отвечает требованию, чтобы одна формула определения понятия, относилась к одному понятию.

Формулировка определения понятия во многих случаях предопределяет распознавание или даже постановку проблемы. В определении обнаруживает себя сущность явления, которая может быть первого, второго и т. д. порядка. Например, понятие капитала первоначально определялось экономическими терминами. Сегодня существует иное его определение, учитывающее современные проблемы и потребности социально-экономического развития, например функционирование понятия "человеческий капитал". Аналогичные примеры можно привести по таким важным понятиям в менеджменте, как потенциал, цель, эффективность.

Разработка определения понятия — **это один из методов исследования. Без определений понятий невозможно описание проблемы, оценка ситуаций, доказательства результатов,**

презентация идеи.

Существуют **явные и неявные определения**. Явные построены на поиске наиболее удачных с точки зрения практики синонимов, т. е. таких понятий, которые представляются бесспорными, которые известны, функционируют в системе знаний. Но определения не строятся только на сопоставлении понятий. Они конструируются посредством дополнения этих понятий, их ограничением, выделением существенных свойств. И это не менее важная часть определения, нежели сопоставление. Проанализируем с этих позиций, скажем, определение понятия "управление".

Управление **—** это целенаправленное воздействие, согласующее совместную деятельность людей. Оно построено на сопоставлении понятий "управление" и "воздействие". Но не всякое воздействие может быть управлением. Есть воздействия случайные, непредвиденные. Поэтому необходимо выделить вид воздействия и его назначение — целенаправленное и согласующее деятельность. Аналогичным образом конструируется любое определение, но не всегда оказывается простым делом сконструировать определение. В конечном итоге практика подтверждает реальность и точность такой конструкции.

Существуют **неявные определения**. При таких его видах сущность и смысл явления передаются через использование понятия в контексте других понятий, в его концептуальных связях, функциях в системе объяснений и обоснований. Такое определение всегда неполно и неустойчиво, односторонне и туманно. Но с этим приходится мириться, как правило, до поры до времени. **Ведь бывают такие явления, которые при исследовании первоначально можно только обозначить некоторым названием или термином, и только впоследствии возникает возможность определить их более точно**. Это происходит **в процессе последовательного формирования концепции.** Примером тому могут быть понятия "менеджмент", "качество управления", "экологический менеджмент" и др.

Существуют правила разарботки определения, которые нельзя нарушать, если стремиться к адекватности реальности, научной корректности, концептуальной значимости.

1. Правило соразмерности определяемого и определяющего понятий. Например, можно сказать, что метод управления — это вид воздействия. Но виды воздействия практически выделяются не только

по методам управления, но и по функциональному содержанию, по силе воздействия, по реакции на него. Это явно не полное определение, сопоставляющее несоразмерные понятия.

2. Правило исключения порочного круга. Согласно этому правилу нельзя определять понятие либо через само себя, либо через другое понятие, которое, в свою очередь, определяется через исходное понятие. Например, можно определить понятие управляющая система следующим образом: управляющая система — это субъект управления. Но далее понятие субъекта управления определять через понятие управляющей системы. Кстати, нередко даже в словарях по управлению такие определения встречаются.

3. Правило ясности и конкретности всех понятий определяющей части. Это значит, что в определяющей части необходимо использовать только понятия известные, практически выверенные, общепринятые, понятные. Здесь не следует использовать метафоры или слова, допускающие многозначное толкование. Например, управление — это решающий фактор прогресса. Такое утверждение можно рассматривать как некий прием убеждения, дополняющее суждение, но не как определение ключевого для исследования или концепции понятия.

4. Правило различения определения-описания и определения-предписания. Первое относится к определениям понятий, функционирующих уже в деятельности, но требующих уточнения, второе — к понятиям истин, которыми оперируют по некоторой договоренности, в определенных условиях, в рамках некоторой концепции. В исследовании систем управления такие определения-предписания необходимы, например при использовании понятий "организация управления" и "управление организацией".

5. Определять понятие можно только посредством понятий определенных, иначе говоря, известных, понятных, принятых, проверенных. Нельзя определять понятие через неизвестное понятие. Например, контроль исполнения — это мониторинг качества. А что такое мониторинг? Что такое качество? И качество чего здесь предполагается?

Определение понятий является сильным формально-логическим методом исследования, без которого невозможно построить

концепцию объяснения тех или иных явлений, невозможно отстаивать идеи и мысли, доказывать и обосновывать их значимость и практическую ценность. А все это необходимые элементы исследовательской деятельности. **Любой исследователь должен хорошо владеть этими методами.**

Особым видом использования **формально-логических операций** являются **методы мыслительного эксперимента**, который построен на **мыслительном моделировании объекта исследования и установлении характера его поведения, при изменении каких-либо параметров или условий функционирования**. При этом *эффект этих методов управления значительно повышается, если они сочетаются с* ***имитационным моделированием*** *с помощью компьютера и проигрыванием вариантов поведения объекта.*

Мыслительно-логические методы исследования в значительной своей части построены на использовании приемов **формальной логики**, которыми исследователь должен владеть в полной мере. Поэтому **к мыслительно-логическим методам исследования можно отнести и методы классификации и построения типологии, методы доказательства и конструирования гипотез, метрологические методы (методы оценок).**

В практике исследований большую роль играет также признание и понимание выводов и рекомендаций, сделанных или разработанных исследователем. Поэтому *к арсеналу методов исследования надо также отнести* ***методы научного обсуждения и научной полемики***. Многие исследовательские проекты и рекомендации возникали в результате успешно построенного и поставленного обсуждения проблем, научной полемики.

Это общая схема системы общенаучных методов исследований. Но некоторые из них требуют дополнительного объяснения и конкретизации, особенно те, которые играют наиболее существенную роль в исследовании систем управления, например, ***методы доказательства***

Понятие доказательства в практике исследовательской деятельности рассматривается как приведение любых аргументов, подтверждающих некоторое положение. Такими **аргументами могут быть факты,**

проверенные положения, заключения, точки зрения признанных авторитетов, результаты эксперимента.

Не все и не всегда можно доказать при помощи фактов, да и не всегда существуют доступные восприятию факты. В этом случае доказываемые положения выводятся из других, достоверность которых полагается установленной.

Надежность доказательства определяется аргументацией, фактологией, методологией его построения, формально-логическим следованием, готовностью к восприятию аргументов и фактов.

Доказательство — это интеллектуальная операция, состоящая в установлении истинности некоторого суждения, посредством его вывода из других суждений, истинность которых полагается установленной до этой операции и независимо от нее, а также посредством подтверждения фактами и практической деятельностью.

В зависимости от характера и особенностей предмета исследования и возможностей его проведения формы доказательства могут быть различными.

Существуют доказательства фактологические, опирающиеся в основном на фактический материал; формально-логические, главной опорой которых являются законы формальной логики; экспериментальные — построенные на эксперименте; эмпирические — опирающиеся на осмысленный и обобщенный опыт.

Корректность доказательства определяется его строением. В каждом доказательстве существует **три элемента**: **тезис, аргументы (основания), демонстрация:**

тезис *— это суждение, истинность и приятие которого устанавливается в доказательстве;* ***аргументы*** *— суждения, из которых выводится тезис;* ***демонстрация*** *— логическая форма связи названных двух элементов, обусловливающая необходимость выведения одного из другого, тезиса из аргумента.*

Существует множество разнообразных приемов и способов доказательства.

1. Доказательство от определения. Оно построено на четком определении ключевых категорий – понятий, так, чтобы определения этих категорий не вызывали сомнений относительно их адекватности реальным явлениям и практическому опыту.

2. Доказательство от обратного. Если принимаются аргументы об абсурдности обратного, противоположного доказываемому, то считается, что первоначальное суждение истинно или, по крайней мере, корректно.

3. Доказательство, построенное на анализе свойств исследуемого объекта.

4. Доказательство по принципу приведения к нелепости, абсурдности. Это прием опровержения допуска истинности, которая оказывается нелепостью.

5. Доказательство на основе классификации факторов, позволяющей установить свойства объекта исследования и причины его оригинального поведения.

6. Аксиоматическое доказательство. Первоначально формулируется аксиома — бесспорное, понятное и принятое положение, затем строится доказательство, базирующееся, как правило, на нескольких аксиомах.

7. Фактологическое доказательство, в котором главную роль играет систематизация фактов.

8. Доказательство по рабочей гипотезе или концепции (гипотетическое, концептуальное доказательство).

9. Экспериментальное доказательство. Здесь главная опора — эксперимент и его результаты.

10. Доказательство по концентрации фактов. То или иное положение, вывод или идею могут доказывать не отдельные или

разрозненные факты, а их определенная концентрация и конструкция. Факты надо накапливать и систематизировать.

Эффективность доказательства определяется правильным выбором его приемов в соответствии с предметом и характером исследования, особенностями и назначением его результатов. В обобщенном представлении эффективность доказательства зависит от множества факторов — гносеологических, методологических, социально-психологических, риторических. Но наиболее важную роль играют факторы, отражающие содержание доказательства.

Тезис или доказываемое положение должны соответствовать правилу точности формулировки, неизменности на всех этапах доказательства. В практике нередко приходится наблюдать подмену тезиса, подмену понятий истин. Эта ошибка проявляется в том, что выдвинутый в начале доказательства тезис в процессе доказательства заменяется другим. Бывает подмена количественных характеристик тезиса (доказанное относительно части объекта переносится на весь объект), подмена модальности (вероятность выдается за достоверность).

В обеспечении эффективности доказательства необходимо следовать и **правилу истинности аргументов**. Часто встречаются ошибки недоказанного основания. Одной из распространенных ошибок является "круг в доказательстве". Она заключается в замкнутости аргументов, не выходящих на тезис. Принципом, предостерегающим от этих ошибок, является принцип доказательственной независимости аргументов. Если аргументационная процедура не является логически строгим доказательством, но обеспечивает некоторому суждению определенную степень вероятности, ее называют обоснованием.

2.3.8.3 Правила составления многозвенной формулы определения понятия

Требование общности в формуле определения понятия в соответствии с логическим законом обратного отношения вступает в противоречие с требованием полноты и частично определенности. **Если в соответствии с требованием общности формула определения понятия истины должна состоять из минимально возможного числа общих признаков, то в соответствии с требованием полноты она должна включать все возможные существенные признаки предмета понятия. Эти противоречия**

преодолеваются правилом, по которому формула определения понятия может состоять из нескольких пунктов (так называемая многозвенная, или многопозиционная, формула определения понятия). При составлении первого пункта формулы определения понятия предпочтение отдается требованию общности, т. е. в этот пункт включается минимальное число обобщенных признаков предмета понятия, совокупность которых, однако, достаточна для характеристики сущности понятия. **В остальных пунктах предпочтение отдается требованию полноты**. Первый пункт составляется с такими необходимыми общими признаками, чтобы объем понятия выраженный совокупностью этих признаков, охватывал все последующие пункты, но сам первый пункт не зависит от последующих. Таким образом, ***первый пункт формулы определения понятия является главным, самостоятельным несущим основную смысловую нагрузку и не зависимый от последующих.*** Признаки предмета понятия, включаемые во все следующие за главным пункты формулы определения понятия, должны **развивать, конкретизировать и добавлять (но не изменять!) признаки**, изложенные в основном пункте формулы определения понятия. Поэтому все последующие пункты излагаются так, чтобы в них **отражалась синтаксическая и семантическая подчиненность, зависимость от первого или от предшествующих пунктов формулы определения понятия.**

Рассмотрим теперь синтаксический и семантический смысл многозвенной формулы определения понятия. Обычно вновь разрабатываемые понятия формируются по результатам законченной научной или научно-технической работы, которая содержит признаки предметов новых понятий, максимально конкретизированное для реализации образования новоых понятий или уточнения существующих. Причем вариант конкретизации зависит от условий и целей работы. Понятия, содержащиеся в работе, могут быть полезны для целой группы предметов, **поэтому в первом пункте формулы определения понятия формулируется обобщенное понятие, а в последующих — конкретизированные его варианты.** Часто разработчики понятия стремятся изложить формулу определения понятия так, чтобы она характеризовала один наиболее разработанный и оптимальный, по их мнению, предмет понятия; по существу этот предмет нередко представляет собой только один из вариантов вновь разрабатываемого понятия. Таким образом, ряд связанных в определенной последовательности отличительных признаков, которые изложены в описании понятия, должен быть сформулирован в такой редакции, **чтобы формула определения понятия характеризовала**

смысл понятия во всех возможных вариантах, точнее — в общем виде и во всех существенных частностях и дополнениях. Тогда будут выполнены предъявляемые к формуле определения понятия важнейшие и противоположные по своей сущности требования общности и полноты.

Многозвенная формула определения понятия имеет большое информационное значение, так как сущность понятия, выраженная обобщенно в первом пункте формулы определения понятия, не всегда может быть точно понята пользователем. Конкретизация предмета понятия в последующих пунктах имеет серьезное информационное значение. Из изложенного следует также, что нет необходимости записывать в формулу определения понятия дополнительные пункты, характеризующие признаки предмета не на уровне дополнительного понятия. Как правило, во многих случаях к признакам дополнительных пунктов допускается менее строгий подход. Это объясняется тем, что дополнительные пункты не сужают синтаксис и семантику понятия по сравнению с первым.

Связи между пунктами многозвенной формулы определения понятия образуются указанием в начале каждого дополнительного пункта того пункта формулы определения понятия, признак которого развивается или дополняется и которому он непосредственно подчинен, например: «Объект по п. 1, отличающийся тем, что...» или «Объект по п. 2, отличающийся тем, что...».

В дополнительном пункте формулы определения понятия, характеризующей один предмет, во всех случаях под понятием «Объект по п. 1» подразумевается полное содержание первого пункта формулы определения понятия, т. е. совокупность всех без исключения признаков, приведенных в его ограничительной и отличительной частях.

В первом пункте излагается основная, «скелетная», конструкция, в которой воплощается идея понятия, а во втором пункте приводится одна из модификаций реальной конструкции понятия.

2.3.8.4. Общие правила составления первого звена формулы определения понятия или однозвенной формулы определения понятия

Первое звено формулы определения понятия представляет собой определение предмета совокупностью его существенных признаков.

Для достижения однозначности и лаконичности рекомендуется каждое звено формулы определения понятия излагать одним предложением без точек с запятыми. Лишь в тех исключительных

случаях, когда характеризующих признаков очень много и их трудно изложить одним предложением, допускается отступление от указанного правила.

Первое звено предмета понятия делится на две основные части словом «отличающийся» («отличающееся», «отличающаяся»). **Первая, доотличительная, часть содержит известные признаки, общие с признаками прототипа, а вторая, отличительная, часть — новые признаки.** Иначе говоря, доотличительная часть формулы определения понятия характеризует известный тип (или разновидность) предмета понятия, к которому относится данное понятие и который совершенствуется данным понятием, а отличительная часть содержит признаки, которые внесены в этот тип предмета понятия.

Обе части характеристики, доотличительная и отличительная, описываются только в сочетании, причем в совокупности всех признаков, которые записаны в характеристике. Доотличительную часть формулы определения понятия можно называть **ограничительной** из тех соображений, что она содержит признаки, расширяющие смысл понятия. **Открывается ограничительная часть всегда именем понятия.**

После слов «...отличающийся тем, что, с целью...» приводится цель разработки (функционирования) понятия.

Требование общности распространяется прежде всего на первое звено формулы определения понятия, поэтому совершенно недостаточно распределить признаки понятия в зависимости от их существенности по разным звеньям; необходимо, чтобы каждый признак включался в первое звено формулы определения понятия в максимально обобщенном виде, так чтобы формулировка первого звено охватывала все возможные частные случаи понятия.

В первое звено формулы определения понятия не следует включать точное соотношение размеров объектов, точные соотношения применяемых компонентов, точные технические характеристики процесса (температуру, давление) и т. п.; указания о выполнении того или иного предмета из определенного материала, если он может быть сделан из другого материала; дополнительные факультативные признаки понятия, т. е. такие, которые должны быть изложены во втором или последующих звеньях.

Чем больше признаков записано в первом звене формулы определения понятия, тем подробнее и конкретнее его редакция. Формула определения понятия должна быть составлена так, чтобы не выходить за пределы данного понятия, не охватывать неправомерно

уже известное, а относиться к группе предметов, в которой это понятие может быть использовано.

Нельзя обобщать признаки предмета понятия настолько, чтобы формула определения понятия потеряла определенность. Для определенности понятия формула определения понятия не должна допускать произвольных догадок и предположений. **Она должна характеризовать не постановку задачи, не идею, а решение поставленной задачи в виде объекта, схемы технологического процесса и т. п.**

Редакция первого звена формулы определения понятия не должна позволять обхода формулы возможным исключением какого-либо признака из числа указанных в ней или заменой одного признака другим без дополнительного научного обоснования. По этой же причине нельзя вводить в формулу слова, выражающие не общие, а частные понятия, неправильно ограничивающие объем понятия. Например, вместо слов «привинчен», «припаян» следует применять слова «прикреплен», «соединен» и т. д., выражающие более общие понятия. Термины же, обозначающие частные понятия, следует применять лишь в тех случаях, когда они выражают суть предмета понятия, т. е. в том случае, когда нужно подчеркнуть, например, что предметы не просто скреплены один с другим, а склеены.

Нельзя вносить в формулу определения понятия указание на размеры, если сама сущность понятия не заключается в выборе определенного размера или замене одного материала другим.

Нельзя вводить в формулу определения понятия слова и выражения, вызывающие неопределенное представление, например: «толстый», «холодный», «достаточно легкий», «достаточно прочный», «небольшое количество» и т. п. Однако общепринятые выражения можно вводить в формулу определения понятия, когда ими традиционно характеризуется в той или иной области науки и техники конкретное понятие. Например, «процесс ведут при незначительном нагревании», «слабый раствор кислоты» и т. п. Указания в сравнительной степени, например «более толстый», «более прочный» и др., допускаются, **если четко показаны объекты сравнения**. Нельзя вводить в формулу определения понятия для характеристики какого-либо элемента слова «специальный», «особенный», «новый». Такие общие слова не дают никакого представления об.

Формула определения понятия должна быть изложена четко, простым и ясным языком, с применением терминов, общепринятых в научно-технической литературе. Термины, эпитеты, марки, условные сокращенные наименования, не имеющие широкого употребления или носящие рекламный характер, например «кран

АМЖК», «сплав 5793», «высокопроизводительный ткацкий станок», «сверхпрочный полиэтилен» и т. п., не должны применяться. Общеупотребительные же термины и понятия, например «способ скоростного резания металлов», «жароупорная сталь», «твердый сплав», допустимы. Не следует писать «легкий токарный станок», но общепринятые термины «быстроходный токарный станок» или «легкий сплав» можно применять. **Не рекомендуется использование местных и жаргонных терминов и оборотов.**

Признаки в каждой части формулы определения понятия (ограничительной и отличительной) приводятся в порядке их важности для раскрытия понятийного замысла.

Каждый признак, впервые упоминаемый в формуле определения понятия, вводится в нее при помощи слов «состоит из», «снабжен», «имеется» и т. д. так, чтобы было ясно, введен ли новый признак, или изменена форма известного признака, или установлена новая взаимосвязь признаков.

Нельзя в отличительной части формулы определения понятия конкретизировать признак, если он не был упомянут в ограничительной части.

Следует избегать употребления слова «применен» в формуле определения понятия при характеристике предмета как совокупности признаков, так как это слово свидетельствует не о связи с известными признаками. Не следует употреблять его и в дополнительных пунктах формулы определения понятия.

2.3.8.5. Структура первого звена формулы определения понятия или однозвенной формулы

Первое звено формулы определения понятия состоит из имени понятия, ограничительной части, цели понятия и отличительной части.

Имя понятия — это имя той истины, для которой разрабатывается понятие, характеризующее узкую область, где используется понятие.

Имя понятия должно удовлетворять следующим требованиям.

1. Являться родовым признаком по отношению к остальным существенным признакам, составляющим формулу определения понятия. Например, имя понятия «Контейнер для хранения источников гамма-излучений» находится в родовидовом отношении к признакам, указанным в формуле определения понятия (корпусу, разгрузочной трубке, центральному барабану, крышке и т. п.).
2. Быть общим для прототипа и предложения.
3. Вписываться в рубрику Международной классификации понятий.

4. Должно отвечать сущности понятия. Нельзя называть понятия «Процесс...», если оно касается объекта или вещества, и наоборот. При несовпадении имени и содержания формула определения понятия часто становится двусмысленной и создает трудности в определении объема понятия.

5. Точно соответствовать объему понятия. Например, если все отличительные признаки понятия относятся только к одному понятию «узел», то формировать понятие необходимо на этот узел, назвав понятие наименованием этого узла.

Например, совершенно правильно понятие названо «Радиационная головка аппарата для лучевой терапии», а не «Аппарат для лучевой терапии», так как понятие и, следовательно, все признаки формулы определения понятия относятся к одному узлу аппарата — радиационной головке.

Имя понятия может включать ограничительные определения, если понятия относится к ограниченному кругу тех, которые определяются общим термином.

Например, «Защитный транспортный контейнер», а не «Контейнер».

6. Должно минимальным числом слов определять общее назначение и тип предмета разрабатываемого понятия, т. е. имя должно представлять собой общее определение предмета понятия: машины, прибора, сооружения, конструктивного элемента, вещества, технологического процесса и при этом быть терминологическим или описательным. Терминологическое имя более предпочтительно. Например, лучше назвать «Тележка», чем «Устройство для перевозки грузов», название «Автопогрузчик» лучше, чем «Устройство для погрузки» и т. д.

Совершенно неудовлетворительным представляется имя понятия «Установка с источником гамма-излучения ^{60}Co». Это и не терминологическое, и не описательное название. Оно говорит о том, что установка содержит какую-то часть (источник гамма-излучения), и эта часть конкретизирована (^{60}Co). Не говоря уже о том, что такая конкретизация значительно снижает объем понятия (почему кобальтовый источник, а не любой другой, например цезиевый?), имя понятия через целое и часть не дает ему возможности быть родовым признаком по отношению ко всем остальным признакам. Поэтому формула определения понятия оказывается составлена не по логическим правилам определения. Правильным было бы одно из следующих названий: терминологическое — «Радиационная установка», описательное — «Установка для облучения».

7. Должно быть общеупотребительным, не содержать специальных, условных или сокращенных названий. Применяемые термины должны

носить общепринятый в нучной литературе характер. Неудачно, например, названо понятие «Устройство для работы с закрытыми радиоактивными препаратами». Термин «закрытый радиоактивный препарат» не употребляется в атомной технике. Остается только гадать, что имели в виду авторы описания: герметично упакованный радиоактивный препарат, радиоактивный препарат без пылеотделения или радиоактивный препарат, находящийся в защитном транспортном контейнере? Даже после изучения описания понятия эта неопределенность не проясняется.

Совершенно излишне включать в имя понятия слова «конструкция», «схема», например «конструкция манипулятора», «схема радиоприемника» и т. д.

Имя должно излагаться в единственном числе: «Радиационная установка», «Защитный транспортный контейнер» и т. д. Множественное число для имен понятий следует применять лишь тогда, когда этого нельзя избежать по правилам русского языка: «Защитные откатные ворота» и т. д.

Имя должно характеризовать, как правило, видимый предмет или видимый технологический процесс. Следует избегать включения в название цели понятия в виде технического, физического, биологического эффекта, например, нежелательно писать «Объект для повышения устойчивости деформаций штока гидравлического плунжера». Лучше написать «Гидравлический плунжер».

8. Должно быть предельно кратким и конкретным, соответствующим сущности понятия. Этому требованию не отвечает понятие, которое названо «Терапевтическое средство для подвижного облучения рентгеновскими гамма-лучами и другими видами проникающего излучения». Абсолютно не конкретно названо понятие «Пневматическое управляющее средство». Управляющее чем? Из описания понятия можно уяснить, что это управляющее средство можно использовать лишь при транспортировке шарообразной радиоактивной ампулы из места хранения к месту просвечивания, т. е. для вполне конкретных утилитарных целей. Давать широкое и неопределенное название объекта, приспособленного для столь узких целей, — неправомерно.

9. Должно использовать повествовательную без инверсий очередность слов. Например, «Поршневой насос», а не «Насос поршневой», «Транспортный защитный контейнер», а не «Контейнер защитный транспортный» и т. д.

10. При описательном названии понятия должно включать частицу «для». Необходимо писать, например, «Средство для перемешивания», а не «Средство перемешивания».

11. Имя дополнительного понятия должно в точности соответствовать основному.
12. Недопустимо указывать в имени отличительные признаки предмета понятия.

Ограничительная часть открывается <u>именем понятия</u>, которое может быть конкретизировано, но лишь за счет терминов, размещаемых вслед за собственно именем понятия. Непосредственно после конкретизированного имени в первое звено формулы определения понятия включают совокупность существенных признаков, общих для предложения и прототипа. Эта совокупность описывается в формуле определения понятия во взаимосвязи и вводится в нее при помощи слов «содержание», «включающий», «включающее», «состоящий», «методом», «на основе», «вводят», «берут» и т. п. Выражения «включающий» и «содержащий» в ограничительной части указывают, что помимо существенных признаков, перечисленных в формуле определения понятия, предмет понятия имеет другие признаки, которые не упомянуты в формуле определения понятия. При использовании в ограничитечьной части таких выражений, как «состоящий из» или «составленный из», предполагается, что предмет понятия характеризуется только совокупностью признаков, которая перечислена в формуле определения понятия.

В ограничительной части первого звена формулы определения понятия не обязательно указывать все признаки предмета понятия, общие для прототипа и предложения. Иногда бывает достаточно указать существенные признаки, взаимосвязанные с новыми признаками предложения, и существенные признаки, без которых данное понятие не может быть использовано, а признаки, не взаимосвязанные с новыми признаками и очевидные для данного предмета понятия, можно опустить. При этом чем известнее предмет понятия, тем больше признаков можно опустить, если они не связаны с новыми.

В ограничительной части формулы определения понятия не допускается указание на признаки, общие с несколькими прототипами или с так называемым сборным прототипом. Не рекомендуется также вводить в ограничительную часть формулы определения понятия качественные характеристики или пояснения к принципу действия объекта или физическим принципам, положенным в основу процесса, и т. д.

Нередко допускается ошибка, которая заключается в том, что в доотличительной части формулы определения понятия отмечаются признаки прототипа, не являющиеся фактически признаками формулируемого определения понятия. Например, включаются такие

элементы прототипа, которые были исключены из него благодаря новым, отличительным особенностям предмета понятия или прямо заменены новыми особенностями. В таком случае формула определения понятия не будет соответствовать предмету понятия.

Если понятие не имеет прототипа более близкого, чем тот, который характеризуется самим наименованием понятия, или если сущность понятия может быть использована в нескольких прототипах, общим признаком которых является только наименование, то в формуле определения понятия ограничительная часть может отсутствовать, т. е. она будет представлена самим наименованием понятия. Это, однако, вовсе не означает пионерское понятие, так как само название в большинстве случаев и характеризует собой прототип, например «Рентгеновский аппарат».

Цель понятия формулируется за ограничительной частью формулы определения понятия после слов «отличающийся (отличающаяся, отличающееся) тем, что...». Она указывает на эффект, который может быть получен в результате использования понятия. Этот эффект должен содержать указание на новое свойство (или свойства) предмета, которое причинно обусловлено всей совокупностью существенных признаков, включенных в формулу определения понятия.

Указание на эффект в формуле определения понятия обосновывает существенность отличий предложения и правомерность разработки понятия.

Однако, хотя это целесообразно для понимания смысла понятия и оценки его уровня, указание на цель не влияет на объем понятия.

Поскольку указание цели в формуле определения понятия, принятой в науке, имеет значение обоснования понятия, следует стремиться как можно лучше оправдать это значение, насколько позволяет лаконичность формулы определения понятия. Во многих случаях достичь этого бывает весьма трудно, поэтому следует записывать цель понятия в формуле определения понятия в виде непосредственного технического, физического, химического или другого результата, используемого в понятии: «с целью разгрузки вала от изгибающих моментов», «с целью исключения коррозии», «для достижения газонепроницаемости» и т. д.

Если подробного указания первопричины полезности окажется недостаточно, можно это указание развить: «с целью разгрузки вала от изгибающих моментов для снижения веса его».

Указание цели в формуле определения понятия необходимо для обоснования целесообразности понятия, поэтому не следует характеризовать цель разработки понятия общими словами, не

отражающими новых функциональных или технологических свойств предмета. Например, не рекомендуется употреблять выражения «с целью улучшения конструкции», «с целью совершенствования процесса», «с целью повышения качества продукции» и т. п. ***Цель понятия как можно более четко должна раскрывать существенные отличия***. Ее следует излагать лаконично. Указание на цель понятия не вводится в формулу определения понятия в тех случаях, когда само имя содержит целевую установку, например «Процесс повышения четкости изображения». В данном случае цель понятия истины сформулирована в самом наименовании.

Отличительная часть первого звена формулы определения понятия приводится после изложения цели понятия. **В отличительной части характеризуются существенные отличия предмета понятия**. **Эти отличия излагаются во взаимосвязи и связи с существенными признаками ограничительной части.** При этом недопустимо указание на какую-либо связь нового признака с известным, если этот известный признак не отражен в ограничительной части первого пункта формулы определения понятия. В отличительной части первого звена формулы определения понятия содержатся новые существенные признаки предмета понятия, отличающие его от прототипа, которые в совокупности с признаками, приведенными в ограничительной части формулы определения понятия, необходимы и достаточны для решения задачи с достижением цели понятия.

Существенные отличия понятия должны характеризоваться в отличительной части формулы определения понятия не постановкой задачи, а конкретными средствами ее решения. Например, нельзя, определяя в понятии существенную новизну конструкции самолета, в отличительной части формулы определения понятия указывать, что крыло имеет профиль, который уменьшает сопротивление воздушному потоку. В данном случае необходимо дать конкретную форму профиля крыла, с помощью которого достигнут требуемый эффект.

Пример.

«Винтовой конвейер, содержащий грузовые тележки, каждая из которых несет платформу, взаимодействующую с приводным винтом, отличающийся тем, что, с целью обеспечения возможности транспортирования грузов по лабиринтной трассе с перпендикулярными поворотами, винт каждого прямолинейного участка трассы снабжен свободно перемещающейся по нему гайкой с выступом, входящим соответственно в один из перпендикулярно расположенных пазов платформы грузовой тележки.»

В этой формуле определения понятия каждый отличительный признак связан или с другим новым признаком, или с уже упомянутым признаком ограничительной части.

2.3.8.6. Структура дополнительных звеньев формулы определения понятия

Как уже указывалось, дополнительные (зависимые) звенья характеризуют различные конкретные формы формирования предмета понятия (обычно оптимальные) и содержат признаки, которые развивают и уточняют признаки первого звена.

Признаки, приведенные в дополнительных звеньях, развивают и конкретизируют как новые существенные признаки понятия, указанные после слова «отличающийся» в первом звене формулы определения понятия, так и известные, включенные в ограничительную часть. Развитие признаков понятия ограничительной части допустимо при условии, если модификации понятия, охарактеризованные с привлечением этих признаков, могут использоваться только в совокупности с новыми существенными признаками предмета понятия.

Таким образом, в дополнительных звеньях многозвенной формулы определения понятия определяются понятия, подчиненные понятию предмета понятия в первом звене. Это означает, что дополнительные звенья не расширяют объем понятия, а лишь раскрывают его.

Ограничительная часть дополнительного звена формулы определения понятия состоит из имени понятия, часто укороченного до одного или двух первых слов и ссылки на первый или предыдущие дополнительные пункты. Например «Объект по п. 1» или «Процесс по п. 1 и 2».

Указание на цель понятия не обязательно. Оно проставляется, если цель введения новых признаков в дополнительное звено отличается от цели, которую решает предмет понятия, изложенный в главном пункте.

Отличительная часть начинается словом «отличающийся» (отличающаяся, отличающееся), вслед за которым, если цель не указывается, приводится новая совокупность признаков, характеризующих дополнительное понятия.

Пример.

«1. Средство для радиационно-химических исследований в газовой, паровой и жидкой фазах, состоящая из последовательно соединенных пульта питания, коммуникационных труб, реакционной зоны и пульта управления и регистрации, отличающаяся тем, что, с целью

повышения точности и воспроизводимости проведения экспериментов при высоких давлениях и температурах, на отводящей трубе установлен регулятор давления «до себя».
2. Средство по п. 1, отличающаяся тем, что коммуникационные трубы выполнены токопроводящими и на концах их установлены электроизолирующие разъемы.»

2.3.8.7. Формула определения дополнительного понятия

Формула определения дополнительного понятия включает в себя:
- название дополнительного понятия, которое берется из формулы определения основного понятия;
- ссылку на описание основного понятия вместо перечисления ограничительных признаков. Это возможно, потому что все признаки основного понятия обязательно должны быть включены в ограничительную часть формулы определения понятия дополнительного понятия;
- характеристику цели понятия, соответствующую тому совершенствованию, которое вносится в основное понятие;
- существенные отличительные признаки, которые привносятся для совершенствования основного понятия.

Если понятие совершенствует более раннее понятие, являющееся в свою очередь совершенствованием еще более раннего, то в формуле определения понятия указывается имя понятия того понятия, которое непосредственно совершенствуется данным.
Таким образом, правила составления формулы определения дополнительного понятия почти не отличаются от правил составления второго и последующих звеньев многозвенной формулы определения понятия. Различие заключается лишь в том, что в зависимом понятии вместо ссылки на первое или предыдущее звено в многозвенной формуле определения понятия дается ссылка на понятие, так как дополнительные звенья в многозвенной формуле определения понятия по существу и характеризуют собой дополнительные понятия.

2.3.8.8. Особенности составления формулы определения понятия на различные предметы понятия

Особенности составления формулы определения понятия на объект. В формуле определения понятия-объекта (объектного понятия) **признаки объекта понятия отражаются в статическом, а не динамическом состоянии**. Поэтому в формуле определения понятия не

должно быть глаголов изъявительного наклонения, выражающих незавершенное действие.
Пример.
«Буровое дисковое долото, включающее лапы и два смонтированных на оси с помощью подшипников качения и смещенных по отношению друг к другу диска, отличающееся тем, что, с целью повышения стойкости долота, внутренняя обойма каждого периферийного подшипника выполнена в виде замковой втулки, закрепленной неподвижно на оси.»
Формула определения понятия-объекта должна характеризовать объект понятия не эффектами, а признаками понятия, так как эффекты являются лишь следствием признаков понятия и находят свое отражение в изложении цели понятия.
Когда в формуле определения понятия-объекта признаки понятия характеризуют не конкретное выполнение элементов или объекта в целом, а лишь наличие элементов и связи между ними, его изображают в виде схемы, в которой реальные элементы представлены в виде общепринятых условных обозначений, а связи между элементами — в виде соединительных линий.
Схемные изображения объекта понятия подразделяются на виды и типы.
В зависимости от видов элементов и связей, характеризующих объект, схемы подразделяются на следующие виды: электрические, гидравлические, пневматические, оптические, комбинированные.
А в зависимости от специфичности характеристики объекта понятия, схемы подразделяются на следующие типы: функциональные, структурные, принципиальные (полные, развернутые), общие схемы, схемы соединений (монтажные), схемы подключения, схемы расположения. Один и тот же объект может быть представлен одновременно несколькими схемами различных видов и (или) различных типов. Например, объект, в состав которого входят элементы и связи разных видов, характеризуемый с какой-то одной точки зрения, может быть изображен в нескольких схемах соответствующих видов одного типа (схема электрическая принципиальная и схема гидравлическая принципиальная). Этот объект может быть изображен в виде комбинированной схемы, содержащей элементы и связи разных видов (схема электрогидравлическая принципиальная). А объект, состоящий из элементов и связей одного вида, но характеризуемый с различных точек зрения, может быть изображен в одном графическом документе несколькими схемами этого вида, но различных типов, например схема электрическая принципиальная и схема электрическая монтажная.

2.3.8.9. Использование функциональных признаков в формуле определения понятия для характеристики предметов понятий

Как указывалось, в первое звено формулы определения понятия включается как можно меньше максимально обобщенных признаков предмета понятия, совокупность которых, однако, достаточна для характеристики сущности понятия.

В некоторых случаях можно добиться предельной степени обобщения признака использованием вместо него любого другого признака, выполняющего ту же функцию, поскольку свойства предмета понятия, понятийный уровень решения совершенно **не зависят от формы признака, а только от выполняемой им функции**. В этом случае признак можно характеризовать посредством выполняемой им функции. Такие ***признаки, которые характеризуются не по конкретной форме выполнения, а лишь по их функциональному назначению, называются функциональными признаками***. В этом случае в формуле определения понятия структурные признаки выражаются при помощи словосочетании «механизм..., осуществляющий определенную функцию,» например: «механизм для подъема источника излучения»; «средство для...», например: «средство для передачи вращательного движения»; «приспособление для...» и т. п.

К функциональным признакам не относят терминологические признаки, такие, как привод, облучатель, захват и т. п., поскольку под такими признаками подразумевают определенный в общем виде тип вполне конкретных известных элементов. Не относятся к функциональным признакам и такие, как, например, «кран поворотный на 360°», так как в этом случае подразумеваются определенные конструктивные признаки, дающие возможность крану поворачиваться на 360°, а не функциональные.

К функциональному признаку предъявляются следующие требования.

1. Форма выполнения функционального признака никак не влияет на предмет понятия, понятийный уровень решения.
2. Функциональный признак представляет собой предельное обобщение нескольких конкретных форм воплощения при помощи признаков I, II и III групп.
3. Разработчику определения понятия совершенно ясно, какие конкретные средства и в какой форме могут быть использованы для реализации функционального признака в предложенном понятии.

4. Как правило, функциональный признак используется по своему известному назначению. Если же назначение функционального признака ново, то конкретная форма выполнения функционального элемента предложенного понятия обязательно раскрывается в последующих пунктах формулы определения понятия.
5. Если функциональный признак находится в отличительной части формулы определения понятия, то одно только его наличие в сочетании с другими признаками, а не конкретные формы его выполнения, обеспечивает существенные отличия понятию.

Но, поскольку границы понимания функциональности признака условны и не столь определенны, использование в формуле определения понятия функциональных признаков часто служит предметом разногласий между разработчиком понятия и рецензентом этого понятия, требующим конкретизировать характеристику внешними, видимыми признаками. Критика содержания определения понятия часто мотивируется тем, что представленная формула определения понятия неопределенна, что характеристика понятия не решает научно-технической задачи. Включать же в основное звено формулы определения понятия конкретные решения признака, приведенные в виде примеров в описании или предусмотренные в последующих пунктах, нежелательно для разработчика определения понятия, так как эти решения не исчерпывают всех возможных вариантов понятия, и формула определения понятия с такими признаками будет недостаточно широкой. Возможность или невозможность исполнения функционального признака специалистом тоже не всегда определяется объективно или с достаточной бесспорностью. Иначе говоря, **оценка возможности применения того или иного функционального признака в формуле определения понятия субъективна.** Нельзя предвидеть мнение рецензента понятия по каждому частному вопросу, поскольку практика отношения к функциональным признакам различна в разных областях науки и техники и даже у разных рецензентов понятия одной и той же области науки и техники. **Поэтому, если составить достаточно широкую формулу определения понятия без этих признаков не представляется возможным, следует основное, главное, звено формулы определения понятия истины изложить с использованием функциональных признаков, а второе и последующие звенья представить в более конкретном выражении.**

Использованию функциональных признаков для характеристики предмета понятия не придается должного значения. Как правило, эти признаки используются лишь в крайнем случае, когда без них невозможно обойтись. А между тем ***функциональный признак — это***

мощное средство выражения понятийной идеи. И дело здесь не только в том, что использование этих признаков позволяет сформулировать предмет понятия в наиболее общем виде (хотя это тоже очень важно для научного значения формулы определения понятия). **Каждое определенное понятие — новое продвижение по пути научно-технического прогресса, и формула определения понятия должна показывать это. Поэтому она должна описывать не частный случай воплощения понятийной идеи, а воплощение ее в общем виде.** Для этих целей иногда без функциональных признаков не обойтись. ***В этом большое научно-техническое значение функциональных признаков , они помогают нам понять и осознать сущность понятийной идеи, значение каждого определения понятия, его место в научно-техническом прогрессе.*** Естественно, из этого не следует, что функциональные признаки нужно применять везде. Их использование допустимо только тогда, когда они удовлетворяют указанным выше пяти требованиям.

Недостаточно широкое использование функциональных признаков в формуле определения понятия приводит к тому, что формулы определения понятий некоторых понятий характеризуются частными признаками и, следовательно, информируют научно-технический мир о создании в какой-то мере утилитарного, частного, хотя и нового, передового научно-технического решения.

Если формула определения понятия выражена частными признаками, то случается, что такое понятие проходит незамеченным, используется только в месте его разработки и не оказывает того влияния на научно-технический прогресс, которое оно могло бы оказать, если бы было образовано в более общих выражениях. Это происходит потому, что частное понятие легко использовать только в тех конкретных условиях, для которых оно разрабатывалось, а таких конкретных условий в другом месте может и не быть. **Перейти от одного частного понятия к другому можно только через общее решение, но для этого уже требуется научное творчество.** Вот и получается, что если в двух разных местах, каждое из которых имеет свои специфические условия, имеется необходимость в решении какой-то научно-технической задачи и эта задача решена в одном из этих мест на уровне разработки нового понятия, но выражена в формуле определения частными признаками применительно к своему случаю, то информационное значение такой формулы определения понятия может оказаться недостаточным, и в другом месте могут не заметить, что решена по сути дела стоящая перед ними задача, только выражена она в другом варианте. Если бы решение этой задачи было выражено в более общем виде, то применять его к частному, конкретному, случаю было бы

делом простым, не требующим научного творчества. **Отсюда видно, насколько велико информационное значение функциональных признаков при использовании их в формуле определения понятия.**

При решении научно-технической задачи, необходимость в которой давно назрела, редко бывает так, что вместе с общим решением исследователь сразу дает и оптимальное частное решение. И не всегда исследователь, решивший общую задачу, способен найти ее лучшее частное воплощение. **Анализ понятия показывает, что если в формуле определения понятия отражено общее решение, то скоро будет образован ряд ценных понятий, конкретизирующих это решение применительно к разным целям**. Если решена какая-то проблема и в формуле определения понятия отражено частное решение, то появление «свиты» у этого понятия происходит намного медленнее.

Итак, **необходимо шире использовать функциональные признаки при формулировке предмета понятия, что поможет отражать воплощение научной идеи в более общем виде.** Что же препятствует использованию функциональных признаков в формуле определения понятия? Такими препятствиями являются две причины:

1) использование функциональных признаков в формуле определения понятия значительно увеличивает объем понятия, и это якобы перекрывает дальнейшее научное творчество по данной проблеме;

2) при помощи функциональных признаков зачастую не удается выразить научно-техническое решение, которое можно было бы осуществить без дальнейшего научного творчества.

Использование функциональных признаков в формуле определения понятия не только не препятствует развитию научного творчества, а, наоборот, способствует ему, так как помогает сформулировать сущность понятия в общем виде и тем самым дает идею для новых понятий.

О второй причине уже говорилось. Функциональный признак может быть использован в формуле определения понятия, если без дополнительного научного творчества совершенно ясно, какие конкретные средства и в какой форме могут быть использованы для его реализации. Эти формы выполнения функционального признака могут быть или общеизвестны, или вытекать из характеристики самого функционального признака, или быть описаны в дополнительных пунктах формулы определения понятия. В противном случае это не функциональный признак, а попытка охарактеризовать предмет понятия достигнутым эффектом.

2.3.8.10. Отражение в формуле определения понятия альтернативных признаков

При составлении формулы определения понятия разработчик понятия нередко сталкивается с возможностью использования различных элементов для характеристики одного признака понятия в формуле определения понятия без какого-либо ущерба для понимания понятия. Это возможно, если различные элементы могут взаимно заменять друг друга в научно-техническом решении, так как являются эквивалентами. Чаще всего с этим приходится сталкиваться при получении химических веществ, когда приходится характеризовать сами вещества или процессы их получения, используя для этих целей различные компоненты реакции: растворители, восстановители, окислители, катализаторы, а также различные значения радикалов одной структурной формулы. Если, стремясь к однозначности формулы определения понятия, при характеристике подобного признака указывать только один из возможных для его осуществления элементов, то неоправданно сужается объем понятия, так как создаются предпосылки для обхода формулы определения понятия без дополнительного научного творчества заменой такого элемента эквивалентным.

Кроме того, включение в разрабатываемое понятие только одного элемента из нескольких возможных может дать рецензенту повод для вынесения отрицательной рецензии на том основании, что разработанное понятие якобы не выполнено на достаточном научном уровне разработки понятий, так как один из описанных новых существенных признаков с достаточной очевидностью можно заменить на другой с достижением того же научного эффекта. Чтобы искусственно не сужать объем понятия, необходимо в формуле определения понятия охарактеризовать все элементы признака. Такие признаки называют *альтернативными*. При использовании в первом звене формулы определения понятия нескольких альтернативных признаков понятие остается одно и единство понятия не изменяется. Это подтверждается и тем обстоятельством, что введение альтернативных признаков не сужает, как это всегда происходит при нарушении единства понятия, а, наоборот, расширяет объем понятия. Объем понятия по формуле определения с несколькими признаками, связанными союзом «или», равен сумме объемов понятия по нескольким формулам определения, включающим эти признаки по отдельности.

Способы выражения альтернативных признаков в формуле определения понятия можно свести к следующим.

1. Если подлежащие включению в формулу определения понятия альтернативные признаки, будучи научными или техническими эквивалентами, входят в объем обобщающего их родового понятия, то в формулу определения понятия должен быть внесен признак, определяемый этим родовым понятием, с последующим приведением указанных эквивалентов со словом «например» и перечислением их через союз «или».

Пример.

«..2. Процесс по п. 1, отличающийся тем, что в качестве десульфирующих соединений используют соли или окислы тяжелых металлов или окислители, например перекись водорода или натрия, азотистую кислоту.»

Во втором звене формулы определения понятия определено родовое понятие — окислители — для альтернативных признаков перекиси водорода или натрия, которые вводятся в формулу определения понятия при помощи слова «например».

Следует отметить, что если в понятии возможно обобщение альтернативных признаков, то иногда при небольшом их числе альтернативы вообще можно избежать включением обобщающего признака в первое звено формулы определения понятия, а альтернативных — в последующие.

2. Если подлежащие включению в формулу определения понятия альтернативные признаки не могут быть обобщены из-за того, что такое обобщение неправомерно или не приводит к достижению поставленной цели, то они могут быть включены в формулу определения понятия при помощи союза «или», если являются эквивалентами.

Пример.

«Процесс получения лекарственного средства из иловых грязей материковых озер путем экстракции с последующим отгоном растворителя и стерилизации, отличающийся тем, что, с целью повышения биологической активности целевого продукта, экстракцию проводят смесью этилового спирта и диэтилового эфира в соотношении 1 : 5 или смесью этилового спирта и дихлорэтана в том же соотношении.»

3. Если подлежащие включению в формулу определения понятия альтернативные признаки не могут быть обобщены и настолько многочисленны, что многократное использование союза «или» при введении их в формулу определения понятия приводит к чрезмерному ее усложнению, то в этом случае для введения альтернативных эквивалентных признаков в формулу определения понятия следует воспользоваться выражением, при котором альтернативные эквивалентные признаки вводятся в формулу определения понятия

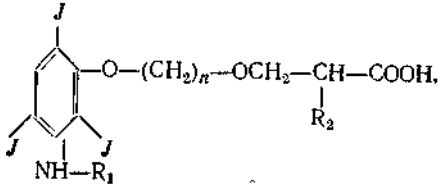
простым перечислением вслед за словами: «выбранное из группы, содержащей...» или «выбранного из ряда, состоящего из...».

Пример.

«Процесс получения полимеров акриловых и метакриловых соединений путем полимеризации соответствующих мономеров или их смесей в присутствии катализатора, отличающийся тем, что, с целью упрощения и интенсификации технологического процесса, в качестве катализатора применяют оловоорганическое соединение, выбранное из группы, состоящей из оловоорганических галогенидов, содержащих станкоксановую связь, и солей двухвалентного олова и органических кислот.»

4. Если альтернативные признаки являются конкретными заместителями радикалов группы химических соединений, описываемых одной структурной формулой, и все представители этой группы (с указанными заместителями как альтернативой) могут быть получены способами-аналогами и проявляют одинаковые свойства, определяющие их одинаковое назначение, то они в формулу определения понятия вводятся при помощи союза «или».

Пример. «3-[(3-ациламино-2,4,6-трииодфенокси)-алкокси]-2-алкилпропионовая кислота общей формулы



где R1 — ацетил или пропионил; R_2 — метил или этил; *n* — целые числа 2, 3 или 4.»

5. Не допускается включение в формулу определения понятия альтернативных признаков, если:

а) альтернативные признаки, выражающие конкретные эквиваленты, явно сужают границы понятия, т. е. не исчерпывают научных средств, возможных для решения данной задачи;

б) альтернативные признаки являются по существу эквивалентными, но один из них выражен абстрактным понятием, например «машина или подобное средство», так как под последним можно подразумевать какое угодно средство, что неправомерно расширяет объем понятия.

Разделительный логический союз «или» имеет два значения: соединительно-разделительное и исключающе-разделительное.

Союз «или» имеет соединительно-разделительное значение, если связанные им признаки не исключают друг друга, т. е. существуют в

объекте одновременно. Например, если в рассмотренном процессе для очистки газа могут одновременно применяться все указанные вещества, то союз «или» имеет в данной формуле понятия соединительно-разделительное значение. Если же использование одного вещества исключает другое, союз «или» применяется как исключающе-разделительдый. В зависимости от значения этого союза объем понятия, выраженный формулой определения понятия, может быть шире или уже. Чтобы исключить неточность в толковании объема понятия по формуле определения понятия, содержащей альтернативные признаки, целесообразно для выражения соединительно-разделительной связи использовать сложный союз «и (или)», а для исключающе-разделительной — союз «или».

2.3.8.11. Отражение в формуле определения понятия математических зависимостей

Следует сказать, что введение математических зависимостей в первое звено формулы определения понятия снижает ее значение, поскольку, во-первых, на практике трудно убедиться, что используется именно записанная в формуле определения понятия зависимость, а не другая; во-вторых, можно применить другие математические зависимости и обойти формулу определения понятия звено Поэтому следует составлять первое звено формулы определения понятия без использования математических зависимостей, которые могут быть отражены в дополнительных звеньях формулы определения понятия.

Однако иногда новые понятия, разработанные с использованием глубоких теоретических исследований и отвечающие принятым критериям научной новизны, не могут быть охарактеризованы без использования математических формул, так как их существенные отличия заключаются в новых математических зависимостях между параметрами изучаемого объекта. В принципе расчетная формула может быть включена в формулу определения понятия , если она является не единственным отличительным признаком понятия, а непосредственно связана с другим (или другими) отличительным признаком. **Под <u>расчетной формулой</u> понимается выражение в виде уравнения или неравенства, служащее для расчета какой-либо искомой конкретной величины, получаемой одноразовой подстановкой соответствующих величин в правую или левую часть уравнения или неравенства.**

Математические зависимости в формуле определения понятия должны использоваться для выражения реальных признаков субъекта.

Например, если субъект понятия — объект, с помощью математических зависимостей можно описывать форму выполнения его отдельного элемента или объекта в целом, соотношение размеров, а в отдельных случаях и абсолютные размеры его элементов.

Во всех случаях символы и обозначения, входящие в математическую зависимость, должны быть расшифрованы в формуле, как это сделано в приведенных ниже примерах.

Иногда при характеристике истин, описывающих процессы контроля, измерения и т. п. в формуле определения понятия используют математические зависимости, определяющие методы расчета как конечную операцию процесса. Если эта расчетная операция — единственная отличительная особенность процесса, он не может быть признан понятием, толкующем его однозначно. Если же расчетная операция, характеризуемая определенной математической зависимостью, — не единственная отличительная особенность процесса, указанная математическая зависимость может быть введена в формулу определения понятия для более полной характеристики процесса как законченной последовательности операций.

Пример.

«Криостат.

Криостат, состоящий из сосуда Дьюара, заполненного хладагентом и опущенного в сосуд хвостовика, соединенного с экспериментальной камерой в виде обоймы с образцами, отличающийся тем, что, с целью получения линейной зависимости температуры обоймы от времени без внешнего регулирования, хвостовик выполнен с переменным по его длине сечением, площадь которого $\delta(x)$ определяют в зависимости от его длины (L, M) по следующему соотношению:

$$\frac{\sigma(x)}{L} = \frac{2aS}{\lambda}\left(\frac{x}{L}\right)^{3/2},$$

где

$$L = \frac{aSv}{2\pi r^2 \beta\rho}\left(\frac{\theta_0 - \theta_x}{V} - \frac{cM}{aS}\right)^2;$$

a — коэффициент теплоотдачи с поверхности обоймы, *ккал/(м²·град·ч); S* — открытая поверхность обоймы, *м²; r* — внутренний радиус сосуда Дюоара, *м; c* — удельная теплоемкость материала обоймы, *ккал/(кг·град); M* — масса обоймы, *кг;* λ—-коэффициент теплопроводности материала хвостовика, *ккал/(ч·м·град);* β — удельная теплота испарения хладагента, *ккал/кг;* ρ — удельный вес хладагента, *кГ/м³;* θ_0, θ_x — температура окружающей среды и

хладагента, °С; v — требуемая скорость роста температуры обоймы, °С/ч; x — текущая координата от нижней точки хвостовика, *м.»*

2.3.8.12. Теория эквивалентов и формула определения понятия

Объем понятия характеризуется кругом предметов, на которые распространяется действие понятия, поскольку эти предметы содержат совокупность всех признаков, указанных в первом звене формулы определения понятия. Чем больше признаков в первом звене и чем каждый из них подробнее охарактеризован, тем меньше круг поредметов, содержащих эту совокупность признаков, и у́же объем понятия.

При толковании формулы определения понятия возникают в основном два практических вопроса:

а) распространяется ли объем понятия на предметы, которые являются дальнейшим усовершенствованием предмета понятия, но не обладают существенными отличиями;

б) можно ли считать понятие общепризнаваемым, если в реализуемом на практике предмете один или несколько признаков понятия заменены взаимозаменяемыми элементами.

На первый вопрос можно ответить утвердительно. С точки зрения формальной логики, объем понятия, характеризующегося большим по сравнению с формулой определения понятия количеством признаков, у́же объема предмета понятия и является его частью, так как в понятие реализованного предмета входят все без исключения признаки понятия.

Для ответа на второй вопрос следует обратиться к доктрине эквивалентов.

Из этой теории вытекает, что если два объекта (средства) выполняют одну и ту же работу одним и тем же способом и дают по существу один и тот же результат, то они одинаковы, даже если отличаются одно от другого по имени, виду и форме. Если реализуемый объект отличается от объекта понятия отсутствием некоторого несущественного признака, включенного в первое звено формулы определения понятия, то по смыслу указанного разъяснения вытекает, что и в таком объекте понятие имеет право на жизнь.

При правильно составленной формуле определения понятия, особенно ее первом звене, не должны возникнуть вопросы, требующие применения теории эквивалентов. Однако на практике еще часто образовывают понятия с формулами определения понятия, где неполно

отражен объем понятия. Для того чтобы от этого не страдали разработчики понятий, и используется теория эквивалентов.

2.3.8.13. Выбор вида предмета понятия для отображения его в формуле определения понятия

В большинстве случаев сущность понятия заключается в модификации или образовании понятия определенного вида, т. е. понятие объекта, процесса, или вещества. В этом случае выбирать вид предмета понятия не приходится — его надо определить на основе анализа существенных признаков понятия.

Но так бывает не всегда. В так называемых стыковых случаях разработанное научно-техническое решение в равной степени может быть охарактеризовано в одном случае признаками объекта или процесса, в другом — процесса или вещества. Соответственно и описание понятия может ориентировано или на объект, или на процесс, или на овеществленный объект. При разработке таких технических решений перед исследователем встает вопрос выбора вида предмета понятия.

Когда разработано несколько научно-технических решений, то вопрос решается следующим образом. Если эти научно-технические решения относятся к разным видам предметов, но служат единой цели и могут быть применены лишь совместно, формулируются цель и задача на комплексное понятие.

Если научно-технические решения могут применяться не только совместно, но и раздельно, — осуществляются описания несколько самостоятельных понятий.

Если разработано одно научно-техническое решение, которое, однако, может быть отнесено к разным видам предметов, то необходимо выбрать один предмет понятия и на него разрабатывать понятие.

Выбор вида предмета понятия определяется следующими факторами.

1. Предмет понятия должен быть выбран так, чтобы максимально проявилась существенность отличий предложенного понятия от известных. Большое значение в данном случае имеет возможность выбора в качестве прототипа понятия широкой известности.
2. Предмет понятия должен позволить составить формулу определения понятия с использованием наименьшего возможного числа максимально обобщенных признаков.
3. Предмет понятия должен обеспечивать охват формулой определения понятия наибольшего круга предметов.

4. Предмет понятия должен обеспечивать большую по сравнению с другими возможными объектами вероятность его использования и быть с научной точки зрения наиболее перспективным.

2.3.8.14. Единство понятия

Принцип единства понятия заключается в том, что в один документ, в том числе и в описание понятия, не могут быть включены два или более независимых понятия.

Другими словами, каждое отдельное описание понятия и каждый отдельный документ должны быть объединены единым понятийным замыслом и относиться только к одному решению, одной задаче.

Поскольку сущность понятия выражается в его формуле определения понятия, то соблюдение или нарушение требования единства понятия определяется по формуле определения понятия. Поэтому вопрос единства понятия рассматривается на основе анализа формул определения понятия.

Единство понятия считается соблюденным, если научно-техническим решением одной единственной задачи является один предмет понятия (объект, процесс или вещество); несколько предметов понятия, если они служат единой цели и могут быть использваны для описание понятия лишь совместно (комплексное понятие).

Одним предметом понятия считается описанное разработчиком целое, т. е. единство частей, существующее только благодаря их взаимосвязи, которая носит устойчивый характер и обеспечивает появление у целого новых свойств, не присущих разобщенным частям. **Части целого называются признаками Понятия целого и части — это соотносительные понятия. Любая составная часть целого является в то же время целым по отношению к составляющим ее частям**.

Поэтому описания понятий могут быть представлены для рецензирования на целое как на систему, так и на отдельные ее части.

Единство понятия не нарушается, если в описании и формуле определения понятия (в первом звене формулы определения) описан один предмет как целое совокупностью его существенных признаков (главных частей целого) и дополнительно охарактеризовано содержание этих частей, т. е. указаны их существенные признаки (части частей целого).

Такое отображение предмета в многозвенной формуле определения понятия допускается при условии, что части целого или части частей целого, отраженные в дополнительных звеньях формулы определения

понятия, не являются сами по себе понятиями, которые могут быть использованы отдельно, самостоятельно, или в других предметах.

Единство понятия не нарушается также в том случае, если отличительный существенный признак (признаки) предмета выражен в первом звене формулы определения понятия общим понятием и, кроме того, в дополнительных звеньях формулы определения указаны его видовые понятия, т. е. указаны технически эквивалентные значения этого существенного признака.

Единство понятия соблюдено, если существенный признак предмета понятия выражен в описании (в описании и формуле определения понятия) перечислением эквивалентов при условии, что они не могут быть выражены обобщающим их понятием.

Единство понятия будет нарушено всегда, если в описании и в формуле определения понятия отображена искусственно собранная в кажущееся целое сумма отдельных предметов (элементов, средств, веществ и т. п.), не взаимосвязанных для достижения общего положительного эффекта, свойственного только целому.

Указанные требования, предъявляемые к формуле определения понятия для решения вопроса о том, относится ли предложение к одному решению одной научно-технической задачи, можно дополнить следующими пояснениями.

Для соблюдения требования единства понятия в первое звено формулы определения или в формулу определения, состоящую из одного звена, должны быть включены только такие отличительные признаки предмета, которые имеют между собой обязательную связь и без которых понятия нет.

Требование единства понимается не буквально в том смысле, что формула определения понятия в целом, в том числе и многозвенная, должна характеризовать только одно понятие, один предмет, а в том, что каждый последующий пункт должен характеризовать не более как дополнительное понятие, т. е. такое, которое не может существовать без главного понятия или без главного не обладает существенными отличиями. Иначе говоря, единство понятия соблюдается, если признаки, отмеченные в дополнительных звеньях формулы определения понятия, конкретизируют или дополняют отличительные признаки первого звена формулы определения понятия.

Однако вполне возможны и такие случаи, когда развиваются доотличительные признаки и при этом единство не нарушается. Это бывает тогда, когда развитие доотличительных признаков не может быть выполнено без отличительных или имеет существенное значение только при наличии отличительных признаков.

Ошибочно считать требование единства удовлетворенным, если второй или другие звенья развивают какой-либо признак главного звена, в том числе и доотличительного, без выполнения изложенных выше условий; такое развитие относилось бы к прототипу и не имело бы обязательной связи с данным понятием.

Альтернативные признаки, относящиеся к значению радикалов *(R)* одной структурной формулы, не являются нарушением единства понятия в тех случаях, когда все соединения, описанные этой структурной формулой с каждым из значений радикалов, получены принципиально одним способом и свойства всех производных с данными значениями радикалов одинаковы, т. е. налицо эквиваленты.

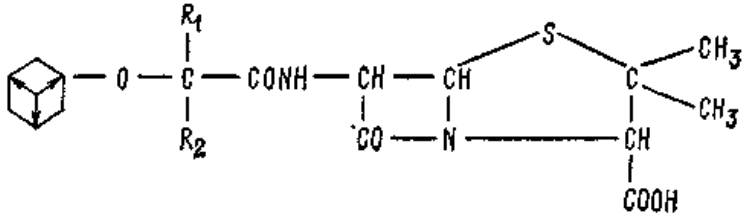
Единство понятия для химического соединения считается соблюденным, если формула определения понятия содержит признаки ряда соединений, выраженных одной общей структурной формулой; если доказана идентичность их химической структуры; если они могут быть получены принципиально одним способом и имеют принципиально одно конкретное название с одинаковым эффектом, обусловленное идентичными свойствами; если признаками, отраженными в звеньях формулы определения понятия, следующих за звеньями с существенными признаками самого соединения и процесса его получения, являются соответственно конкретные соединения, отвечающие общей структурной формуле, признаки, развивающие, конкретизирующие процесс получения соединения, и признаки, развивающие конкретное назначение, указанное в первом звене формулы определения понятия.

Единство понятия считается нарушенным, если вместе с новым химическим соединением описано новое промежуточное вещество, исходное для получения этого соединения; если вместе с новым химическим соединением описан новый продукт, в который это соединение входит как один из ингредиентов; если вместе с новым химическим соединением описаны два или более новых процесса его получения и указаны два или более его конкретных назначения.

Единство понятия в формуле определения, содержащей часть и целое и альтернативу, нарушено, если формула описывае ряд соединений, а свойствами, обнаруженными у новых соединений, обладают только некоторые представители этого ряда, что отражено в описании.

Пример.

Производные 6-амино-пенициллановой кислоты и (или) их соли общей формулы:



где R_1 — водород, алифатический радикал, содержащий до 4 углеродистых атомов; циклоалифатический радикал, содержащий до 6 углеродистых атомов или меноциклический ароматический радикал; R_2 — водород или алкил, содержащий до 4 углеродных атомов, который может быть также связан с R_1 в двухвалентный алифатический радикал, содержащий от 2 до 8 углеродных атомов.

Эта формула иллюстрирует нарушение единства понятия из-за наличия части и целого и альтернативных признаков в формуле определения понятия, но при условии, что в описании отсутствует подтверждение одинаковых свойств произвольных 6-аминопеницилановой кислоты и их солей, а лишь установлено наличие указанных свойств у производных кислот (но не у их солей).

Последствия нарушения единства понятия сказываются прежде всего на неоправданном сужении объема понятия. Действительно, если бы два разных научно-технических решения были описаны в одной формуле определения понятия (или первом звене многозвенной формулы определения понятия), то это явилось бы нарушением единства понятия и такая формула определения понятия описывала бы только совокупное использование независимых научно-технических решений, а это довольно редкий случай.

Нарушение единства понятия в дополнительном звене формулы определения понятия также приводит к сужению объема понятия, но в меньшей степени. Это происходит потому, что если в первом звене формулы определения понятия охарактеризовано одно научно-техническое решение, а в дополнительном — другое независимое научно-техническое решение, которое в соответствии с правилами составления многозвенной формулы определения понятия имеет ссылку на первое звено, то объем понятия практически характеризуется научно-техническим решением, указанным в первом звене. В этом случае объем понятия, характеризующийся одним научно-техническим решением, все же больше, чем объем понятия, характеризующийся совокупным применением двух независимых научно-технических решений.

2.3.8.15. Комплексные понятия

Иногда встречаются такие многосторонние научно-технические задачи, которые требуют разработки, например, нового технологического процесса и нового оборудования для его проведения или нового материала.

В результате решения такой комплексной задачи может быть разработано несколько понятий на различные предметы или, можно сказать, несколько разновидностей понятия. Эти несколько понятий, хотя и являются результатом решения одной комплексной задачи, могут рассматриваться как самостоятельные, если каждое из них (или некоторые из них) может использоваться не только в комплексе, но и вне связи с другими. Тогда говорят, чго между понятиями нет обязательной связи, нет единства понятия. Такие понятия необходимо оформлять отдельными описаниями, как и всякие самостоятельные понятия.

Но если одно или несколько понятий, использующихся при решении комплексной задачи, можно применить только совместно с другим или только для выполнения другого, значит разработано комплексное понятие, состоящее из нескольких разновидностей понятий, развивающих одно понятие.

Допускается объединение в одном описании двух или более понятий, относящихся к разным предметам (объект, процесс, вещество), если они служат одной цели и могут быть использованы лишь совместно.

Оговоренные в этой формулировке условия единства понятий требуют пояснения.

В качестве положения, необходимого для соблюдения единства понятия при описании одной формулой определения объемов разных видов, вводится так называемый единый понятийный замысел. ***Под единым понятийным замыслом следует понимать задуманную разработчиком и воплощенную в описании совокупность средств, которая решает одну задачу (проблему).***

Условие совместного применения предметов разных видов (разнородных предметов) будет соблюдено в том случае, если предмет одного вида не может решить задачу без предмета другого вида указанной совокупности. Последнее условие совместной реализации следует понимать так, что новый овеществленный объект, например, может быть описан только одним понятием вместе с ним процессом, а сам процесс пригоден для получения только этого нового овеществленного объекта (вещества).

Указанные условия необходимы и достаточны для признания правомерности включения в одну формулу определения понятия двух

и более предметов разных видов, т. е. для признания единства понятия соблюденным.

Все звенья многозвенной формулы определения понятия, характеризующие разнородные предметы, являются одинаково существенными в решении одной задачи (каждый предмет необходим, вместе — достаточны) и представляют собой равноценные существенные признаки.

Место каждого объекта в многозвенной формуле определения понятия (первое, второе и третье звенья), таким образом, определяется не его значимостью в едином понятийном замысле, а только тем, какая задача (проблема) решена совокупностью разнородных объектов.

Именно решенная задача (проблема) обусловливает факт, что в первом звене многозвенной формулы определения понятия содержится характеристика вещества, процесса или объекта в зависимости от истинных границ понятийного замысла.

Итак, формула определения понятия, включающая два или более разнородных предметов понятия, составляется в виде многозвенной формулы, при этом:

а) в первом пункте формулы определения понятия характеризуется тот предмет понятия, который является доминирующим в данном сочетании;

б) во втором звене формулы определения понятия характеризуется предмет понятия, находящийся в связи с предметом первого звена, что выражается, например, словами «процесс получения вещества по п. 1». Поскольку объект, характеризуемый в з. 2, служит достижению единой цели, указываемой в з. 1, цель предмет понятия в з. 2, как правило, не указывается;

в) в третьем звене формулы определения понятия характеризуется предмет понятия, находящийся в связи с предметом второго звена, что выражается, например, словами «средство для осуществления процесса по п. 2». Цель понятия в этом случае не указывается;

г) если в дополнительных звеньях формулы определения понятия развиваются, уточняются признаки, содержащиеся в независимых звеньях формулы определения, то такие дополнительные звенья формулируются как в обычной многозвенной формуле определения понятия, характеризующей один предмет понятия, т. е. с подчинением их соответствующим звеньям формулы определения и расположением соответственно подчинению.

2.4. Переменные и параметры

Формой выражения знаний об объектах и системах объектов ИИ будем называть ***представлением.***

Представление формы знаний о моделях объектов и систем объектов ИИ будем осуществлять с использованием понятий: ***переменная и параментр.***

Под ***переменной*** будем понимать **операционное представление свойства объекта**, т. е. **образ свойства, который определяется конкретной процедурой наблюдения или измерения.** Каждое понятие **переменной** имеет **уникальное имя (метку),** отличающее ее от других переменных, и связывается с определенным **множеством величин,** посредством **которых она себя проявляет**. Эти величины будем называть ***состояниями* (или значениями) переменной**, а все множество — ***множеством состояний***.

Под ***параметром*** будем называть **операционное представление базы**. Каждый **параметр** имеет **уникальное имя**, и с ним связывается некоторое **множество**; будем называть его ***параметрическим множеством,*** а его **элементы** — ***значениями параметра***.

По аналогии со **свойствами и базами** будем предполагать, что **разные наблюдения одной и той же переменной различаются по значениям параметров**. **Если используются два и более параметра, то их общим параметрическим множеством является декартово произведение отдельных параметрических множеств.** Требуется , чтобы каждое **конкретное значение параметра (из общего параметрического множества) идентифицировало одно и только одно наблюдение соответствующих переменных.**

На отдельных множествах состояний или параметрических множествах могут быть определены некоторые математические отношения, такие как, **отношение порядка или расстояние**. Они отражают **фундаментальные характеристики свойств и баз** в той степени, в какой они присущи соответствующим измерительным процедурам. **Различия в подобных свойствах среди переменных или параметров, которые имеют существенное методологическое значение, будем называть *методологическими отличиями*.**

В дополнение к ***конкретным переменным и параметрам,*** **представляющим соответственно определенный признак или базу**, будем также рассматривать ***обобщенные переменные и параметры***. Последние представляют собой **абстрактные величины**, т. е. **величины, не определенные через какие-либо свойства или базы**. Их

множества состояний и параметрические множества, а также различные отношения, определенные на этих множествах, представляются определенным образом.

Обобщенной переменной дается **интерпретация**, когда множество ее состояний отображается **изоморфно (т. е. отображается тождественно (один в один) с сохранением всех существенных математических отношений, определенных на нем)** в элементы множества состояний конкретной переменной; **то же относится к обобщенным и конкретным параметрам и их параметрическим множествам**. Любое изоморфное отображение такого рода будем называть ***конкретизацией*** **обобщенной переменной** (или **обобщенного параметра**), а обратное отображение назовем ***абстрагированием*** **конкретной переменной (или конкретного параметра).**

Для формализации понятий обобщенных и конкретных переменных и их параметров введем следующие обозначения в дополнение к введенным в предыдущем разделе; v_i , V_i, V_i, означающие соответственно **обобщенную переменную, ее множество состояний и множество математических свойств, определенных на v_i.**

Обозначим через v'_i , V'_i, V_i' те же характеристики **конкретной переменной**, являющиеся конкретизацией переменной v_i. Обозначим через w_j, W_j, и W_j соответственно **обобщенный параметр, его множество состояний и множество математических свойств, определенных на параметре w_j,** а через w'_j, W'_j, и W'_j — те же характеристики **конкретного параметра**, полученные конкретизацией параметра w_j.

Назовем ***каналом наблюдения*** **любую операцию, представляющую конкретную переменную как образ свойства**. Канал наблюдения, с помощью которого свойство a_i представляется переменной v'_i, **реализуется функцией**

$$o_i: A_i \rightarrow V'_i. \qquad (2)$$

Эта функция **гомоморфна** относительно предполагаемых свойств множеств A_i и V'_i. Аналогичная функция

$$\omega_j: B_j \rightarrow W'_j \qquad (3)$$

задает представление базы b_j параметром w'_j , она также должна быть гомоморфной относительно соответствующих свойств базы (например, времени) и свойств множества W'_j .

Для некоторых **свойств и баз каналы наблюдения могут представлять собой явно заданные функции** o_i и ω_j. Однако в других случаях, когда множества A и B неизвестны, возможно явно задать эти функции, не прибегая к неким абстрактным допущениям. При этом

представления свойств и баз вводятся физически (операционно), а не с помощью математических определений.

За исключением тривиальных случаев, когда функции o_i и ω_j определены ясно, **канал наблюдения представляет собой физическое устройство и процедуру, описывающую его применение.** Это устройство обычно называется **измерительным прибором или инструментом**. **Процедура представляет собой набор команд, определяющих то, как следует использовать инструмент в разных условиях.**

Термин ***измерительный инструмент*** будем понимать широко. В некоторых областях, таких, как психология, общественные науки или этика, сам исследователь (или его группа) может рассматриваться в качестве инструмента или вместо него самого для оценки таких признаков, как мнения, позиции или способности людей, используются анкеты или тесты. **Любой измерительный инструмент должен уметь взаимодействовать с измеряемым свойством и преобразовывать это взаимодействие в вид, непосредственно представляющий состояния соответствующей переменной (например, показания указателя на шкале буквенно-цифрового дисплея или просто запись значений).**

Несмотря на то, что измерительные инструменты и процедуры, образующие каналы наблюдения, должны соответствовать некоторым общим принципам измерения, они существенно зависят от того, что они измеряют. Поэтому их изучением, созданием и использованием занимаются, главным образом, в рамках традиционных научных дисциплин. Теория и практика измерений, как и техника, медицина, управление и т. д., является частью традиционных областей науки и не входит в сферу решения наших задач. **Поэтому каналы наблюдения используются в схеме наших задач только как компоненты, необходимые для полного определения любой реально существующей системы ИИ.** Будем считать, что каналы наблюдения непосредственно связаны с конкретными явлениями и в этом качестве должны изучаться и разрабатываться в рамках традиционных дисциплин.

Мы будем работать в системах ИИ как с конкретными, так и с обобщенными переменными и параметрами. Заданная обобщенная переменная v_i конкретизируется переменной v'_i тогда и только тогда, когда функция

$$e_i: V_i \rightarrow V'_i \qquad (4)$$

существует и изоморфна относительно математических свойств V_i.

Аналогично обобщенный параметр w_j конкретизируется параметром w'_j тогда и только тогда, когда функция

$$\varepsilon_j : W_j \rightarrow W'_j \qquad (5)$$

существует и изоморфна относительно W'_j. Каждый конкретный изоморфизм e_i (или ε_j) задает конкретизацию v_i с помощью v'_i (или соответственно w_j с помощью w'_j). Функции, обратные e_i и ε_j, т. е.

$$e_i^{-1} : V'_i \rightarrow V_i, \qquad (6)$$

$$\varepsilon_j^{-1} : W'_j \rightarrow W_j \qquad (7)$$

задают абстрагирование соответственно v'_i и w'_j.

Пример 1. Для иллюстрации введенных понятий положим, что a_i — это установленный ежегодный доход налогоплательщика Украины за последний год, как сообщается в его налоговой декларации за этот год. Тогда A_i— это всевозможные суммы денег (в грн.) от нуля до максимально представимой суммы, скажем до 100 000.00 грн. Это множество конечно, так как минимальная имеющая хождение денежная единица — 1коп. Мы понимаем также, что это множество полностью (линейно) упорядочено. Для вычисления подоходного налога достаточно рассматривать только интервалы (диапазоны) облагаемого налогом дохода, где каждому интервалу соответствует определенный процент дохода, который следует выплатить в качестве подоходного налога. Для упрощения будем этими интервалами считать интервалы 0—4,999.99, 5,000.00 — 9,999.99, ..., 90,000.00 —94,999.99**,** 95,000.00— 100,000.00 и пусть множеством состояний V'_i - конкретной переменной v'_i, представляющей свойство a_i, будет множество минимальных значений этих интервалов. Содержательное представление a_i с помощью v'_i можно ввести с помощью функции o_i, которая для каждого интервала любому значению из интервала присваивает минимальное значение в этом интервале, например o_i (52357) =50 000 или o_i (796)=0. Очевидно, что функция o_i гомоморфна относительно полного упорядочения A_i, так как для любой пары α, $\beta \in A_i$, если $\alpha \leq \beta$, $o_i(\alpha) \leq o_i(\beta)$. Из методических соображений обобщенная переменная v_i может быть для конкретной переменной v'_i определена с помощью абстрагирующей функции $e_i^{-1} : V'_i \rightarrow V_i$. Эта функция должна быть изоморфной относительно упорядочения на V'_i. Предположим, что нужно, чтобы множество V_i представляло собой набор значений целых чисел. Тогда e_i^{-1}можно, вероятно, наиболее естественным образом, задать следующим уравнением:

e_i^{-1} (5000k) $= k$ (k =0, 1, ..., 19).

Базой в этом примере является множество налогоплательщиков определенной категории, скажем множество жителей г. Киева. Данное множество не обладает никакими математическими свойствами. Таким образом, ω_j: $B_j \rightarrow W'_j$ может быть любой взаимно однозначной функцией, которая каждому налогоплательщику ставит в соответствие уникальный идентификатор, например номер его идентификационного

кода. Методологически удобно абстрагирование $\varepsilon_j^{-1} : W'_j \rightarrow W_j$ представить в виде взаимно однозначной функции, ставящей в соответствие целым числам из множества N_n, где n — число налогоплательщиков в этой группе, значения номеров идентификационных кодов.

Рассмотрим более подробно понятие «**канал наблюдения»**. До сих пор мы его определяли через функции o_i и ω_j, определенные соответственно в уравнениях (2) и (3). Эти функции индуцируют разбиения множеств A_i и B_j, например, разбиения A_i/o_i и B_j/ω_j. Элементы любого блока в этом разбиении эквивалентны в том смысле, что они не различаются с точки зрения введенной **процедуры наблюдения**. В таком разбиении **каждый блок представляет одно состояние переменной v'_i или одно значение параметра w'_j.** Когда наблюдение свойства a_i проводится при некотором значении параметра, то наблюдаемое свойство получает определенное проявление (значение) из множества A_i .Это проявление является **элементом одного и только одного блока разбиения A_i/o_i. Функция o_i присваивает его определенному состоянию переменной v'_i** . Таким образом, предполагается, что любое наблюдение позволяет нам определить, к какому блоку из A_i/o_i принадлежит данное проявление, даже если отдельное проявление и нельзя идентифицировать.

Предположение о том, что различие блоков A_i/o_i может быть обнаружено по результатам наблюдений, оправдывается только в том случае, когда ошибки наблюдения исключены. Подобные случаи, как показано в примере 1, встречаются, но относительно нечасто. Тем не менее это предположение можно считать практически оправданным и в других случаях, когда блоки A_i/o_i существенно больше ожидаемых значений ошибок наблюдения. При этом блок A_i/o_i правильно определяется во всех случаях, кроме тех, когда фактическое проявление оказывается близко от границы между блоками, т. е. в пределах ожидаемой ошибки наблюдения. Поскольку свойства (по крайней мере некоторые из них) не контролируются исследователем, невозможно предотвратить проявления свойств в нежелательной близости от границ между блоками в A_i/o_i и, следовательно, можно только сократить возможность определения неправильных блоков по наблюдениям благодаря правильному выбору канала наблюдения o_i. **Исключить такую возможность полностью нельзя.**

В результате появления возможности ошибок измерения с проявлениями возле границ между блоками в A_i/o_i связана определенная **недостоверность наблюдения**. Будем отличать два варианта интерпретации этой недостоверности.

1. **Блоки разбиения**, определенные на множестве A_i, рассматриваются как **множества без четких границ**. В терминах теории нечетких множеств эти блоки представляют собой **нечеткие подмножества множества A_i.** Предполагается, что множество A_i является четким. Каждый элемент множества A_i принадлежит любому его нечеткому подмножеству с определенной степенью принадлежности. **Согласно такому подходу подмножества определяются только степенями принадлежности, а не функцией o_i.**

2. **Разбиение множества A_i задается функцией o_i.** Это то же самое разбиение A_i/o_i, что рассматривалось выше. Ясно, что его блоки являются четкими подмножествами A_i. Достоверно неизвестно, к какому блоку A_i/o_i принадлежит заданный элемент A_i. Эта недостоверность может быть задана функцией, сопоставляющей любой паре (элемент A_i, блок A_i/o_i) число (обычно между 0 и 1). Определенное таким образом число в заданном контексте выражает степень достоверности того, что данный элемент принадлежит данному блоку.

В дальнейшем мы будем придерживаться второго варианта. Согласно этого варианта требуется, чтобы сначала была задана функция o_i, как в уравнении (2). Будучи заданной, она определяет на множестве A_i разбиение A_i/o_i. Затем определяется функция

$$\tilde{o}_i: A_i \times A_i/o_i \to [0, \quad 1], \qquad (8)$$

где $\tilde{o}_i\,(x, y)$ задает степень достоверности того, что x принадлежит y. Однако, поскольку каждый блок A_i/o_i однозначно представляется (помечается) состоянием из множества V'_i (в соответствии с функцией o_i), функцию $\tilde{o}_i$ можно задать в более удобном виде

$$\tilde{o}_i: A_i \times V'_i \to [0, \quad 1], \qquad (9)$$

где $\tilde{o}_i(x, y)$ задает степень достоверности того, что x принадлежит блоку из разбиения A_i/o_i, представляемому состоянием y переменной v'_i.

Определенная в уравнении (9) функция $\tilde{o}_i$ характеризует наблюдения свойства a_i в смысле их недостоверности. Ее также можно рассматривать как функцию степени принадлежности, определяющей нечеткое отношение на декартовом произведении $A_i \times V'_i$. В этом смысле $\tilde{o}_i$ будем называть *нечетким каналом наблюдения.* Чтобы исключить недоразумения, o_i будем называть *четким каналом наблюдения.*

Ясно, что **для определения нечеткого канала наблюдения необходимо сначала задать четкий канал наблюдения o_i.** Четкий канал

наблюдения можно также рассматривать как частный случай нечеткого. В самом деле, если

$$\tilde{o}_i(x, y) = \begin{cases} 1, \text{ если } o_i(x) = y \\ 0 \quad \text{в противном случае} \end{cases}$$

то $\tilde{o}_i$ задает четкую функцию из A_i в V'_i, идентичную o_i.

При рассмотрении баз введем функцию

$$\tilde{\omega}_j : B_j \times W'_j \to [0, 1], \qquad (10)$$

подобную функции (9) и основанную на соотношении (3). Здесь $\tilde{\omega}_j\,(x, y)$ — степень достоверности того, что x принадлежит блоку разбиения B_j/ω_j, который представлен значением y параметра w'_j. Как правило, эту функцию используют редко. Так, если B_j — является группой, то функция ω_j является взаимно однозначной, и, следовательно, недостоверность наблюдений отсутствует. Если B_j является временем или пространством, то реальное наблюдение контролируется исследователем, т. е. исследователь принимает решение о том, где или когда должно быть проведено наблюдение. Такой контроль реальных наблюдений, а также относительная свобода выбора соответствующей функции ω_j позволяют исследователю избежать всякой недостоверности, за исключением неустранимых ошибок при измерении времени или пространства. Если, например, он решает, что температура, относительная влажность и т. д. должны фиксироваться на метеостанции 24 раза в сутки, в половине каждого часа, он может задать функцию ω_j так, чтобы любой блок результирующего разбиения B_j/ω_j - представлял собой одночасовой интервал [0 ч—1 ч), [1 ч — 2 ч), ..., [23 ч — 0 ч). Тогда, если взять конкретное измерение наблюдаемых свойств, скажем в 1 ч 30 мин, то обычно совершенно достоверно можно сказать, что это измерение представляет блок [1—2 ч). В принципе и в этом случае возможны ошибки (такие, как грубое нарушение правила измерения или поломка часов), однако такие случаи для обычного канала наблюдения не рассматриваются.

Подытожим наши соображения относительно каналов наблюдения. **Для любых практических надобностей достаточно использовать четкий канал наблюдения ω_j для баз, будь то группа, время или пространство. Однако для свойств применимы как четкие, так и нечеткие каналы наблюдения (o_i и $\tilde{o}_i$), и при разных обстоятельствах более подходящим может быть тот или иной тип канала.**

Пример 2. Пусть свойством a_i является возраст человека из группы B_j. И пусть элементами A_i будут номера лет в интервале от 0 до 100. Положим, что V'_i= {очень молодой, молодой, средних лет, старый, очень старый}, и пусть o_i — это взаимно однозначная функция $A_i/o_i \rightarrow V'_i$, определенная следующим образом:
[0, 1, ..., 14] —очень молодой,
[15, 16, ..., 29] —молодой,
[30, 31, ..., 49] —средних лет,
[50, 51, ..., 74] —старый,
[75, 76, ..., 100]—очень старый.
При использовании четкого канала наблюдения очень плохо описываются люди, чей возраст близок к границам между блоками A_i/o_i. Например, 49-летний человек помечается как человек средних лет, а 50-летний, как старый. При использовании нечеткого канала o_i, например такого, какой описан на рис. 1, приведен оказывается более подходящим, поскольку не дает таких резких скачков.

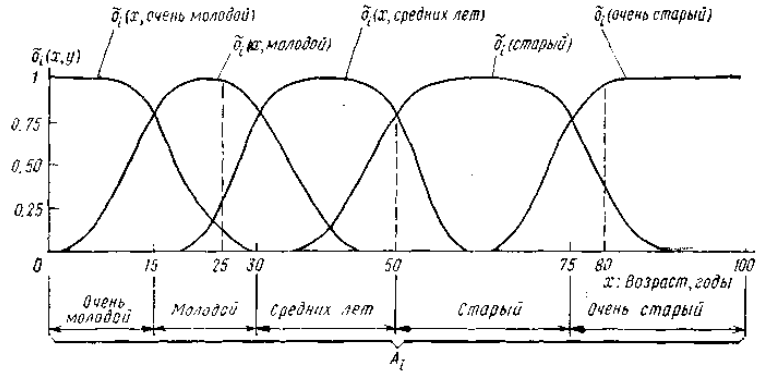


Рис. 1. Четкий и нечеткий каналы наблюдения для полностью упорядоченного признака «возраст человека»

Важно отметить, что нечеткий канал наблюдения дает не одно состояние V'_i для одного наблюдения, как четкий канал, а набор значений $\tilde{o}_i(x, y)$ для всех $y \in V'_i$. Так, например, при наблюдении 25-летнего человека через нечеткий канал будут получены следующие 5 значений:

$\tilde{o}_i$ (25, очень молодой) =0.1
$\tilde{o}_i$ (25, молодой) =0.97
$\tilde{o}_i$ (25, старый) =0
$\tilde{o}_i$ (25, очень старый) =0.

2.5. Методологические отличия

Приведем цитату Дж. Пфанцагль *«Когда мы говорим, что свойство имеет выраженную структуру, то имеем в виду, что структура определена эмпирическими отношениями между эмпирическими объектами... Чем больше отношений принимается во внимание при определении шкалы значений, тем больше значения шкалы говорят нам о реальном мире... Построить шкалу, гомоморфную относительно отношения порядка, но не сохраняющую, скажем, отношение аддитивности, которое можно определить эмпирически, значит потерять информацию.»*
Термин ***методологическое отличие*** **используется в данной работе для описания особенностей интеллектуальных задач,** по которым различаются равные типы интеллектуальных задач внутри одной эпистемологической категории интеллектуальных задач.
Методологические отличия касаются как систем ИИ, так и требований к ним. Такие модификации, как введение в систему нового методологического отличия, его удаление или замена одного методологического отличия другим, **не изменяют эпистемологического уровня системы ИИ**. Однако они могут повлиять на наборы методологических отличий, подходящих к различным требованиям. Таким образом, **методологические отличия для систем ИИ должны выбираться до того, как такие отличия будут выбраны для требований.**
Типы интеллектуальных задач разнятся только некоторыми методологическими отличиями, требуют разных методов решения, но **имеют один и тот же статус в эпистемологической иерархии систем ИИ.** Таким образом, **методологические отличия представляют собой вторичные критерии классификации интеллектуальных задач**.
Они служат дополнением к первичной классификации по эпистемологическим критериям. Эпистемологических типов задач слишком много для всякого конкретного методологического исследования. Поэтому методологические отличия предназначены для уточнения типов методологически разрешимых интеллектуальных задач.
Набор приписываемых в рамках определенной концептуальной схемы методологических отличий систем ИИ связан с набором принятых эпистемологических типов систем ИИ **отношением «применимо к»; аналогичное отношение имеется и для требований**. В то время как одни методологические отличия подходят только к определенным эпистемологическим типам, другие применимы ко всем.

В данном разделе рассматриваются методологические отличия, относящиеся к переменным и их параметрам. Так как переменные и параметры являются компонентами любой системы ИИ независимо от ее эпистемологического уровня, эти эпистемологические отличия применимы к системам ИИ всех эпистемологических уровней.
Методологические отличия для переменных и параметров — это характеристики их множеств состояний и, соответственно, параметрических множеств. Если **переменная (или параметр) представляет свойство (или базу),** то эти свойства не могут быть произвольными. Характеристики, очевидным образом не подходящие свойству или базе, не следует выделять и в соответствующем множестве состояний или параметрическом множестве. Однако некоторые предполагаемые характеристики свойства (или базы), не имеющие отношения к рассматриваемой задаче, также не следует признавать свойствами соответствующей переменной (или параметра).
Во избежание возможных недоразумений приведем следующее замечание для прояснения смысла методологических отличий для самого нижнего эпистемологического уровня — **уровня свойств, баз и их абстрактных аналогов (переменных и параметров).** Для решения интеллектуальных задач **методологические отличия определяются на нижнем эпистемологическом уровне только для переменных и параметров (как конкретных, так и общих), а не для соответствующих свойств и баз.** Таким образом, на нижнем эпистемологическом уровне методологические отличия определены в **терминах математических свойств множеств состояний и параметрических множеств.** Нужна, разумеется, гарантия, что выделенные свойства отражают фундаментальные характеристики соответствующих свойств и баз. Это, однако, вопрос эмпирический, связанный прежде всего с методикой измерения и не входящий в проблематику решения интеллектуальных задач.
Всякая переменная связана с одним или несколькими параметрами, и изменения состояний переменной наблюдаются на полном параметрическом множестве. Таким образом, комбинация свойств множества состояний и полного параметрического множества определяет самый элементарный тип методологических отличий.
Если имеется более одного параметра, то полное параметрическое множество представляет собой декартово произведение отдельных параметрических множеств. Для представления распознаваемых свойств этого декартова произведения свойства отдельных параметров должны сочетаться соответствующим образом. Эти свойства полного параметрического множества (декартового произведения) совместно со свойствами соответствующего множества состояний используются

затем для описания элементарного методологического отличия. Если все отдельные параметрические множества имеют одни и те же свойства, то их легко скомбинировать, и полученные общие свойства будут однородны на всем декартовом произведении. Сложнее, когда отдельные параметрические множества имеют разные свойства. В этих случаях по крайней мере некоторые общие свойства не распространяются на все декартово произведение.
Будем сначала для простоты считать, что мы имеем дело с одним параметрическим множеством независимо от того, является оно отдельным параметрическим множеством или декартовым произведением нескольких, и что выделенными свойствами обладают все это множество.
Одним из фундаментальных методологических отличий является *отсутствие математических свойств* у множества состояний или соответствующего параметрического множества. Это крайний случай, и он плохо подходит для переменной (или параметра), предназначенной для представления свойства (или базы) и имеющей явно выраженные и существенные для задачи характеристики. Однако во многих случаях такое предельное методологическое отличие вполне уместно и даже необходимо. Такие, например, переменные как семейное положение (одинокий, женат, разведен, вдовец), политическая принадлежность (демократ, республиканец, независимый), группа крови *(A, B, O, AB)* или пол (мужской, женский), **заданные на элементах некой общественной группы, демонстрируют существенность этого методологического отличия**. В литературе по измерениям переменные такого рода обычно называют *переменными с номинальной шкалой.*
Наиболее фундаментальным из выделяемых свойств множеств состояний и параметрических множеств является *упорядоченность.* Методологически следует различать два типа упорядоченности — частичную и линейную.
Частичная упорядоченность — это бинарное отношение на множестве (в нашем случае на множестве состояний или параметрическом), являющееся рефлексивным, антисимметричным и транзитивным.
Линейная упорядоченность сильнее частичной, так как это частичная упорядоченность, обладающая свойством связности (т. е. любая пара элементов множества так или иначе упорядочена).
Формально частичная упорядоченность Q, например, множества V_i — это бинарное отношение

$$Q \subset V_i + V_i,$$

удовлетворяющее следующим требованиям:

1. *(x, y)* ∈ *Q* (рефлексивность);
2. если *(x,* z) ∈ Q и *(y,* x) ∈ Q, то *x=y* (антисимметричность);
3. если *(x, y)* ∈ *Q* и *(y,* z) ∈ Q, то (x, z) ∈ Q (транзитивность).

Если *(x,y)* ∈ Q, то *x* называется *предшественником y, z,* а — *преемником x.* Если *(x,* y) ∈ Q и не существует *z* ∈ *Q,* такого, что *(x,* z) ∈ Q и (z, *x)* ∈ *Q,* то *x* называется *непосредственным предшественником y,* а *y — непосредственным преемником x.* В дополнение к требованиям рефлексивности, антисимметричности и транзитивности отношение линейной упорядоченности удовлетворяет следующему требованию связности: для всех *x, y* ∈ V_i, если *x≠y,* то или *(x, y)* ∈ *Q,* или *(y, x)* ∈ *Q.*

Примерами переменных с частично упорядоченным множеством состояний являются служебное положение или образование человека (определенные, например, на группе государственных служащих). Примерами переменных с линейно упорядоченными множествами состояний являются шкала твердости Мооса, высота как характеристики звука или экзаменационные оценки, определенные на группе студентов. Примером упорядоченности параметрического множества является время. Хотя в большинстве случаев такое упорядочение линейно, имеют смысл и частично упорядоченные временные множества, например при исследовании отдельных пространственно разделенных процессов (таких, как распределенные вычислительные машины, которые обмениваются друг с другом информацией и для которых задержка при передаче сообщения сравнима со временем изменения состояний переменных из отдельных процессов). Полезно определить упорядочение и для некоторых групп. Например, группа людей может быть упорядочена по таким отношениям, как «быть старше», «быть потомком», «занимать более высокое положение по работе». Обычно частичные упорядочения и их существенность зависит от характера группы и всего контекста задачи. Переменные с линейно упорядоченными множествами состояний называются *переменными с упорядоченной шкалой.*

Кроме частичных или линейных упорядочений существуют и другие математические свойства, определение которых для множеств состояний или параметрических множеств оказывается во многих случаях очень полезным. **Одним из наиболее существенных свойств является расстояние между парой элементов изучаемого множества. Эта мера определяется функцией, сопоставляющей любой паре элементов этого множества число, определяющее, на каком расстоянии друг от друга находятся эти элементы с точки зрения некоторого фундаментального упорядочения.**

Для данного множества, скажем множества V_i, расстояние определяется функцией δ вида

$$\delta: V_i \times V_i \rightarrow R.$$

Однако для того, чтобы эта функция отвечала интуитивному представлению о расстоянии, она должна удовлетворять следующим условиям для всех *x, y, z* ∈ V_i:

(δ1) δ (*x , y)≥0* (условие неотрицательности);
(δ2) δ (*x, y*)=0 тогда и только тогда, когда *x=y* (условие нулевого расстояния, называемое также условием невырожденности);
(δ3) *δ(x, y)= δ(y, x)* (симметричность);
(δ4) δ(x, *z)≤*δ*(x, y)* +δ(*y, z)* (неравенство треугольника).

Любая функция, удовлетворяющая условиям (δ1) — (δ4), называется *метрическим расстоянием* на множестве V_i, а пара (V_i, δ)— *метрическим пространством.* **Метрическое расстояние можно, разумеется, определить как на множестве состояний, так и на параметрическом множестве.**

Примерами переменных с выраженными и существенными метрическими расстояниями являются почти все переменные в физике, например длина, масса, давление, электрическая проводимость, напряжение, сила звука, однако и помимо физики есть множество примеров таких переменных, скажем, количество денег, объемы производства, число дефектов, число несчастных случаев и т. д. Совершенно очевидно, что и **пространство, и время — это параметры, к которым вполне естественно применимо понятие метрического расстояния**. Однако редко удается определить метрическое расстояние на группах. Одним из таких примеров является группа студентов, линейно упорядоченная по показателям их успеваемости. При этом расстояние для каждой пары студентов определяется как абсолютное значение разницы между их позициями в упорядоченном списке. **Переменные, с множеством состояний которых связано метрическое расстояние, обычно называются *метрическими переменными.***

Еще одним свойством множеств состояний и параметрических множеств, имеющим большое значение как методологическое отличие, является *непрерывность.* Это понятие хорошо известно из математического анализа, и нет необходимости рассматривать его здесь подробно. Тем не менее уместно будет привести несколько замечаний относительно некоторых аспектов непрерывности, которые будут использоваться нами в дальнейшем.

Необходимым условием непрерывности множества является его упорядоченность. Так как линейная упорядоченность является частным случаем частичной упорядоченности, то предпочтительнее

определить непрерывность через частичную упорядоченность. Это можно сделать несколькими способами. Одно из определений непрерывного частичного упорядочения опирается на понятие **разреза частично упорядоченного множества**: *разрез частично упорядоченного множества,* скажем множества V_i, это разделение этого множества на два непустых подмножества, например X и Y= V_i —X, такое, что или никакой элемент X не является предшественником (согласно частичному упорядочению, определенному на V_i) никакого элемента из Yи некий элемент Y является предшественником какого-либо элемента X, или никакой элемент из X не является преемником никакого элемента из Y и некоторый элемент Y является преемником некоторого элемента X. *Непрерывное частичное упорядочение* V_i определяется как частичное упорядочение, для которого любой разрез X, Y множества V_i характеризуется неким элементом из X, являющимся предшественником элемента из Y, такого, что или наибольшая верхняя граница X принадлежит Y, или наименьшая нижняя граница Y принадлежит X

Наилучшим примером непрерывного частичного упорядочения является отношение «меньше или равно», определенное на множестве действительных чисел или на его декартовых произведениях. Фактически само понятие *непрерывной переменной* (или *непрерывного параметра)* опирается на требование, чтобы соответствующее множество состояний (или параметрическое множество) было изоморфно множеству действительных чисел.

Из этого следует, что **множество состояний любой непрерывной переменной или параметрическое множество любого параметра бесконечно и несчетно.** Тем самым альтернативой непрерывным переменным и параметрам являются переменные и параметры, заданные на конечных множествах или, возможно, на бесконечных счетных множествах. **Последние называются *дискретными переменными или параметрами***.

Непрерывные переменные и параметры представляются действительными числами, а их дискретные аналоги удобно представлять целыми числами. Это особенно существенно, если множество состояний или параметрическое множество значений дискретной переменной или параметра линейно упорядочено и, следовательно, изоморфно соответствующим множествам значений целых чисел. Для работы с некоторыми переменными и параметрами могут быть использованы метрическое расстояние, определяемое естественным образом как абсолютное значение разницы между целыми, а также целая арифметика.

Для нас **такие свойства, как упорядоченность, метрическое расстояние и непрерывность множеств состояний и параметрических множеств, представляют основу для определения наиболее существенных методологических отличий на уровне переменных и параметров.** Приведем список перенумерованных альтернатив для этих свойств:

Упорядоченность	0 — упорядоченности нет
	1 — частичная упорядоченность
	2 — линейная упорядоченность
Расстояние	0 — не определено
	1 — определено
Непрерывность.	0 — дискретно
	1 — непрерывно

Статус любой переменной (или параметра) для этих трех свойств может быть однозначно охарактеризован **триплетом (упорядоченность, расстояние, непрерывность),** в котором каждое свойство представляется его определенным значением (или его идентификатором). Так, например, триплет (2, 1, 0) описывает дискретную переменную с линейно упорядоченным множеством состояний, на котором определено метрическое расстояние.

Хотя данные три свойства в принципе определяют 12 возможных комбинаций, три из них (0, 0, 1), (0, 1,0) и (0, 1, 1) смысла не имеют. В самом деле, если на множестве не определена упорядоченность, то на нем нельзя ни содержательно определить метрическое расстояние, ни рассматривать его как непрерывное. Таким образом, имеется девять осмысленных комбинаций. Будем называть эти комбинации ***методологическими типами* переменных и параметров**. Они могут быть частично упорядочены с помощью отношения «быть методологически более определенным чем». На рис 2,*а* это частичное упорядочение, образующее решетку, представлено в виде диаграммы Хассе. Упрощенная решетка на рис. 2,б задает схему для свойств упорядоченности и расстояния, но без непрерывности.

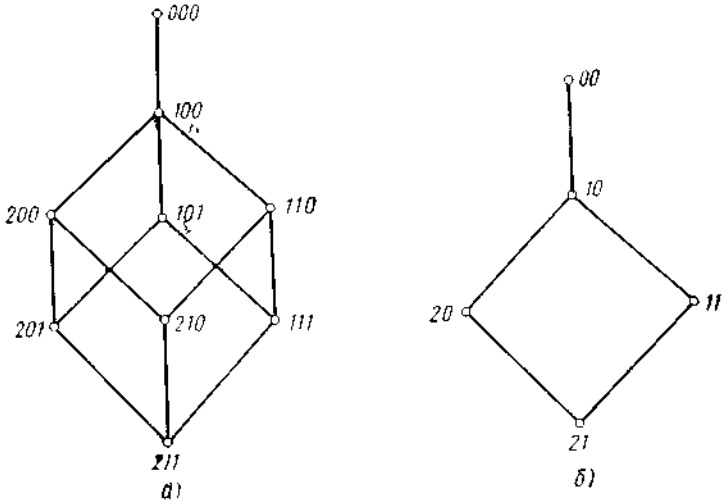


Рис. 2 Решетки методологических типов переменных или параметров

На уровне переменных и параметров методологическое отличие одной переменной представляет собой сочетание методологических типов этой переменной и соответствующих баз. Каждая из них имеет девять типов. Следовательно, если есть только одна база или требуется, чтобы все базы, входящие в комбинацию, имели один методологический тип (наиболее часто встречающийся случай), то число методологических отличий будет равно 81 (так как методологические типы переменных и параметров не накладывают ограничений друг на друга).
Если к тому же в нашей схеме будут учитываться только дискретные переменные и параметры, методологические типы которых приведены на рис. 2,б, то число методологических отличий сократится до 25. Решетка методологических отличий для этого случая приведена в табл. 1.

Таблица 1.

Решетка методологических отличий для дискретных переменных и параметров

Методическое отличие	Непосредственные предшественники в решетке
00/00	10/00 00/10
00/10	10/10 00/20 00/11
00/20	10/20 00/21
00/11	10/11 00/21
00/21	10/21
10/00	20/00 11/00 10/10
10/10	20/10 11/10 10/20 10/11
10/20	20/20 11/20 10/21
10/11	20/11 11/11 10/21
10/21	20/21 11/21
20/00	21/00 20/10
20/10	21/10 20/20 20/11
20/20	21/20 20/21
20/11	21/11 20/21
20/21	21/21
11/00	11/10 21/00
11/10	11/20 11/11 21/10
11/20	11/21 21/20
11/11	11/21 21/11
11/21	21/21
21/00	21/10
21/10	21/20 21/11
21/10	21/21
21/11	21/21
21/21	нове

Теперь предположим, что имеется два или более, например *m,* параметров. Они могут быть одного, двух, трех (независимо от порядка) и т. д. типов. Предположим, что $m \leq 9$, тогда общее число методологических типов полного параметра определяется суммой

$$\binom{9}{1}+\binom{9}{2}+\ldots\binom{9}{m}$$

При сочетании этой суммы с девятью методологическими типами переменных мы получим общее число возможных методологических отличий одной переменной и ее параметра, это число определяется формулой

$$9\times\sum_{i=1}^{m}\binom{9}{i}$$

2.6. Дискретное и непрерывное

Теоретические модели обычно содержат непрерывные функции или бесконечные последовательности, хотя подтверждающие их данные по существу в высшей степени дискретны и конечны.

Как уже говорилось в предыдущем разделе, в формулировку методологических отличий на уровне переменных и параметров входит дихотомия непрерывных и дискретных множеств. В схеме ФРИЗ определены и те, и другие. Однако, если говорить о реализации ФРИЗ, то данная работа почти исключительно посвящена дискретным системам ИИ, т. е. системам ИИ с дискретными переменными и дискретными параметрами. Мы будем лишь при случае касаться вопросов, связанных с непрерывными системами ИИ.

Есть несколько соображений, по которым мы решили ограничиться только дискретными системами ИИ. Прежде всего, сфера применения ФРИЗ столь широка, что не представляется возможным описать все аспекты его применения в работе разумного объема. Во всяком случае **основная цель работы— это описание архитектуры ФРИЗ, а не его применений.** Поэтому применение ФРИЗ описывается не сами по себе, а для обогащения описания его архитектуры. В этом смысле, на наш взгляд, лучше включать в текст работы достаточно полные описания возможных применений ФРИЗ к некоторым содержательным примерам**, а не давать поверхностное описание всего спектра применений.** Класс дискретных систем ИИ и соответствующих интеллектуальных задач как раз и является таким подходящим и ясным подмножеством.

Хотя может показаться, что непрерывные системы ИИ в равной степени подходят для описания архитектуры ФРИЗ, мыубеждены, что для этой цели по многим соображениям больше подходят дискретные системы ИИ. Приведем некоторые из этих соображений:

(I) Независимо от того, считаем ли мы мир по существу дискретным, непрерывным или смешанного типа, факт остается фактом, что с большинством, если не со всеми наблюдениями связана некоторая **неисключаемая конечная ошибка**. Значение этой ошибки определяет некую конкретную верхнюю границу уровня разрешения для данных, собираемых через определенный канал наблюдения. **Это означает, что данные всегда оказываются дискретными независимо от философских убеждений или состояния технологии.**

(II) В случаях, когда эмпирические соображения, описанные в (I), оказываются несущественными и желательно использовать непрерывные переменные, **всегда можно выбрать определяющий дискретные переменные конечный уровень разрешения, позволяющим аппроксимировать непрерывные переменные с наперед за-**

данной точностью; точно так же можно аппроксимировать и непрерывные параметры. Этот подход был очень наглядно продемонстрирован в ряде работ, в которых показано, что как и классическая, и релятивистская физика могут быть переформулированы в терминах дискретных переменных, и что в этой формулировке могут быть получены результаты, сколь угодно близкие к результатам, полученным в традиционной формулировке, основанной на непрерывных переменных и дифференциальных уравнениях.

(III) В то время как дискретные переменные (и параметры) всегда могут быть заданы так, чтобы аппроксимировать непрерывные переменные с требуемой точностью, **непрерывные переменные применимы только к определенному типу свойств (атрибутов)** . В частности, **множество проявлений соответствующих атрибутов должно иметь структуру, изоморфную множеству действительных чисел. Это очень строгое ограничение.** Таким образом, область применения дискретных переменных и параметров оказывается существенно шире области применения их непрерывных аналогов.

(IV) Если явления реального мира и описываются с помощью непрерывных переменных и параметров (обычно в виде набора дифференциальных уравнений), то редко оказывается так, что для работы с этими описаниями удается использовать методы непрерывной математики. Дифференциальные уравнения, описывающие явления реального мира, обычно или не могут быть решены аналитически (т. е. это нелинейные дифференциальные уравнения), или их аналитическое решение чересчур трудоемко. **Следовательно, или по необходимости, или для удобства приходится для их решения использовать численные методы и цифровые вычислительные машины, что, очевидно, требует, чтобы непрерывные переменные и параметры были преобразованы в свои дискретные аналоги.**

Обычно при получении научных знаний сначала проводятся эксперименты, дающие дискретные наборы данных. Затем теоретики анализируют эти данные и в классическом духе вводят непрерывные модели. Если уравнения в этих моделях нелинейны, они решаются численными методами на компьютерах, и результатами снова являются дискретные данные. Средний этап этой работы идеологически несовместим с первым и третьим. В самом деле, было бы проще и удобней вывод непрерывной модели заменить выводом дискретной и таким образом совершенно отказаться от понятия бесконечности. Понятие бесконечности и логически следующие из него понятия предела, производной и интеграла приемлемы в чисто математическом исследовании действительных чисел и

действительных функций, однако они *не подходят* для моделирования физических понятий и явлений

(V) Работа с непрерывными переменными и параметрами имеет ряд чисто математических сложностей и ограничений, которые требуют не только более высокой подготовки, но, что еще важнее, затемняют реальные понятия.

(VI) Легко заметить, что характерное для докомпьютерной эры главенство методов непрерывной математики постоянно сокращается. Непрерывно возрастающая мощность вычислительных машин превосходит возможности аналитического исчисления. Процесс этот, вероятно, будет продолжаться и дальше, и в результате в системных исследованиях будут доминировать системы, построенные на дискретных переменных.

(VII) В то время как точность созданных человеком непрерывных систем (например, аналоговых вычислительных машин или результатов) ограничена и **никакими средствами не может быть поднята выше определенного предела**, точность созданных человеком дискретных систем (цифровых компьютеров, регуляторов, систем связи и т. д.) является только вопросом стоимости.

(VIII) Созданные человеком дискретные системы могут быть в силу своей природы спроектированы так, чтобы они обладали свойствами самокоррекции, что невозможно для созданных человеком непрерывных систем.

(IX) Дискретные функции (например, выражающие зависимости переменных от их параметров или от других переменных) более гибки, чем их непрерывные аналоги в смысле способа представления. Это положение хорошо описал Э. Барто:«... очень удобно использовать символьные выражения для задания дискретных функций... Можно также определить операторы для этих функций через символьные преобразования этих формул. Однако преимущество использования символьных выражений состоит *не только* в возможности полностью специфицировать функции, как в случае с непрерывными переменными. Дискретные функции могут быть полностью определены списком своих значений, например, с помощью запоминания в памяти ЭВМ так, чтобы «адресам» соответствовали аргументы функции, а «содержанию» — ее значения. Можно также задать алгоритм, входом которого является аргумент функции, а выходом — ее значение... Примитивные команды, используемые при спецификации алгоритмов (например, цикл или условное ветвление), позволяют кратко определять функции, которые невозможно или очень неудобно определяются с помощью обычных алгебраических средств.»

2.7. Представляющие и исходные системы ИИ

Свойства, конкретные и общие переменные, а также базы, конкретные и общие параметры являются компонентами соответственно трех примитивных систем ИИ—**системы объекта, конкретной представляющей (image) системы ИИ и общей представляющей системы ИИ**, **которые вместе с отношениями между ними образуют исходную компьютерную систему ИИ**. **Одна** из этих **трех систем ИИ** введена в разд. 2.1 и формально определяется уравнением (1). Оставшиеся две примитивные системы ИИ имеют тот же вид, что и система объекта, но **их компонентами являются переменные и параметры, а не свойства и базы.**

Пусть **I′** и **I** — это, соответственно **конкретная** и **общая представляющая системы ИИ**. Тогда

$$\mathbf{I}'=(\{(v'_i, V'_i)|i \in N_n\}, \{(w'_j, W'_j)|j=N_m\}), \quad (1)$$

$$\mathbf{I}=(\{(v_i, V_i)|i \in N_n\}, \{(w_j, W_j)|j=N_m\}), \quad (2)$$

где соответствующие символы имеют тот же смысл, что и в разд. 2.4. Теперь нужно определить **отношения** между **тремя примитивными системами ИИ O**, **I′**, **I**. Для упрощения нотации условимся, что для любых $i \in N_n$ и $j \in N_m$ свойство a_i соответствует переменным v'_i, v_i, а база b_j — параметрам w'_j, w_j.

Отношение между системой объекта и конкретной представляющей системой ИИ задается в виде полного канала наблюдения, состоящего из отдельных каналов наблюдения, по одному для каждого свойства или базы из системы объекта. Обозначим через $\mathcal{Q}$ ***четкий полный канал наблюдения***. Тогда

$\mathcal{Q}=(\{(A_i, V'_i, o_i)|i \in N_n, o_i$ определяется уравнением (2 р.2.4) и должны быть *гомоморфны* относительно свойств A_i и $V'_i\}$,
$\{(B_j, W'_j, \omega_j)|i \in N_m$ определяются уравнением (3 р.2.4) и должны быть *гомоморфны* относительно свойств B_j и $W_j\}$), (3)

где все символы имеют тот же смысл, что и в разд. 2.4.

Отношение между конкретной и общей представляющими системами ИИ задаются набором отображений конкретизации (абстрагирования, по одному для каждой переменной и параметра из этих систем). Будем называть этот набор ***каналом конкретизации абстрагирования*** и обозначать его $\mathcal{E}$. Тогда

$\mathcal{E}=\{\{V'_i, V_i, e_i)|i \in N_n, e_i$ - определяются уравнением (4 р.2.4) и должны быть *изоморфны* относительно свойств V'_i и $V_i\}$,

$\{(W'_j, W_j, \varepsilon_j) | j \in N_m, \varepsilon_j$ - определяются уравнением (5 р.2.4) и должны быть изоморфны относительно свойств $W'_j, W_j \})$.

(4)

Можно рассмотреть канал наблюдения из системы объекта непосредственно в общую представляющую систему ИИ. Однако этот канал можно получить из двух каналов, определяемых уравнением (3) и (4). Он состоит из триплетов

$$(A_i, V_i, o_i \circ e_i^{-1}) \text{ и } (B_j, W_j, \omega_j \circ \varepsilon_j^{-1}),$$

где символ ○ обозначает **композицию**.

Теперь можно определить ***исходную систему ИИ*, как пятерку**

$$\mathbf{S}=(\mathbf{O}, \mathbf{I}', \mathbf{I}, \mathcal{Q}, \mathcal{E}). \quad (5)$$

На рис. 1 изображены **эти пять компонент, а также их связи с дометодологическими посылками (исследователь, объект, цель исследования и т. д.) и с системами ИИ более высоких эпистемологических уровней.**

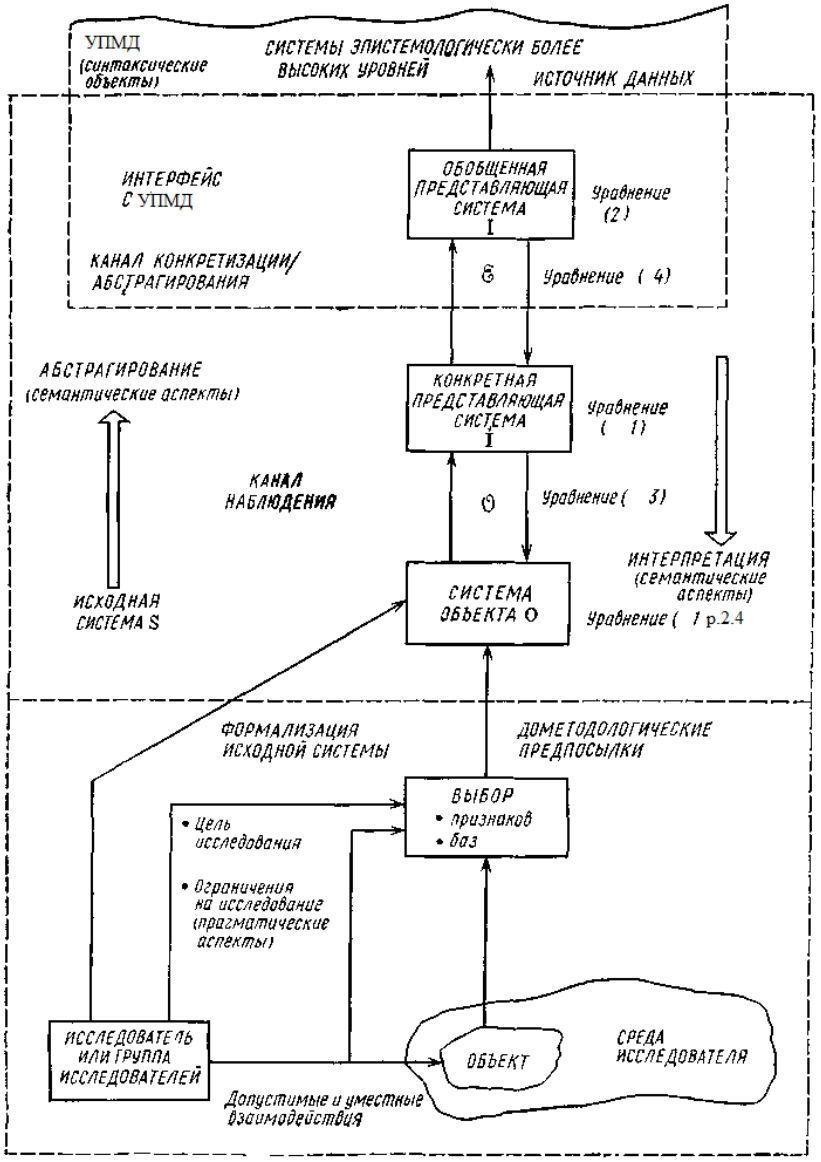


Рис. 1. Концептуальные элементы, используемые для определения исходной системы

Рисунок также дает представление об **основных методологических понятиях**, связанных с **исходной системой ИИ**:

1. С одной стороны, исходная система ИИ представляет связи с реальным миром. Они проходят через **систему объекта О и канал наблюдения Q**. С другой стороны, **исходная система ИИ связана с УПМД через общую представляющую систему ИИ I и канал конкретизации/абстрагирования E.** Эти две компоненты (**I** и E) представляют **интерфейс между конкретной предметной областью и УПМД** . Данный интерфейс, находящийся на самом нижнем эпистемологическом уровне, очень важен, поскольку любой интерфейс на более высоком уровне опирается на него.
2. По существу **концептуальная схема УПМД —это специальный язык для описания важных интеллектуальных задач.** Область УПМД ограничена *синтаксическими аспектами* решения интеллектуальных задач. Эти аспекты представляются через разные методологические отличия общих представляющих систем ИИ и их эпистемологически более нысоких аналогов. Таким образом, **реализация УПМД может быть разработана и описана только в терминах общих представляющих систем ИИ и их развития на более высоких эпистемологических уровнях**. При применении УПМД в конкретном исследовании или для какой-то другой цели соответствующие *семантические аспекты* вводятся через **O**, **I′**, Q и E данной исходной системы ИИ. С абстрагированием связаны функции o_i, ω_j, e_i^{-1} и ε_j^{-1}; конкретизация характеризуется функциями e_i, ε_j и разбиениями $o_i^{-1} = A_i/o_i$, $\omega_j^{-1} = B_j/\omega_j$. *Прагматические аспекты* вводятся на дометодологическом уровне. К ним относятся **цель и ограничения на определенные действия** (научные исследования, системное проектирование и т. д.). Некоторые из этих прагматических аспектов находят свое отражение в формулировках языка УПМД.
3. Исходная система ИИ называется исходной по двум причинам. С одной стороны, например, для использования ИИ в научных исследований она является *источником эмпирических данных,* т. е. **источником описанных на языке УПМД абстрактных представлений явлений реального мира.** С другой стороны, для таких работ, как инженерное проектирование, она является *источником интерпретаций абстрактных данных,* которые или определяются пользователем, или выводятся УПМД.

При **определении системы на объекте полезно различать два типа переменных (или соответствующих признаков).** Мы назвали их **входными и выходными переменными.** Все, что в дальнейшем будет говориться об отличии между входными и выходными переменными,

равным образом относится к конкретным и обобщенным переменным, а также соответствующим свойствам.

Дихотомия входных и выходных переменных возникла из практических соображений. Она отражает в основном точку зрения пользователя, которая, в свою очередь, повлияла, а в некоторых случаях и определила цель, с которой задавалась система ИИ.

Выходные переменные исходной системы ИИ рассматриваются пользователем как переменные, значения которых при соответствующих значениях параметров определяются внутри системы ИИ, в отличие от *входных переменных,* значения которых задаются извне. Все **факторы, влияющие на определение входных переменных, будем называть *средой системы ИИ*.**

Системы с входными и выходными переменными будем называть *направленными системами ИИ,* а системы, у которых переменные не классифицированы таким образом, *нейтральными системами ИИ.* Согласно этой терминологии исходная система ИИ, определяемая уравнением (15), также как и три ее примитивные системы ИИ (**O, I′, I**) являются нейтральными. Чтобы преобразовать их в направленные системы ИИ, нужно, чтобы **в их определении все переменные (и свойства)** были объявлены как входные или выходные. Пусть, например, для системы ИИ **I** такое объявление сделано с помощью функции

$$u{:}N_n \rightarrow \{0,\ 1\},$$

такой, что если $u(i)=0$ или $u(i)=1$, то это значит, что переменная v_i является соответственно входной или выходной. Любую n-ку

$$\mathbf{u}=(u(1), u(2),\ldots,u(n)),$$

задающую определенный статус для всех переменных системы ИИ, назовем *определителем входа-выхода.* Ясно, что для n переменных всего может быть 2^n объявлений входов-выходов, каждый из которых имеет свой определитель входа-выхода **u**.

Разумеется, **переменным v'_i и признакам a_i** соответствует тот же определитель входа-выхода. Определение любой из трех примитивных систем ИИ **O, I′, I** можно легко превратить в определение ее направленного аналога, если добавить к нему конкретный определитель входа-выхода. Обозначим направленные аналоги нейтральных систем ИИ теми же символами, но с добавлением знака ^. Тогда

$$\hat{\mathbf{O}} = (\{a_i, A_i) \mid i \in N_n\}\ \mathbf{u}\ \{(b_j, B_j) \mid j \in N_m\}), \qquad (6)$$

$$\hat{\mathbf{I}} = (\{o,, V,) \mid i \in N_n\}\ \mathbf{u}\ \{(ш,, W_j) \mid j \in N_m\}), \qquad (7)$$

$$\hat{\mathbf{I}} = (fa, V_f) \mid i \in N_n\}\ \mathbf{u}\ \{(\hat{}, W,) \mid j \in N_m\}), \qquad (8)$$

где $\hat{\mathbf{O}}, \hat{\mathbf{I}}', \hat{\mathbf{I}}$ — направленные аналоги нейтральных систем ИИ **O, I′, I**. Направленная исходная система ИИ определяется пятеркой

$$\hat{S} = (\hat{\mathbf{O}}, \hat{\mathbf{I}}', \hat{\mathbf{I}}, Q, E). \quad (9)$$

Отличие входных и выходных переменных на уровне исходных систем ИИ выражено не очень ярко. Оно становится более явным на более высоких эпистемологических уровнях, **на которых описываются разного рода отношения между переменными**. Поскольку считается, что входные переменные не определяются внутри системы ИИ, то их состояния рассматриваются как условия, определяемые средой и имеющие влияние на выходные переменные. Таким образом, **отношения между переменными в направленной системе ИИ выражаются** условными утверждениями вида «если *x*, то *y*», **где *x* описывает состояние входных переменных (определяемое средой), а *y* — некоторое свойство этой системы ИИ.** В отличие от направленных, **в нейтральных системах ИИ отношения между переменными описываются простыми утверждениями вида «*y* истинно», где *y* — также описывает свойство системы ИИ.** Выходные переменные направленной системы ИИ также могут влиять на ее входные переменные, но это влияние, если оно имеет место, осуществляется не через систему, а, как это показано на рис. *2,а,* через среду.

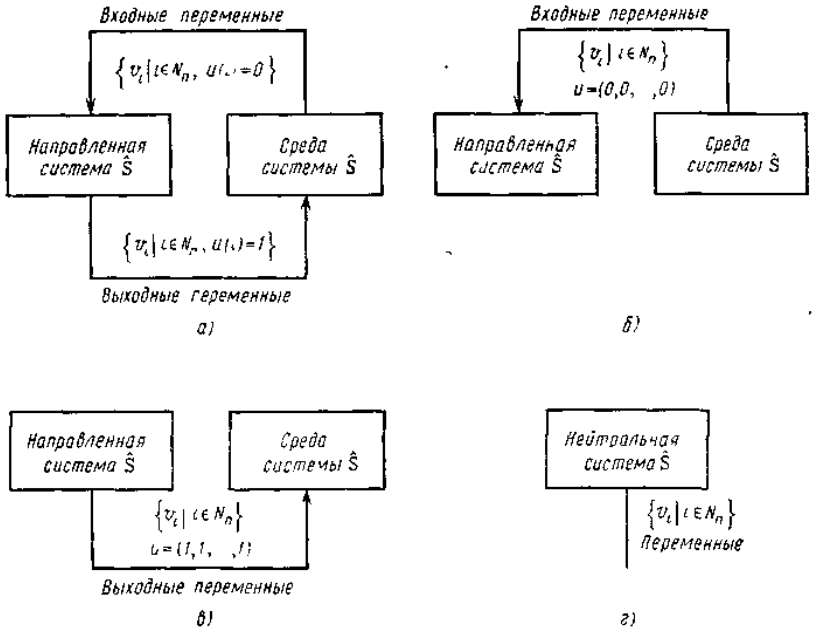


Рис. 2. Методологические отличия направленных и нейтральных исходных систем ИИ

Таким образом, свойства входных переменных направленной системой ИИ не являются предметом исследования в рамках этой системы. Существует два типа *вырожденных направленных систем.*

1**. Направленные системы без выходных переменных (рис. 2,б), т. е. системы с u= (0, 0,..,0).** Эти системы методологически бесплодны. В самом деле, любая такая система имеет только входные переменные, которые по определению полностью задаются средой, и, следовательно, их свойства невозможно представить и исследовать внутри самой системы ИИ. Таким образом, в системе нечего описывать и изучать. Любое утверждение, которое можно сформулировать внутри системы, бессмысленно, так как оно **содержит только условие, но не следствие**. Направленные системы этого типа исключены из схемы УПМД как бессмысленные. Следовательно, для *n* переменных имеется только 2^n—1 осмысленных объявлений входа-выхода.

2. **Направленные системы ИИ без входных переменных (рис. 2,в), т.е. системы с u =(1, 1,..., 1).** Эти системы методологически интересны, поскольку для них можно сформулировать **содержательные утверждения**. Однако эти утверждения не могут быть условными,

поскольку в таких системах нет входных переменных, на мнорых можно было бы сформулировать эти условия. Таким образом, эти утрвеждения оказываются столь же существенны, как и у терждения, сформулированные для аналогичных нейтральных систем ИИ, т. е. для нейтральных систем с такими же наборами переменных, параметров и других компонентов. Однако, как это показано на рис 2,*в* и *г,* между системами этих двух типов есть некоторой разница. В то время как все переменные в таких вырожденных направленных системах ИИ объявляются как выходные переменные, переменные в соответствующих нейтральных системах ИИ никак не объявляются. А поскольку они никак сначала не объявлены, то в процессе исследования их при необходимости или при желании можно объявить как входные или выходные переменные и сделать систему направленной. Отметим, что хотя эти два типа систем ИИ концептуально различны, они являются эквивалентными в том смысле, что могут быть преобразованы одна в другую просто за счет включения или исключения **u**.

Свойства среды направленной системы ИИ могут быть и неизвестны, и, следовательно, при данном наборе параметров нельзя предсказать состояния ее входных переменных. Однако такое отсутствие информации на саму систему ИИ не влияет, так как состояния входных переменных участвуют в утверждениях только как условия. Для некоторых направленных систем ИИ состояние ее входных переменных, определяемое средой, может быть полностью известно. Эта информация также не влияет на описание системы, но может быть использована при решении задач, в которых участвует данная система ИИ. Бывает и так, что **входные переменные полностью контролируются исследователем, т. е. <u>он сам представляет собой среду</u>.**

Для нейтральных систем ИИ никакой среды нет (рис. 2,г). При замене нейтральной системы ИИ на направленную ИИ вводится среда, и если **u**≠*(*1, 1, ..., 1), некая информация, содержавшаяся в системе ИИ, перемещается в среду. Таким образом, полученная направленная система ИИ содержит меньше информации, чем исходная нейтральная. Нейтральную систему ИИ с n переменными можно заменить 2^n—1 разными направленными системами ИИ (по числу разных n-мерных определителей входа-выхода *u,* но для системы без входных переменных (рис. 2*,в)* довольно одной тривиальной замены. Следовательно, мы можем считать, что нейтральная система ИИ может быть заменена направленной 2^n —2 нетривиальными способами.

Отличия между нейтральными и направленными системами ИИ и между четкими и нечеткими каналами наблюдения — <u>это еще два</u>

<u>методологических отличия исходных систем ИИ.</u> Они независимы друг от друга и от отличий в **свойствах множеств состояний и параметрических множеств**. Любая исходная система ИИ является или нейтральной, или направленной, а каналы наблюдения ее переменных или все четкие, или все нечеткие, или разных типов. Таким образом, новые отличия дают 2×3=6 возможностей. Кроме того, в исходную систему ИИ могут входить переменные разных методологических типов.

Обозначим общее число методологических отличий, определенных для уровня исходных систем ИИ, через #S. Тогда при вполне разумном предположении, что число параметров не превышает 9 *(m*≤9*),* мы получим

$$\#S=6\times\sum_{i=1}^{k}\binom{9}{i}\times\sum_{j=1}^{m}\binom{9}{j} \qquad (20)$$

где $k = \min(9, n)$.

Если запретить, чтобы в исходных системах ИИ переменные и параметры были смешанных методологических типов, то число методологических отличий станет равным 6×9×9 = 486. Если кроме этого рассматривать только дискретные переменные, то это число сократится до 6×5×5=150. В этом диапазоне и показаны в данной работе некоторые реализационные аспекты УПМД.

Методологические отличия, определенные для исходных систем ИИ, весьма важны, поскольку они приложимы и ко всем системам ИИ более высоких эпистемологических уровней. Проиллюстрируем эти отличия, а также другие свойства исходных систем ИИ на следующих двух примерах.

Пример 1. Пусть объектом исследования являются лесопосадки деревьев твердых лиственных пород на западе Киевской области. Лесники обычно помечают деревья для выборочной рубки через довольно регулярные промежутки. Главные цели маркировки деревьев для рубки — поддержание или улучшение качества леса и вывоз лесоматериалов, стоимость которых достаточно велика, чтобы это было выгодно и владельцу и лесообрабатывающему предприятию.

<u>Цель определения исходной системы ИИ для данного объекта</u> — получение характеристик деревьев, маркированных для рубки, их оценка и разработка более подходящих и точных руководств для маркировки деревьев в будущем.

Базой в данном примере является **группа деревьев**, представляющая собой выбранный для исследований лесной массив. Пусть каждое исследуемое дерево помечается целым числом. Тогда функция ω дает отображение «один в один», равно как и функция ε.

Пусть для исследования объекта было отобрано семь свойств. Приведем их описания и определим соответствующие переменные.
Вид дерева: свойство a_1. Для исследования в данном лесном массиве для всех видов деревьев выделено только четыре класса деревьев. Следовательно, для представления этого свойства нужна **конкретная переменная с четырьмя состояниями**. На рис. 3,*а* определена функция o_1, связывающая свойство с этой переменной.

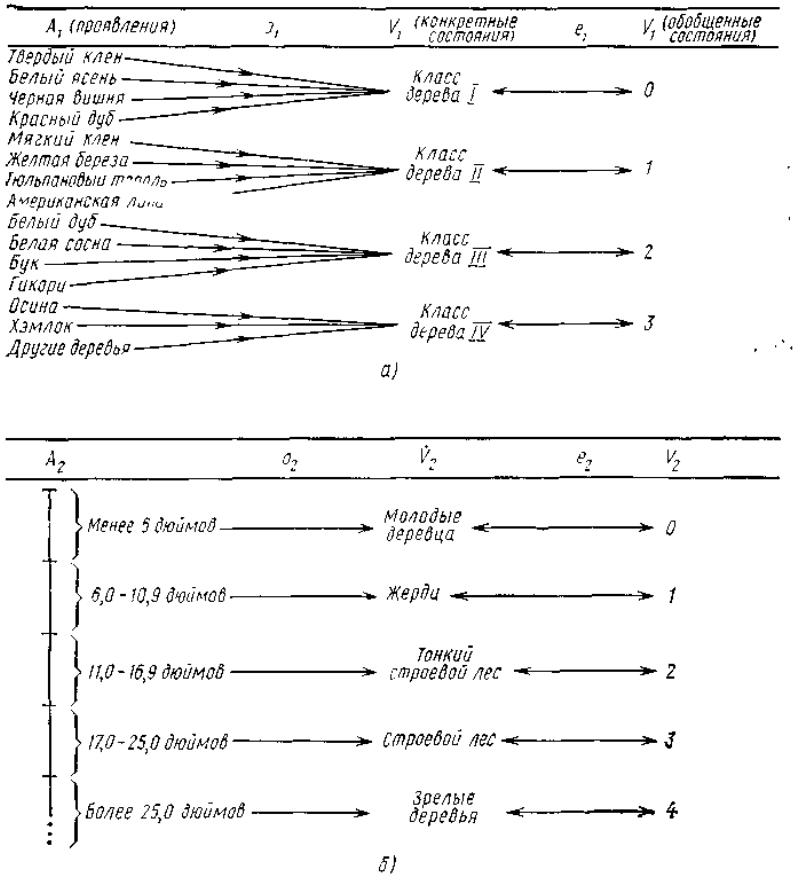


Рис.3. Определение переменных для признаков a_1 и a_2 из примера 1

На том же рисунке определяется функция e_1, которая, как всегда, представляет собой простую схему переобозначения. Множества A_1 и V'_1 никакими свойствами не обладают, и, следовательно, всевозможные свойства целых чисел из множества V_1 и не могут быть использованы.

Канал наблюдения является четким, т. е. непосредственно представляется функцией o_1.
Диаметр (диаметр на уровне груди): свойство a_2. Более точно это свойство определяется как диаметр ствола на уровне четыре с половиной фута от земли. Хотя с помощью рулетки диаметр может быть измерен с точностью до 0,1 дюйма, при оценке объема древесины этот диаметр оценивается или измеряется с точностью до 1—2 дюймов. Однако для выборочной рубки деревьев достаточно разбить диаметры стволов всего на пять категорий. Они определяются на рис.3,б вместе с функциями o_2 и e_2.
Несмотря на то, что возле границ блоков разбиения A_2/o_2 может иметь место некоторая нечеткость измерения, канал наблюдения o_2 можно рассматривать как четкий, так как эта нечеткость практического значения не имеет. Множества A_2, V'_2, V_2 можно рассматривать как линейно упорядоченные с метрическим расстоянием, и, следовательно, если нужно, то можно для множества V_2 воспользоваться свойствами целых чисел.
Товарная высота: свойство a_3. Несмотря на то, что этот параметр может быть измерен довольно точно, достаточно только оценить его значение и выделить три диапазона: меньше 24 футов, от 24 до 48 футов, больше 48 футов. Эти диапазоны можно представить состояниями 0, 1, 2 множества V_3. Порядок и расстояние для целых чисел из V_3 можно использовать в исследованиях.
Класс кроны: свойство a_4. Класс кроны описывает размер и положение верхней части дерева относительно верхушек соседних деревьев. Подавленные деревья заслонены другими деревьями, относительно мало влияют на соседей, и им в борьбе за существование плохо помогает освобождение от соседей. С другой стороны доминирующие деревья оказывают большое влияние на соседей. Это свойство считается очень важным показателем того как дерево реагирует на освобождение от соседей и насколько оно способно дать хорошую древесину. Лесоводы разработали четкие стандарты для классификации класса кроны на четыре состояния: доминирующая, кодоминирующая, средняя, подавленная которые могут быть соответственно представлены состояниями 0, 1, 2, 3, сохраняющими в V_4 линейный порядок.
Сорт согласно стандартам ЛСУ (Лесная служба Украины): свойство a_5. Значение этого свойства зависит от числа, размера и взаимного расположения ветвей, сучков и других знаков наличия в дереве узлов. Выделяется четыре сорта (состояния V_5). Они хорошо определены и опираются на стандарты Навдональнои ассоциации производителей лесоматериалов и Лесной службы министерства сельского хозяйства

Украины. Они называются сортами ЛСУ 1 2 3 и местного применения. Их в V_5 можно представить в виде целых чисел 0, 1, 2, 3, сохраняющих отношение порядка.

Дефекты: свойство a_6. Значения этого свойства говорят о дефектах дерева, таких, как гнилостность, искривления и разветвленность, уменьшающие объем древесины, которую можно получить из дорева того же размера, но без дефектов. В V'_6 выделяют три состояния: без дефектов или с небольшими дефектами, частично испорченное, бракованное дерево. Эти состояния в V_6 можно, сохраняя порядок, представить значениями 0, 1 и 2. Так как эта переменная представляет собой оценку внутренних дефектов по внешним признакам, она может быть ошибочной, даже если ее делает очень компетентный наблюдатель. Поэтому в данном случае желательно использовать нечеткий канал наблюдения, позволяющий наблюдателю в каждом отдельном наблюдении выразить степень неуверенности в достоверности его оценки. При этом функция o_6 явно не задается, а определяется самим наблюдателем.

Маркировка дерева: свойство a_7. Исследуемые деревья либо маркируются для рубки, либо нет. Пометим проявление этого свойства в множестве V_7 цифрами 1 и 0. Множество V'_7 никакими свойствами не обладает.

Мы видим, что определенная в этом примере исходная система является нейтральной. Однако для формулирования правил маркировки деревьев, предназначенных на сруб, система должна быть переопределена как направленная с входными переменными v_1—v_6 и выходной переменной v_7. Один канал наблюдения нечеткий, а остальные четкие, поэтому в исходной системе смешаны четкие и нечеткие переменные. Множество параметров свойствами не обладает, а множества состояний имеют два, а, возможно, и три типа: без свойств, линейно упорядоченные и, может быть, линейно упорядоченные с метрическим расстоянием.

Пример 2. Пусть объектом исследования является женщина, страдающая пневмонией. Целью исследования является наблюдение так называемого комплексного анализа крови пациентки в течение определенного периода времени. Это позволяет определить, улучшается ли ее состояние само по себе или следует установить определенный режим лечения.

Параметром является время. Измерения делаются один раз в день в 7 ч утра в течение всего сентября 2016 г. Конкретное параметрическое множество представляет собой дни месяца 9.01.16, 9.02.16, ...,9.30.16, которые в соответствующем обобщенном параметрическом множестве

можно представить в виде целых чисел 0, 1, 2, ...,29, что сохраняет порядок и расстояние.

Каждое измерение представляет собой определение состояний четырех описанных ниже переменных по 10 см^3 крови, взятой у пациентки. Все переменные являются численными и определяются через четкие каналы наблюдения o_i, такие, что соответствующее разбиение o_i/A_i состоит из блоков одинакового размера. Переменная принимает значение, равное середине интервала, в который попадает значение свойства. По данным четырем переменным вычисляются другие переменные, входящие в комплексный анализ крови.

Число красных кровяных телец: свойство a_1 определяется как число разных кровяных телец в 1 мм^3 крови и обычно принимает у женщин значения в диапазоне 4.2—5.4 млн/мм^3. Оно измеряется с точностью до 10 000. Множество состояний V'_1 состоит из значений 4.20, 4.21,...,5.4 млн/мм^3. Согласно предшествующему общему замечанию относительно природы каналов наблюдения в данном примере, блоки $o_1^{-1}(x)$ разбиения A_1/o_1, представляемые этими значениями $x \in V'_1$ будут иметь такой вид:

$4.105 \le o_1^{-1}(4.20) < 4.205$,

$4.205 \le o_1^{-1}(4.21) < 4.215$,

.....

$5.395 \le o_1^{-1}(5.40) < 5.405$.

Конкретные состояния 4.20, 4.21 ...,5.40 с помощью функции o_1^{-1} могут быть отображены в целые 0, 1,..., 120, так что упорядоченность и расстояние в V'_1 сохраняются во множестве целых V_1.

Число белых кровяных телец: свойство a_2 определяется как число белых кровяных телец в 1 мм^3 крови и обычно у женщин находится в диапазоне

5—10 тыс./мм^3. Измеряется с точностью до 100. Множество состояний V'_2 состоит из значений 5.0, 5.1,... ..., 10 тыс/мм^3, представляющих блоки

$4.95 \le o_2^{-1}(5.0) < 5.05$,

$5.05 \le o_2^{-1}(5.1) < 5.15$

.....

$9.95 \le o_2^{-1}(10.0) <: 10.05$.

Функция e_2 отображает целые числа 0, 1,...,50 на числа 5.0, 5.1,..., 10,0 соответственно.

Гематокрит: свойство a_3 определяется как процент красных кровяных телец от общего объема крови, взятой на анализ. Ожидаемый диапазон 37—47%. Измеряется с точностью до 1%. Множество состояний V'_3 состоит из значений 37, 38, ...,47%, представляющих следующие блоки:

$36.5 \le o_3^{-1}(37) < 37.5$,
$37.5 \le o_3^{-1}(38) < 38.5$
.....
$46.5 \le o_3^{-1}(47) < 47.5$

Функция e_3 отображает целые числа 0, 1,..., 10 на числа 37, 38,... 47 соответственно.

Гемоглобин: свойство a_4 оценивается как количество гемоглобина и граммах на 100 мм крови, диапазон 12—16 г/мм. Измеряется с точностью до 0,01 г. Множество состояний V'_4 состоит из значений 12.00, 12.01,..., 16.00; функции o_4 и e_4 определяются так же, кпк дли прн.шакон a_1, a_2, a_3.

И комплектный анализ крови входят еще несколько переменных, значения которых вычисляются по определенным формулам из значений введенных базовых переменных. Одна из таких переменных, называемая *средний размер частицы* (СРЧ), определяется как средний объем красного кровяного тельца и измеряется и кубических микрометрах. Ожидаемый диапазон 82—92 мкм^3. Она вычисляется с точностью до 1 мкм^3 по формуле

$$СРЧ = \frac{Гематокрит \ 10}{Число \ красных \ кровяных \ телец}$$

Если исходная система содержит переменные, определяемые через другие переменные, как СРЧ из предыдущего примера, то она содержит и введенные исследователем искусственные отношения между переменными. Эти искусственные отношения должны быть описаны при определении исходной системы и отделены от подлинных отношений, возникших непосредственно при исследовании явления.

2.8. Системы данных

Исходная система ИИ — это схема, по которой могут быть сделаны наблюдения отобранных признаков. Если канал наблюдения четкий, то любое реальное наблюдение представляется в виде **упорядоченной пары, состоящей из значения полного параметра, при котором было сделано наблюдение, и зафиксированного полного состояния переменных.** Так как при одном значении параметра может быть сделано только одно наблюдение, **множество этих упорядоченных пар является функцией, отображающей полное параметрическое множество в полное множество состояний.** Эта **функция и представляет собой данные или, точнее, четкие данные**.

В УПМД всегда предполагается, что **данные должны быть представлены как обобщенные параметры и переменные** (см. рис.1

р.2.5). Следовательно, при формализации понятия данных мы можем ограничиться рассмотрением только обобщенной направляющей системы ИИ **I**, как она определена в (12). Пусть

$\mathbf{W} = W_1 \times W_2 \times \ldots \times W_m$,
$\mathbf{V} = V_1 \times V_2 \times \ldots \times V_n$.

Тогда *четкие данные* представляются функцией

$$d: \mathbf{W} \to \mathbf{V}. \qquad (21)$$

Функция *d* любому значению полного параметра ставит в соответствие одно полное состояние переменных.

В то время как представляющая система ИИ **I** описывает **только потенциальные состояния переменных**, функция *d* дает **информацию об их действительных состояниях при неограниченном параметрическом множестве.** Система **I** в соединении с функцией *d* можно рассматривать как систему ИИ более высокого эпистемологического уровня (уровня 1). Будем называть такую систему ***системой данных*** и обозначать **D**. Тогда

$$\mathbf{D} = (\mathbf{I}, d). \qquad (22)$$

Несмотря на то, что в этом определении отсутствует какой-либо семантический оттенок, оно достаточно и вполне удобно для разработки и описания методологических характеристик УПМД. Однако для любого конкретного применения в формулировке должен быть отражен и **смысл данных *d***. Это можно сделать, заменив представляющую систему ИИ **I** в уравнении (22) соответствующей исходной системой ИИ **S**. Получившуюся в результате этой замены систему назовем ***системой данных с семантикой*** и обозначим $^S\mathbf{D}$. Таким образом,

$$^S\mathbf{D} = (\mathbf{S}, d), \qquad (23)$$

где *d* — та же самая функция, что и в уравнении (22). Однако в данном случае функция *d* связана с системой **S** следующим образом: если наблюдение, описываемое с помощью $o_i \circ e^{-1}(x_i) = y_i$ для всех $i \in N_n$ (где x_i — предполагаемое проявление свойства a_i, а y_i — соответствующее состояние переменной v_i), связывается со значением полного параметра $w \in \mathbf{W}$, то

$$d(\mathrm{w}) = \mathbf{v},$$

где $\mathbf{v} = (y_1, y_2, .., y_n) \in \mathbf{V}$. В зависимости от рассматриваемой задачи функция *d* на самом деле может быть определена по крайней мере тремя разными способами. **Во-первых**, она может быть результатом наблюдений или измерений, как это бывает при всевозможных эмпирических исследованиях. **Во-вторых**, ее можно вывести из систем более высоких уровней, как это будет показано в последующих разделах. **В-третьих**, она может быть из каких-то соображений

определена самим пользователем, как это бывает в интеллектуальных задачах.

Системы данных **D** и ^{S}**D** нейтральные, так как они определены, соответственно через нейтральную представляющую систему ИИ **I** и нейтральную исходную систему ИИ **S**. Превращение этих систем в их направленные аналоги $\hat{\mathbf{D}}$ и $^{S}\hat{\mathbf{D}}$ труда не представляет. Нужно только заменить **I** на $\hat{\mathbf{I}}$, а S на $\hat{S}$. Таким образом,

$$\hat{\mathbf{D}}=(\hat{\mathbf{I}}, d), \tag{24}$$

$$^{S}\hat{\mathbf{D}}=(\hat{S}, d) \tag{25}$$

— это *направленные системы данных* без семантики и с семантикой соответственно.

Если переменные определяются через нечеткие каналы наблюдения, то каждое наблюдение записывается как упорядоченная пара, состоящая из **значения полного параметра, с которым связано наблюдение, и *n*-ки $(h_1, h_2,...,h_n)$ функций**

$$h_i: V \rightarrow [0, 1],\ i \in N_n, \tag{26}$$

где $h_i(y)$ выражает **степень уверенности** в том, что y является наблюденным состоянием переменной v_i. **Формализуем понятие нечетких данных**. Пусть

$\tilde{\mathbf{V}} = \{V_1 \rightarrow [0,1]\} \times \{V_2 \rightarrow [0,1]]\} \times ... \times \{V_n \rightarrow [0,1]]\}$.

Тогда *нечеткие данные* представляются функцией

$$\tilde{d} : \mathbf{W} \rightarrow \tilde{\mathbf{V}}\ . \tag{27}$$

Для любого значения полного параметра $w \in \mathbf{W}$

$$\tilde{d}\ (\mathbf{w})=\mathbf{h},$$

где $\mathbf{h} = (h_1, h_2, .., h_n) \in \tilde{\mathbf{V}}$.

Если данные являются нечеткими, то определения систем данных следует модифицировать, заменив в (22) — (25) функцию d функцией $\tilde{d}$. С каким типом данных — четким или нечетким — мы имеем дело, всегда ясно по контексту, поэтому не имеет смысла для систем данных с четкими и нечеткими данными использовать разные обозначения.

Если в определении системы данных с нечеткими данными входит исходная система ИИ **S**, то функции d и **S** связаны следующим образом, если наблюдение связано со значением полного параметра, описываемого как

$\tilde{o}_i(x_i,\ y'_{i,k})=z_{i,k}$,

и

$e^{-1}(y'_{i,k})=y_{i,k}$

для всех $i \in N_n$, где x_i — возможное проявление признака a_i, то

$h_i(y_{i,k})= z_{i,k}$

для всех $y_{i,k} \in V_i$, и всех $i \in N_n$.

Четкие данные могут быть представлены в самом разном виде. Пусть **<u>стандартной формой представления</u> дискретных переменных и параметров будет <u>матрица</u>**

$$\mathbf{d}=[v_{i,w}]$$

элементами которой $v_{i,w}$ являются состояния переменных v_i, наблюденные при соответствующих значениях полного параметра w (рис.1,*а*).

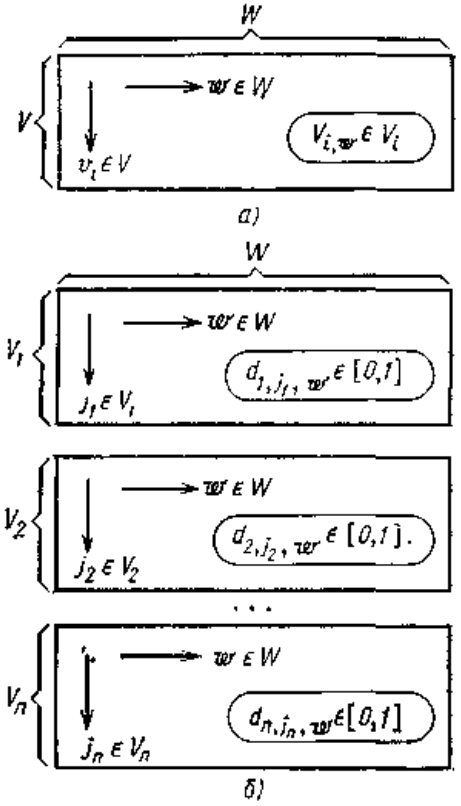


Рис 1. Стандартные формы представления для дискретных переменных а) матрица **d** четких данных d, б) трехмерный массив **d** нечетких данных $\tilde{d}$

Каждый столбец матрицы **d** задает полное состояние, наблюденное при данном **w**, а каждая строка — все наблюдения одной переменной на параметрическом множестве **W**. Если **W** линейно упорядочено, то и столбцы в матрице **d** должны быть упорядочены точно таким же образом. **Если используются несколько параметров, например группа — время, пространство нескольких измерений или пространство — время, то может оказаться удобнее использовать другие формы представления.** Некоторые из этих форм будут продемонстрированы в различных примерах.

Для нечетких данных **стандартной формой представления**, подобной матрице **d**, является **трехмерный массив**

$$\tilde{\mathbf{d}} = [\tilde{d}_{i,j_i,\mathbf{w}}],$$

элементами которого являются **значения степени уверенности в том, что при значении параметра w наблюдалось состояние j_i переменной v_i.** Понятно, что $i \in N_n$, $j_i \in V_i$, **w=W**, а $d_{i,j,\mathbf{w}} \in [0,1]$.

Массив представляет собой **d** матриц (страниц, плоскостей), по одной для каждой переменной. Столбец **w** в матрице переменной v_i задает функцию h_i, определяемую уравнением (26) и соответствующую наблюдению, идентифицируемому **w**.

Для пояснения различных характеристик систем данных и их представлений рассмотрим более детально некоторые примеры **конкретных систем данных (т. е. систем данных с семантикой).**

Пример 1. При изучении поведения животных этологи используют такие методы, которые как можно меньше беспокоят исследуемых животных, находящихся в естественной среде обитания. Одним из таких методов изучения поведения группы животных является съемка фильмов, а затем определение уже по фильму характера поведения животных. Для каждого конкретного вида животных обычно определяются **характерные позы и движения**. Этологи часто **специфицируют их рисунками, сопровождаемыми часто вербальными описаниями**. Например, на рис. 2,*а* изображены важнейшие позы чаек, которым даны поясняющие названия; «отдых», движение «вперед» и т. д.

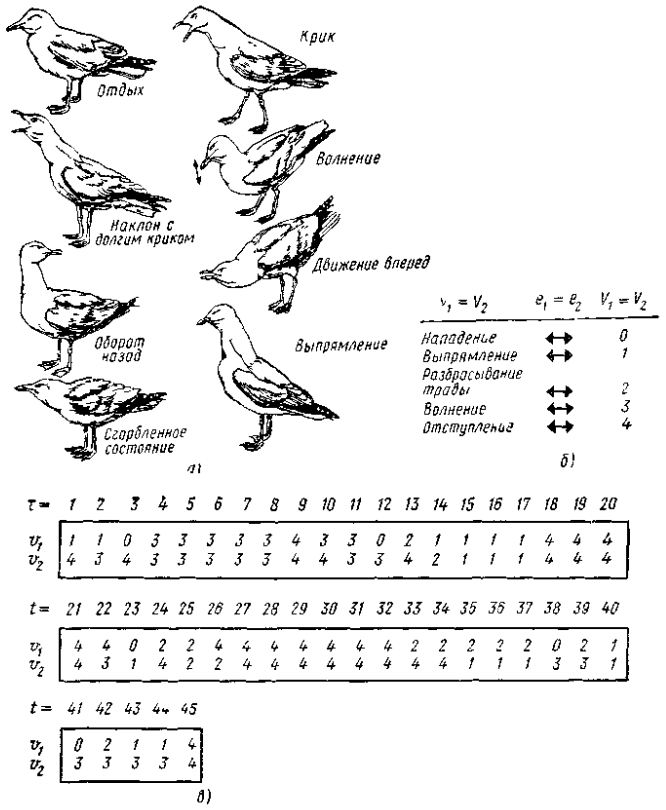


Рис 2. Данные, описывающие пограничное столкновение между двумя чайками (пример 1)

Каждая из них также описывается вербально, например «волнение начинается с того, что чайка наклоняется над гнездом (или любой другой вмятиной в земле, похожей на гнездо, например, над следом ноги), а затем начинает ритмично покачивать головой вверх-вниз».

Объектом исследования в данном примере являются две чайки: чайка I и чайка II. Переменные определены на следующих характеристиках:

a_1 - тип поведения чайки I,

a_2 - тип поведения чайки II.

Характеристики наблюдаются во времени. Период наблюдения 90 с. Он разделен на интервалы по 2 с, каждый из которых представляет адно наблюдение. Таким образом, конкретное временное множество (являющееся **параметрическим множеством** для данного примера), скажем
$T'=\{ t_1, t_2,.., t_{45}\}$, может быть определено разбиением периода времени в 90 с с помощью следующего канала наблюдения:

$0 \le \omega^{-1}(t_1) < 2$ с,
$2 < \omega^{-1}(t_2) < 4$ с,
……..
$88\ \omega^{-1}(t_{45}) < 90$ с.

Время T' линейно упорядочено и имеет **метрическое расстояние**. При отображении T на множество целых чисел, скажем на множество $T=N_{45}$ с помощью функции

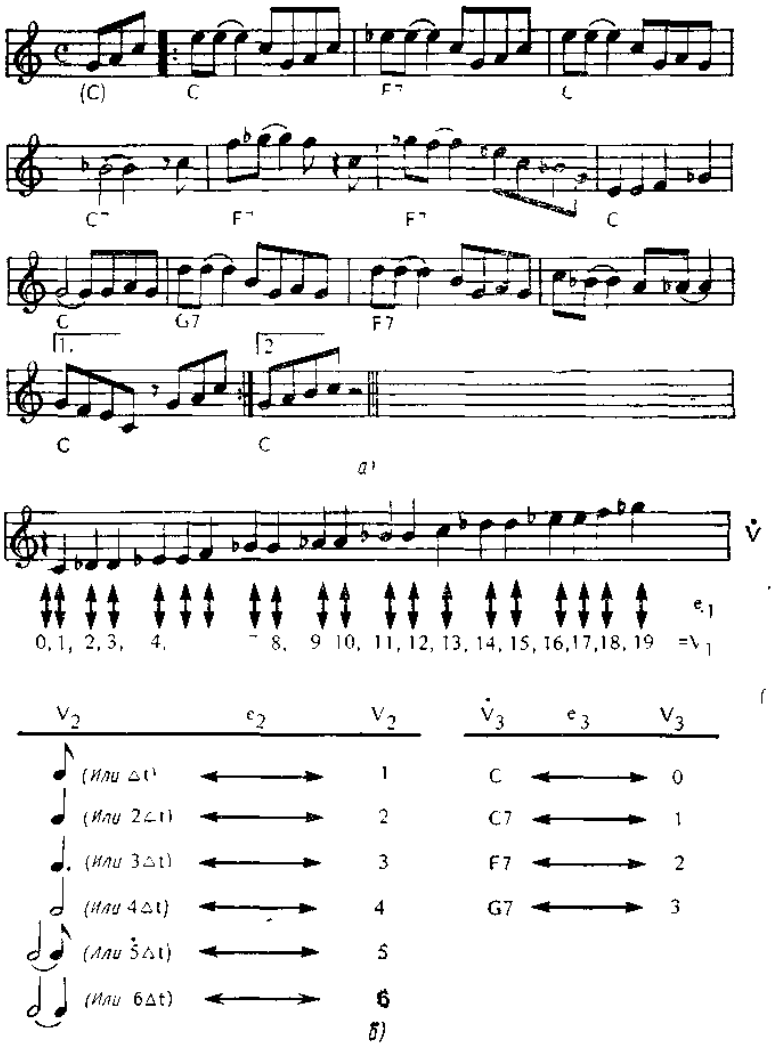
$$\varepsilon(k)=t_k(k \in N_{45}),$$

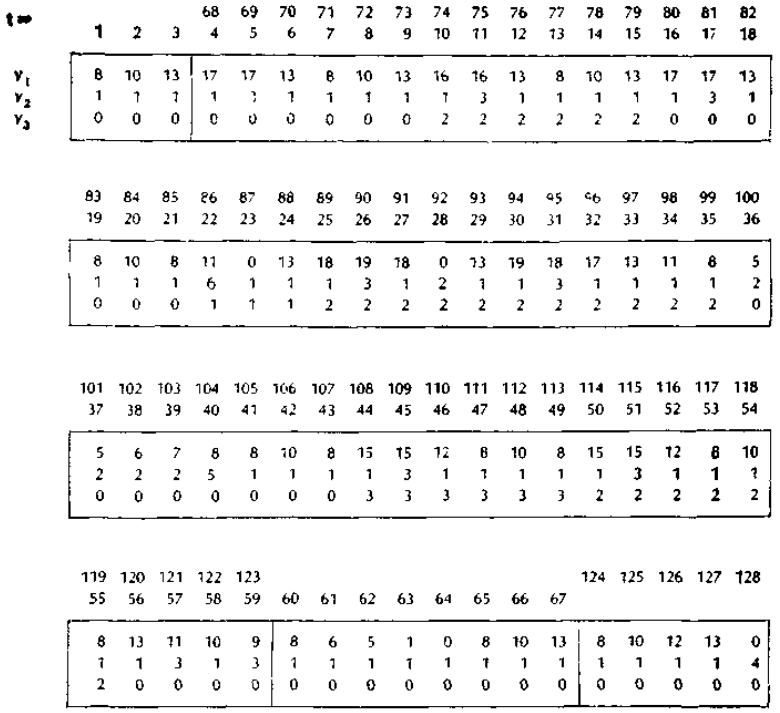
порядок и метрическое расстояние сохраняются.
Каждое наблюдение данных двух характеристик представляется теми кадрами из фильма, которые относятся к данному периоду в 2 с. Эти кадры изучаются этологом, и для каждой чайки определяется один из нескольких перечисленных выше типов поведения (состояния из множеств состояний V'_1, V'_2). В данном примере, в котором исследуется пограничный конфликт между двумя чайками, для обеих характеристик достаточно пяти одинаковых типов поведения (т. е. $V'_1 = V'_2$). Названия этих типов, а также их целые обозначения (элементы обобщенных множеств состояний V_1, V_2) приведены на рис. 2,*б*. Типы поведения, называемые «выпрямление» и «волнение», входят в число основных поз, изображенных на рис. 2,а. «Разбрасывание травы» определяется так: «неистово клюет землю, вырывает растения и разбрасывает их в стороны движениями головы». Два оставшихся типа поведения «нападение» и «отступление» хорошо характеризуются своими названиями. На рис. 2,*в* приведена матрица данных, полученная по реальному фильму. При этом $t \in T$ и $v_{1,t}, v_{2,t} \in V_1 (= V_2)$.

Определенная в этом примере система является **нейтральной системой с семантикой**. **Ее переменные дискретны, а параметром является время. Параметрическое множество (время) линейно упорядочено и обладает метрикой. Данные четкие.**
По поводу этого примера необходимо сделать два замечания. Во-первых, оба канала наблюдения

$$o_i: A_i \rightarrow V'_i \ (i=1,2),$$

где A_i —это предполагаемый набор возможных действий, одинаковы, представлены самим исследователем, и их нельзя определить математически. Они определяются по сочетанию картинки и вербального описания.
Во-вторых, любое **наблюдение (столбец матрицы данных)** представляет собой сделанное исследователем заключение относительно того, что произошло в соответствующий 2-секундный период времени. Этот подход довольно сомнителен, поскольку действия чайки (описанные в терминах типов поведения, определенных на множестве состояний) не обязательно начинаются и заканчиваются на границах 2-секундных интервалов. **В подобного рода исследовании лучше было бы временное множество T представить неявно, по изменению состояния переменных.**
Временное множество T' определяется неявно по следующему правилу: весь период наблюдения (в нашем примере 90 с) делится на временные интервалы, в течение которых ни одна из переменных (в нашем случае переменных две) не меняет своего состояния, если по крайней мере одна переменная изменяет свое состояние, один временной интервал кончается и начинается следующий. Если важно знать длительность отдельных действий, то можно ввести новую переменную — **длительность временного интервала** (измеряемую с необходимой точностью) и **запоминать значения этой переменной в качестве части данных**. *Неявное определение временных множеств* (как с введением дополнительной переменной, содержащей длительность неявно заданных временных интервалов, так и без этой переменной) часто бывает более подходящим, чем любое явное определение. Оно также бывает полезно, а часто и желательно для переменных, параметром которых является то или иное пространство.
Пример 2. Типичным объектом изучения в музыковедении является музыкальное сочинение. Пусть этим объектом в нашем примере будет современная мелодия блюза. Ее партитура, приведенная на рис.3,*а,* состоит из двух частей —мелодии и гармонии.





в)

Рис.3. Система данных, представляющая мелодию блюза (пример 2)

Мелодия приведена в обычной нотной записи, а гармония—в терминах так называемой нотации «фейк бук», часто применяемой джазовыми музыкантами. Для определения на этой мелодии блюза (или любого другого музыкального сочинения) содержательной системы нужно рассмотреть **<u>три типа признаков</u>: высоту тона, ритм и гармонию. Все они меняются во времени.** Соответствующие временные интервалы могут быть определены через длительность самой короткой ноты в сочинении. Назовем эту длительность Δt. В данном примере длительность равна 1/8. **Время (<u>как параметр</u>)** может быть определено явно или неявно. При явном задании времени элементами соответствующего временного множества T' будут метки интервалов времени $[0, \Delta t), [\Delta t, 2\Delta t)$ и т. д., а элементами обобщенного временного множества T соответствующие целые числа (например, 1, 2,... что сохранит порядок и расстояние). При неявном

определении временные интервалы, представленные в *T',* определяются длительностью звучания отдельных нот из мелодии. В данном примере принято неявное определение, представляющееся более подходящим.

Три признака — высота тона, ритм и гармония — могут быть представлены переменными самыми разными способами. Например, высота тона в мелодии может быть описана одной переменной, скажем переменной v_1, состояния которой определяют тона в нашей мелодии (см. рис.3,б). Но можно их представить и с помощью двух переменных, одна из которых задает октаву, а вторая —12 тонов хроматической гаммы. Высота тона может быть представлена и тремя переменными — одна задает октаву, вторая — один из семи основных тонов *(a, b,...,g)* в каждой октаве, и одна — для знаков вариации на полтона *b* (бемоль) и # (диез). **Ритм мелодии — это характеристика, зависящая от времени, и, следовательно, определение переменной, ее представляющей, зависит от определения времени как параметра.** Когда параметр «время» определен (как в нашем примере) неявно, переменная для задания ритма, скажем переменная v_2, должна определить длительности отдельных нот, составляющих мелодию. Эти длительности кратны Δt (или 1/8), как это показано на рис.3,*б*. Канал наблюдения o_2, с помощью которого вводятся временные интервалы $[0, \Delta t)$, $[\Delta t, 2\Delta t),...,[4\Delta t, 5\Delta t)$, совершенно ясен. Если говорить о гармонии, то пусть обобщенная переменная v_3 будет связана с соответствующей конкретной переменной v'_3 через канал абстрагирования, определенный на рис.3,*б*. В данном случае канал наблюдения o_3 представлен стандартными определениями символов гармонии из «фэйк бук» (элементы V_3). Обратим внимание на то, что гармонию также можно представить с помощью двух переменных, одна переменная для основной ноты (C, F, G) и одна для типа аккорда (C или C7).

Если переменные, задающие высоту тона, ритм и гармонию, определены так, как это показано на рис.3, то **мелодия блюза полностью описывается матрицей данных на рис.3,б. Повторяющаяся часть мелодии задана в матрице столбцами с двойным указанием времени.**

Пример 3. На рис.4,в приведена периодическая матрица, определяющая нужную последовательность сигналов светофоров на перекрестке.

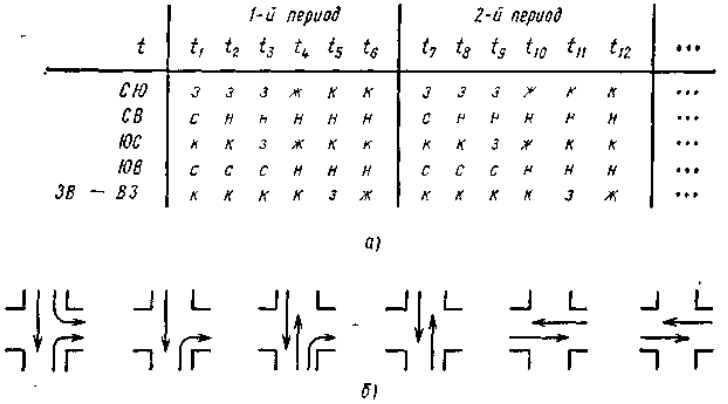


Рис.4. Описание работы светофоров на перекрестке (пример 3)

Конкретные переменные, описывающие сигналы светофоров для транспортных потоков, идущих в направлениях север — юг, юг — север, запад — восток и восток — запад, обозначены соответственно СЮ, ЮС, ЗВ и ВЗ. Каждая из этих переменных имеет три состояния: красный, желтый, зеленый (к, ж и з). Стрелка левого поворота для транспорта, следующего в направлении север—восток, обозначена СВ, а стрелка правого поворота юг-—восток — ЮВ. Эти переменные имеют два состояния: стрелка или горит, или нет (обозначаются с и н). Параметром является время. Временное множество *T'* состоит из шести интерва лов $t_1, t_2,...,t_6$, представляющих следующее разбиение интервала времени в 90 с, налагаемое каналом наблюдения:

$0 \le \omega^{-1}(t_1) < 15$, $50 \le \omega^{-1}(t_4) < 60$,
$15 \le \omega^{-1}(t_3) < 25$, $60 \le \omega^{-1}(t_5) < 80$,
$25 \le \omega^{-1}(t_3) < 50$, $80 \le \omega^{-1}(t_6) < 90$.

Реальные ситуации на перекрестке для каждого из шести временных интервалов $t_1, t_2,..., t_6$ схематически изображены на рис.4,б. Система данных полностью определяется заданием данных для одного периода. Замена матрицы данных, изображенной на рис.4,а, матрицей данных для обобщенных переменных тривиальна: поскольку множества состояний никакими свойствами не обладают, отображение на множество целых чисел может быть произвольным.

Пример 4. Покрытие моря льдом —один из признаков, за которым постоянно наблюдают климатологи. **Признаки такого рода обычно наблюдаются и в пространстве, и во времени**. Данные в этом примере получены по фотографиям со спутников и отражают процент водной поверхности в высоких широтах (от 50 до 76° ю. ш.) южных

морей, покрытых льдом. Наблюдения собраны за 1 год.
Предполагается, что канал наблюдения *о* выделяет шесть состояний, отражающих процент поверхности, покрытой льдом: льдов нет, низкий, средний, высокий, очень высокий (но меньше 100%) и полное покрытие льдами (т. е. 100%). Эти состояния обозначены целыми числами 0, 1,...,5, так что линейный порядок сохраняется. Эти состояния задают следующее разбиение отрезка [0, 100%)]:

o^{-1} (нет)=0%, $50< o^{-1}$ (высокий %)≤75%,
$0<o^{-1}$ (низкий %)≤25%, $75< o^{-1}$ (очень высокий %)<100%,
$25< o^{-1}$ (средний %)≤^50%, o^{-1} (полное покрытие) = 100%.

В данном случае имеются два параметра: время и одномерное пространство — широта. Время представлено месяцами одного года и определяется, как обычно, разбиением всего периода в 1 год на 12 интервалов. Широта измеряется с точностью до 2° и также определяется обычным образом.
На рис. 5 приведены два набора данных: один для Тихоокеанского сектора южных морей, один для Атлантического.

Пространство ⟶ I

Время ↓ T

Градусы широты

Время	50	52	54	56	58	60	62	64	66	68	70	72	74	76
Январь	0	0	0	1	1	1	1	1	2	3	3	4	4	4
Февраль	0	0	0	0	1	1	1	1	2	2	3	3	3	4
Март	0	0	0	0	0	1	1	1	2	2	3	4	4	4
Апрель	0	0	0	0	0	1	1	1	2	3	4	4	5	5
Май	0	0	0	0	1	1	1	2	3	4	4	5	5	5
Июнь	0	0	0	1	1	2	2	3	4	4	5	5	5	5
Июль	0	0	1	1	1	2	3	4	4	4	5	5	5	5
Август	0	1	1	1	2	3	4	4	4	5	5	5	5	5
Сентябрь	0	1	1	1	2	3	4	4	5	5	5	5	5	5
Октябрь	0	1	1	1	2	3	3	4	4	5	5	5	5	5
Ноябрь	0	1	1	1	2	2	2	3	4	4	5	5	5	5
Декабрь	0	0	1	1	1	2	2	2	3	4	4	4	5	5

а)

Градусы широты

	50	52	54	56	58	60	62	64	66	68	70	72	74	76
Январь	0	0	0	0	0	1	1	1	1	2	4	4	4	4
Февраль	0	0	0	0	0	0	0	1	1	1	3	4	4	5
Март	0	0	0	0	0	0	0	0	1	1	3	4	4	4
Апрель	0	0	0	0	0	0	0	1	1	2	4	5	5	5
Май	0	0	0	0	0	1	1	1	2	4	4	5	5	5
Июнь	0	0	0	0	0	1	1	2	3	4	5	5	5	5
Июль	0	0	0	0	0	1	1	3	4	5	5	5	5	5
Август	0	0	0	0	0	1	2	3	4	4	5	5	5	5
Сентябрь	0	0	0	0	1	1	1	3	4	5	5	5	5	5
Октябрь	0	0	0	0	1	1	1	3	4	5	5	5	5	5
Ноябрь	0	0	0	0	0	1	1	3	4	4	5	5	5	5
Декабрь	0	0	0	0	0	1	1	2	2	3	4	5	5	5

б)

Рис.5. Данные, полученные со спутников, приведенные по месяцам (средние за 5 лет, 1973—1977 гг.), о ледовой обстановке в южных морях (пример 4):
а — тихоокеанский сектор южных морей; *б* — атлантический сектор южный морей

Это значит, что на самом деле в данном примере определены две разные системы данных. Элементы данных представляют собой средне-месячные показатели за пять лет (1973—1977 гг.). Оба набора данных представлены в обычной матричной форме. Строка матрицы дает «моментальный снимок» — одновременное наблюдение ледового покрытия на разных широтах. Столбец матрицы описывает динамику

ледовой ситуации на одной широте в течение года. Если есть несколько переменных, то каждый элемент матрицы данных будет представлять собой *n*-ку, содержащую конкретные состояния всех переменных (т. е. элемент **V).**

Пример 5. Добавим к исходной системе, определенной в примере 2 р.2.5 (комплексный анализ крови), еще один параметр — группу пациентов, страдающих анемией. Остальное остается без изменений. При этом данные удобно представить в виде матрицы, изображенной на рис.6**.**

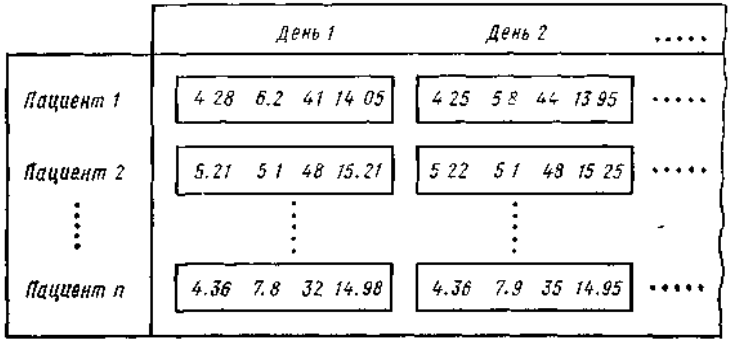


Рис.6. Медицинские данные (примеры 2 р.2.5 и 6)

Каждый элемент матрицы представляет собой четверку, являющуюся полным состоянием четырех переменных, которые наблюдались в определенный день у определенного пациента. На этом примере иллюстрируется систематическая подготовка медицинских данных для дальнейшей обработки.

Пример 6. Для демонстрации работы с нечеткими данными рассмотрим две переменные из примера 1 р.2.5 (посадки деревьев): товарную высоту *(v_3)* и дефекты (v_6). Пусть обе эти переменные определяются через нечеткие каналы наблюдения, которые, однако, не заданы явно, а представлены самим наблюдателем. Все остальное так же, как в примере 1 р.2.5. При этом массив нечетких данных будет иметь такой вид, как это показано на рис.7,а.

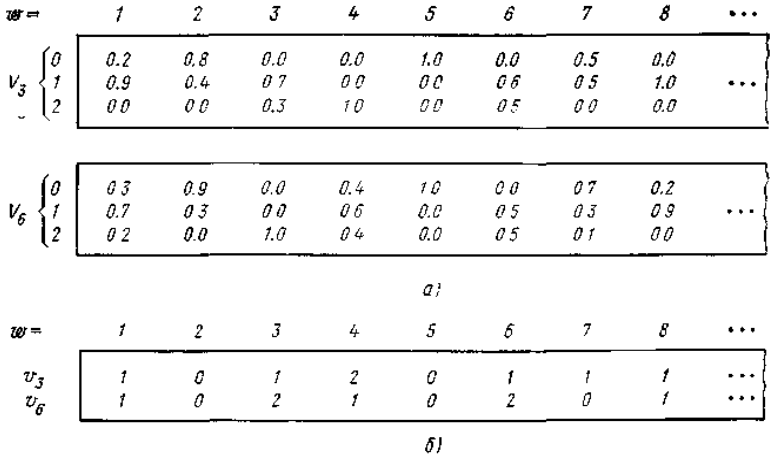


Рис.7. Нечеткие данные (из примера 6): *а* — нечеткие данные, *б* — соответствующие четкие данные

Каждый элемент массива задает степень уверенности (уверенности исследователя) в том, что данное дерево (помеченное целым числом **w**) характеризуется определенным состоянием одной из данных переменных. Этот трехмерный массив нечетких данных можно сопоставить с матрицей данных на рис.7,б, построенной в предположении, что обе эти переменные определяются с помощью четких каналов наблюдения.

Кроме методологических отличий, введенных для исходных систем ИИ, для систем данных можно выделить еще два. **Первое — это отличие между *полностью* и *не полностью* определенными данными**. **Данные называются полностью определенными тогда и только тогда, когда определены все их элементы матрицы или массива данных; в противном случае данные называются не полностью определенными.** Будем в дальнейшем различать два типа не полностью определенных данных:

1) все случаи, в которых некоторые данные при заданном параметрическом множестве *недоступны* (как это бывает в некоторых экспериментальных или исторических исследованиях);

2) все случаи, в которых *несущественно,* имеются ли некоторые данные (как в некоторых задачах постановки диагноза болезни, в которых подобные элементы данных называются *безразличными условиями).*

Если данные определены не полностью, то отдельные множества состояний должны быть расширены некими подходящими (стандартными) символами для обозначения элементов массивов данных, являющихся или «недоступными», или «несущественными». УПМД должен иметь соответствующие возможности для работы с такими элементами.

Второе методологическое отличие систем данных относится только к системам данных с линейно упорядоченными полными параметрическими множествами. При этом можно говорить о *периодических данных,* т. е. о данных, которые повторяются при расширении параметрического множества. Образцы периодических данных приведены в примерах 2 и 3. **Исходные системы ИИ и системы данных ИИ это эпистемологические типы систем ИИ, имеющие по преимуществу эмпирическую природу.** В этом смысле их связь с различными традиционными дисциплинами науки и другими предметными областями значительно сильнее, чем связь с ними систем более высоких эпистемологических уровней, являющихся по преимуществу теоретическими. В самом деле, **для того, чтобы определить на объекте исходную систему ИИ так, чтобы эта система отвечала целям исследования, необходимы соответствующие знания и опыт работы в определенной предметной области.** Обычно существует много способов определения исходной системы ИИ, и выбор какого-то определенного способа, при котором соответствующие вопросы формулируются и разрешаются наилучшим способом,— важнейшая проблема для исследователя. Но и **после определения исходной системы ИИ для получения содержательных данних необходимы специальные знания и инструменты. Главная цель разбора разнообразных примеров, приведенных в данной главе, состояла в том, чтобы продемонстрировать решение различных вопросов, возникающих в процессе определения исходной системы ИИ и получения для них данных.**

В литературе часто не делается различия между объектом и системой, определенной на объекте Такая терминологическая нечеткость может быть источником недоразумений. В некоторых случаях для наших понятий объекта и системы используют соответственно термины «реальная система» и «модель». Это неудачные термины, поскольку, если им следовать, наука о системах ИИ будет иметь дело с моделями, а не с системами. Мы считаем, что **лучше пользоваться термином «система ИИ» для обозначения только операционально описанных абстрактных представлений множеств свойств и баз. Тогда термином «модель» можно обо-**

значить разного рода отношения подобия между парами сравнимых систем ИИ, т. е. систем ИИ одного эпистемологического уровня

Во многих случаях проблемы определения исходных систем ИИ на исследуемых объектах и сбора данных о них связаны с различными вопросами теории и практики измерений. Эти вопросы неразрывно связаны с традиционными дисциплинами науки. Поэтому они находятся вне сферы УПМД и, следовательно, нами не рассматриваются.

Синтаксические, семантические и прагматические аспекты исходных систем ИИ (см. рис.1 р.2.5) можно кратко охарактеризовать следующим образом. К *синтаксическим аспектам* относится то, что связано с отношениями между знаками, например правила построения предложений из слов, но не смысл и применение этих знаков (слов предложений). ***Семантические аспекты*** **включают отношения знаков к понятиям, по которым определяется смысл этих знаков (слов, предложений), но не их применение. В *прагматические аспекты* входят отношения знаков к понятиям, по которым этим знакам приписывается некоторое применение**. ***Синтактика*** **(или синтаксис),** ***семантика*** **и** ***прагматика*** **— это три основных аспекта семиотики (общей теории знаков).** Слово «семиотика» происходит от греческого sema (знак). Семиотика определяется как общая теория знаков во всех их видах и проявлениях у животных или людей, индивидуумов или обществ.

2.9. База данных
(Структуры данных и структуры управления)

Системное программное обеспечение ИИ формируется из комплексов программ. Эти программы записываются на некоторых (машинных) языках. Таким образом, основные идеи системного программного обеспечения ИИ отражаются в построении языков программирования.

Во-первых, в языках программирования должна существовать возможность отображения структур данных, являющихся объектами обработки.

Во-вторых, язык программирования должен обладать **изобразительными средствами для представления методов обработки этих данных, т. е. алгоритмов**. Эти **изобразительные средства называют структурами управления. Структуры данных и структуры управления тесно связаны между собой и являются двумя важнейшими элементами программирования, относительно**

которых группируются функциональные возможности языков программирования и концепции программного обеспечения.
Система ИИ состоит из запоминающего устройства (памяти) и устройства управления решения интеллектуальных задач.
Запоминающее устройство, в частности оперативная память, обычно имеет однородную структуру, состоящую из ячеек, содержащих некоторое фиксированное количество битов. **С абстрактной точки зрения** память — **это одна (возможно очень большая) цепочка**. **Для указания элементов этой цепочки служат адреса (указатели).**
Это означает, что **в своей основе память системы ИИ обладает лишь простейшей структурой данных, реализованной аппаратным путем.**
Программа, записанная на некотором языке, представляет собой на первый взгляд **последовательность битов**. (**Эта последовательность образуется двоичными кодами команд машинного языка**. Они содержат обычно **адресную и управляющую части**. Одной из функций последней является **управление выполнением операций в арифметико-логическом устройстве и обменом между его регистрами и памятью.** Имеются также команды ветвления, позволяющие выбирать порядок выполнения команд программы в зависимости от результатов операций.).
Если посмотреть на нее с позиций функционирования устройства управления вычислительным процессором, то она обеспечивает **два типа деятельности: выполнение операторов данных и управления.**
Язык описания данных в зависимости от выполняемых команд может интерпретировать числовые данные (с фиксированной и с плавающей точкой), символьные данные и другие. Кроме того, в некоторых языках можно описывать адреса (указатели). Все это —**базовые типы данных**, которыми располагает система ИИ, однако при этом, **пользуясь указателями, можно создавать сложные концептуальные структуры данных.**
Разумеется, в арифметическом устройстве определены четыре арифметических действия и другие вычислительные операции, причем типы используемых операций зависят от конкретной машины.
Совокупность команд, включающая команды ветвления и организации их последовательного выполнения по программе, и представляет собой **структуру управления системой ИИ**.
Все это — наиболее определяющие и простейшие элементы системы ИИ. **Более сложные концептуальные структуры управления формируются путем сочетания этих базовых элементов (при участии памяти).**

Первичные программы системного программного обеспечения записываются именно на машинном языке (и исполняются на нем же), однако программирование на машинном языке требует чрезвычайно много времени. **Перед декомпозицией задачи на машинные команды мы обычно пользуемся некоторыми образами функций, которые для своей реализации требуют выполнения нескольких машинных команд.** Такой способ программирования называют способом "снизу вверх". Существует противоположный способ программирования: "сверху вниз", когда речь идет о реализации некоторых понятий, промежуточных между машинным языком и поставленной задачей, и использовании этих понятий при решении задачи. **В зависимости от состава таких понятий предложено множество различных языков.** Эти языки выполняют свои задачи с точки зрения практики их применения, и в то же время отражают ряд концепций создания программного обеспечения. **Эти концепции делятся на две большие группы: концепции структур данных и управления.**
Следующая возникающая проблема состоит в том, как на данной системе ИИ эффективно (с точки зрения затрат времени и объема памяти) построить сложные структуры, группируя более мелкие встроенные структуры.

2.9.1. Структуры данных

В данном разделе рассматриваются структуры данных, а также обсуждаются способы абстрактного и конкретного представления данных. Кроме того, приводятся **примеры основных абстрактных типов данных**.

2.9.1.1. Абстрактные типы данных

Теории, касающиеся различных структур данных, будут рассмотрены несколько позднее. Здесь же мы остановимся на **абстрактном и конкретном подходах к данным**. Эти подходы носят общее название: "Абстрактные типы данных" и были предложены впервые в 1970 г. Эти идеи получили широкое распространение и на их основе был создан язык CLU и др. Абстрактные типы данных вошли в состав различных систем описания ситуаций.
Основной идеей в абстрактных типах данных является различение двух вещей:
а) вида представления данных пользователю;

б) вида организации данных из структур данных, встроенных во внутримашинное представление. Другими словами, **вид структуры данных не должен зависеть от конкретных приложений**.

К формальной записи абстрактных структур данных можно применить **алгебраический подход**. **Он проявляется в том, что задается множество операций над данными**. **Данные определяются как структуры, порождаемые этими операциями.**

Использование выразительных средств алгебры связано с возможностью **применения к данным математических методов**, в частности с применения к данным **теории ассоциативных и коммутативных групп.**

В качестве примера рассмотрим тип данных "список". В качестве операций над списками возьмем **операцию создания списков** *cons,* **операции декомпозиции списков** *car* и *cdr,* **операцию *null,*** дающую возможность определить: **пуст список или нет**, а также **операцию *nil,* задающую пустой список.**

(Операцию *nil* можно по форме записи отождествить с конкретными данными, которые обозначают функцию, дающую пустой элемент.)

Эти операции записываются следующим образом:

DATA_TYPE *LIST* IS
NIL:→LIST
*CONS:*α× *LIST→LIST*
*CAR:LIST→*α
CDR:LIST→LIST
NULL: LIST→BOOLEAN
END *LIST*

Символом α в приведенных выше выражениях обозначается **тип данных для каждого непустого элемента списка**. (Тип данных "список" называют **полиморфным типом данных**, поскольку он может содержать внутри себя переменные для типов, как, например, α.)

Если использовать только эти определения, то остаются неясными связь между операциями *cons, car* и *cdr,* а также вид выражения *null.* Поэтому помимо задания операций записываются **выражения (называемые обычно аксиомами)**, которым должны удовлетворять эти операции. Имеются **три аксиомы списков**:

AXIOMS *X:LIST,Y:* α
CONS(CAR(X),CDR(X))=X
NULL(NIL)=TRUE
NULL (CONS (*Y,X*)) = *FALSE*

С помощью определенных таким образом операций и аксиом формально определяется **абстрактный тип данных: "список".**

Создание конкретных списков осуществляется таким образом, чтобы

удовлетворять этим правилам. В большинстве случаев списки создаются с помощью **указателей**, о которых речь пойдет ниже, и часто на практике оперируют лишь **образами списков**. При этом существуют другие способы организации списков и примеры практической реализации этих способов.

Независимо от варианта реализации, на котором мы остановились, действует правило замены, показанное на рис. 1.

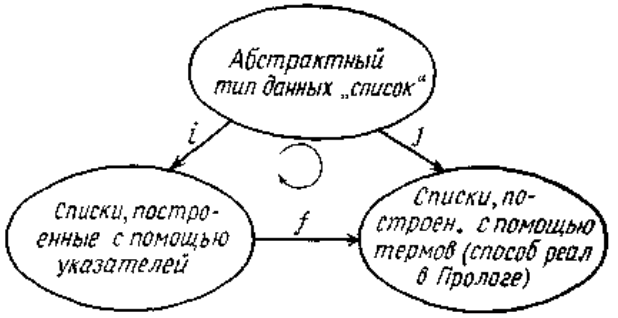


Рис **1.** Абстрактное и конкретное определения списков.
i, *j* — два способа реализации, *f* — отображение, задающее переход от способа реализации списков с помощью указателей к способу с помощью термов. Поскольку выполняется условие *j=i·f,* способ реализации с помощью термов можно не рассматривать (Он рассматривается в разделе 2.7.1.5)

Этот пример иллюстрирует лишь малую часть тех возможностей, которые обеспечивают абстрактные определения.

2.9.1.2. Базовые типы данных

Вообще говоря, часто **при абстрактном подходе сначала выдвигаются некоторые посылки, а затем на их основе ведется рассуждение.** Точно так же при рассмотрении типов данных сначала **аксиоматически задаются некоторые исходные их типы** (их можно называть также **примитивными, базовыми или же элементарными типами данных**). Например, такими данными могут быть следующие:

1) бит (bit);
2) булевский (boolean);
3) символьный (character);
4) целый (integer);

5) вещественный (real).
Им также можно дать абстрактные определения. Например, определение типа " булевский" может иметь вид
DATA-TYPE *BOOLEAN* **IS**
OPERATIONS:
TRUE:→BOOLEAN
FALSE:→BOOLEAN
NOT:BOOLEAN→BOOLEAN
AND:BOOLEAN×BOOLEAN→BOOLEAN
OR:BOOLEAN×BOOLEAN→BOOLEAN
AXIOMS:
NOT (TRUE) = FALSE
NOT(FALSE)=TRUE
AND(TRUE,TRUE)=TRUE
AND(TRUE,FALSE)=FALSE
AND(FALSEJRUE) = FALSE
AND (FALSE,FALSE) =FALSE
OR(TRUE,TRUE)=TRUE
OR(TRUE,FALSE) =TRUE
OR(FALSE,TRUE)=TRUE
OR(FALSE,FALSE) = FALSE
END *BOOLEAN*
Короче говоря, это определение задает **логическую таблицу истинности**. Двоичные нуль и единица удовлетворяют указанному выше абстрактному определению, и поэтому представляют собой **конкретную реализацию истинностных значений**. Это — упрощенный способ объяснения смысла абстрактных типов данных. Другой пример, в языке Лисп для реализации значений истинности используют элемент *nil* и любой другой.
Важное значение имеет понимание того, что **базовые типы данных также имеют абстрактные определения и для них можно сформулировав различные виды конкретной реализации.**

2.9.1.3. Списки

Выше уже давалось определение списка, однако здесь для полноты понимания мы приведем его вновь.
DATA-TYPE LIST (α) IS
OPERATIONS
NIL:→LIST(α)
CONS: α *XUST(α)→LIST(α)*
CAR: LIST (α)→ α

CDR: LIST (α)→LIST (α)
NULL: LIST (α) *→BOOLEAN*
AXIOMS:
VARX: α *L:LIST(α)*
CONS (CAR (L), *CDR (L))=L*
NULL(NIL)=TRUE
NULL (CONS (X,L)) = FALSE
END *LIST*
В качестве одного из **методов реализации списков** используют **указатели.**
В данном случае **указатель** представляет собой структуру, с помощью которой реализуется ссылка на адрес в компьютере; конкретно это — просто **указатель адреса**. На рис. 2 они показаны стрелками. Хранилища для этих указателей называют ячейками.

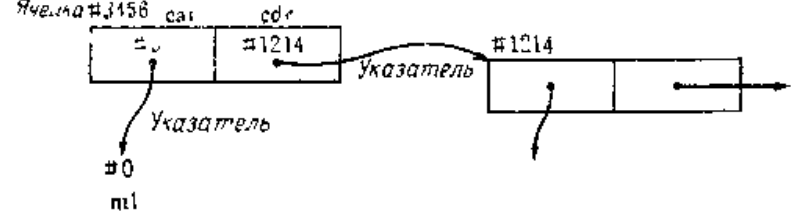


Рис. 2. Реализация списков с помощью ячеек и указателей.
#0, # 3458, # 1214 —указатели адресов.

В случае списков ячейка состоит из частей *car* и *cdr,* каждая из которых хранит соответствующий адрес.
Для части *nil* используется особый **атом (адрес).** Указатель, показывающий на него, изображен на рис. 2. Например, в языке Лисп, впервые разработанном в Массачусетском технологическом институте, атом *nil* выражался нулевым адресом.
Операция *cons* соединяет два указателя, из которых в одной ячейке формируются части *car* и *cdr.* Операции *car* и *cdr* возвращают содержимое *car*- и *cdr*-областей ячейки соответственно. Операция *null* реализует сравнение произвольного указателя ячейки на предмет, является ли он указателем *nil* или нет.
Посмотрим, удовлетворяет ли такая реализация аксиомам списков. Можно говорить, что в выражении

cons (car (L), cdr(L)) = L

фактически задается отношение эквивалентности списков (" = "). В конкретной реализации (рис. 3) указатель ячейки, полученной в результате операции *cons,* может не совпадать с исходным указателем *L.*

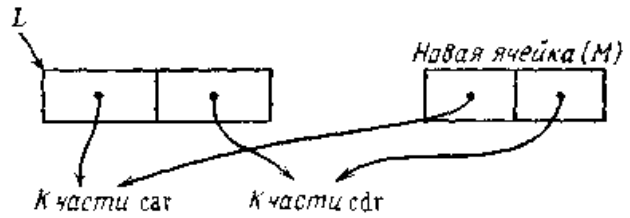


Рис. 3 Реализация операции cons с помощью указателей
M=cons(car(L), cdr(L))=L

С помощью этой аксиомы скорее задается эквивалентность двух списков *L* и *M* по эквивалентности пар результатов *car* и *cdr* для каждого из них.
Далее, очевидно, что *nul (nil) = true* и *null (cons (X, L)) =false,* поскольку результатом операции *cons* не будет указатель, равный *nil*: *nil* не реализуется в качестве специальной ячейки.
Теперь поговорим о трудностях, которые возникают в языке Лисп и других языках при взятии функций *car* и *cdr* от *nil.*
В абстрактном определении объекты (можно называть их списками), задаваемые функциями *car* и *cdr* в аксиоме *cons (car (L), cdr(L))=L,* могут быть лишь списками, созданными с помощью функции *cons.*
Короче говоря, **имеют смысл лишь списки, для которых функция *null* возвращает значение *false.*** Помимо этого, в конкретных реализациях специально не оговариваются значения функций *car* и *cdr* от *nil.*
Если считать, что все **операции**, **определяемые абстрактно**, являются **математическими отображениями**, то необходимо, чтобы операции *car* и *cdr* были определены для произвольного описка. Если это так, то необходимо определить функции *car, cdr* и для элемента *nil.* В первом из приведенных определений об этом ничего не говорится.
Следовательно, ничего необычного не будет, если в реализациях принять *car(nil) = nil, cdr (nil) = nil.* Это не противоречит определению

$$cdr: list\ (\alpha) \rightarrow list\ (\alpha).$$

Чтобы подчеркнуть это в явном виде, необходимо отдельно ввести множесшо непустых списков и множество списков, содержащее пустой список, *(nil),* следующим образом:
DATA_TYPE *NON_EMPTY_LIST* (π) IS
OPERATIONS:
CONS: α ×*NON_EMPTY_LIST (*α *)* → *NON_EMPTY_LIST (*α *)*
*CAR:NON_EMPTY_LIST (*α *)* → α
CDR:NON_EMPTY_LIST { α *)* → *LlST(*α *)*

NULL: *NON_EMPTY_LIST (*α *)* → *BOOLEAN*
AXIOMS:
VAR *X:* α *,M:NON_EMPTY_LIST{* α *)*
CONS(CAR(M),CDR(M))=M
NULL(M)= FALSE
END *NON_EMPTY_LIST*
*DATA_TYPE_LIST (*α *) IS*
OPERATIONS:
NIL: → *LIST(a)*
CONS:a×*LIST(*α *)* → *NON_EMPTY_LlST(*α *)*
*NULL: LIST (*α *)* → *BOOLEAN*
AXIOMS:
VAR *X:* α *,L:LIST(*α *), M:NON_EMPTY_LIST(*α *)*
CONS(CAR(M),CDR(M))=M
NULL(NIL)=TRUE
NULL (CONS (X,L)) = FALSE
CDR(CONS(X,L))=L
CAR(CONS(X,L))=X
END *LIST*

Теперь рассмотрим **еще одну реализацию списков**, о которой упоминалось ранее; а именно **метод термов**. В этом случае операция *cons* из двух элементов образует новый терм, т. е.

$$cons(X,L) \Rightarrow X{\cdot}L.$$

Здесь знак "." является **конструктором терма**. Возможна и такая запись. *.(X, L).*
Функции *car* и *cdr* представляют собой операции, возвращающие соответственно левую и правую части терма, созданного упомянутой операцией *cons* (в языке Пролог это возвращение реализовано через операцию унификации) (здесь мимоходом упоминается важная особенность языка Пролог, а именно - возможность **задания инструкций процедурного (или в данном случае функционального) языка через операторы представления знаний** непроцедурного на первый взгляд языка, каким является Пролог. Эта возможность как раз и обеспечивается упомянутым механизмом унификации.)
Атом *nil* выражается термом: *nil.* Функция *null* определяет, является ли возвращаемое значение термом или нет.
Посмотрим, удовлетворяются ли при этой реализации аксиомы списков:

$$cons\ (car\ (L),\ cdr(L))=L.$$

Если положить *L=M.N,* то получим

cons (car (L), cdr (L)) = cons (M, N)
= M·N
= L.

Далее, очевидно, что

null (nil) = true,

а также, что

null (cons (X, L)) = false.

На практике в методе термов абстрактные определения списков реализуются в универсуме самих этих термов.

2.9.1.4. Деревья

Структуры данных типа дерева *(tree)* многообразны, однако сначала мы дадим здесь общее определение.

DATA_TYPE *TREE* IS
OPERATIONS:
CONS:MODEx TREES→TREE
NIL: → TREE
CHILDREN: *TREE→TREES*
MODE: TREE→NODE
LEAF?: TREE→BOOLEAN
AXIOMS:
VAR *N:NODE, C:TREES,T:TREE*
LEAF? (NIL) =T RUE
LEAF? (CONS (N,C)) = FALSE
CONS (NODE(T) .CHILDREN(T)) = T
END *TREE*
DATA_TYPE TREES IS
OPERATIONS:
HEAD: TREES→TREE
TAIL :TREES→TRLIS
NIL_SEQ: -+TRLLS
NULL: TREES→BOOLEAN
CONS: *TREE TREES→TREES*
AXIOMS:
VAR *T:TREES E: TREE*
CONS(HEAD(T),TAIL(T))=T
NULL (NIL_SEQ) = TRUE
NULL (CONS (E,T)) = FALSE
END *TREES*

Здесь тип *node* и *list* указываются в виде **параметров**. Кроме того, здесь используются наименования **операций (функций),** совпадающие

с наименованиями *cons, null, nitl* и т. д., участвующими в определении списков. В принципе это — случайное совпадение и можно считать, что эти операции полностью отличны от одноименных им в случае списков. Одинаковые наименования используются по той причине, что одноименные операции в обоих случаях схожи по смыслу.

Использование одних и тех наименований в разных, не вполне тождественных случаях ничем не отличается от общепринятой практики словоупотребления в естественных языках. Аналогичной **полисемией (многозначностью)** обладает, например, фраза: "**модульный подход**" (подробнее об этом ниже).

Таким образом, мы получили определение деревьев, однако при внимательном рассмотрении операции и аксиомы в этом определении совпадают с операциями и аксиомами списков. **Главным отличием является то, что дерево превращается в список деревьев: структура данных *trees* становится структурой данных *list (tree).***

Этот факт на уровне абстрактных типов данных указывает на возможность реализации одной структуры данных через другую. Это — не что иное, как практический метод реализации конкретных структур данных.

В системе OBJ2 указывается подобное соотношение между структурами данных с использованием механизмов наследования и подстановки (так называемого **механизма перспективы**). Схожий с ним метод описания выглядит следующим образом

VIEW *TREES_AS_LIST IS*
LIST(TREE)
WITH *HEAD = CAR,*
TAIL=CDR,
NIL_SEQ=NIL
END *TREES_AS_LIST*

Само **дерево можно описать, с помощью списка** (например в этом случае необходимо с привлечением функции *cons* списков создать функцию *cons* дерева). Таким образом, эти **две функции будем различать, используя в качестве индексов слова *tree* и *list*.**

Операции дерева можно определить следующим образом-

$CONS_{TREE}=CONS_{LIST}$
CHILDREN=CDR
NODE=CAR
$NIL_{TREE} = NI_{LIST}$
LEAF?=NULL

В этом случае также можно использовать определение списков без изменений. Здесь предполагается, что в этом определении структура *trees* реализуется как структура *list (tree).*

Структура *tree* определяется как структура *list (node),* однако, строго говоря, этот случай не совсем совпадает со случаем *trees.* В общем случае дерево имеет структуру, представленную на рис. 4, однако на практике часто используют деревья с фиксированным числом ветвлений: *n*-ричные, деревья, или деревья *n*-го порядка.

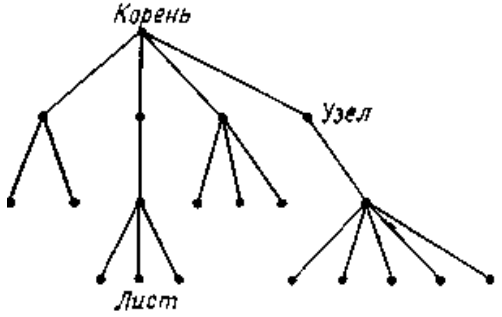


Рис. 4. Графическое изображение дерева

В частности, широко используют простейшие, двоичные деревья (рис. 5).

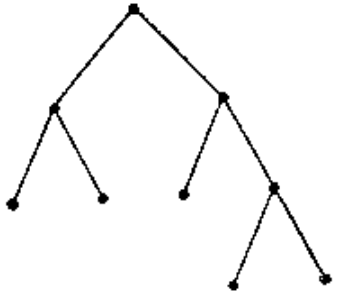


Рис.5. Двоичное дерево

Определение двоичного дерева содержится в определении дерева как частный случай, однако это определение можно выделить в самостоятельное, тогда оно примет следующий вид:

DATA_TYPE *BINARY_TREE* IS
OPERATIONS:
CONS NODE×BINARY_TREE×BINARY-TREE→BINARY_TREE
LEFT :B1 NARY_TREE→BINARY_TREE
RIGHT:BINARY_TREE-+BINARY_TREE
NIL: →BINARY_TREE

LEAF?: BIN ARY_TREE→BOOLEAN
MODE:BINARY_TREE→MODE
AXIOMS:
VAR:T,R,L:TREE,N:NODE
CONS(NODE(T),LEFT(T),RIGHT(T))=T
LEAF? (NIL) =TRUE
LEAF?(CONS) (N,R,L))=FALSE
END *BINARY_TREE*

Попробуем переписать это определение в таком виде, чтобы оно не содержало элементов *node,* а в качестве элемента *leaf* содержало бы произвольную структуру данных α. Такое двоичное дерево обозначают *sbt (simplified — binary — tree)* и определяют следующим образом:

DATA_TYPE *SIMPLIFIED_BIN'ARY_TREE(α)IS*
OPERATIONS:
CONS:SBT(α)×SBT(α) → SIMPLIFIED_BINARY_TREE(α)
LEFT:SIMPLIFIED_BINARY_TREE(α) →SBT (α)
RIGHT:SIMPLIFIED_BINARY_TREE(α) →SBT(α)
NIL: →SIMPLIFIED_B1NARY_TREE (α)
LEAF?:SBT(α) → BOOLEAN
AXIOMS:
VAR *T:SIMPLIFIED_BINARY_TREE*
CONS (LEFT(T)RIGHT(T))=T
END *SIMPLIFIED_BINARY_TREE*

Здесь запись *sbt* соответствует выражению *α' simplified — binary— tree (α).* Определения функций *cons* и *leaf}* даются над *sbt.*

DATA_TYPE *SBT* (α) *IS*
OPERATIONS:
CONS:SBT(α) ×SBT(α) →SIMPLIFIED_BlNARY_TREE(α)
LEAF?:SBT(α) → BOOLEAN
AXIOMS:
VAR *T:SIMPLIFIED_BINARY_TREE, X:* α *,R,L:SBT*
CONS(LEFT) (T),RIGHT(T))=T
LEAF? (NIL) =TRUE
LEAF?(X)=TRUE
LEAF(X)=TRUE
LEAF? (CONS (R,L)) = FALSE
LEFT (CONS (R,L))=R
RIGHT (CONS (R,L))=L
END *SBT*

Используя это *sbt,* можно определить через него и списочную структуру.

В этом случае

cons = cons,
car = left,
cdr =right,
а для операции *null* вновь введем соотношения, определяемые следующими аксиомами:
null (nil) = true,
null (X: α) *= false,*
null (cons (R,L)) = false.
Поскольку для функции *leaf?* выполняется условие: *leaf? (X* : α*) = true,* то при использовании ее в описании списочной структуры возникает избыточность.
На практике лисповские списки реализуются в виде этих упрощенных двоичных деревьев. В позиции *cdr* в ячейке на рис. 2 может находиться и тип данных α.
В лисповской нотации операции *leaf?* соответствует предикат АТОМ. Здесь возможности использования в программировании на языке Лисп предиката АТОМ вместо функции *leaf?* соответствуют соотношению между *leaf?* и *null.*

2.9.1.5. Графы

Более сложной структурой данных по сравнению со списками и деревьями являются графы. Общий вид графа представлен на рис. 6.

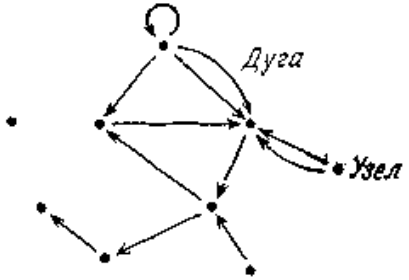


Рис. 6. Граф

Узловые точки графа называют вершинами, между двумя вершинами можно провести произвольное количество дуг или стрелок. В графах могут быть изолированные вершины, а также дуги, направленные к той же вершине, из которой они исходят.
Как правило, дугам графа задается направление, однако это направление может быть и не определено (считают, что при этом

определены оба направления). Их называют ориентированным графом (орграфом) и неориентированным графом соответственно.
Абстрактное определение графа выглядит следующим образом:
DATA_TYPE *GRAPH* IS
OPERATIONS:
V ERT ICES :GRAPH→SET (VERT EX)
ARROWS:*GRAPH→SET (ARROW)*
ADD_NODE:VERTEX×GRAPH→GRAPH
ADD-ARROW:ARROW×GRAPH→GRAPH
DELETE_NODE: VERTEX×GRAPH→GRAPH
DELETE_ARROW:ARROW ×GRAPH→GRAPH
END *GRAPH*
DATA_TYPE *VERTEX* IS
OPERATIONS:
CREATE: IDENTIFIER→VER TEX
CHILDREN: VERTEX→SET (*VERTEX)*
ARROW S:*VERTEX→SET (ARROW)*
IDENTITY: VERTEX×VERTEX→BOOLEAN
END *VERTEX*
DATA_TYPE *ARROW* IS
OPERATIONS:
CREATE:*DESCRIPTION× VERTEX× VERTEX→ARROW*
SOURCE:*ARROW→VERTEX*
DESTINATION:*ARROW→VERTEX*
END *ARROW*
Здесь тип данных *set* обозначает тип данных "**множество**". **В общем случае списки и деревья являются частными случаями графов!**
В качестве метода реализации используют **матрицы смежностей**, **указатели и другие методы**. Чтобы практически реализовать определенный здесь граф общего вида, необходимо **создать базу данных, в которой отдельно описывать вершины и дуги.**
На рис. 7 представлено описание графа с помощью матрицы смежностей.

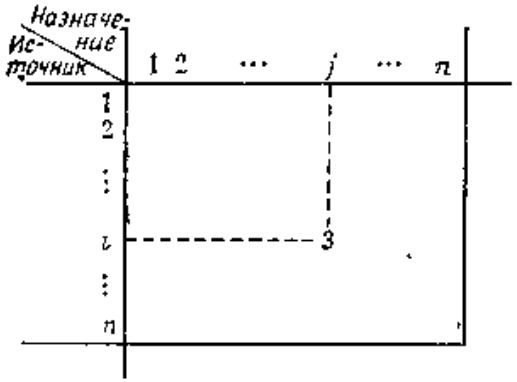


Рис. 7. Представление графа с помощью матрицы смежностей.
(i, j)=3 означает, что из вершины i в вершину j проведено три ребра.

В этом случае размер матрицы определяется количеством вершин графа. При добавлении произвольного числа вершин досточно продолжить матрицу, при удалении вершины необходимо менять расположение элементов матрицы смежностей.
Количество дуг, идущих из вершины i в вершину j, заносится в ij-й элемент матрицы. Полный граф, изображенный на рис. 8, представляется с помощью матрицы, показанной на рис. 9, в которой все элементы являются единицами.
В этом методе нельзя с каждой дугой связывать какую-либо информацию. Как и в изображении списков, при использовании указателей стрелка выражается адресом.
Обычно в этом случае количество дуг, выходящих из одной вершины, определяет число следующих за ней вершин, однако в каждом случае необходимо фиксировать верхнюю границу этого числа.

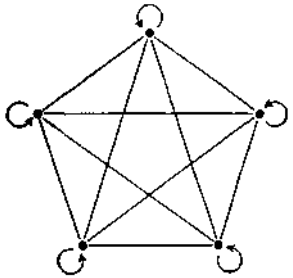


Рис. 8. Полный граф с пятью вершинами.
Все вершины связаны между собой.

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Рис. 9. Матрица графа, изображенного на рис. 8.

В случае использования ориентированных дуг или стрелок вершину, в которую входит дуга, называют *концом* или *стоком* дуги, а вершину, из которой она выходит, — *началом* или *истоком* дуги.
В методе указателей вместо дуг используется множество вершин, являющихся непосредственными потомками данной вершины.
При использовании данного метода легко удалять и добавлять вершины. Дело с удалением и добавлением дуг также обстоит просто, если при этом не превышать максимальное количество последующих вершин. Как и в случае матриц, здесь нет возможности связывать с дугами какую-либо информацию.

2.9.2. Структуры управления

В данном разделе рассмотрены структуры управления в их соотнесенности со структурами данных. **Управление** можно разделить на две группы: **последовательное и параллельное**. В данном случае рассматриваются лишь последовательные структуры управления, используемые в традиционных, так называемых **фон-неймановских машинах**, и их обобщения.

2.9.2.1. Структуры управления и циклы в блок-схемах

Классической структурой последовательного управления является блок-схема, пример которой приведен на рис. 10.

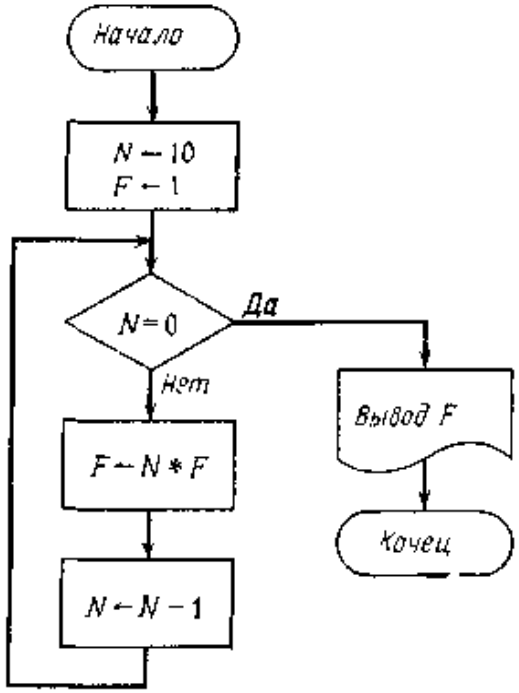


Рис. 10. Пример блок-схемы.

На нем показаны **группировка управления, условные ветвления, циклическое выполнение операций с помощью передачи управления и безусловный переход**. Основным элементом в этой структуре является **счетчик команд**. **Он осуществляет управление выполнением программы, задавая конкретное место (адрес) команды, которая должна выполняться.**
Типичными для рассмотренной блок-схемы могут служить некая условная программа, представленная на рис. 11, и соответствующая блок-схема, приведенная на рис. 10.


```
#1  N←10; F←1
#2  IF N=0 THEN GOTO 6
          ELSE GOTO 3
#3  F←N*F
#4  N←N—1
#5  GOTO 2
#6  PRINT F
#7  END
```

Рис. 11. Некоторая условная программа, соответствующая блок-схеме, представленной на рис. 10.

Исходя из рис. 10 и 11, можно считать, что счетчик команд задает **номера операторов** (рис. 11), а блок-схема на рис. 10 иллюстрирует порядок исполнения программы, изображенной на рис. 11.
Степень подробности структуры, задаваемой счетчиком команд, зависит от уровня языка программирования. На уровне машинного языка или микропрограмм нижнего уровня эта структура задается чрезвычайно подробно, однако при использовании языков высокого уровня такое структурирование, естественно, оказывается слишком мелким.
Самым большим преимуществом изображения структур управления в виде блок-схемы является то, что при этом становится весьма наглядной структура управления. Например, ту же самую программу (рис. 11) (т. е. выполняющую те же операции) можно представить в виде, показанном на рис. 12.

```
#1  N←10
#2  F←1
#3  IF N=0 THEN GOTO 7 ELSE GOTO 5
#4  END
#5  PRINT F
#6  GOTO 4
#7  F←N*F
#8  N←N—1
#9  GOTO 3
```

Рис. 12. Программа, эквивалентная программе, представленной на рис. 11.

При этом из сравнения текстов не сразу становится понятным, что суть обеих программ одна и та же, однако изображенные в виде блок-схемы они имеют одну и ту же структуру, что и делает эквивалентность этих программ очевидной.

Конкретная работа счетчика команд состоит в том, что при выполнении линейных участков программы сверху вниз его значение после выполнения каждой команды увеличивается на единицу. При выполнении команд, указываемых операторами *goto* в программах, изображенных на рис. 11 и 12, ему присваивается значение, соответствующее адресу требуемой команды.

Основная причина того, что блок-схемы понимаются легче программ, заключается в подобных изменениях значения счетчика команд, т. е. в выполнении команд, указываемых оператором *goto.*

В связи с этим существует **методика структурирования программ**, т. е. **структурное программирование**. При таком структурировании благодаря ограничению, накладываемому на изменения содержимого счетчика команд, появляется возможность с помощью текста программы добиться такой же эффективности в изобразительных средствах, какая достигается с помощью блок-схемы.

Изменение значения счетчика команд происходит главным образом при условных ветвлениях и в циклах. Таким образом, структурирование касается именно этих частей программы. Для записи команды условного ветвления используется следующий оператор:

if *C* **then** S_1 else S_2 fi (Условный оператор).

Здесь *C* выражает условие, а на местах S_2 и S_2 непосредственно записываются выполняемые операции. Для записи циклов предложено несколько форм. Хорошо известны формы

while C **do** S (Ожидающий оператор цикла)

и

repeat S **until** *C* (Итерационный оператор цикла),

а кроме того конструкции с условным ветвлением.

В **ожидающем операторе цикла сначала проверяется условие, а затем** в течение всего времени, пока оно проверяется, **выполняется некоторое действие**.

В **итерационном операторе цикла сначала выполняется действие, а затем проверяется условие**; при его выполнении наступает выход из цикла. Таким образом, **выполнение заданного оператора продолжается до тех пор, пока не выполнится предписанное условие**.

Кроме этих двух операторов существует еще так называемый **замкнутый оператор цикла**, в котором выход из цикла осуществляется по специальному указанию. Этот оператор имеет вид

loop S_1; S_2; ...; S_n end_loop (Замкнутый оператор цикла).

При появлении среди выполняемых операторов ключевого слова *exit* выполнение цикла прекращается и происходит выход из него. Как правило, слово *exit* используется в сочетании с условным оператором.

Например, ожидающий и итерационный операторы цикла записываются с помощью замкнутого оператора цикла следующим образом:

```
LOOP     IF C THEN S ELSE EXIT  END_LOOP
LOOP      S; IF C THEN EXIT END_LOOP
```

Поэтому, используя структурирование, блок-схему, изображенную на рис. 10, можно представить в виде программ, показанных на рис. 13 (ожидающий оператор цикла) или рис. 14 (замкнутый оператор цикла).

```
BEGIN   N←10; F←l;
WHILE N≠0 DO BEGIN F←N*F;
                   N←N—1 ,
 END
PRINT F
 END
```

Рис. 13. Структурированная программа с использованием ожидающего оператора цикла.

```
BEGIN   N←10; F←1;
LOOP   IF N =0 THEN EXIT FI
                   F←N*F;
                   N←N—1
END_LOOP PRINT F
 END
```

Рис. 14. Структурированная программа **с** использованием замкнутого оператора цикла.

Здесь ключевые слова begin и end соответствуют словам НАЧАЛО и КОНЕЦ блок-схемы. Они используются для обозначения одного программного блока. На некотором более высоком уровне ожидающий и замкнутый операторы цикла могут рассматриваться как "один оператор". Поэтому на таком уровне можно считать, что счетчик команд получает единичное приращение на весь оператор цикла.

2.9.2.2. Структура вызова процедур и возврата

"Общий" подход, рассмотренный в предыдущем разделе в виде блок-схем, обеспечил высокий уровень наглядности при просмотре структуры программ в целом. Была рассмотрена проблема циклов с тех же структурных позиций.

При этом возникает **понятие подпрограмм, т. е. "структурно-выделенных цепочек формальных процедур**". В частности, важными вопросами являются присвоение имен выделенным таким образом процедурам и передача связанной с ними **информации через параметры.** При этом применяется такое хорошо известное в математике понятие, как **функция**.

Процедура вызывается с участием двух элементов: **ее имени и передаваемых ей значений (имен) параметров.** После выполнения необходимых действий происходит возврат в вызывающую программу. Некоторый типичный вид вызова процедуры и возврата из нее показан на рис. 15.

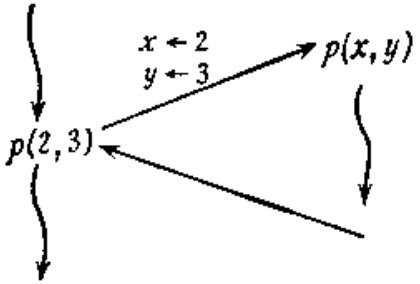


Рис. 15. Типичный вид вызова процедуры и возврата из нее.

Для счетчика адреса команд вызов процедуры эквивалентен переходу по команде goto. При возврате из процедуры в вызывающую программу (**называемую главной программой**) значение счетчика команд также восстанавливается с увеличением на единицу, поскольку на вызов процедуры в вызывающей программе отводится одна команда. И в данном случае на некотором более высоком уровне **процедура может рассматриваться как оператор, соответствующий одной команде главной программы**. Затем осуществляется переход к выполнению следующей команды.

Строго говоря, существуют различия между процедурой и функцией: процедура может не возвращать значение. Однако можно считать, что как та, так и другая работают одинаково. Вызов процедур, который имеет вид, показанный на рис. 15, за исключением процесса передачи информации через параметры, можно считать одним из вариантов работы с использованием оператора *goto*.

Однако существует следующая особая форма вызова процедур:

factorial(*n*)≡if *n*:=0 then l

else *n :factorial (n* — 1)

Эта функция в своем определении содержит вызов самой себя. Такой вызов называют *рекурсивным.* С рекурсивным вызовом также связана следующая более сложная форма:

ACK(M,N) IF M=0 THEN N+l

ELSE IF *N* =0 THEN *ACK(M*-1,1)

ELSE *ACK(M*-1,*ACK(M,N*-1*))*

При сброботке такого рекурсивного вызова **необходима специализированная структура управления.**

Если при выполнении программы происходит прерывание какой-либо деятельности, то после устранения причины прерывания, например окончания исполнения прерывающей программы, необходим возврат к той деятельности, которая была прервана.

(Особенность рекурсивного вызова состоит в том, что процедура вызывает саму себя. Обычная программа существует лишь в одном экземпляре. Поэтому если при рекурсивном вызове устанавливать счетчик команд на начало процедуры, то такое выполнение будет не рекурсивным, а циклическим. Обращаясь к примеру с факториалом, можно увидеть, что выход из процедуры произойдет лишь при обращении к факториалу от единицы; если не использовать механизм прерывания (восстанонления), а лишь оператор *goto,* то в ответе получим единицу независимо от значения, с которым было обращение к процедуре Этого можно избежать, создавая каждый раз при вызове новую копию вызываемой процедуры, однако такое решение на практике оказывается малопригодным.)

В качестве управляющей структуры данных, пригодной для реаизации управления прерыванием (восстановлением), используют стек. Для стека определены следующие операции:

push:α stack* (α) →*stack* (α)

pop: stack (α) →<α, *stack* (α) >

Действие этих двух операций *push* и *pop* противоположно друг другу и удовлетворяет следующим аксиомам:

X:α,S:stack(α)

pop(push(X,S))=<X,S>

В управляющий стек помещаются значения счетчика команд, параметры и другая информация о локальной вычислительной среде. Обозначив их соответственно через *PC (program counter)* и *env (enviroment)*, вызов процедуры можно описать следующим образом:

PC*:PC,E:ENV.S:STACK(PCUENV)*

PC ←CURRENT_*PC*+l; Присвоение очередного значения счетчику команд

E←CURRENT_ENVIRONMENT; Текущая вычислительная среда

S←PUSH(E,S); Помещение в стек

S←PUSH(E,S);
S←PUSH(PC,S);
E←PARAMETER_SET (PROCEDURE_CALL)
PC *←ST ART _ADDRESS (PROCEDURE_CALL)*
Здесь ключевое слово procedure — call соответствует вызову процедуры и при нем задаются значения параметров и начальный адрес процедуры.
При возврате из процедуры выполняются следующие операции:
<E,S>←pop(S);
<PC,S>←pop(S);
При вызове функции ее значение можно передать несколькими способами. При передаче через верхушку стека в самом начале выполняются следующие действия:
<Value, S> ←pop(S);
В записи для операций *push* и *pop* можно явно не указывать стек и описывать их в сокращенной форме: *push(X)* и *pop.*
Вместо занесения в стек и выталкивания из стека значений локальной вычислительной среды можно активно использовать содержимое стека в качестве такой вычислительной среды. В этом случае стек становится не только объектом операций *push* и *pop,* но и представляет из себя некую сложную конфигурацию, т. е. списочную структуру. Другими словами, существование указателя стека *p* обеспечивает с помощью функции *p[N]* возможность доступа к *N-му* элементу из *p.*(Можно строго определить операцию: *access(Stack,Pointer,Bias).* Поскольку существует возможность записи в стек, можно, естественно, определить операцию: *assign(Stack,Pointer,Bias,Value).)*
Для задания указателя стека существует специальная операция, которую можно записать, например, в виде
set_ staek_pointer (S ,P)
В этих обозначениях вызов процедуры можно записать следующим образом:
PC*:PC,* V*:VALUE,***SP***:STACK_POINTER,S:STACK*
S←PUSH(SP,S);
SETSTACK_POINTER(S,SP);
V←ARGUMENT(1);
S← PUSH (V, S)- Число повторений равно числу аргументов
S←PUSH (PC+Δ,S);
PC←START_ADDRESS (PROCEDURE_CALL);
(PC+Δ указывает на продолжение программы}
<V,S>←POP(S) {V —значение)
<SP,S>←P0P(S) с восстановлением *SP* происходит восстановление локальной вычислительной среды

При этом обрабатываемые значения в качестве параметров передаются в вызываемую программу с помощью *SP* и после обработки посредством операции
<PS,S>←pop(S)
из стека выталкивается значение счетчика команд и происходит возврат к участку программы, выполнявшемуся перед этим.

2.9.2.3. Управление бэктрекингом

Использование управляющих структур данных в виде стеков для управления процедурами, содержащими рекурсивный вызов в общем виде, позволяет реализовать основные виды управления.
Однако, как это имеет место в языках логического программирования, на уровне языка появляются **управляющие структуры, содержащие не только последовательную обработку, но и обработку, допускающую распараллеливание, и при этом требуются иные структуры управления**.
В качестве примера языка логического программирования рассмотрим язык Пролог. Например, программа на нем может иметь вид
p:-q,s
p:-r,t.
Эту программу с использованием понятия процедуры можно прочитать следующим образом. **При выполнении процедуры *p* выполняются процедуры *q* и s**. Если они завершаются успешно, то и процедура *p* считается успешно завершенной. Если это не так, то выполняются процедуры *r* и *t.* В этом случае процедура считается успешно завершенной, если обе процедуры завершаются успешно. В противном случае процедура *p* терпит неудачу.
Такой алгоритм обработки можно реализовать на дереве специального вида: И-ИЛИ-дереве (AND-OR-tree), которые можно изобразить в виде, показанном на рис. 16.

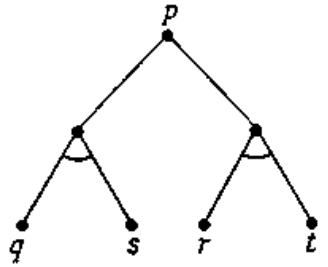


Рис .16. И ИЛИ-дерево.
∧ — узел ИЛИ, А — узел И

В каждой своей вершине И-ИЛИ-дерево имеет метку И или метку ИЛИ. **Вершина И успешно разрешима только в том случае, когда ее вершины-потомки успешно разрешимы**. **Вершина типа ИЛИ успешно разрешима в том случае, когда хотя бы одна из ее вершин-потомков успешно разрешима.**

При возможности параллельного выполнения этого И-ИЛИ-дерева легко удается реализовать распараллеливание, которое подробнее будет рассмотрено ниже и о котором мы здесь вкратце упомянем. Метод такого распараллеливания состоит в том, чтобы, начиная с корня дерева (самая верхняя вершина *p* на рис. 16), вести ветвящийся процесс относительно каждой точки ветвления. При этом результаты выполнения вершин-потомков следует анализировать следующим образом. Если мы находимся в вершине типа И, то следует проверить, не терпит ли хотя бы одна из вершин-потомков неудачу, а если мы находимся в вершине типа ИЛИ, то проверить, не завершается ли хотя бы одна из вершин успехом.

В случае, приведенном на рис. 16, сначала можно проверять на успех или неудачу листья дерева (терминальные вершины), а затем, поднимаясь обратным ходом вверх по дереву, проверить, как сочетаются между собой результаты.

Если распараллеливание невозможно, то применяется последовательная обработка вершины И-ИЛИ-дерева, при этом процедуры (q, s; *r, t*) вызываются, как описано в разд. 2.7.2.2.

К идеальному случаю последовательного выполнения принципиально параллельного И-ИЛИ-дерева относится поиск по нему без потерь. Под идеальным здесь понимается тот случай, когда при рассмотрении И-вершины, если хотя бы одна из вершин-потомков терпит неудачу, при поиске мы сразу выходим на нее, а при рассмотрении ИЛИ-вершины мы также сразу выходим на успешную дочернюю вершину, если таковая имеется. В общем случае идеальной стратегии поиска вершин не существует

Рассмотрим две стратегии поиска по И-ИЛИ-дереву: стратегии поиска в глубину и в ширину.

При поиске в глубину ведется последовательный обход И-ИЛИ-дерева сверху вниз слева направо, как показано на рис. 17.

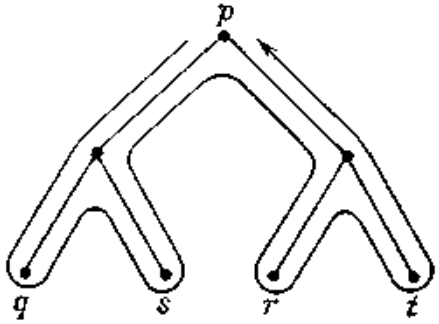


Рис. 17 . Поиск в глубину.

Если при обходе дерева какой-либо потомок вершины типа ИЛИ разрешается успешно, то обработка остальных дочерних вершин (поддерева, находящегося справа) приостанавливается и считается, что данная ИЛИ-вершина разрешилась успешно.

Если какой-либо из потомков И-вершины терпит неудачу, то обработка остальных ее потомков полностью прекращается и считается, что данная вершина терпит неудачу. Следует заметить, что обработка ИЛИ-вершины не прекращается, а лишь приостанавливается. Это связано с тем, что впоследствии возможно повторное обращение к этой вершине, и тогда оставшиеся про запас ветви, которые еще не обрабатывались, могут повторно привести к успеху этой вершины.

Этот процесс называется *бэктрекингом*. В простом примере, приведенном на рис. 16, существо бэктрекинга просматривается не очень наглядно. Поэтому разберем более сложный пример (**необходимость в бэктрекинге возникает тогда, когда переменные получают значения в результате унификации**),

Рассмотрим следующую программу:

P(X):-Q(X),S(X)
P(X):-R(X),T(X) (
Q(A). ,
Q(B).
R(A).
R(D).
S(C).
S(D).
T(B).
T(D).

Эту программу можно изобразить в виде дерева, показанного на рис. 18. Оно напоминает дерево, изображенное на рис. 16.

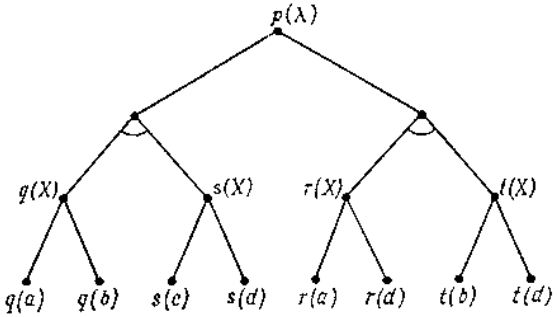


Рис. 18. И-ИЛИ-дерево для программы 2

Важным элементом в поиске по И-ИЛИ-дереву является способ задания значений переменной X в выражении *p(X).* Например, если ведется проверка выполнимости выражения *p(a)*,каждому вхождению переменной *X* сопоставляется значение *a,* и выполнимость проверяется вне связи с бэктрекингом (рис. 19),

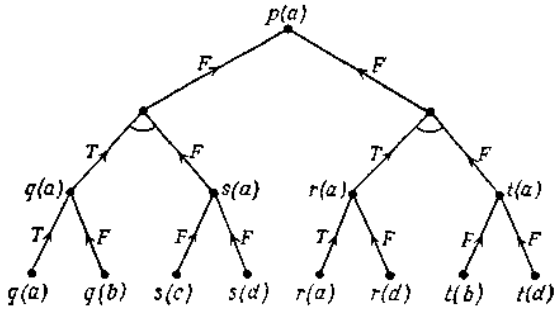


Рис. 19. Определение выполнимости выражения p(a).
T — выполняется, F — не выполняется

Если же необходимо установить выполнимость предиката *p(X),* т. е. определить, существует ли подходящее значение *X,* доставляющее истинное значение предикату *p(X),* то можно выполнять следующую процедуру. В общем случае неизвестно, конечна ли область значений *X.* В случае данной программы она конечна, и достаточно проверить четыре значения: *X = a, X = b, X = c, X=d.*
При последовательном поиске по И-ИЛИ-дереву программы переменной *X* присваиваются значения и проверяется, приводят ли они к успеху рассматриваемой в данный момент вершины. На рис. 20 показано место, где при поиске в глубину впервые задается значение *X=a.*

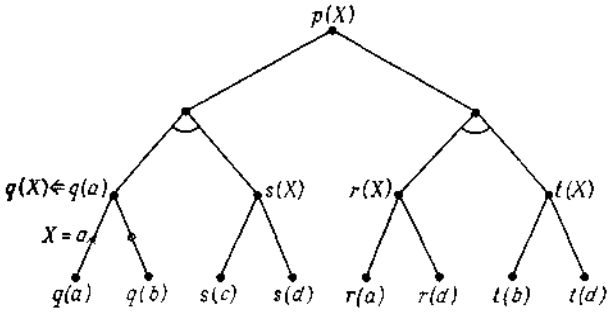


Рис. 20. Присвоение значения X=a
~~θ~~ —часть дерева, оставченрая в резерве

При этом часть потомков ИЛИ-вершины, оставшаяся после присвоения *q(X)* ⇐ *q(a),* а именно часть *q(b),* не проверяется и оставляется в резерве. Вершина ИЛИ удовлетворяется одним только значением *q(a).*
Однако при этом, как показано на рис. 21, в части дерева, соответствующей присвоению *s(X)* ⇐ *s(a),* все потомки ИЛИ-вершины терпят неудачу, и сама вершина тоже терпит неудачу.

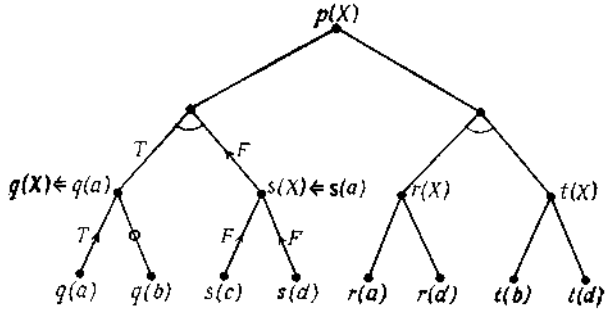


РИС.21. s(X) терпит неудачу при X=a.

Это приводит к тому, что И-вершина с дочерними вершинами *q(X)* и *s(X),* при присвоении *X=a* окончательно терпит неудачу. Теперь, возвращаясь к сохраненной ветви *q(b),* можно сделать еще одну попытку. При этом переменная *X* теряет ранее полученное значение *a* (происходит возврат назад), и в этом случае производится попытка присвоить ей значение *b.*

На рис. 22 показано, что при присвоении переменной *X* значения *b* также имеет место неудача в *s(X)* ⇐ *s(b)*.

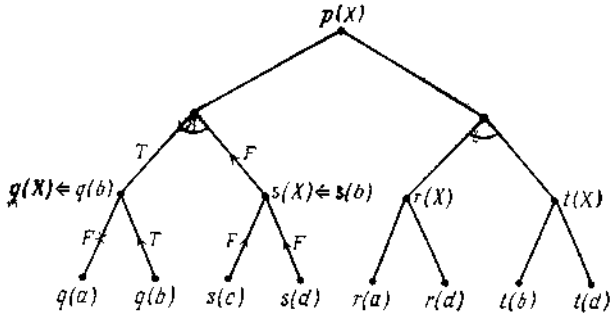


Рис. 22. s(X) терпит неудачу при X=b.

Поскольку вершина, связанная с *q(a),* при этом сразу терпит неудачу, там стоит литера, обозначающая неудачу.
При этом исчерпываются все возможности для успешного означивания *q(X).* Поэтому все левое поддерево ИЛИ-вершины *p(X)* терпит неудачу. Теперь впервые происходит обращение к правому поддереву вершины *p(X).* Вначале здесь также испытывается *X=a,* и это означивание приводит к неудаче. *X=d* приводит к успеху. На рис. 23 показано успешное означивание *X = d.*

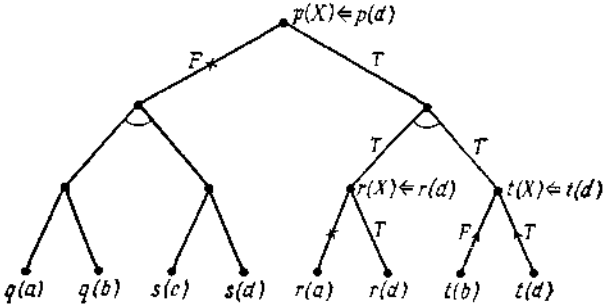


Рис. 23. При X=d на вершине p(X) достигается успех**.**

Выше было описано управление бэктрекингом, сопровождающим поиск в глубину по И-ИЛИ-дереву.
Простейшим способом реализации такого управления бэктрекингом является непосредственное использование И-ИЛИ-дерева.
При создании реальных систем метод непосредственной работы с

И-ИЛИ-деревом неэффективен. Фактически в реальных Пролог-процессорах, например в Пролог-процессоре системы DEC-10, вместо И-ИЛИ-дерева используется сложный стек, обеспечивающий высокую скорость обработки. Вопрос в том, как в этом случае реализовать управление сложного стека, построенного, исходя из И-ИЛИ-дерева.

2.9.2.4. Прерывания и процессы

При анализе управления бэктрекингом в разд.7.2.2.3 было рассмотрено, как можно последовательным путем реализовать недетерминированный параллелизм. Помимо параллелизма такого рода существует параллелизм, необходимый для работы последовательных структур. Примером такого параллелизма является параллелизм процессов, организуемых операционными системами. Операционные системы выполняют несколько заданий пользователей как бы "одновременно". Другими словами, активно используя высокое быстродействие ЭВМ, система в некотором промежутке времени выполняет последовательно несколько заданий. В общем случае такие системы **называют системами с мультипроцессированием**, а каждую **программную единицу, требующую обработки, — процессом**.
На рис. 24 представлена схема одновременного существования нескольких процессов: $p_1 \div p_n$.

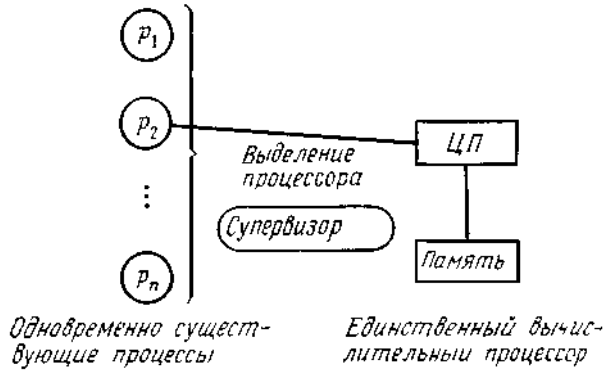


Рис. 24 Схема мультипроцессирования.

Машина передается в пользование этим процессам на некоторый фиксированный интервал времени в соответствии с некоторой заданной дисциплиной обслуживания.
Система, управляющая этими процессами, называется *планировщиком* или *супервизором.*

Смену процессов, требующих обработки, называют *переключением* процессов. Для повышения эффективности работы всей системы в целом в условиях мультипроцессирования эти переключения необходимо выполнять за минимальное время.
При управлении системой с мультипроцессированием следует принимать во внимание следующее:
1) В силу важности операций ввода-вывода необходимо вовремя принимать заявки на их обработку.
2) Операции ввода-вывода выполняются на порядок медленнее вычислительных операций в процессоре.
3) При выполнении операций с равными приоритетами операции, начатые раньше, следует и раньше завершать.
Первый пункт особенно важен в проектировании так называемых систем реального времени. Известно, что быстродействие человека значительно ниже быстродействия операций ввода-вывода, обслуживающих вычислительную машину. Однако если реакция системы на запросы требует некоторого времени (порядка 2—3 с), то у человека пропадает желание продолжать взаимодействие с такой системой. Поэтому, как правило, наиболее срочными являются процессы, связанные с устройствами ввода-вывода (по аналогии с человеческим организмом с рецепторными органами).
Эти процессы не являются компонентами вычислительной системы и запускаются человеком в результате изменения окружающей среды. Их называют *процессами обработки прерываний,* а явления, в результате которых запускаются данные процессы, — прерываниями или событиями.
Например, может возникнуть прерывание в результате нажатия человеком клавиши на клавиатуре. Вслед за этим запускается процесс считывания с клавиатуры одного знака, который и является процессом обработки прерывания.
Прерываниям, как правило, присваивают приоритеты, и при обработке предпочтение отдается прерываниям с более высоким приоритетом.
Прерываемый процесс в момент прерывания переводится в состояние ожидания. После окончания процесса, запущенного в результате прерывания, процесс, находящийся в состоянии ожидания, запускается снова.
Примером системы реального масштаба времени может служить, например, банковская система обработки счетов; если не учитывать времени, затрачиваемого на обработку прерываний, то работа программы распараллеливается.
Быстродействие системы обработки банковских счетов ограничивается скоростью, с которой человек нажимает на клавиши, и составляет

величину порядка нескольких минут на один счет, однако за это время вычислительная система выполняет аналогичные операции для нескольких десятков или сотен клиентов.
Для реализации принципа FIFO (first-in, first-out — первым пришел — первым обслужен), упомянутого в п. 3, необходима **структура данных, называемая *очередью.***
Как указывалось в разд.7.2.2.3, для обработки параллельных заявок можно было бы использовать стек, однако он реализует принцип LIFO (last-in, first-out — последним пришел — первым обслужен). Очередь же реализует принцип обслуживания FIFO. Схематически очередь представлена на рис. 25.

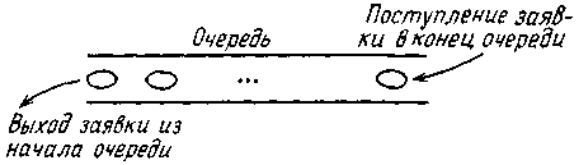


Рис. 25. Схема очереди.

Рисунок напоминает очередь, образованную людьми, стоящими к некоторому окошку. Над очередями определены две операции:
enqueue: queue × *α→queue*
dequeue: queue→<α,queue>
В отличие от стеков соотношение
dequeue (enqueue (Q,X)) = <X,Q>
не выполняется.
Введя для определения длины очереди операцию
length: queue→N, получим

$$\underbrace{dequeue(dequeue(\ldots dequeue(enqueue(Q,X))\ldots))}_{length(Q)+1} = \langle X, nil\rangle$$

Эффективная реализация очередей на компьютере требует определенных усилий. Если реализовывать простейшую очередь путем расположения ее элементов в ряд, то в цепочке *Q* длиной *L* первый элемент будет началом очереди, а L-й элемент — ее концом. Если переменной *INDEX* присвоить значение длины *L* цепочки, то при выполнении операции *enqueue* (постановки заявки в очередь) будут выполняться следующие действия:
INDEX←INDEX+1
Q[INDEX]←X,

где *X* — элемент, добавляемый к очереди (например, процесс). В этом случае добавление элемента к очереди осуществляется очень просто, однако при удалении элемента из очереди в ней многое меняется. При выполнении операции *dequeue* удаления заявки из очереди необходимо выполнить следующие действия:

```
X←Q[1];
Q[1]←Q[2];                    }
Q[2]←Q[3];                    }
    .                         }
    .                         } Передвижение вперед по одному
    .                         }
Q[INDEX−1]←Q[INDEX];          }
INDEX←INDEX−1
```

Таким образом, сокращение цепочки, состоящей из людей, выполняется просто, однако та же самая операция со структурами данных, находящимися в машине, весьма обременительна.
Упростить работу с очередями позволяет способ, показанный на рис. 26.

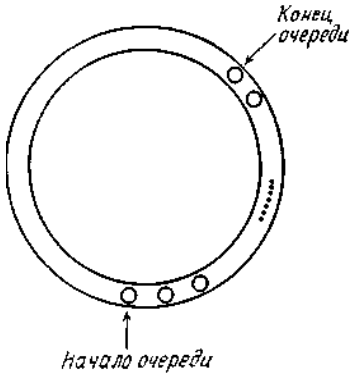


Рис. 26. Организация кольцевой структуры данных с перемещением указателей.

В этом случае данные объединены в кольцо, начало и конец очереди помечены с помощью указателей. При этом операции *enqueue* и *dequeue* значительно упрощаются.
ENQUEUE(Q,X) = TAIL←TAIL+1;
Q[TAIL] ← X;
DEQUEUE(Q) =X←Q[HEAD];
HEAD←HEAD+1*;*

На первый взгляд кажется, что образование кольцевых структур в памяти машины представляет собой сложную задачу, однако ее можно осуществить, например, взятием индексов элементов очереди по модулю длины очереди, т. е. значение индекса определять в виде *index* mod *length.* В этом случае при превышении длиной очереди значения *length,* индекс принимает значение нуль, что эквивалентно "закольцовыванию".
В очередях могут выполняться не только операции постановки в очередь и удаления из нее, т. е. *enqueue* и *dequeue;* ее элементам могут устанавливаться различные приоритеты в зависимости от содержательной стороны заявок. При этом получается очередь с приоритетами.

7.9.2.5. Управление конкурирующими процессами и их синхронизация

Методы управления и синхронизации обрабатывающих единиц (процессов), находящихся между собой в конкурентных отношениях, делятся на две большие группы. Одна из них включает в себя методы, основанные на общих переменных, вторая — на посылке заявок. Ниже изложены оба этих метода.
1) Методы управления и синхронизации, основанные на общих переменных.
Существуют несколько методов управления и синхронизации, основанных на общих переменных. Одним из них является **метод семафоров Дейкстры**, о котором и пойдет речь ниже.
Для реализации механизма семафоров используют переменные семафоров и функции *p* и *v.* Обозначим переменную семафора через *s*. Ее значениями могут быть целые неотрицательные числа. Если при выполнении *p(s)*-функции в работающем в данный момент процесс значение переменной s окажется равным нулю, то процесс останавливается, переводится в состояние ожидания, ставится в очередь, связанную с данным семафором, и будет находиться в этом состоянии до тех пор, пока начнет выполняться условие s>0. Если при этом имеется какой-либо другой процесс, готовый к выполнению, то он выполняется. Если s >0, то оператор *p(s)* уменьшает значение s на единицу. Разумеется, проверка значения *s* и вычитание единицы выполняются нераздельно. Соответствующую команду можно записать как test & set.
Если выполняется оператор *v(s),* то к значению переменной s прибавляется единица, и если в очереди к семафору s имеются процессы, то один из них берется из этой очереди, делается активным,

а от значения переменной отнимается единица. Операции *p* и *v* называют *взаимоисключающими,* они должны быть доступны внешним прерываниям.

Таким образом, через посредство внешней переменной осуществляется управление конкурирующими процессами и их синхронизация.

2) Управление и синхронизация посредством посылки заявок.

Метод посылки заявок, как следует из его названия, наряду с механизмами синхронизации имеет также механизмы передачи данных.

При использовании для связи и синхронизации метода посылки заявок процесс вместо записи в общую переменную (семафор) и считывания из нее выдает и получает сообщения. Если какой-то процесс получает сообщение, то тем самым реализуется **связь по информации**: сообщение не могло бы быть получено, если бы его не посылали, а этим как раз и реализуется синхронизация. Для передачи сообщений используются методы:

— назначения канала связи,
— синхронизации обмена;

и имеются различные пути их реализации, о которых речь пойдет ниже.

1) Метод назначения канала связи.

Для указания источника сообщения и процесса, выполняемого перед его посылкой, используется канал связи.

Функция канала связи состоит в посылке сообщений

— от одного процесса к одному процессу,
— от одного процесса к нескольким процессам,
— от нескольких процессов к нескольким процессам.

В данном разделе мы рассматриваем посылку сообщений от одного процесса к одному процессу.

В каналах типа "один процесс — один процесс" допускается посылка сообщений лишь от одного источника сообщений (т. е. процесса, **порождающего сообщения**) к одному процессу, получающему и использующему для своей работы посланное сообщение (т. е. процессу — **потребителю сообщения**). Существуют несколько способов назначения имен каналу связи при использовании этого метода. Одним из них является присвоение имени самому каналу связи, используемому между источником и потребителем. Пусть каналу связи между источником *P* и потребителем *C* присвоено имя *channel* 1. В процессе-источнике при выполнении

*channel*1!*X*

значение *X* может быть направлено в канал связи *channel* 1, а при выполнении в процессе-потребителе *C*

*channel*1?*Y*

значение, посланное по каналу *channel* 1, попадает в переменную У.

2) Синхронизация с помощью канала связи.

Посылка заявок по каналам связи может быть асинхронной и синхронной. Если предположить, что между процессом-источником и процессом-потребителем, соединенными между собой каналом, имеется бесконечной длины буфер, то процесс-источник не будет находиться в ожидании при посылке сообщения. Поэтому этот процесс может продолжать свою работу независимо от процесса-потребителя. Такую посылку заявок называют *асинхронной,* и именно этот метод используется в конкурентном Прологе.

С другой стороны, если предположить, что между процессом-источником и процессом-потребителем, соединенными между собой каналом связи, отсутствует какой бы то ни было буфер, то тогда процесс-источник должен подождать с посылкой заявки до тех пор, пока процесс-потребитель не будет готов к ее приему. Иными словами, посылка заявки от источника к потребителю будет возможна лишь тогда, когда откроется обмен между источником и потребителем по их совместной готовности.

Такую посылку сообщений называют синхронной; она применяется в языке Оккам.

3) Язык Оккам.

Одним из языков, снабженных функцией обмена-синхронизации на основе посылки заявок, является язык Оккам. Его мы и рассмотрим.

Язык Оккам был спроектирован и разработан английской фирмой Inmos на основе идей профессора Оксфордского университета Хоара о "последовательных процессах, взаимодействующих через обмен информацией". В языке Оккам отразилась философия самого профессора Хоара, нацеленная на создание простого и ясного языка программирования. В этой связи уместно вспомнить, что Оккам — имя средневекового философа, прославившегося своим принципом: **"Не увеличивать число сущностей сверх необходимого"**.

Программы на языке Оккам состоят главным образом из **примитивов**, т. е. **базовых процессов, конструкторов, допустимых выражений и формул.**

Существуют **три примитива**. Один из них — **процесс подстановки, соответствующий постановкам в формулы в существующих языках программирования. Два других процесса являются процессами ввода-вывода, характерными для языка Оккам.** Например, процесс вывода

c!e

выводит значение формулы *e* в канал связи *c.*

Далее, процесс ввода
c?*v*
присваивает введенное из канала значение переменной *v.*
Конструкторы используются для связывания процессов друг с другом, образуя, таким образом, более крупные процессы.
Рассмотрим основные конструкторы языка Оккам:"
— **конструктор *SEQ.*** Процессы, связанные конструктором *SEQ,* выполняются последовательно;
— **конструктор *PAR.*** Все процессы, связанные конструктором *PAR,* выполняются параллельно. Этот конструктор является одной из разновидностей оператора управления параллельным потоком.
— **конструктор *WHILE***. Процессы, соединенные конструктором WHILE, образуют структуру вида
WHILE <ФОРМУЛА><ПРОЦЕСС>
Здесь ПРОЦЕСС выполняется до тех пор, пока не станет ложным результат выполнения ФОРМУЛЫ.
Рассмотрим на примере программы, изображенной на рис. 27, обмен и синхронизацию на языке Оккам.

```
PROC GENERATE(VALUE INI, CHAN OUT) =
        VAR N:
        SEQ
            N—INI
                WHILE TRUE
                    SEQ
                        OUT ! N
                        N:=N+1
PROC OUTSTREAM(CHAN IN) =
        VAR I:
        WHILE TRUE
            SEQ
                IN ? I
                WRITE(I):
CHAN CH1
PAR
        GENERATE(1.CH1)
        OUTSTREAM(CH1)
```

Рис. 27. Пример программы.

В этой программе имеются два процесса. Процесс *generate* выводит в канал целые числа, начинающиеся со значения *int.* Процесс *outstream* выводит на устройство вывода значения, получаемые из канала. Поскольку среди процессов, связанных конструктором *PAR,* находящимся в конце программы, имеется процесс

generate (1,*ch*1)
то процесс *generate* передает через канал *ch*1 процессу *outstream* целые числа, начиная с единицы.
Поскольку, как уже говорилось выше, передача сообщений в языке Оккам осуществляется синхронно, процесс *generate* передает одно целое число, процесс *outstream* принимает его и после этого процесс *generate* порождает очередное целое число с помощью оператора
Часть программы, начинающаяся термином *PROC,* является идентификатором процесса. Кроме него в программе имеются идентификатор переменной *(VAR),* идентификатор значения *(VALUE),* идентификатор канала *(CHAN).*
Write(i) представляет собой процесс вывода на выходное устройство целого числа *i.* Будем считать, что он определен в другом месте.
Поскольку значение формулы, находящейся в операторе, начинающемся с конструктора *WHILE,* равно *TRUE,* эти два процесса *generate* и *outstream* не заканчиваются, а продолжают совместно работать. Так, в языке Оккам с помощью различных конструкторов задается поток управления и осуществляется синхронизация и обмен через канал между конкурирующими процессами.

3. Объекты и отношения

В данном разделе мы рассмотрим вопросы построения информационной модели окружающего нас мира, связывая эту задачу не только с концепциями, изложенными в предыдущем разделе, но и с архитектурой новой (не фон-неймановской) машины, позволяющей более естественно реализовать эти концепции.

3.1. Модель мира

3.1.1. Вещи и отношения

Окружающий нас мир предельно конкретен, однако если абстрагировать его с позиций познания, то мы придем к **понятиям "вещей" (объектов) и отношений, существующих между ними** Эти два типа понятий весьма глубоки и признаны как бы "атомарными", т. е. не допускающими дальнейшего дробления.
Таким образом, основываясь на **базовых понятиях объектов и отношений**, можно (по крайней мере с позиций интеллектуальных

задач) построить некоторую **модель мира**. При этом **существуют две основные точки зрения, различающиеся между собой по тому, что считать более важным: вещи или отношения.**
Одна точка зрения заключается в том, что **вещи обладают внутренней структурой и связаны с другими вещами посредством различных отношений.** Это хорошо согласуется с нашими непосредственными наблюдениями.
Расчленяя эти вещи на их составные части, мы видим, что казавшиеся единичными объекты имеют сложную структуру, распадающуюся на ряд отношений, существующих между этими более простыми компонентами. Продолжая расчленение, мы в конце концов приходим к простейшим "вещам", которые уже **в данной теории не обладают внутренней структурой и существуют в виде "точечных" объектов, связанных отношениями с другими объектами.**
От этой точки зрения можно перейти к другой, **ставящей в центр внимания отношения.**
Те же самые две точки зрения существуют и у программистов. **Точка зрения, рассматривающая в качестве основы вещи, проявляется в объектно-ориентированном программировании, а точка зрения, рассматривающая в качестве основы отношения, — в логическом программировании.**
Однако по сути своей эти две точки зрения не противоречат друг другу. Следует понимать, что они взаимно дополняют друг друга в постижении окружающего нас мира. **Представления, основанные на обеих точках зрения, должны переходить одно в другое путем взаимных преобразований**. Однако исследования в этой области пока еще не доведены до уровня технической реализации методов таких преобразований. Поэтому в настоящее время выбор делается на основе индивидуальных приоритетов и порой ведутся жаркие дискуссии о том, какая точка зрения предпочтительнее. По-видимому, по мере развития исследований острота этих дискуссий будет спадать.

Типы отношений

***Отношением (R)* называется взаимозависимость или взаимодействие двух и более объектов либо явлений абстрактного или конкретного типа.** При решении задач ИИ существенны объективные, определенные **отношения**, **которые поддаются описанию и соответствии с физическими или логическими законами**. **Отношения связывают отдельные элементы в различные системы.** Выражение «объект *X* находится в отношении *R* к объекту *Y* » символически обозначается *R(X, Y)* или *XRY*. Отношение может быть

рефлексивным, симметричным или транзитивным. Эти **типы** отношений можно охарактеризовать следующим образом:
а) рефлексивность — каждый объект эквивалентен самому себе;
б) симметричность — если один объект эквивалентен второму, то второй объект эквивалентен первому;
в) транзитивность — два объекта эквивалентны между собой, если они по отдельности эквивалентны третьему.
Если выполняются все три условия, то отношение называется *отношением эквивалентности.* Отношение между двумя объектами будет также называться корреляцией. Корреляция — это математическая модель отношения в обобщенной форме.

Виды отношений

Подобие. ***Подобие* — это отношение сходства между двумя или более системами (объектами, процессами, высказываниями), определяемое некоторыми общими свойствами**. Вообще говоря, возможен диапазон степеней подобия **от полного равенства *(идентичности)* до частного *сходства.* Можно говорить о функциональном, структурном и других видах подобия**. Обычно **подобие объектов** понимается как **одинаковость формы** (но, как правило, **не равенство по величине**). Отношение подобия имеет большое значение при математическом и физическом моделировании. Законы подобия позволяют определить условия, при выполнении которых результаты модельных экспериментов справедливы для реальных условий. Например, течения газа или жидкости подобны при равных числах Рейнольдса. **Область подобия может быть определена как пересечение множеств свойств, участвующих в данном отношении.**
Аналогия. Соответствие существенных **признаков**, **свойств, структур или функций объектов** или **явлений** будем называть ***аналогией***. Этот термин часто употребляется в том же смысле, что и подобие.
Гомоморфизм. Отношение между двумя системами, когда каждую составную часть и каждое отношение одной системы можно отобразить на некоторую составную часть и некоторое отношение второй системы (но не обратно), называется *гомоморфизмом.* В этом случае выполнение соответствующих условий подобия позволяет перенести результаты модельных экспериментов на натуру. **Область подобия может быть определена как пересечение множеств свойств.**
Изоморфизм. *Изоморфизмом* называется отношение между двумя системами, когда каждой составной части одной системы может быть поставлена в соответствие определенная составная часть другой

системы и наоборот (симметричность), а также, когда для каждого отношения между двумя соответствующими составными частями имеется такое же отношение в другой системе и наоборот.

Идентичность. Это отношение между объектами или процессами, **характеризующимися одинаковыми свойствами (признаками)** . *При абсолютной идентичности* должны быть одинаковыми все свойства, при *относительной* — только некоторые (в этом случае имеет место подобие).

Эквивалентность. Объекты или процессы называются эквивалентными, если между ними имеется отношение эквивалентности, т. е. **равноценности**. **Эквивалентность полнее идентичности, так как для последней характерна только рефлексивность**. Применительно к ИИ оба понятия будут использоваться как синонимы, т. е. **под эквивалентностью будет подразумеваться абсолютная идентичность**.

Математические функции. Важный класс отношений выражают *математические функции* как закономерные зависимости от переменной: $y = f(x)$. Такого рода математические функции выражают точно установленное отношение между x и y, т. е. **детерминированную связь.**

Причинность. Между причиной и вызванным ею действием существует асимметричное отношение. **Причина вызывает действие**. Существует строгая (детерминированная типа «если..., то») или ослабленная форма причинного отношения. **Причинная цепь имеет место, если действие выступает в качестве причины дальнейших действий.**

Связь. Если определенные выходы элемента (системы) одновременно являются входами какого-либо элемента (системы), то такого рода отношение называется *связью.* Связь может быть прямой (последовательной либо параллельной), обратной или комбинированной (рис. 1.4); **она может быть материальной, энергетической или информационной.**

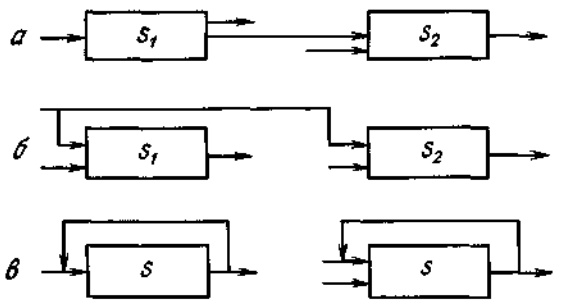


Рис.1. Виды связей между системами.
а последовательная, *б* — параллельная, *в*—обратная и комбинированная связи

Отношение цель — средство. Это — двухместное асимметричное отношение между системой целей (назначением, задачей) и средством их реализации.

Пространственное отношение. Отношение такого рода характеризует взаимное положение элементов отношения в пространстве. Пространственные отношения изучаются в топологии.

Логическое отношение. Логическим отношением (в логике — двух- или многоместным предикатом) называется отношение между объектами типа «l_1 меньше, чем l_2», или «l_3 находится около $_4$». Известными константами (**функторами**) являются: И; ИЛИ; И-ИЛИ; НЕ-ИЛИ; ТАК, ЧТО; ИЛИ-ИЛИ; ЕСЛИ-ТО; ТОЛЬКО ЕСЛИ-ТО; ТОЛЬКО ТОГДА-КОГДА; РАВНО. Из этого перечисления ясно, что многие описанные выше отношения являются также логическими отношениями. В ЭВМ реализация отношений такого рода осуществляется логическими элементами.

Временное отношение. Отношение такого рода описывает упорядочение процессов и событий во времени.

3.1.2. Исчисление предикатов

В исчислении предикатов за основу берутся отношения. Построение логики предикатов характерно для всей логики в целом, а разные логические системы строятся на основе разных формул. Предложено много различных логических систем. Установление соотношений между ними является одной из задач логики как науки. Какова же причина того, что логика предикатов продолжает занимать особое место в логических системах? Скорее всего это связано с тем,

что в качестве элементов этой логики, т. е. исчисления предикатов, используются простые формулы, а правила вывода, используемые в качестве логических операций, выражаются четко и определенно.
Можно рассмотреть также и объектно-ориентированные логические системы. По сравнению с исчислениями предикатов они обладают более сложными синтаксисом и правилами вывода. Исторически сложилось так, что логика Аристотеля, положившая начало теории правильных рассуждений, была объектно-ориентированной. Построенная им система силлогизмов из трех суждений (большой посылки, малой посылки и заключения) отличается большой сложностью. Однако, после того как были разработаны исчисления отношений, стало ясно, что многочисленные сложные фигуры силлогизмов по трем суждениям являются лишь различными модификациями базовых правил вывода.
Если исходить из задач математического исследования логических систем, то желательно, несколько абстрагируясь от действительности, располагать простыми и ясными формулами. Однако если исходить из интеллектуальных задач (в том числе и программирования), то следует иметь в виду необходимость обеспечения доступности системы ИИ пользователю, а также то, что объектно-ориентированный подход больше соответствует обыденным человеческим взглядам.
Конечно, при этом возникает множество разных проблем, требующих исследования, которые включают в себя также и технические аспекты, однако при этом общая картина могла бы иметь такой вид.
Элементами, образующими исчисление предикатов (для одной формулы), являются:

1) константные термы $c_1, c_2, \ldots$;
2) переменные термы $x_1, x_2, \ldots$;
3) функциональные буквы $f_1, f_2, \ldots$;
4) предикатные буквы $p_1, p_2, \ldots$;
5) логические символы $\rightarrow$, $\exists$ ... ;
6) специальное высказывание $\square$.

Число термов для каждого предиката фиксировано и называется его арностью. Термы определяются следующим образом:

1) константный терм есть терм;
2) переменный терм есть терм;
3) для функциональной, а также для предикатной буквы определена арность как целое неотрицательное число. Если арность функциональной буквы f есть n, а $t_1, t_2, \ldots, t_n$ — термы, то $(t_1, t_2, \ldots, t_n)$ также терм.

Приписывая термы предикатным буквам, получаем следующие формулы:

1) $\square$ есть формула;
2) если арность предиката p есть n, то

$$p(t_1,t_2,\ldots t_n)$$

— формула.
3) для двух формул p_1 и p_2 $p_1 \rightarrow p_2$ — формула;
4) если формула p содержит свободную переменную x, то

$$\exists x \cdot p(x)$$

— формула. (Формула $\exists x \cdot p(x)$ читается так: "существует x, при котором выполняется $p(x)$".) При этом x в этой формуле называют *связанной переменной.* Не связанные переменные являются свободными переменными.
Формулу, не содержащую свободных переменных, называют *замкнутой формулой* или *высказыванием.* Замкнутая формула может быть *истинной* или *ложной.* Символ $\square$ является ложной замкнутой формулой; он выражает понятие "противоречие". Формула $p(t_1, \ldots, t_n)$ говорит о том, что между термами $t_1, \ldots, t_n$ выполняется "отношение".
$A \rightarrow B$ означает, что если истинно A, то истинно B. $A \rightarrow \square$ означает, что A ложно. Это записывается как: $\neg A$.
Среди формул выделяются тождественно истинные формулы или аксиомы. Примеры аксиом

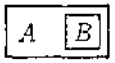
$\neg\neg A$ эквивалентно A A
$\rightarrow (B \rightarrow C)$ эквивалентно $B \rightarrow (A \rightarrow C)$

и т. д. (Эквивалентность в данном случае означает, что левая и правая части формулы принимают одинаковые логические значения при одинаковых наборах значений входящих э них переменных.).
Последняя аксиома, например, говорит о независимости порядка следования посылок, что позволяет записать:
$A_1A_2 \ldots A_n \rightarrow B$.
Примером правила вывода является правило *modus ponens:* если тождественно истинна формула A и при этом тождественно истинна формула: $A \rightarrow B$, то формула B тождественно истинна. Кроме этого правила, существуют правила вывода, учитывающие наличие квантора существования. Существуют также аксиомы, содержащие квантор $\exists$.
Строгое описание манипуляций с формулами и правилами вывода дается в учебниках по логике. Здесь же мы дадим графическое представление формул. Его называют диаграммами Пирса.
В его обозначениях формула $\neg A$, т. е. $A \rightarrow \square$, изображается в виде:

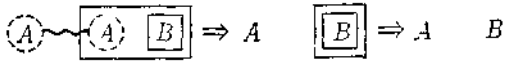
Таким образом, рисунок вида

соответствует формуле А.

Формула $A \rightarrow B$ изображается (в соответствии с формулой $\neg (A \& \neg B)$) в виде

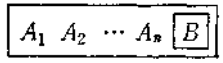


Если выполнимо *A*, то, согласно правилу вывода, выполнимо *B*. Эта операция изображается в виде



(Здесь *A,* находящееся вне прямоугольника, стирает *A,* находящееся внутри прямоугольника.)

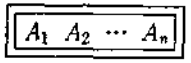
Поскольку несколько посылок можно записать подряд, то возможна также запись вида



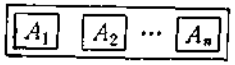
Запись без рамки

$$A_1 A_2 \ldots A_n$$

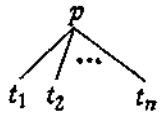
выражает тот факт, что выполняется каждая из формул: $A_1, A_2, \ldots, A_n$, т. е. имеет место конъюнкция. Это можно считать сокращенным видом записи



Дизъюнкция имеет изображение (соответствует формуле $\neg (\neg A_1 \& A_2 \ldots \& \neg A_n)$)

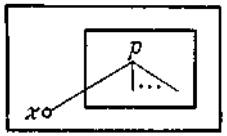


Далее, отношение выражается в виде дерева



Переменные термы изображаются кружочками, что дает рисунок вида

Например, формулу $\forall x.\ p(x)$ (т. е для всех *x* выполняется *p(x))* можно задать в виде $\neg \exists x.\ \neg p(x)$, и потому она может быть изображена на диаграммахПирса в виде



Эти графические представления являются одной из разновидностей так называемых семантических сетей.

Пусть имеется несколько истинных высказываний, задаваемых (трехместным) предикатом *p* (a_i, b_i, c_i — константные термы).

$p(a_1 b_1 c_1)$

$p(a_2, b_2, c_2)$

.

.

.

$p(a_m, b_m, c_m)$

Эту запись можно сделать в виде таблицы. В ней сведены вместе элементы отношения, задаваемого предикатом *p* и справедливого для *a, b, c* и других троек термов.

p:

1	2	3
a_1	b_1	c_1
a_2	b_2	c_2
⋮	⋮	⋮
a_m	b_m	c_m

Полная таблица, выражающая истинные высказывания, называется *базой данных.* Поскольку в ней основным понятием является понятие отношения, ее называют *реляционной базой данных.*

В качестве языка программирования можно рассматривать логическую систему Хорна, являющуюся частью исчисления предикатов. Ее так и называют *языком логического программирования.* Вкратце об этом написано в разд. 3 3. Одним из примеров языков подобного типа является язык Пролог.

Логика Хорна шире теории реляционных баз данных. При этом языки логического программирования естественным образом включают в себя (как часть) реляционные базы данных (разд. 3.2).

Языки программирования можно строить на основе понятия функции (общепринятого в математике). Функцию можно рассматривать как особый вид отношения, на которое наложены некоторые ограничения. В этом смысле языки функционального программирования можно считать особым видом языков логического программирования. Функциональное программирование также кратко рассмотрено в разд. 3.3.

3.1.3. Объектно-ориентированное программирование

Отношения, как правило, выполняются между двумя объектами. Однако в исчислении предикатов и в реляционных базах данных это понятие расширено и считается, что отношение выполняется между *n* объектами. Такое отношение называют *n*-арным. Обычное же отношение является бинарным отношением. С другой стороны, существуют и **унарные отношения**. **Унарное отношение обычно называют «свойством»**. Например:

МУЖЧИНА -{ИМЯ МУЖЧИНЫ}
ЖЕНЩИНА -{ИМЯ ЖЕНЩИНЫ}
ПРИЯТНЫЙ -{имя МУЖЧИНЫ или ЖЕНЩИНЫ}
УПОРНЫЙ -{ИМЯ МУЖЧИНЫ ИЛИ ЖЕНЩИНЫ}

Пусть

ОТЕЦ-ДОЧЬ {ИМЯ МУЖЧИНЫ, ИМЯ ДОЧЕРИ}

— схемы бинарного отношения. Зафиксируем один из элементов первого домена. Тогда получим унарное отношение

ОТЕЦ-ДОЧЬ {СЕРГЕЙ, ИМЯ ДОЧЕРИ}

Оно может быть пустым или содержать несколько элементов — имена всех дочерей СЕРГЕЙ , если у него есть хотя бы одна дочь. Отношение

ОТЕЦ-ДОЧЬ {ИМЯ МУЖЧИНЫ, ЕЛЕНА }

имеет единственный элемент.

Так из *n*-арных отношений извлекаются обобщенные зависимости, и, если имеется несколько отношений, где в качестве имени атрибута входит "Имя мужчины", аналогичным образом можно получить всю информацию, имеющуюся, например, о Сергее в следующих отношениях:

СЕРГЕЙ -МУЖЧИНА,
СЕРГЕЙ -ОТЕЦ (ЕЛЕНА),
СЕРГЕЙ -УПОРНЫЙ
и т.д.

Реляционные базы данных составляются, таким образом, из множества отношений-таблиц. В такой базе данных, например, имя лица, именуемого СЕРГЕЙ , разбросано по различным таблицам. Если база данных обладает цельностью, то она должна отражать всю информацию о Сергее. Однако может потребоваться собрать воедино всю информацию о Сергее. При этом всю информацию, извлеченную из реляционных таблиц, можно собрать в одно место, как описано выше, и создать базу данных по отдельным лицам.

Тогда свойства соберутся следующим образом:

СЕРГЕЙ —МУЖЧИНА,
— УПОРНЫЙ
и т. д.

Для *n*-арных отношений (включая сюда и бинарные) можно ввести обобщенные "свойства" (зависимости). Например, на основании отношения ОТЕЦ-ДОЧЬ (СЕРГЕЙ, ЕЛЕНА) можно сказать, что Сергей имеет свойство ОТЕЦ-ДОЧЬ (*, ЕЛЕНА), или это можно переписать в виде ОТЕЦ (ЕЛЕНА).

Таким образом, для индивида по имени Сергей можно собрать группу соответствующих ему свойств. (Существует и такое философское учение, согласно которому индивид полностью задается совокупностью его свойств.)

Это — объективно-ориентированный подход; он дополняет реляционный подход.

Категории свойств в объектно-ориентированном представлении знаний часто выражают в явном виде. Например:

СЕРГЕЙ ~ пол: МУЖСКОЙ,
характер- УПОРНЫЙ

Это соответствует присвоению имен ролям термов в отношении. При этом сведения о конкретном индивиде не записываются непосредственно в базу данных, а выводятся из знаний общего вида, подобно знаниям, понимание которых реализуется на основании общих законов.

Например, если ввести для человека общие категории:

ЧЕЛОВЕК ~ число рук 2,
число ног 2

и если справедливо утверждение "Сергей — человек", иными словами, если можно записать формально

СЕРГЕЙ —ЧЕЛОВЕК,

то в базу данных уже можно не заносить сведения о количестве рук и ног у Сергея. Таким образом, разбив всех индивидов на классы, можно записать в обобщенном виде категории для каждого класса. Индивид,

принадлежащий некоторому классу, наследует свойства из этого класса.
Классы можно также классифицировать и структурировать по-разному. Например, можно указать, что человек является живым существом. Таким образом, объективно-ориентированные структуры данных представляют собой более структурированные образования, чем модели, построенные просто на основе реляционных баз данных. Однако если пополнить реляционную модель общими законами, то и та и другая модели станут эквивалентными.
При рассмотрении логических программ, содержащих в себе реляционные базы данных, можно установить соответствие их с объектно-ориентированными моделями.
Хотя в этих двух подходах нет существенной разницы, их особенности обусловлены по крайней мере традицией научных исследований в этой области. **В объектно-ориентированных языках есть возможность выражать не только статические свойства индивидов и классов, но и учитывать их динамику. Одни объекты могут посылать сообщения другим объектам. Благодаря этому удается построить модели с изменяющимся внутренним состоянием**. Обычное (1 порядка) исчисление предикатов отражает статическую сторону упрощенной модели мира. Однако ведутся исследования по обобщению логики предикатов, позволяющему моделировать динамические свойства. Примерами могут служить логические системы, называемые временными логиками и т. д.
Механизмы (интерпретации), позволяющие осуществлять обмен сообщениями, вводятся и в языки логического программирования.

3.2. Реляционные модели

3.2.1. Реляционное представление знаний

Реляционная модель была предложена Коддом в 1970 г. Она представляет собой модель данных, в которой база данных представлена набором *n*-арных отношений.

n-арным отношением *R* называют произвольное подмножество декартова произведения $D_1 \times D_2 \times ... \times D_n$ множеств $D_1, D_2, ... D_n (n \geq 1)$. При этом $D_1, D_2, ..., D_n$ называют *доменами,* а элемент отношения — кортежем $\langle v_1, v_2, ..., v_n \rangle \in R(v_i D_i, i=1, ... n)$. С математической точки зрения отношение может быть бесконечным множеством, однако в реляционных моделях его предполагают конечным.

Домены D_i и $D_j (i \neq j)$, которые служат основой для задания одного отношения, могут быть одним и тем же множеством. Например, при моделировании расписания авиарейсов естественно предположить, что среди множеств <"номер рейса", "аэропорт отправления", "аэропорт назначения", "время отправления"> и т. д. в качестве множеств D_2 и D_3 взять множество АЭРОПОРТ. Чтобы различать эти множества D_i, им присваивают имена и называют их *именами атрибутов* (или просто *атрибутами).* Множество имен атрибутов $\{A_1, A_2, ..., A_n\}$ отношения *R* (в порядке их следования) называют *схемой отношения.* Схему отношения записывают также в виде $R\{A_1, A_2, ..., A_n\}$.,
Отношения удобно изображать в виде таблиц, как представлено в табл. 1.

Таблица 1.

Отношение: «Места, посещаемые туристами»

Наименование местности	Центр префектуры	Расстояние от Токио, км
Камакура	Канагава	51
Атами	Сидзуока	105
Ито	Сидзуока	122
Симода	Сидзуока	135

Данная таблица представляет отношение: "Места, посещаемые туристами", состоящее из трех атрибутов: "наименование местности", "наименование центра префектуры", "расстояние от Токио".
При изображении отношений с помощью таблицы **ее строки соответствуют кортежам, а ее столбцы — атрибутам.** Поскольку отношение было определено как подмножество, порядок следования кортежей, т. е. положение строк в таблице, роли не играет. По сравнению с ним порядок следования атрибутов (столбцов) входит в схему отношения. Можно определить отношение независимо от порядка следования атрибутов, однако для сохранения связей с логическим программированием будем считать существенным порядок следования атрибутов в отношении.

Подмножество $K = \left\{ A_{i_1}, A_{i_2}, ..., A_{i_k} \right\}$ заданной схемы $\{A_1, A_2, ..., A_n\}$ отношения *R* будем называть ключом, если в двух различных кортежах отношения *R* значения в каждом атрибуте из *K* не все одинаковы, и ни одно собственное подмножество *K* этим же свойством не обладает.

Другими словами, **ключ — это такое множество атрибутов, которое при заданных значениях этих атрибутов однозначно определяет кортеж (и при этом не содержит лишних атрибутов).**
В обычных языках программирования файл часто считают множеством записей, а запись — множеством полей**. Отношение можно представить файлом, кортежи отношений — записями файла, атрибуты кортежа — полями записи, таким образом, отношение можно считать строгим определением файла.** Поскольку понятия записи, файла и т. д. используются в других моделях данных, одно только задание отношений нельзя считать достаточным для определения реляционной модели. Как правило, базу данных считают собранием файлов (отношений), однако при отображении реального мира посредством базы данных файлы не следует помещать на диски беспорядочно, их необходимо организовать в виде **некоторого набора файлов, органически связанных между собой**. **Модель данных приобретает свою индивидуальность именно в зависимости от того, каким образом установлены эти связи.**
В реляционной модели базу данных рассматривают как набор файлов. **Структура базы данных выражается набором схем отношений и называется *схемой базы данных.*** В схеме реляционной базы данных в отличие от других моделей данных связи между отношениями не записываются в явном виде (за исключением ограничений на целостность базы данных), о чем говорится в разд. 3.2.5. Таким образом, структура реляционной базы данных выражается чрезвычайно просто. **Операции над базой данных выполняются с помощью одной или нескольких реляционных операций над отношениями.** В других моделях данных операции выполняются над отдельными записями, а особенностью реляционных операций является то, что они выполняются над множествами в целом.
Теория реляционных баз данных является частью исчисления предикатов. Совокупность операций, порождающих новые таблицы из совокупности существующих таблиц, задающих отношения, определяет реляционную алгебру.
Реляционные операции делятся на две большие группы: поиск (запрос) и обновление данных. Метод формирования поисковых предписаний изложены в следующем разделе. Примерами методов могут служить **реляционная алгебра, представляющая собой систему операторов, задающих операции над множествами, а также реляционное исчисление, основанное на исчислении предикатов первого порядка.** Обновление же базы данных, как правило, задается операциями, ограничивающимися одиночными отношениями. При этом возникает ряд проблем, одной из которых

является опасность нарушения целостности базы данных в случае, когда производится обновление нескольких отношений.
К операциям обновления относятся:
а) операция добавления;
б) операция модификации кортежей, удовлетворяющих поисковому предписанию;
в) операция удаления кортежей, удовлетворяющих поисковому предписанию.
Все эти три операции выполняются над отдельно взятым отношением. Что касается объекта, над которым выполняется обновление, то в качестве источника формирования поискового предписания может быть взято другое отношение.

3.2.2. Первый вид операций над отношениями: реляционная алгебра

Поскольку отношения являются множествами, одним из естественных способов выполнения операций над отношениями является использование системы операций над множествами. В качестве такой системы операций была предложена реляционная алгебра. **Важной особенностью реляционной алгебры является простое соответствие между выразительными возможностями различных языков формирования запросов и процедурами, реализующими эти запросы**. Таким образом, если еще можно отстаивать выразительные возможности языков запросов, в которых не используется реляционная алгебра, то возможности выполнения запросов чаще всего реализуются с **помощью языков, использующих реляционную алгебру**.
Реляционная алгебра включает пять основных операций и четыре вспомогательных: сумму, разность, пересечение, декартово произведение, выборку, проекцию, Θ-соединение, обыкновенное соединение и деление.
1**. Сумма**. Суммой отношений R и S : $R \bigcup S$ является множество всех кортежей, содержащее кортежи из R и S.
2. **Разность**. Разностью отношений R и S: $R — S$ является множество всех кортежей, содержащее кортежи из R и не содержащее кортежи из S .
3. **Пересечение**. Пересечением $R \bigcap S$ отношений R и S является множество кортежей, одновременно содержащихся в отношениях R и S.

4. **Декартово произведение**. Декартовым произведением $R \times S$ отношений R и S является такое множество кортежей $t = < r_1, r_2, ..., r_m, s_1, s_2, ..., s_n >$, что $< r_1, r_2, ..., r_m > \in R$ и $< s_1, s_2, ..., s_n > \in S$.

Четыре перечисленных выше операции совпадают с обычными операциями над множествами. Операции суммы, разности и пересечения выполнимы лишь тогда, когда отношения R и S являются подмножествами декартовых произведений одних и тех же доменов.

Кроме того, для пересечения $R \bigcap S$ можно задать соотношение:

$$R \bigcap S = R — (R — S).$$

Имеются также особые операции над отношениями.

5. **Выборка**. Выборкой $\alpha_F(R)$ из отношения R является множество всех кортежей отношения R, для которых истинен предикат F. Если мы будем называть **компаративными термами** вычислительные операторы сравнения ($=, >, <, \geq, \neq$) атрибутов с константами, либо атрибутов с атрибутами, то предикат F выразится комбинацией булевских операторов ($\wedge, \vee, \neg$ } над такими термами.

6. **Проекция.** Пусть $R \subseteq A_1 \times A_2 \times ... \times A_n$ — я-арное отношение.

Проекцией $\pi_X(R)$ отношения на множество $X = \left\{ A_{i_1}, A_{i_2}, ..., A_{i_\kappa} \right\}$ доменов называется такое множество всех кортежей

$$t = <v_1, v_2, ..., v_k>,$$

что существует кортеж

$$r = < r_1, r_2, ..., r_m >$$

отношения R, в котором $r_{i_j} = v_j$ (j=1, 2, ..., k).

Другими словами, проекция представляет собой операцию создания множества кортежей путем отбора из каждого кортежа отношения R указанных атрибутов. При этом могут возникнуть кортежи, у которых все значения атрибутов полностью совпадают. Поскольку отношение определено как множество, все повторения в этом случае удаляются. Комбинированием основных операций реализуют ряд других операций реляционной алгебры.

7. **Θ-соединение**. Θ-соединение $R \overset{\triangleright\triangleleft}{A_i \theta A_j} S$ отношений R и S, где m — число доменов в R, определяется так: пусть Θ — вычислительный оператор сравнения; A_i — имя атрибута с i-м номером в отношении R; A_j — имя атрибута с j- м номером в отношении S. Тогда

$$R \overset{\triangleright\triangleleft}{A_i \theta A_j} S = \sigma_{A_i \theta A_{m+j}} (R * S).$$

Здесь A_{m+j} — имя атрибута с номером $m+j$ в $R \times S$, где m — число доменов в R.

Частный случай Θ-соединения, когда Θ представляет собой равенство, называется *эквисоединением.*

8**. Обыкновенное соединение**. Обыкновенное соединение $R \triangleright\triangleleft S$ отношений R *и* S определяется следующим образом.

а) Пусть количество одинаковых имен атрибутов в R и S равно k. Чтобы различать атрибуты того и другого отношения, будем обозначать их точечной записью в виде: $R.A_{i_e}$. Обозначим

$$T = \sigma_{R \cdot A_{i_1} = S \cdot A_{j_1} \wedge \ldots \wedge R \cdot A_{i_k} = S \cdot A_{j_k}} (R * S),$$

где $R.A_{i_e}$ и $S.A_{i_e}$ (l=1, ..., k) — имена одинаковых атрибутов отношений R и S соответственно.

б) Если обозначить количества атрибутов в отношениях R и S через m и n соответственно, то количество имен атрибутов среди $S.A_j$, не равных именам атрибутов из R, будет равно $n — k$. Обозначив их через $S.A_{s_1}, ..., S.A_{s_{n-k}}$, получим

$$R \triangleright\triangleleft S = \pi_{R \cdot A_1, ..., R \cdot A_m, S \cdot A_{s_1}, ..., S \cdot A_{s_{n-k}}} T.$$

Как видно, при заданных R и S выражение $R \triangleright\triangleleft S$ определяется однозначно, и поэтому, как и в эквисоединении, нет необходимости указывать атрибуты, по которым выполняется соединение. Обыкновенное соединение играет важную роль при создании отношений.

9. Деление. Пусть количества атрибутов в отношениях R и S равны $m + n$ и n соответственно. Обозначим кортеж из R через $r = <u, v>$. Здесь u — сокращенная запись для $r_1, ..., r_m$;
v — сокращенная запись для $r_{m+1} ..., r_{m+n}$. Аналогично обозначим кортеж из S через $<w>$. Тогда частным от деления R на S, $R \div S$, будет такое множество $<u>$, что для всех кортежей $< w >$ из S существует кортеж $<u, w>$ из R. Деление можно выразить через базовые операции реляционной алгебры

$$R \div S = \pi_{A_1, ..., A_m}(R) — \pi_{A_1 .., A_m}\left((\pi_{A_1, ... A_m}(R) * S) — R\right).$$

Данную операцию называют *делением* в связи с тем, что для нее выполняется условие

$$(R \times S) \div S = R.$$

При этом соотношение $(R \div S) \times S = R$ не выполняется.

Итак, мы ввели девять операций реляционной алгебры. При определении этих операций и в комментариях к ним были показаны способы образования всех остальных из пяти базовых операций. Среди них особенно важным является понятие соединения, в частности эквисоединепия, часто используемого при формировании запросов. Помимо этих существуют также и другие операции, например полусоединение, используемое в связи с оптимизацией обработки запросов. При формулировании реляционных операций с помощью реляционной алгебры одних и тех же результатов можно добиться путем сочетания различных операторов. Это происходит в силу ряда причин, например, использования в определении операции выборки операции соединения, реализованной через булевские операторы над компаративными термами. В частности, выполняются соотношения:

$$\sigma_{A_i=a}\left(\sigma_{A_j=b}(R)\right) = \sigma_{A_i=a \vee A_j=b}(R)$$

и

$$\sigma_{A_i=a}(R) \cup \sigma_{A_i=b}(R) = \sigma_{A_i=a \vee A_i=b}(R).$$

Эти свойства, а также свойство коммутативности, выполняемое для некоторых операторов, используются для оптимизации обработки при запросах.

3.2.3. Второй вид операций над отношениями: реляционное исчисление

Реляционное исчисление является одним из видов исчисления предикатов первого порядка, используемого в качестве основы для создания языка запросов к базе данных. В реляционной алгебре операции над отношениями выражаются в виде комбинации нескольких операторов. В реляционном же исчислении задаются формулы исчисления, которым должны удовлетворять отношения, являющиеся результатами операций. Существуют **два вида реляционных исчислений: реляционное исчисление кортежей**, в котором используются переменные для кортежей, а также **реляционное исчисление доменов**, в котором используются переменные для доменов. Начнем с реляционного исчисления кортежей, предложенного Коддом

В реляционном исчислении кортежей (для краткости будем его называть исчислением кортежей) запрос имеет вид

{t|f(t)}

где *t* — переменная для кортежей; *f* — формула исчисления. Смысл данного запроса состоит в том, чтобы найти множество кортежей, для

которых формула *f* истинна. Вначале введем элементы исчисления, а затем определим понятие формулы исчисления кортежей.

1. Если *t*—переменная для кортежей, то *R(t)* обозначает, что *t* принимает значения из множества кортежей отношения *R*.

t[i] обозначает *i*-ю компоненту кортежа *t* (значение атрибута).

2. Если *t* и *u* — переменные для кортежей, а θ — знак вычислительной операции сравнения, то монадические термы объединения будут определяться одним из следующих способов:

а) *t*[*i*] θ<константа> и (константа) θ*t[i],*

б) *t[i]* θ и [*j*].

Тогда правильно построенная формула (ППФ) исчисления кортежей рекурсивно определится следующим образом:

1. *R(t)* есть *ППФ.*
2. Терм объединения есть *ППФ.*
3. Если *f — ППФ,* то *(f)* и *(¬f) —ППФ.*
4. Если *f* и *g — ППФ,* то *(f∧g) —ППФ* и *(f∨g)—ППФ.*
5. Если *f — ППФ* и *t —* свободная переменная в *f,* то ∃ *t(f)* и ∀ *t(f)— ППФ.*
6. Других *ППФ* нет.

Здесь переменные в *ППФ* могут быть свободными или связанными. Свободные и связанные переменные определяются согласно следующим правилам.

1. Переменные для кортежей *в R(t)* ив термах объединения являются свободными переменными.
2. Переменные для кортежей в формулах: (*f*) и (¬)*f*) являются свободными или связанными переменными в зависимости от того, являются ли они свободными или связанными в формуле *f*. В формулах *(f∧g)* и *(f∨g)* переменные для кортежей являются свободными или связанными в зависимости от того, связаны ли они в формулах *f* и *g*.
3. Переменная для кортежа *t,* свободная в формуле *f*, является связанной в формулах *(∃ t) (f)* и *(∀ t)(f)*. Связанные переменные, отличные от *t,* связаны в формулах (∃ *t*) (*f*) и *(∀ t) (f)* в зависимости от того, связаны ли они в формуле *f*.

Если *f(t)*—ППФ, в которой *t* — единственная свободная переменная для кортежей, то выражение

{*t*|*f*(0}

называют формулой исчисления кортежей. Запросы к базе данных выражают в виде формул исчисления кортежей, например в виде

ПРИМЕР СУММА $R \cup S:\{T|R(T) \vee S(T)\}$

ПРОИЗВЕДЕНИЕ $R \cap S: [T|R(T) \wedge S(T)\}$

РАЗНОСТЬ $R\text{-}S:\{T|R(T) \wedge \neg S(T)\}$

ПРЯМОЕ ПРОИЗВЕДЕНИЕ $R \times S:\{T| \exists U)(\exists V)(R(U) \wedge S(V) \wedge T_1 = U_1 \wedge \ldots \wedge T_M = U_M \wedge T_{M+1} = V_1 \wedge \ldots \wedge T_{M+N} = V_N)\}$

Аналогично можно определить проекцию и выборку и таким образом выразить все **пять базовых операций реляционной алгебры через формулы исчисления кортежей**.

Ясно, что с помощью только что определенных формул исчисления кортежей можно выражать запросы, однако всегда ли этими формулами можно выразить отношения? Например, какой смысл имеет формула

$$\{t| \neg R(t)\}?$$

Рассмотрим, в каких же случаях формулы *f(t)* исчисления кортежей соответствуют отношениям, т. е. в каких случаях запросы имеют смысл.

Прежде всего, чтобы $\{t|f(t)\}$ становилось множеством, при значении t_0 переменной *t* значение $f(t_0)$ формулы *f(t)* должно быть истинным или ложным. Например, в формуле $f(t) = (\forall u)\ (t \neq u)$ область значений переменной *u* явно не задана, и значение истинности формулы не определено. В формулах *f(t)* вида

$$f(t) = (\forall u)(\neg R(u) \vee (t \neq u)) \quad \text{или} \quad \neg(\exists u)(R(u) \wedge (t = u))$$

область значений *u,* доставляющих истинное значение формуле *f(t),* ограничена, и поэтому задание значений *t* определяет истинность или ложность всей формулы. Следовательно, **необходимо задавать области значений как связанных, так и свободных переменных**.

Если в качестве областей значений переменных кортежей взять множества, задаваемые декартовым произведением доменов, то формула $\{t|f(t)\}$ соответствует, как правило, множеству, однако если это множество не является конечным, то при реальной работе с базой данных оно теряет смысл. **Поэтому области значений переменных для кортежей всегда ограничивают с помощью специально заданных выражений в явном виде.**

Обозначим через *DOM(f)* объединение множества констант в формуле *f* и множества компонент кортежей, фактически существующих в отношениях, входящих в формулу *f*. Поскольку отношения являются конечными множествами, то *DOM(f)* — также конечное множество. Если при этом выполняются следующие три условия, то формула $\{t|f(t)\}$ называется соответственной **диапазонной ППФ**.

1. Если функция *f(t)* истинна, то каждая компонента кортежа *t* должна быть элементом *DOM(f);* для кортежа *t* с компонентами, значения которых не являются элементами множества *DOM(f),* формула *f(t)* не должна быть истинной.
2. Некоторая подформула формулы *f*, имеющая вид $\exists (u)\ (g(u)),$ истицна, если подформула *g(u)* истинна для любых значений свободных переменных иных, нежели *u,* при некотором значении *u,* которое должно быть элементом множества *DOM(f).*
3. Если некоторая подформула формулы *f* имеет вид $(\forall u)\ (g(u)),$ то ее можно преобразовать к эквивалентному виду $\sim (\exists u)\ (g(u))$; при этом для подформулы $(\exists u)(\ (g(u))$ выполняется условие 2.

Условие 1 является ограничением на свободные переменные, а условия 2 и 3 — ограничениями на связанные переменные. Например, формула $\{t| \neg R(t)\}$ противоречит условию 1, а формула $\{t|R(t) \wedge (\forall u)\ (t \neq u)\}$— условию 3. Поэтому обе они не являются соответственными диапазонными *ППФ.*

Известно, что соответственные диапазонные *ППФ* исчисления кортежей по своей выразительной силе эквивалентны реляционной алгебре.

Кроме реляционного исчисления и исчисления кортежей известно исчисление доменов. В исчислении доменов вместо переменных для кортежей используются переменные для доменов, определяемые почти так же, как в исчислении кортежей. **Введено три типа исчислений доменов, эквивалентных исчислению кортежей**; все они по выразительной силе эквивалентны друг другу. Языки запросов, обладающие по крайней мере эквивалентной им выразительной силой, называют *реляционно-полными.*

Среди языков, основанных на исчислении кортежей, имеются такие языки, как Alpha, QUEL и т. д. Языком, основанным на исчислении доменов, является язык QBE. Кроме того, существует язык SQL, который сочетает в себе свойства исчисления кортежей и реляционной алгебры.

В реляционных операциях, имеющих самое непосредственное отношение к реляционному исчислению, используется **форма Хорна**. В основу ее применения положен единый взгляд на два факта: истинность предиката $r(x_1, \ldots, x_n)$ и принадлежность кортежа $<x_1, \ldots, x_n>$ множеству кортежей отношения *R.* Например, если выполняется условие

$r(X,Y):\text{-}p(X,Z)q(Z,Y),$

то существуют отношения *P* и Q, соответствующие предикатам *p* и *q*. При этом множество всех кортежей, являющихся результатом поиска

по запросу [:—$r(X, Y)$], совпадает с результатом эквисоединения $R = P \underset{A_2=A_1}{\triangleright\triangleleft} Q$. Способ выражения множества реляционных операций через формы Хорна позволяет выразить пять базовых операций реляционной алгебры (а следовательно, и все составные операции), и поэтому он реляционно-полон. **Он позволяет также выражать рекурсивные операции и поэтому обладает большей выразительной силой по сравнению с реляционным исчислением**. Если в этом методе форму Хорна считать дедуктивным правилом, то обработку запроса можно мыслить себе как процедуру дедукции. Следовательно, СУБД, в которых реализован этот метод, можно называть дедуктивными СУБД.

3.2.4. Управление реляционной базой данных

В реляционных СУБД необходимы специальные механизмы управления, чтобы избежать противоречий, которые могут возникать при порче содержимого отношений. Имеются следующие **четыре важные функции управления:**

1. Управление целостностью.
2. Восстановление данных (при сбоях и т. д.).
3. Синхронизация выполнения процессов, имеющих общие ресурсы.
4. Разграничение прав доступа к данным.

Эти функции необходимы не только в системах, которые используют реляционную модель данных, но и в системах с другими моделями данных.

Обеспечение целостности базы данных накладывает ограничения на отношения (называемые ***правилами целостности***). Функцией системы обеспечения целостности является недопущение противоречий в данных относительно этих условий (противоречия могут возникать в результате ошибок пользователей при вводе данных путем модификации базы данных).

Восстановление данных представляет собой функцию приведения данных в некоторое исходное состояние при опасности возникновения противоречий в данных. Это может произойти при ошибках пользователя или при незавершенных операциях модификации по причине сбоев в системе.

Синхронизация процессов, претендующих на общие ресурсы, представляет собой функцию управления, используемую для предотвращения противоречий в данных, которые могли бы возникнуть в результате конкуренции процессов обработки при одновременной работе нескольких пользователей с базой данных.

Разграничение прав доступа есть функция, не допускающая изменения данных посторонними лицами и чтения ими секретных данных. В этой работе мы остановимся главным образом на представлении реляционной модели и на операциях с ней.

Основными ограничениями, накладываемыми функцией обеспечения целостности данных, являются следующие.

1. **Однозначность кортежей**: в одном отношении недопустимо существование одного или более кортежей, все значения атрибутов которых одинаковы. Это — естественное ограничение при определении отношений как множеств.

2. Выше уже говорилось, что множество атрибутов, однозначно определяющих кортеж, называется *ключом.* Ключ, используемый для выделения кортежей, называется *главным ключом.* Он должен удовлетворять следующим требованиям:

2. а) однозначность ключа: значения ключа для двух или более кортежей отношения должны быть различными;

б) непустое значение главного ключа: даже в тех системах, где допускаются пустые значения атрибутов (незаданные или неизвестные), главный ключ не должен иметь пустых значений.

Ограничение 2 вытекает из ограничения 1.

3. Существуют также ограничения на связи между отношениями.

Главными доменами называются домены, присутствующие в главных ключах нескольких отношений.

3. **Целостность связей** заключается в том, что в атрибутах над главными доменами не должно появляться значений, отличных от значений главных ключей в кортежах, фактически существующих в отношениях (в том числе пустых значений). В качестве примера целостности по связям приведем три отношения: отношение "ТОВАР" (номер_товара,...), которое приводится в разд. 3.2.5 в качестве примера проектирования базы данных, отношение "ПОСТАВЩИК" (номер_поставщика...), отношение "ПОСТАВКА" (номер_поставщика, номер_товара...). Здесь номер_товара и номер_поставщика являются главными ключами отношений ТОВАР и ПОСТАВЩИК соответственно. Если при этом номер _поставщика (либо номер_товара) в отношении ПОСТАВЛЕННЫЕ ТОВАРЫ принимает значения, не содержащиеся в кортежах отношения ПОСТАВЩИК (либо отношения ТОВАР), то это означает, что имеется поставщик (или товар), не существующий на самом деле.

Однозначность кортежей в сочетании с ограничениями на ключ называют ***существенной целостностью.*** Если язык запросов к СУБД реляционно-полон и при этом обладает функциями, позволяющими

реализовать существенную целостность и целостность по связям, то такую СУБД называют ***полностью реляционной***.
Помимо основных ограничений существуют еще и дополнительные ограничения. Они обычно задаются в виде дополнительной информации к схеме базы данных. Вот некоторые из них.
1. Перечисление значений для тех атрибутов, значения которых ограничены некоторой областью.
2. Указание отношения < между атрибутами или задание арифметического выражения для них, например:

$$A_1 < A_2,\ A_1 + A_2 + A_3 = 1.$$

3. Указание связей между значениями, существовавшими до модификации базы данных, и значениями после модификации базы данных, например:

```
(Новая) зарплата≥ (Старая) зарплата
(Новая) зарплата< (Старая) зарплата*2
```

4. Задание условий, которым должны удовлетворять кортежи при модификации, например "при выполнении операции заведения нового кортежа должно выполняться условие: $''A_1 = 0$" или
"Нельзя вычеркивать кортежи, у которых не выполняется: A_1=0" и т. д.
Важное значение имеет проверка выполнимости ограничений. Это связано с тем, что при сложной модификации базы данных не удается модифицировать все связанные данные сразу, и поэтому в отдельные моменты времени они могут оказаться противоречимыми. Однако по завершении обработки все условия могут выполняться. Бывает и так, что значения, удовлетворяющие ограничениям, получают в результате работы процедур, которые поступают в систему и регистрируются в ней тогда, когда возникают какие-либо условия на данные. При этом проверка данных на непротиворечивость по отношению к ограничениям не производится. Пусть, например, в базе данных существуют атрибуты, определенные в виде суммы и среднего значения атрибутов некоторого отношения. Тогда при изменении значений исходных атрибутов должны автоматически вноситься изменения в сумму и среднее.
Проблема восстановления данных связана с понятием транзакции. **Транзакция представляет собой связанную последовательность реляционных операций, которые, однажды начавшись, должны выполняться без перерывов.** При прерывании транзакции могут возникнуть противоречия в базе данных. Например, при переводе денег с одного счета на другой операции поднятию денег с одного счета и помещению этих денег на другой счет представляют собой одну транзакцию, которая приобретает смысл лишь после того, как завершены обе операции: снятие денег и помещение денег. Если по

каким-либо причинам продолжение транзакции прерывается и ее продолжение с прерванного места не представляется возможным, то данные с помощью функции восстановления приводятся к состоянию, предшествовавшему транзакции. Для реализации функции восстановления данных используются файлы частичных изменений, вспомогательные файлы текущего дублирования и т. д.
Управление одновременно выполняющимися процессами осуществляется в многопользовательском режиме, когда имеет место конкуренция за выполнение операций над одними и теми же отношениями. В принципе эта проблема аналогична той, которая возникает в операционных системах при управлении конкурирующими процессами. Однако в базах данных имеются свои особые методы, связанные, с тем, что объемы ресурсов велики и конкурирующие процессы представляют собой транзакции, которые не должны прерываться в течение сравнительно длительного времени. Примерами могут служить методы, основанные на последовательности начал транзакций, например метод, использующий **механизмы захвата, и метод меток времени.**
Функция разграничения прав доступа к данным использует, как правило, методы назначения паролей пользователям аналогично файловой защите. В этом случае защита распространяется как на отношения в целом, так и на отдельные его атрибуты. При этом доступ ограничивается лишь тем множеством кортежей, которые удовлетворяют некоторому заданному условию. В системах обработки статистических данных используются специальные методы защиты, поскольку, объединяя результаты статистической обработки таких множеств кортежей, можно до некоторой степени восстановить исходные данные.

3.2.5. Создание отношений

При проектировании схемы базы данных главной проблемой является **выбор отношений и их атрибутов**. В хорошо спроектированной схеме структура базы данных оказывается легко доступной для анализа и при модификации базы не возникает противоречий в данных. В данном разделе мы рассмотрим следующие вопросы проектирования баз данных:
1. Методы создания множества отношений, отображающих окружающий мир.
2. Методы оценки качества заданной схемы отношений и ее совершенствования в случае, если в ней легко возникают противоречия (методы нормализации).

В качестве метода проектирования множества отношений используют модель связей сущностей. Эта модель формируется из множества сущностей окружающего мира и связей, отражающих взаимоотношения элементов этого множества сущностей.
***Сущностями* называют объекты, отделимые от других объектов либо существующие независимо.**
Для примера рассмотрим задачу проектирования базы данных для магазина детской одежды. В качестве рассматриваемых сущностей возьмем какой-либо товар, например детское платье. Атрибутами товара будут наименование_товара, размер, цвет. Далее, естественно, также принять за сущность изготовителя (поставщика) платья. В качестве атрибутов изготовителя возьмем наименование_фирмы, адрес, телефон, ответственное_лицо. Между детским платьем и изготовителем существует связь "поставка_товара". Кроме этих можно рассмотреть еще много различных сущностей. Часть из них представлена на рис. 2.
На основе этой схемы формируется схема отношений. Для облегчения работы с базой данных возьмем в качестве ключей индивидуальные номера. При этом появляются "номер-товара"

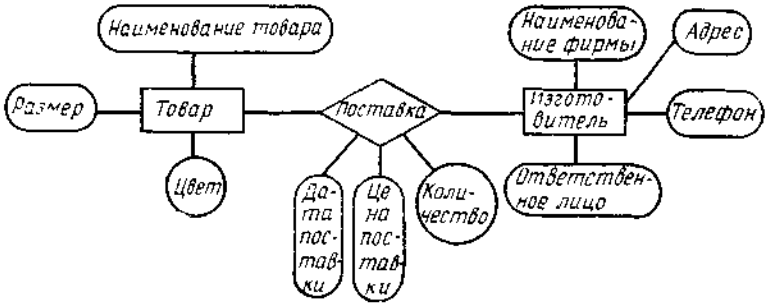


Рис. 2 Схема связей сущностей "детское платье" и "изготовитель — поставщик".
▭ — сущности, ◊ —связи; ( ) —атрибуты.

и "номер_поставщика". Задавая в качестве отношений множество сущностей и множество связей, получим следующую несложную реляционную схему:

ТОВАР. {Номер_товара, наименование_товара, размер, цвет}

Далее рассмотрим нормализацию, с помощью которой оценивается качество полученной схемы базы данных и производятся преобразования с целью ее совершенствования. При нормализации важную роль играет понятие функциональной зависимости. Пусть в реляционной схеме $R.\{A_1, A_2, ..., A_n\}$ имеются подмножества X и Y. Если

при этом в двух произвольных кортежах из R с одинаковыми значениями компонент X значения компонента Т также равны, то говорят, что "Y функционально зависит от X'', и обозначают это как $X \rightarrow Y$. При этом X называют определяющим подсписком.
Например, в отношении ТОВАР от атрибута "номер_товара" функционально зависят все атрибуты.
Функциональная зависимость есть свойство реляционной схемы, которое определяется во время ее формирования и выполняется для любого состояния отношений. Можно считать функциональную зависимость одним из ограничений на целостность базы данных; при сложной функциональной зависимости управление базой данных значительно усложняется. Поэтому **прибегают к нормализации базы данных, разбивая реляционную схему с целью получения одних лишь простых функциональных зависимостей.**
Предложено несколько нормальных форм, основанных на понятии функциональной зависимости. Самой важной из них является третья нормальная форма. Говорят, что **отношение находится в третьей нормальной форме, если все, говоря кратко, дескрипторы являются ключами.** Для пояснения процедуры нормализации вернемся к примеру с детской одеждой.
Ниже перечислены все функциональные зависимости рассмотренной реляционной схемы:
для отношения ТОВАР: номер_товара→наименование_товара, размер;
для отношения ИЗГОТОВИТЕЛЬ: номер_изготовителя→название_фирмы (двусторонняя функциональная зависимость, т. е. взаимно однозначное соответствие);
номер_изготовителя→адрес (будем считать, что адрес единственный) ;
название_фирмы→адрес;
для отношения ПОСТАВКА: номер_изготовителя, номер_товара→количество.
В приведенной реляционной схеме все дескрипторы являются ключами, и поэтому эти отношения находятся в третьей нормальной форме.
Рассмотрим отношение $R.\{$номер_изготовителя, наименование_фирмы, адрес, номер_товара, количество$\}$, полученное объединением отношений ИЗГОТОВИТЕЛЬ и ПОСТАВКА. Ключом отношения R является подмножество {номер_изготовителя, номер_товара}. Поскольку при этом указанные зависимости сохраняются, номер_изготовителя и наименование_фирмы остаются дескрипторами, однако их комбинация не является ключом отношения R. **Таким образом, R не находится в третьей нормальной форме.** При внесении изменений в отношения, не находящиеся в нормальной

форме, могут возникать нежелательные эффекты. Например, в отношении *R* один изготовитель входит в несколько кортежей. Следовательно, при изменении адреса изготовителя соответствующие изменения необходимо внести во все кортежи. В случаях когда происходит удаление ненужных кортежей, например по причине того, что годовой объем поставок оказался равным нулю, может оказаться, что одновременно количество всех товаров, выпускаемых какой-либо фирмой, также равно нулю, и тогда будут удалены все кортежи, но при этом потеряется информация о наименовании фирмы и ее адресе. Если же разделить отношение *R* на два отношения (ИЗГОТОВИТЕЛЬ и ПОСТАВКА), то этой проблемы не возникнет.

Если отношение не находится в третьей нормальной форме, то его можно нормализовать, расщепляя (разбивая) (как было показано в примере) на два отношения, у которых часть атрибутов является общей. Это разбиение можно провести путем выполнения операции проекции. Наоборот, из двух отношений, полученных путем разбиения, можно получить исходное отношение путем выполнения операции обыкновенного соединения. При этом информация не теряется.

С момента появления реляционных моделей было предложено несколько нормальных форм (среди них первая нормальная форма и вторая). Третья нормальная форма определяется на базе понятия функциональной зависимости, и поэтому она накладывает самые сильные ограничения. Предложены также сильные нормальные формы, основанные на понятиях многозначной и связной зависимостей.

При проектировании отношений не все они обязательно должны быть нормализованы. Поскольку переход расщепленных при нормализации отношений к исходным отношениям путем выполнения операций соединения обладает большой трудоемкостью, то отношения часто хранят в нерасщепленной форме. Если при модификации базы данных проблем не возникает» выигрыш, получаемый от нормализации, невелик.

3.2.6. Другие модели данных

Реляционные модели нашли широкое применение благодаря простоте анализа модели и работы с языком запросов.

Находит применение и ряд других моделей. Примерами могут служить иерархическая и сетевая модели.

В иерархической модели база данных представлена в виде дерева. Другими словами, некоторый тип записей считается корнем дерева, и с ним в подчиненной связи находятся другие типы записей. Поэтому

пример с магазином детской одежды, описанный в разд. 3.2.5, мог бы иметь вид, показанный на рис. 3.

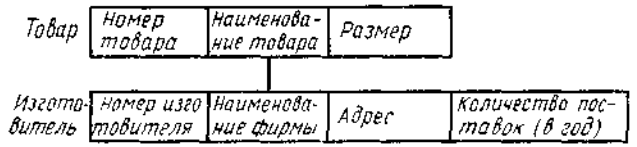


Рис.3. Пример иерархической базы данных для магазина детской одежды.

В этом примере запись "ТОВАР" является корнем, содержащим *n* записей, где *n* — количество изготовленных товаров. С каждой из записей типа "ТОВАР" связана запись об изготовителях, которые этот товар поставляют.

Таким образом, получается *n* экземпляров дерева. Если один и тот же товар поставляют несколько фирм-изготовителей, то с одной корневой записью связано несколько записей. Наоборот, если один поставщик поставляет несколько товаров, то существует несколько записей для этого поставщика, и они связаны с записью для соответствующего товара. В данном примере имеются два уровня, однако их может быть и больше. Иерархическая модель данных обладает следующими особенностями.

1. В дереве с отношениями "отец-сын" доступ к записи-сыну возможен только через запись-отца. В приведенном примере невозможен непосредственный доступ к записи "изготовитель". К ней можно обратиться лишь через родительскую запись "товар". Таким образом, отсутствует симметрия в доступе к записям разного типа.
2. Благодаря древовидной структуре удается хорошо выразить связь-"один ко многим", однако, как видно из предыдущего примера, связь типа "многие ко многим" оказывается неориентированной. Если, например, в приведенном примере в записях "изготовитель" вычеркнуть те из них, поставки которых равны нулю, то исчезнут все записи "изготовитель", не имеющие поставок ни по одному товару.
3. Поскольку метод моделирования не является объектно-ориентированным, то время реакции очень сильно зависит от типа запроса.

В сетевых моделях база данных создается путем связывания всех записей между собой. В отличие от иерархической базы данных в данном случае возможна не только древовидная структура. Поэтому можно реализовать связь "многие ко многим". Пример с детским магазином можно представить в виде, показанном на рис.4.

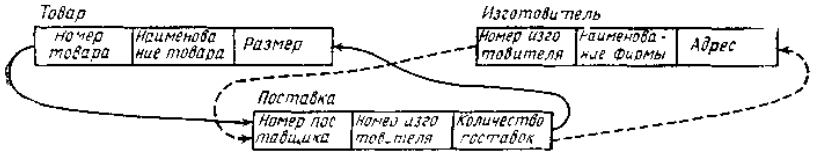


Рис. 4. Пример сетевой модели для магазина детской одежды
→ — цепь «товар — поставка»; — → — цепь «производитель — поставка»

В данном примере имеются две связи: одна, когда отцом является товар, а сыном — поставки товара, и другая, когда отцом является производитель, а сыном — поставки. Эти связи играют почти ту же самую роль, что и в двухуровневой иерархической модели. Если в данном примере мы захотим узнать имя фирмы-изготовителя, поставляющей товар № 1, то мы должны войти в запись "поставка", связанную с экземпляром записи "товар", имеющим номер, равный 1, по цепочке товар-поставка. Далее, из найденной записи "поставка" по цепочке "изготовитель-поставка" необходимо найти запись "изготовитель" и извлечь из нее необходимую информацию. Получив название одной фирмы, можно по цепочке "товар-поставка" перейти к следующей записи типа "поставка" и повторить аналогичные операции.
В сетевых моделях, как показано в примере с записями "товар" и "изготовитель", записи различных типов обрабатываются симметрично. При наличии связей "многие ко многим" трудностей с модификацией базы данных не возникает, как это имеет место в иерархических структурах. При этом логическая структура базы данных и реализующая ее физическая структура оказываются сложнее иерархической модели.
Как в иерархической, так и в сетевой моделях структура связей между записями выражается на логическом уровне, и пользователь при доступе к базе данных должен в явном виде указывать операции перехода по этим связям. Следовательно, схема данных, предоставляемая пользователю, в этом случае оказывается сложнее реляционной модели. Для повышения эффективности в системах с базами данных на основе реляционной модели часто используют внутренние структурные связи, однако пользователь, выполняющий запросы или модификацию базы данных, может об этой структуре не знать.
Помимо названной здесь модели данных предложено много других. Одна из них — модель связей сущностей, упомянутая в разд. 3.2.5,

посвященном проектированию отношений в реляционной схеме данных.

3.2.7. Реляционные СУБД-машины

Поскольку работа с базами данных занимает значительное место в общем объеме работ, проводимых средствами вычислительной техники, разработаны специализированные СУБД-машины, ведущие работу с базами данных. Они подключаются к обычным вычислительным машинам, в результате чего улучшаются характеристики всей системы в целом. Большинство из предложенных и реализованных СУБД-машин представляют собой реляционные СУБД-машины, в которых используются реляционные модели. Причины, по которым именно реляционные модели, а не какие-либо другие модели данных стали предметом особого внимания со стороны создателей СУБД-машин, состоят в следующем.
1) Поскольку язык запросов представляет собой язык высокого уровня, допускается большая свобода в выборе методов выполнения запросов. Отсюда и большая свобода в проектировании специализированного аппаратного обеспечения, что позволяет повысить эффективность системы.
2) На языке запросов легко выразить сложные запросы. Это приводит к увеличению количества сложных запросов, значительно повышающих трудоемкость обработки, и растет потребность в повышении эффективности.
Кроме того, СУБД-машина с любой моделью данных обеспечивает следующие преимущества:
а) распределение обработки по нескольким процессорам повышает эффективность обработки;
б) легко организуется совместное использование общей базы данных несколькими главными машинами;
в) поскольку база данных отделена от всех других данных, доступ к ней в обход методов доступа затруднен; кроме того, уменьшается влияние на базу данных ошибок и в программах, ие связанных с базой данных.
С другой стороны, при обработке информации сохраняется дополнительный обмен между СУБД-машиной- и центральной машиной, и поэтому время реакции увеличивается, однако интенсивность обмена с главной машиной невысока, и недостатки последней проявляются незначительно.
Реляционные СУБД-машины делятся на две большие группы.

1. СУБД-машины аппаратного типа, имеющие специализированное оборудование для обработки реляционной информации.
2. Специализированное программное обеспечение, при котором для работы с базой данных используется универсальная ЭВМ.

Направление развития исследований СУБД-машин тесно связано со снижением стоимости аппаратного обеспечения, что обусловлено развитием больших интегральных схем. Исследования по снижению стоимости аппаратного обеспечения ведутся по следующим направлениям:

а) распараллеливание обработки путем использования большого количества однородного аппаратного обеспечения;
б) повышение степени интеграции и быстродействия аппаратного обеспечения;
в) разработка архитектуры аппаратного обеспечения, допускающей его простую реализацию на больших и сверхбольших интегральных схемах.

В первых работах по реляционным СУБД-машинам, проведенных в середине 70-х гг., в основном рассматривалась архитектура с параллельной работой ассоциативного процессора, синхронизирующего выполнение реляционных операций (в особенности операций выборки), с памятью на дисках ассоциативного поиска, устройствах с зарядовой связью, на пузырьковых магнитных доменах и т. д. Архитектуры, в которых использовались устройства с зарядовой связью и магнитная пузырьковая память, не имели возможностей к повышению степени интеграции по сравнению с магнитными дисками, и по мере снижений стоимости обычной памяти, а также по ряду других причин они утратили свои позиции.

Для современных исследований по реляционным СУБД-машинам характерно:

1) использование дисков для размещения базы данных;
2) использование больших объемов памяти в качестве буферов и пулов памяти для ускорения доступа к данным;
3) надстраивание многопроцессорных систем над большими объемами памяти путем соединения их через общую шину илв через организацию их в сеть и распараллеливание таким образом их работы.

По параллельной работе процессоров исследования ведутся в двух направлениях, объединяющих в себе

а) использование нескольких процессоров по одному на каждую операцию, либо с универсальным набором функций;
б) создание специализированного аппаратного обеспечения, либо процессора общего назначения.

В программных СУБД-машинах разрабатываются архитектуры, позволяющие повысить эффективность обработки путем объединения нескольких программных надстроек над штатным математическим обеспечением.

3.3. Языки программирования логического и функционального типов

В основе языков программирования функционального типа лежит понятие функции в математическом смысле, а в основе логических языков программирования лежит понятие отношения, восходящее к предикатным логикам.

Если исходить из отношений, то функция

$$f: (x_1, x_2,...,x_n) \rightarrow y$$

представляет собой отношение

$$R\ (x_1, x_2,...,x_n, y),$$

которое для наборов значений $(x_1, x_2, ..., x_n)$ однозначно задает значения $y(= f(x_1, x_2, ..., x_n))$. Наоборот, если исходить из функций, то отношение

$$R\ (x_1, x_2,...,x_n)$$

можно считать функцией

$$R: (x_1, x_2,.., x_n) \mid\rightarrow \text{ИСТИНА или ЛОЖЬ},$$

которая для наборов значений $(x_1, x_2,..., x_n)$ задает логические значения "ИСТИНА" или "ЛОЖЬ". Таким образом, тот и другой подходы в некотором смысле взаимно дополняют друг друга и не противоречат один другому. В данном разделе рассматриваются языки обоих типов.

3.3.1. Языки программирования логического типа

Во всей предшествующей истории культуры наиболее точная формализация человеческой мысли, хотя и в простейших формах, осуществлялась в *математической форме*, в частности в *логической форме с использованием логической дедукции с четким выделением объектов мышления.* Одним из основанных на этом подходов является создание языков программирования по типу предикатных логик. Примером такого языка является Пролог (programming in logic), разработанный в 1972 г. Кольмерауером в Марсельском университете (Франция). В следующих разделах на примере языка Пролог будут рассмотрены процедуры унификации и параллелизм. Рассмотрим некоторые преимущества, обусловливающие

перспективу использования языков программирования логического типа.

Во-первых**, логика предикатов является в некотором роде формализацией человеческого мышления**, в частности **формализацией доказательств в математике**, поэтому полагают, что основанные на ней языки программирования логического типа будут более пригодны в будущем для обработки информации, **в особенности для работы со знаниями,** нежели существующие языки последовательного типа.

Во-вторых, существует проблема верификации программ: исследования того, удовлетворяют ли они предъявляемым требованиям, а также проблема того, как, исходя из предъявляемых требований, составить нужную программу. **Для решения этих проблем желательно осуществить взаимодействие языков, на которых записываются требования к программам, и языков программирования.** Кроме того, необходимо исходить из одних и тех же посылок при выполнении операций по верификации и составлению программ. **Понятие языков программирования логического типа позволяет использовать единую основу — исчисление предикатов для решения всех этих задач.**

В-третьих, если обратиться к проблеме отображения моделей предметных областей в базах данных, то можно с уверенностью говорить о том, что **характер базы данных определяется структурой хранимого в ней множества фактов, а также методами доступа к ним**. **В языках программирования логического типа факты представляются выражениями, в которых константы играют роль термов. Следовательно, множество этих выражений можно рассматривать как базу данных, и теорию реляционных баз данных, как отмечалось выше, можно считать одним из разделов логики предикатов.**

Видимо, должно **происходить постепенное объединение языков программирования логического типа и теории реляционных баз данных на общей основе логики предикатов**. По-видимому, в исследованиях распределенных баз данных как множества моделей предметных областей следует двигаться в этом же направлении.

Языки программирования логического типа связаны и с естественными языками. Одним из истоков символической логики была формализация естественного языка. Логика предикатов как форма выражения этого формального смысла является первым приближением. В качестве семантики естественного языка можно назвать теорию, выдвинутую Монтегю, и хотя эта теория дает формализм перевода предложений естественного языка в формулы

интенсиональной логики, в результате их можно свести к предикатному формализму. В области лингвистики ведутся исследования предикатных логик в условиях их семантической интерпретации. Можно считать, что эти работы вливаются в русло языков программирования логического типа. Успехи в этих исследованиях могли бы породить новые успехи в развитии языков описания спецификаций на основе естественного языка.

В области экспертных систем, использующих методы работы со знаниями, активно применяются языки программирования, называемые *производственными системами,* однако эти языки носят сугубо эмпирический характер. По-видимому, в будущем развитие этой семантической теории и других исследований будет все больше склоняться к логике предикатов.

Перед тем как начать разговор о том, что такое языки программирования логического типа, рассмотрим принцип резолюции и метод унификации.

3.3.2. Принципы резолюции и метод унификации

В 1965 г. Дж. А. Робинсон предложила так называемый **принцип резолюции, в котором используется регулярная процедура последовательного сопоставления с образцом**. Этот метод сопоставления с образцом получил название ***алгоритма унификации***.

При наличии нескольких переменных или логических формул, содержащих функции, эти переменные или функции выделяются, часть из них подвергается процедуре унификации и производится их подстановка в исходные формулы. Если при этом среди них оказываются тождественно истинные формулы вида: "А или НЕ-А" $(A \vee \neg A)$, то они удаляются. Метод резолюции состоит в том, чтобы, повторяя эту операцию, показать невыполнимость множества исходных логических формул. Поясним на примерах принцип резолюции и унификацию. Запишем выражение: *"x*— человек" в виде *man(x)***. Здесь *man* — предикатная буква, *x* — терм (в данном случае терм-переменная).** Кроме того, запишем выражение *"x* смертен" в виде *mortal(x).* Таким образом, выражение "Человек смертен" можно записать в виде

$$man\ (x) \rightarrow mortal\ (x)$$

Попробуем теперь, задав выражение "Сократ — человек", т. е.

$$man\ (\text{СОКРАТ}),$$

где СОКРАТ — константный терм, доказать методом резолюции выражение *mortal* (СОКРАТ).

Прежде всего добавим к исходному множеству формул отрицание того утверждения, которое мы хотим доказать ⌉*mortal* (СОКРАТ), как показано на рис. 2.4, cl—c3.

(c1) ⌉*man(x)* ∨ *mortal(x)*
(c2) *man (Сократ)*
(c3) ⌉*mortal (Сократ)*

Рис.5.

Формула ⌉*man(x)* ∨ *mortal(x)* логически эквивалентна формуле *man(x)→mortal(x).* Поэтому необходимо осуществить унифицирующую подстановку *x* -СОКРАТ относительно формулы cl или c2. В результате удаляется тождественно истинная формула

⌉ *man* (СОКРАТ) ∨ *man(СОКР*АТ) и остается лишь выражение *mortal* (СОКРАТ) (рис.6).

(c1∨c2) ⌉*man(x)*∨*mortal(x)*∨*man (Сократ)*
↓ *Унификация x и Сократ*
(c1∨c2) ⌉*man (Сократ)*∨*mortal (Сократ)*∨*man(Сократ)*
↓
(c1∨c2) *mortal (Сократ)*

Рис. 6

После этого, присоединив к нему через связку "ИЛИ" формулу c3, снова получим тождественно истинное выражение

mortal (СОКРАТ) ∨ ⌉*mortal* (СОКРАТ), которое также удаляется, давая при этом пустое множество формул (рис.7).

(c1∨c2∨c3) *mortal (Сократ)* ∨⌉*mortal (Сократ)*
↓ *Удаление истинной и ложной формул*
(c1∨c2∨c3) *пустой дизъюнкт*

Рис. 7.

Это показывает невыполнимость формул cl—c3. Отсюда методом обоснования вывода выводится формула *mortal* (СОКРАТ).

3.3.3. Выражения Хорна

Логика создана для того, чтобы четко выражать человеческую мысль, однако если использовать в языках программирования исчисление предикатов (1 порядка) в непосредственном виде, то это приведет к чрезвычайно большим объемам вычислений, что неприменимо на практике. **Поэтому разработан язык программирования *Пролог,* в котором ограничиваются специальными выражениями, называемыми *выражениями Хорна,* и только к ним применяют описанный выше метод резолюции.** В данной работе на примере языка Пролог мы будем рассматривать суть языков логического программирования, а также параллелизм, существующий в языке Пролог (И- и ИЛИ-параллелизм).

Выражение Хорна имеет вид

$$\neg P_1 \vee \neg P_2 \vee \dots \neg P_n \vee P_m.$$

Как видно из этого выражения, в нем имеется единственная логическая формула без отрицания, и все формулы связаны знаком логической суммы (ИЛИ). Это можно описать в другом виде

$$P_1 \wedge P_2 \wedge \dots \wedge P_n \longrightarrow P_m.$$

Это выражение читается так: "Из P_1 и P_2 и так далее до P_n следует P_m". Здесь слово "и" соответствует английскому слову "AND". На языке Пролог в синтаксисе системы DEC-10 Пролог это записывается в другом виде

P_m : —P_1, … P_n.

Пример с Сократом из предыдущего раздела (формулы cl—c3) выразится в следующем виде:

mortal(x): —*man(x)*
man (СОКРАТ)
?- *mortal* (СОКРАТ)

Выражение *man* (СОКРАТ) говорит о том, что предикат *man* (СОКРАТ), безусловно, выполним (является фактом). Выражение ?- mortal (СОКРАТ) является сокращением выражения
□:- mortal (СОКРАТ)
Это означает, что если выполняется mortal (СОКРАТ), то не выполняется ничего, следовательно, не выполняется и само выражение *mortal* (СОКРАТ). Это говорит о том, что в языке Пролог, так же как и в методе резолюции, взяв за отправную точку отрицание формулы *mortal* (СОКРАТ), можно методом обоснования вывода доказать утверждение *mortal* (СОКРАТ).

Прежде всего в левой части ищется предикат, совпадающий с (целевым) утверждением *mortal* (СОКРАТ), и производится

унификация переменной *x* и константного терма СОКРАТ. Затем ищется выражение, содержащее в *левой* части предикат, совпадающий с новой целью (в данном случае *man* (СОКРАТ)), образующейся в правой части текущего выражения в результате его унификации, и затем аналогично производятся дальнейшие унификации. **Программа на языке Пролог представляет собой некоторое множество выражений Хорна, и выполняется она путем проведения рассуждений от целевого утверждения в обратном направлении.** Если логических формул, подлежащих унификации, много, то производится унификация для одной из них. Если эта унификация удовлетворительна, то программа выполняется дальше. Если унификация оказывается неудовлетворительной, то происходит **возврат (бэктрекинг)** и выполняются другие возможные унификации. **Таким образом, в языке Пролог рассуждение ведется в обратном направлении методом проб и ошибок.**

3.3.4. Параллелизм

Рассмотрим прологовский параллелизм. Обратимся вначале к ИЛИ-параллелизму. Как следует из предыдущего примера, при существовании нескольких логических формул, в которых общим переменным можно придать значение, это делается лишь для одной из них.

Однако если используется не один процессор, а несколько, то замену общих переменных некоторым значением можно выполнить над несколькими такими формулами одновременно, т. е. **над формулами с одинаковыми предикатными буквами и одинаковыми термами**. В языке Пролог все логические формулы (члены формы Хорна) находятся в отношении ИЛИ, тогда параллельное выполнение операций по приписыванию значений общим переменным над несколькими формулами, находящимися в отношении ИЛИ, является *ИЛИ-параллелизмом*.

С другой стороны, прологовское правило-оператор имеет следующий вид:

$$P_m : — P_1, \ \ldots \ P_n.$$

и его правая часть читается как P_1 и P_2 и ... и P_n. Таким образом, подцели в правой части связаны отношением И. **При этом можно выполнять P_1, P_2, ..., P_n параллельно, а не в порядке следования слева направо.** В этом случае, поскольку P_1, P_2, ..., P_n находятся в отношении И, необходимо проверять непротиворечивость P_1, P_2, ..., P_n (например, непротиворечивость переменных с одинаковыми именами). Таким образом, параллельную обработку подцелей в правой части

прологовского правила-оператора можно назвать *И-параллелизмом.* Можно одновременно реализовать И- и ИЛИ-параллелизм.

3.3.5. Языки программирования функционального типа

Под влиянием теоремы Гёделя о неполноте, опубликованной в 1931 г., было осуществлено уточнение важного понятия, связанного с алгоритмами. **Это — определение класса целочисленных функций, носящих название *рекурсивных функций.***

(Проблемой, впервые потребовавшей уточнить понятие алгоритма, **была** десятая проблема Гильберта (1900 г.). Отправляясь от идей Гильберта, Гё-дель впервые описал класс всех рекурсивных функций как класс числовых функций, определенных в некоторой формальной системе)

А. Чёрч, создавший с помощью λ-формализма и β-преобразования новый универсум, названный им *λ-исчислением,* установил, что так называемые λ-определимые функции эквивалентны рекурсивным функциям. Независимо от этого А. Тьюринг установил, что так называемые "функции, вычислимые на машине Тьюринга", также эквивалентны рекурсивным функциям. Кроме них многие другие исследователи разными способами давали определения вычислимых функций, однако все они, как оказалось, также эквивалентны рекурсивным функциям.

В этих условиях к 1936 г. Чёрч выдвинул тезис о том, что все функции, которые мы можем интуитивно рассматривать как вычислимые, являются рекурсивными. **Для любой рекурсивной функции существует алгоритм ее вычисления, с другой стороны, числовая функция, имеющая общий алгоритм ее вычисления, является рекурсивной функцией.**

Таким образом, языки функционального программирования подводят практическую основу для идеи λ-исчисления, упоминавшегося выше. Одним из них является язык Лисп (на самом деле, язык Лисп дополнен возможностями, ухудшающими его качества как языка функционального программирования).

Существует несколько языков функционального программирования, отличающихся по конструкции, идеологии и т. д. Из них трудно выделить какой-нибудь один (кроме Лисп Мак-Карти, примерами могут служить Лисп KRC Тэрнера, FP Бэкуса и др.).

3.3.6. Логические и функциональные машины

Можно рассмотреть архитектуру машин нового типа, в основу которых положены языки программирования логического и функционального типов. Для обычных машин это — языки высокого и даже сверхвысокого уровней, однако они же являются машинными языками в рассматриваемых здесь машинах.
На современном уровне развития техники и эти машицы, видимо, будут последовательными, однако при этом по крайней мере будет доказана их осуществимость
При рассмотрении архитектур машин параллельного типа, превосходящих по своим возможностям последовательные машины и являющихся машинами будущего, этими языками можно руководствоваться при исследовании архитектур. В следующем разделе мы коснемся этого вопроса.

3.4. Параллельные структуры управления

3.4.1. Поток управления и поток данных

1. **Поток управления**. Нам уже известно понятие потока управления. Примером такого понятия может служить традиционный (фон-Неймановский) поток управления. В последовательном потоке управления передача управления неявно и притом единственным образом определяется счетчиком команд по принципу: "От команды, исполняемой в настоящий момент, к очередной исполняемой команде". Разумеется, есть возможность задать передачу управления в явном виде (с помощью команд типа GOTO).
С другой стороны, вводя операторы параллельной передачи управления, можно организовать параллельный поток управления, т. е. организовать одновременную активизацию нескольких потоков. Среди операторов параллельного потока управления есть, например, такие, как FORK-оператор (оператор разветвления), JOIN-оператор (команда синхронизации разветвленных потоков управления), а также ряд других операторов.
2. **Поток данных**. Как указывалось выше, в параллельных потоках управления возможна организация параллельного выполнения команд, однако для этого в программе должны быть явно указаны операторы параллельного выполнения команд.

Потоки данных организуются таким образом, чтобы при накоплении всех данных, необходимых для выполнения какого-либо вычисления, оно выполнялось бы немедленно (его называют *спонтанным вычислением).* Результат данной операции передается следующим операциям. Другими словами, потоки данных организуются таким образом, что вычислительные операции запускаются под действием пересылаемых данных. По этой причине вычисления, организуемые через потоки данных, называют также *вычислениями, управляемыми данными.* Главной особенностью вычислений в потоках данных является то, что параллельное выполнение вычислений не задается явно, с помощью операторов параллельного управления, как это делается при организации вычислений через потоки управления, а процессы, допускающие параллельное выполнение, выполняются параллельно при наличии соответствующих данных.

3.4.2. Языки и графы потоков данных

Многие идеи организации потоков данных — согласно которым по накоплении данных, необходимых для выполнения вычислительной операции, происходит спонтанное ее выполнение и результат пересылается очередной вычислительной операции — становятся реальностью с разработкой **новых языков (потоков данных и машин потоков данных).**
Программа, написанная на языке потоков данных, после компиляции преобразуется в граф потоков данных. Устройство, непосредственно выполняющее этот граф, называется *машиной потоков данных.* В данном разделе речь пойдет о языках и о графах потоков данных. После этого мы рассмотрим машины потоков данных.
1. **Языки потоков данных**. Главной особенностью языков потоков данных является **правило единственной подстановки**. Попросту говоря, это правило означает, что засылка значения в переменную разрешается единственный раз. Поясним это на следующем примере.

X:=5
.
. Пример программы 1.
.
X:=X+1

В традиционных языках программирования подстановки, показанные в примере программы 1, допустимы и часто используются. Смысл этих подстановок в том, что производится запись в ячейку памяти, выделенную под переменную *X,* и считывание из нее. Другими

словами, при выполнении оператора *X: = 5* в ячейку *X* записывается значение 5. При выполнении действия *X+1* из ячейки памяти, отведенной под переменную *X,* считывается значение 5. К нему прибавляется единица и результат, равный 6, вновь записывается в ячейку памяти, отведенную под переменную X.

При организации же вычислений по методу потоков данных понятие ячейки отсутствует, и поэтому не допускается более одной подстановки значения переменной, как это имеет место в примере программы 1. При этом следует поступать, например, как показано в примере программы 2.

```
X: = 5
.
.                    Пример программы 2.
.
NewX:=X+l
```

Другими словами, переменные в языках потоков данных не являются указателями ячеек памяти, как это принято в традиционных языках программирования, а указывают назначение результата выполненной операции. При организации вычислительного процесса через потоки данных отсутствие понятия ячейки памяти приводит к серьезной проблеме сохранения ретроактивного следа вычислений.

2. Графы потоков данных. Программа, записанная на языке потоков данных, преобразуется в граф потоков данных.

Например, рассмотрим формулу *A:=* (B+1)* *(B*—C). Граф этой формулы представлен на рис.8.

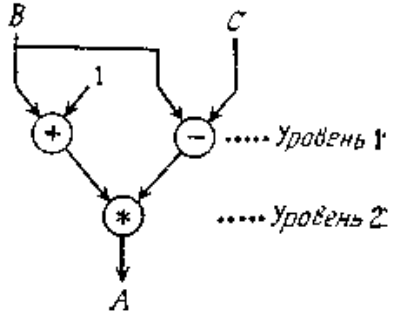


Рис. 8. Граф потока данных

Элементы графа, обозначенные кружочками, называют *узлами,* а отрезки прямых, соединяющие узлы, — *дугами.* Как видно из рисунка, **узлы соответствуют операциям, а дуги — потокам данных**. **Число дуг, входящих в один узел, представляет собой число операндов**

операции. В данном примере все операции имеют по два операнда, например, при совместном попадании значения *B* и значения C в узел происходит спонтанное выполнение вычитания и результат выдается по выходной дуге. Кроме того, модули, находящиеся на том же уровне, могут выполняться параллельно.

При этом следует обратить внимание на то, что в формуле *A:=(B+1)*(B—C)* не указывалось явно, какие ее части могут выполняться параллельно. Из графа, полученного в результате компиляции этой формулы, становится ясно, что при наличии данных, необходимых для вычислений и соответствующего числа процессоров (в данном случае двух), узлы ⊕ и ⊖ могут выполняться параллельно. Другими словами, программист может представлять себе, какие участки в его программе будут выполняться параллельно; явно он их не указывает, однако в результате преобразования программы в граф потока данных возможность параллельного выполнения становится ясной на основании взаимосвязей между данными. Можно также сказать, что ***при организации вычислений по методу потоков данных параллелизм, содержащийся в программе, извлекается из нее естественным образом***.

Кроме четырех действий арифметики в графе потоков данных могут быть и следующие узлы:

— операция сравнения (рис.9).

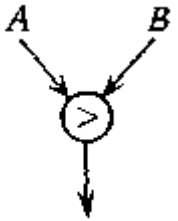


Рис.9. Узел операции сравнения

В данном примере показан граф потока данных для формулы *A*>B. Операция сравнения *A* и B на больше-меньше вырабатывает результат ИСТИНА или ЛОЖЬ.

— переключатель типа ИСТИНА-ЛОЖЬ (рис.10)

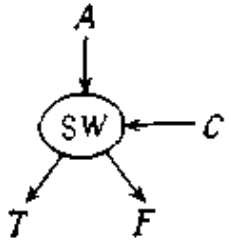


Рис. 10. Узел переключателя "истина— ложь".

В данном примере *A* — значение данных, *C* — управляющая информация (значением является ИСТИНА или ЛОЖЬ). При значении ЛОЖЬ значение данных выдается по дуге *F*.

3.4.3. Архитектура потоков данных

В процессе исследования языков функционального программирования, теоретической базой которых является λ-исчисление, и исследования архитектуры потоков данных выяснилось, что эти два направления связаны между собой. Например, связаны независимости вычисления функций и выполнения операций, запускаемых при накапливании необходимых данных в потоках данных.
Архитектура организации вычислений в потоках данных предлагается как одна из эффективных кандидатур для реализации Пролог-ориентированных архитектур, поскольку при этом параллелизм, содержащийся в алгоритме, извлекается из него естественным образом независимо от того, виден он программисту или нет.
В данном разделе мы рассмотрим наиболее общие структуры при организации вычислений в потоках данных.
Архитектура подобного типа содержит два основных элемента: блок обработки и структурную память (рис.11).

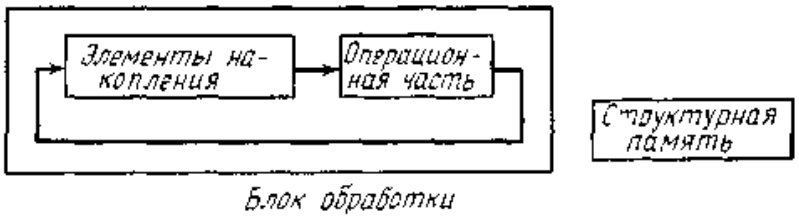


Рис.11. Элементы архитектуры вычислительного процесса в потоках данных.

Искусство создания такой архитектуры проявляется в том, в каком количестве выбрать элементы этих двух типов и как связать их между собой.
Блок обработки содержит загружаемую память и операционную часть. Загружаемая память управляет процессом накопления данных. По накоплении комплекта данных, удовлетворяющего условиям спонтанного вычисления, эти данные и команда пересылаются в операционную часть (бывает и так, что при поступлении хотя бы одного элемента данных осуществляется спонтанное вычисление с опережением).
В операционной части производится выполнение команды над данными и результат вновь пересылается в загружаемую память. Команда в случае работы со структурированными данными вместе с данными (на самом деле вместе с указателем на структурную память) пересылается в структурную память.
При организации потоков данных в чистом виде сами данные передаются по дугам, однако структурированные данные имеют тенденцию к большим объемам и передавать их непосредственно по дугам неэффективно, для этого и вводится структурная память, играющая роль общего поля памяти для структурированных данных. **Структурная память служит для размещения структурированных данных, их хранения и манипулирования ими.** Поэтому по дугам передаются либо непосредственно значения, либо указатели на область структурной памяти.

3.5. Объектно-ориентированное программирование

О структурах данных уже говорилось в разделе, посвященном абстрактным типам данных. **В анализе структур данных и управления ими центр внимания можно перевести со структур управления на структуры данных.**
Если при таком переносе отойти от проблем описания данных и перейти к действиям по созданию программ, т. е. применить этот подход к программированию, то мы получим объектно-ориентированное программирование, о котором и пойдет речь в данном разделе.

3.5.1. Модели объектно-ориентированного программирования

Основными особенностями объектно-ориентированного программирования являются:
1) описание внутреннего содержания объекта;
2) описание действий объекта;
3) формирование объектов путем наследования.

Здесь выражение "действия объектов" можно понимать аналогично выражению "операции над объектами". В абстрактных типах данных сам характер этих данных выражался через множество таких операций.

Слово "операции" неявно предполагает наличие операций, которые можно выполнять. Если исключить эту предпосылку и от "объекта, над которым производятся действия", перейти к рассмотрению объектов, "меняющихся под действием внутренних причин", то аналогично можно сформировать внешний взгляд на всю систему в целом.

Так, от описания факта: "Я уронил чашку со стола и разбил ее" можно перейти к описанию факта: "Я передвинул свою руку и чашка поехала по столу, упала на пол и сама собой разбилась".

Точно так же вычислительные модели сильно меняются при перемене взгляда на объекты, согласно которому они движутся под влиянием внешних воздействий, на ту точку зрения, что объекты движутся самостоятельно.

Одной из методик моделирования, построенной на этом подходе, является теория акторов. С другой стороны, существуют два способа описания внутреннего содержания объектов: сведение всех описаний содержания к описанию "действий" и описание "содержания" независимо от описания действий.

Описание содержания объектов без учета действий весьма напоминает описания типа "записей", используемые в языках Паскаль, Ада и т. д.

Что касается механизмов формирования объектов, то здесь мы определенно выскажемся лишь о механизмах наследования.

Наследование как метод формирования объектов можно проиллюстрировать таким упрощенным примером: "Из объектов *A* и *B* можно получить объект С". В этом выражении мы не будем касаться вопроса о том, что при этом получается и как из исходных объектов получается новый объект.

При практическом использовании метода наследования выразительная сила его проявляется в опускании мелких подробностей при передаче информации через связи наследования. В конкретных системах ИИ

семантика и работа процедур наследования должны быть точно определены.

В описательной классификации методы наследования делятся на единичное и множественное наследования. В общем случае множественное наследование включает в себя единичное.

На практике множественное наследование можно реализовать через единичное.

По семантической классификации существуют **наследования свойств и элементов.** Используются также **термины *таксономия (taxonomy)* и *партномия (partnomy).* При наследовании свойств наследуются существенные свойства объекта, а в иерархии наследований они уточняются.** С другой стороны, наследование элементов называют *наследованием частей.* **При этом наследуются сами объекты**. При многократном наследовании наследуемые объекты усложняются.

Например, в классификации живых существ, показанной на рис.12, используется наследование свойств, а в структурном описании велосипеда, изображенном на рис.13, — наследование элементов.

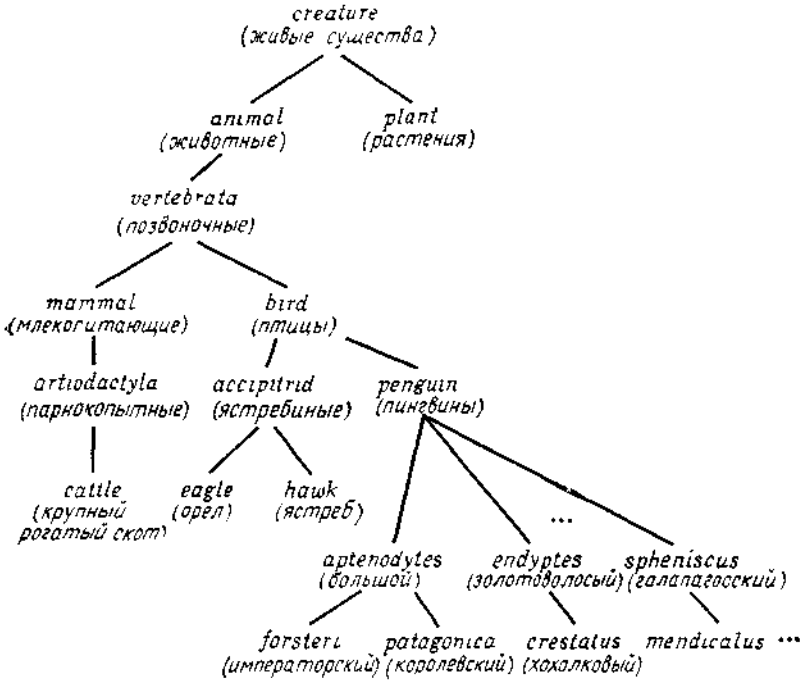


Рис. 12. Наследование свойств в классификации живых существ

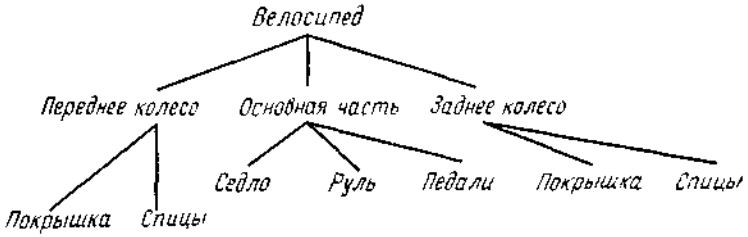


Рис.13. Наследование элементов в структурном описании велосипеда.

При этом наследование свойств является наследованием единичного типа, а наследование элементов представляет собой множественное наследование.
Существуют различные мнения о том, является ли метод формирования объектов главным вопросом в объектно-ориентированном программировании, ставящим объект во главу угла. При рассмотрении объектно-ориентированных языков с позиций построения вычислительных моделей можно также считать, что существенным моментом является посылка сообщений как механизм организации вычислительного процесса.
Однако в моделировании при решении задач создания реального математического обеспечения большую известность, по сравнению с методом посылки сообщений получил метод генерации объектов, основанный на наследовании как разновидности одного из методов модуляризации. В данном разделе в качестве основного содержания объектно-ориентированного программирования мы назовем наследование.

3.5.2. История объектно-ориентированного программирования

Считается, что первоначальные формы объектно-ориентированного программирования были воплощены в языке Симула, представляющем собой язык моделирования, разработанный в Норвегии Далом и др., и имеющем синтаксис, основанный на синтаксисе языка АЛГОЛ-60.
Разработка языка Симула велась с первой половины 60-х гг. Довольно широко используется название Симула-67, в котором фигурирует год выхода рукописи с описанием языка.
Поскольку **Симула является языком моделирования, он требует описания объектов и их действий**. Здесь уже имеются описания,

называемые "класс" (class), и допускаются конструкции, в которых используется наследование объектов.
Наследование объектов, вообще говоря, важная идея, однако помимо описаний, относящихся к объекту, **в языке Симула определена операция, называемая new, которая реализует создание конкретного экземпляра объекта (instantiation).**
Однако создание языка Симула само по себе не преследовало цели создания объектно-ориентированного языка программирования общего назначения (хотя среди пользователей были люди, готовые к его широкому применению). Объектно-ориентированное программирование прочно утвердилось в связи с реализацией функций машинной графики и нашло свое воплощение в языке Смолток, разработанном А. Кеем. Разработка языка Смолток началась с 70-х гг. По литературе можно проследить ряд версий: Смолток-72, Смолток-74, Смолток-76, Смолток-80, Смолток-82.
В языке Смолток впервые была выдвинута идея объектно-ориентированного программирования. В нем реализованы механизмы вычислений посредством описания содержания объектов, описания действий, наследования и передачи сообщений.
Большой интерес представляет реализация языка Смолток как конкретного примера системы объектно-ориентированного программирования в персональных компьютерах.
Под влиянием языка Смолток на фирме «Ксерокс» в язык описания систем Меза были введены функции объектно-ориентированного программирования, в результате чего был разработан язык Трейтс. Этот язык использовался при описании системы Star Workstation, разработанной фирмой «Ксерокс». С другой стороны, на основе Лисп-машины, разработанной в Массачусетском технологическом институте, была построена система Flavors, снабженная функциями объектно-ориентированного программирования. Вначале она была разработана для описания систем машинной графики, но затем применялась при разработке различных других систем.
Существует также язык Objective-C, полученный введением объектно-ориентированных функций в язык описания систем C.
В качестве языков логического программирования, наделенных чертами объектно-ориентированных языков, можно указать язык ESP, разработанный как одна из компонент в проекте ЭВМ пятого поколения, а также **язык Eqlog**, привлекающий к себе внимание как **язык описания ситуаций**.
Сочетание логического и объектно-ориентированного программирования важно также и с точки зрения соединения двух способов программирования. **Представляет интерес трактовка**

такого соединения как связи между абстрагированием структур управления в логическом программировании и абстрагированием структур данных в объектно-ориентированном программировании.

В языке Eqlog учтен опыт создания языков описания ситуаций OBJ, OBJ2 и в нем имеется возможность задавать свойства, которым должны удовлетворять объекты языка, в виде конструкции, называемой theory.

3.5.3. Примеры объектно-ориентированного программирования: языки ESP и Eqlog

В данном разделе мы проиллюстрируем основные принципы программирования на примере языков ESP и Eqlog, основанных на логическом программировании. Синтаксис логических языков программирования и стиль логического программирования описаны в разд. 3.3. Вначале покажем, как можно организовать классификацию живых существ, приведенную на рис.12. Здесь мы будем использовать язык ESP. В языке ESP для описания каждого объекта используется следующий формат:

class	⟨Имя класса⟩
nature	⟨Имя класса наследования⟩
instance	⟨Описание класса⟩
attribute	⟨Список имен свойств⟩
component	⟨Список имен элементов⟩
end	⟨Список описаний действий⟩

Поскольку в языке ESP допускается множественное наследование, после ключевого слова nature через запятую перечислены классы наследования. Описание класса представляет собой описание класса в целом и используется специальным образом.

После ключевого слова instance идут описания отдельных объектов. В языке ESP наследование по свойствам и наследование по элементам выделяются ключевыми словами: attribute и component, и обрабатываются оба наследования. <Список описаний действий> представляет собой программу, справедливую для данного объекта в конкретных обстоятельствах. Параметром к ней является объект, входящий в данный класс. Существует соглашение о предикатах, по которому перед предикатной буквой ставится двоеточие (:).

Итак, чтобы реализовать классификацию живых существ, представленную на рис. 12, пишется программа со связями наследования, показанная на рис.14.


```
CLASS CREATURE HAS
END.
CLASS ANIMAL HAS
        NATURE CREATURE;
END.
CLASS PLANT HAS
        NATURE CREATURE;
END.
CLASS VERTEBRATA HAS
        NATURE ANIMAL;
END.
CLASS BIRD HAS
        NATURE VERTEBRATA,
END.
CLASS MAMMAL HAS
        NATURE VERTEBRATA;
END.
CLASS PENGUIN HAS
        NATURE BIRD;
END.
CLASS KING_PENGUIN HAS
        NATURE PENGUIN;
END.
CLASS EMPEROR_PENGUIN HAS
        NATURE PENGUIN;
END.
CLASS ADELIE_PENGUIN HAS
        NATURE PENGUIN;
END.
CLASS ACCIPITRID HAS
        NATURE BIRD;
END.
CLASS EAGLE HAS
        NATURE ACCIPITRID;
END.
CLASS HAWK HAS
        NATURE ACCIPITRID;
END.
CLASS FALCON HAS
        NATURE ACCIPITRID;
END.
CLASS ARTIODACTYLA HAS
        NATURE MAMMAL;
END.
CLASS CATTLE HAS
        NATURE ARTIODACTYLA;
END.
CLASS DEER HAS
        NATURE ARTIODACTYLA;
END.
```

Рис. 14. Выражение классификации живых существ с использованием языка ESP.

Однако такая программа малоинтересна. Пополнив каждый вид свойствами (с использованием латинских терминов), получим программу, изображенную на рис.15.

```
CLASS CREATURE HAS
        INSTANCE
            ATTRIBUTE
                POSITION,
                BIRTH,
                LIFE_TIME,
                DEATH,
                PARENT,
                CHILDREN,
            MORTAL (CREATURE):-TRUE.
END.
CLASS ANIMAL HAS
        NATURE CREATURE,LOCOMOTIVE;
END.
CLASS PLANT HAS
        NATURE CREATURE;
        INSTANCE
            :PHOTOSYNTHETIC_METABOLIC(PLANT):
END.
CLASS VERTEBRATA HAS
        NATURE ANIMAL;
        INSTANCE
            :HAS_SPINE(VERTEBRATA);
END.
CLASS BIRD HAS
        NATURE VERTEBRATA,WITH_WINGS,OVIPAROUS
END.
CLASS MAMMAL HAS
        NATURE VERTEBRATA,VIVIPAROUS;
END.
```

Рис. 15(1). Программа на языке ESP с подробной классификацией живых существ.


```
CLASS PENGUIN HAS
        NATURE BIRD,SWIMMING;
        INSTANCE
            :LIVE(PENGUIN,PLACE):-MARINE
                        (PLACE),:HEMISPHERE(PLACE,SOUTH)
END.
CLASS KING_PENGUIN HAS
        NATURE PENGUIN;
END.
CLASS EMPEROR_PENGUIN HAS
        NATURE PENGUIN;
        INSTANCE
            :LIVE(X,ANTARCTIC);
            :LARGE_SIZE(X)
END.
CLASS ADELIE PENGUIN HAS
        NATURE PENGUIN;
        INSTANCE
            :LIVE(X,ANTARCTIC);
            :MADIUM_SIZE(X)
END.
CLASS ACCIPITRID HAS
        NATURE BIRD,FLYING;
END.
CLASS EAGLE HAS
        NATURE ACCIPITRID;
        INSTANCE
            :LARGE(EAGLE);
            :LONG_BROAD(FAGLE\WINGS_SHAPE);
            :STRONG_SOARING_FLIGHT(EAGLE);
END.
CLASS HAWK HAS
        NATURE ACCIPITRID;
        INSTANCE
```

Рис. 15(2).

Пользуясь это программой, можно делать следующее. (Имена с префиксом # в языке ESP являются именами классов. :new — предикат создания одного объекта.)

```
?-:NEW(#KING_PENGUIN,KING_PENGUIN),
  :CAN_MOVE(KING_PENGUIN).
```

```
            :HAS_STRONG_CLAWS(HAWK);
END.
CLASS FALCON HAS
        NATURE ACCIPITRID;
        INSTANCE
            :LONG(FALCON!WING_SHAPE);
            :FLY(FALCON,SWIFT);
END.
CLASS ARTIODACTYLA HAS
        NATURE MAMMAL;
        INSTANCE
            ATTRIBUTE
                NUMBER_OF_TOES;
            :HAS_HOOF(X)
            :TEST_TOES(X):-EVEN(X!NUMBER_OF_TOES);
END.
CLASS CATTLE HAS
        NATURE ARTIODACTYLA;
            :DOMESTICATED(CATTLE);
END.
CLASS DEER HAS
        NATURE ARTIODACTYLA;
        INSTANCE
            :HAS_ANTLER(DEER):-MALE(DEER);
END.
CLASS LOCOMOTIVE HAS
        INSTANCE
            ATRIBUTE
                POSITION,
                VELOCITY;
            :CAN_MOVE(OBIECT).
END.
```

Рис. 15(3).


```
CLASS SWIMMING HAS
        NATURE LOCOMOTIVE;
        INSTANCE
            :CAN_SWIM(OBIECT).
END.
CLASS FLYING HAS
        NATURE LOCOMOTIVE;
        INSTANCE
            :CAN_FLY(OBIECT).
END.
CLASS VIVIPAROUS HAS
        INSTANCE
            ATTRIBUTE
                BABY_HOLDER;
END.
CLASS WITH_WINGS HAS
        INSTANCE
            ATTRIBUTE
                WINGS_SHAPE;
            COMPONENT
                LEFT_WING,
                RIGHT_WING,
                TAIL_WING;
END.
CLASS OVIPAROUS HAS
        INSTANCE
            ATTRIBUTE
                EGGS;
END.
```

Рис. 15(4).

В этом примере создается экземпляр класса "Королевский' пингвин" *(king-penguin)* и задается вопрос о том, может ли он, этот пингвин, двигаться. Ответ на этот вопрос, естественно, положительный, однако при этом интересно, что описание: *cati-move* отсутствует в классе *king-penguin,* а выдается в результате наследования по цепочке:
penguin→bird→vertebrata→animal→locomotive. Ответ на вопрос: *can — swim (King-Penguin)* будет также утвердительным, однако ответ на вопрос: *can_swim (Falcon)* будет отрицательным, поскольку наследование при этом пойдет по другому пути.
Таким образом, **с помощью программ можно накапливать знания определенного рода.**

Для иллюстрации возможностей языка Eqlog рассмотрим **математический пример**. В этом языке аналогом объекта языка ESP, обозначаемого словом class, являются объекты, обозначаемые словами sort, module и theory. Объекты sort соответствуют типам данных. В языке Eqlog слова type избегают. Для описания содержания и действий объектов sort используют объекты theory и module. Наиболее близким аналогом конструкции module является конструкция class из языка ESP. Конструкция theory описывает свойства, выполняющиеся для конструкции sort. Есть возможность описать эти свойства в общем виде; как и в случае ESP, нет необходимости в том, чтобы эта конструкция выполнялась сама по себе. Она служит для того, чтобы, задавая совокупность логических условий, гарантировать правильность программы.
Особенностью языка Eqlog является возможность передачи конструкций theory как параметров конструкций module. Благодаря этому можно описать выполнимость приводимой ниже конструкции module относительно конструкции sort, удовлетворяющей некоторой совокупности логических условий.
Для начала выберем **математическую систему, называемую *кольцом*** (**более точно, коммутативным или эбелевым «кольцом»**), и зададим его свойства в виде конструкции fheory (рис.16).

```
THEORY RING IS
        SORTS RING
        FNS
            +:RING,RING->RING(ASSOC COMM ID:0)
            -:RING->RING
            *:RING,RING->RING(ASSOC COMM ID:1)
        VARS
            X,Y,Z RING
        AXIOMS
            X+(-X)=0.
            X*(Y+Z)=(X*Y)+(X*Z).
ENDTH RING
```

Рис.16. Конструкция Theory для кольца.

Здесь естественным образом различаются конструкция с именем *RING,* существующая как теория (theory), и конструкция с именем *ring,* существующая как объект. Поскольку в языке Eqlog допускается функциональная нотация, с помощью ключевого слова fns можно задавать операции. Эквивалентности, описанные в разделе теории, поименованном axioms, а также ключевые слова assoc, comm, id в

разделе с именем fns указывают свойства объекта *ring,* описанного в объекте *RING.*
Здесь assoc обозначает ассоциативность (associativity), comm — коммутативность (commutativity), id — идемпотентность (idempotent). После id: написаны единичные элементы, соответствующие операциям, описанным в разделе fns. Первый член раздела axioms показывает, что операция "—" дает обратный элемент относительно операции " + ", а второй член задает дистрибутивность, являющуюся общим свойством колец.
Рассмотрим подмножество кольца, состоящего из одних только ненулевых элементов. С использованием языка Eqlog его можно задать в виде, показанном на рис.17.

```
THEORY MULT USING RING IS
        SORTS MULT
        SUBSORTS
            MULT<RING
        VARS
            X,Y:RING
        AXIOMS
            MULT(1).
            MULT(X*Y):-MULT(X),MULT(Y).
ENDTH MULT
```

Рис.17. Определение конструкции mult, являющейся подсортом конструкции ring.

При этом theory *MULT* наследует theory *RING,* а описанный в ней sort *mult* является подсортом sort-а *ring.*
Данный тип наследования является наследованием по свойствам, и с теоретико-множественной точки зрения *mult* действительно является подмножеством множества *ring.*
Наличие в разделе axioms одних только ненулевых элементов не отражено в явном виде, а указано существование единицы и замкнутость относительно операции умножения (**что дает полугруппу**).
Проведя указанные подготовительные операции, определим с помощью языка Eqlog дробные числа. Определение дано на рис.18.

```
MODULE FRACTION[M::MULT]
        SORTS INVERTIBLE,FRACT
        SUBSORTS
            RING,UNVERTIBLE<FRACT
        FNS
            /:RING,MULT->FRACT
            *:FRACT,FRACT->FRACT(ASSOC COMM ID:0)
            +:FRACT,FRACT->FRACT(ASSOC COMM)ID:1)
            -1:INVERTIBLE->INVERTIBLE
        VARS
            Y,W:MULT
            X,Z:RING
        AXIOMS
            X=X/1
            X/Y=Z/W:-X*W=Z*Y.
            (X/Y)*(Z/W)=(X*Z)/(Y*W).
            (X/Y)+(Z/W)=((X*W)+(Z*Y))/(Y*W).
            INQVERTIBLE(X/Y):-MULT(X).
            (X/Y)-1=(Y/X).
ENDMOD FRACTION
```

Рис.18. Определение дроби.

Как видно из этого определения, theory *MULT* требует, чтобы знаменатель не равнялся нулю. Поскольку *MULT* является параметром module *FRACTION,* при конкретном создании модуля (объекта) *FRACTION* можно записать, например, следующее выражение:

```
make RATisFRACTION[INT]endmake
```

Здесь *RAT* — имя конкретно созданного экземпляра объекта; *INT* — еще один модуль, выражающий условие, которому должна удовлетворять теория *MULT* (в данном случае выражает целочисленность).

Ниже приводится более подробное описание процедуры создания экземпляра *INT* с описанием конструкции view на языке Eqlog. Полный вид процедуры будет следующим:

```
view NZINT-AS-MULT is INT as MULT by
    int is ring
    nzint is mult
endview
make RAT is FRACTION[NZINT-AS-MULT] endmake
```

Таким образом, на языке Eqlog предполагается реализовать мощные возможности. Он весьма интересен для рассмотрения перспектив объектно-ориентированного программирования.

4. Порождающие системы ИИ

4.1. Эмпирическое исследование ИИ

Для любого содержательного эмпирического исследования ИИ необходимы **три предпосылки**. **Во-первых, должен быть определен объект исследования; во-вторых, должна быть известна цель исследования этого объекта; в-третьих, должны быть определены ограничения, при которых проводится исследование.**

Объект исследования определен нами ранее как часть мира, различаемая как единое целое в течение достаточно длительного периода времени и подходящая для какого-либо конкретного исследования.

Цель исследования можно представить как **набор вопросов об объекте**, на которые исследователь (или его заказчик) хотят получить ответы. Если, например, объектом исследования является г. Киев, то целью исследования могут быть ответы на такие, например, вопросы: каким образом можно сократить преступность п городе или как улучшить движение транспорта; если объектом исследования является вычислительный комплекс, то целью исследования может быть поиск ответов на вопросы: каковы узкие места в комплексе, что можно сделать для повышения его проиводительности — и тому подобное; если исследуется больница, то возможны такие вопросы: как повысить готовность оказать срочную помощь в опасных случаях, как сократить среднее время пребывания пациента в больнице или что можно сделать для сокращения платы за лечение при сохранении его качества? Если музыковед изучает творчество какого-либо композитора, например Игоря Стравинского, то его вопрос будет, вероятно, выглядеть так: каковы основные особенности сочинений Стравинского, отичающие их от произведений других композиторов?

Ограничения в эмпирическом исследовании представляют ограниченные возможности выбора инструментов, ограниченные финансовые возможности и время, людские ресурсы и мощность вычислительной техники, правовые, моральные и другие нормы, которых должен придерживаться исследователь.

Основные этапы эмпирического исследования изображены на рис. 1; они послужат нам руководством при изложении материала данного раздела.

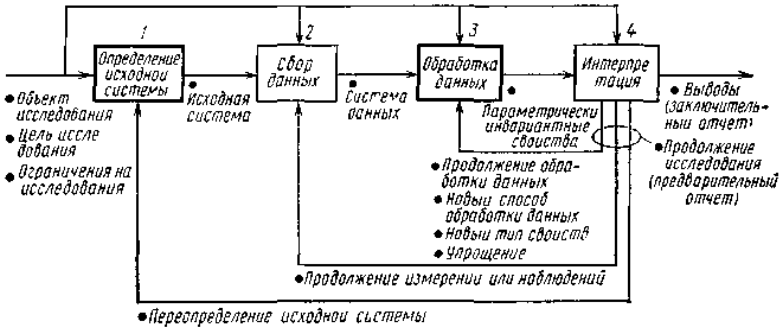


Рис. 1. Основные этапы эмпирических исследований систем ИИ

Первым этапом любого эмпирического исследования ИИ является *определение исходной системы* на соответствующем объекте. Этот этап достаточно подробно рассмотрен в раз. 2.7 и там же изображен на рис.1. Основной проблемой этого этапа исследования является выбор обычно из многих возможностей исходной системы ИИ, наилучшим образом соответствующей цели исследования и удовлетворяющей имеющимся ограничениям. Решение этой проблемы, безусловно, зависит от контекста. Оно требует со стороны исследователя не только опыта и знаний в изучаемой области, но и специального исследования. Перед тем как выбрать какую-либо определенную исходную систему ИИ для ее эмпирического исследования, часто приходится изучить сначала различные гипотетические варианты систем ИИ более высоких эпистемологических уровней.

С самого начала при выборе подходящей исходной системы ИИ **возникают две проблемы: выбор свойств и баз и выбор каналов наблюдения для них**. Каналы абстрагирования рассматриваются позже, когда возникает необходимость перевести систему ИИ на язык ФРИЗ.

Выбор свойств и баз, возможно, самое важное решение, принимаемое в процессе эмпирического исследования ИИ, так как этот выбор влияет на все последующие этапы этого процесса. Принять решение очень непросто и не всегда удается рационально объяснить это решение. **Решение часто опирается на некие априорные идеи, которые, когда они достаточно четко сформировались в сознании исследователя, называются научными теориями (гипотезами).** Некоторые ученые настаивают на том, что любой осмысленный выбор свойств для эмпирического исследования всегда опирается на некую теорию, которая или сформулирована явным образом, или возникла у

исследователя подсознательно, или является частью нашего унаследованного генетически врожденного знания.

Важно, чтобы в процессе определения исходной системы ИИ исследователь знал все возможности методологии систем. В противном случае он может без особой необходимости ограничить свой выбор. Одно из этих ограничений хорошо описано у Эшби:

«Тот, кто имеет некоторый опыт работы с математическими понятиями, легко может приобрести привычку считать, что **под термином *переменная* всегда понимается численная шкала с аддитивной метрикой.** Данное предположение является совершенно излишним ограничением, которое иногда приводит к фатальным заблуждениям. Метеорологи давно работают с пятью «типами облаков», ветеринары — с разными «паразитами у свиней», а гемаматологи — с четырьмя основными «группами крови». **Современная математика, использующая методы теории множеств, умеет работать с переменными такого типа, без которых в науках о поведении обойтись невозможно.**»

Другое часто встречающееся в эмпирических исследованиях ограничение — это использование только четких каналов наблюдения, даже если для некоторых признаков и баз нечеткие каналы наблюдения подходят значительно лучше. Подобное ограничение является совершенно излишним, поскольку имеются методы работы с нечеткими данными.

После выбора свойств и баз исследователь должен определить для них канал наблюдения. Как уже говорилось ранее, **каналы наблюдения задают разбиения заданного множества проявлений свойства или значений параметра**. **Любое такое разбиение будем называть *разрешающей формой (resolution form).*** Хотя и некоторых случаях разрешающие формы не могут быть определены математически (если только не принимать какие-то метафизические допущения), вполне возможно определить, является ли одна разрешающая форма уточнением другой или ее укрупнением (**в смысле стандартного уточняющего упорядочения, определенного на разбиениях заданного множества**). **Подобное сравнение днух разрешающих форм делается не математически, а с помощью сопоставления соответствующих процедур измерения.** Диапазон возможных разрешающих форм всегда имеет верхнюю границу, определяемую разрешающей способностью имеющихся измерительных инструментов. Нижней границей является любая разрешающая форма, состоящая только из двух блоков. Выбор фирмы из этого диапазона зависит от цели исследования.

После определения исходной системы ИИ становится возможным ***сбор данных***. **Этот процесс сводится к наблюдению или измерению отобранных свойств при определенных значениях баз** и записи этих наблюдений в некой подходящей форме (см. разд. 2.8). Если исследователь имеет возможность управлять некоторыми свойствами, он может этим воспользоваться. В этом случае **свойства, которыми он собирается управлять, рассматриваются как входные свойства, что дает направленную исходную систему ИИ.** Затем исследователь предлагает некие эксперименты, в которых согласно осуществимой экспериментально стратегии, связанной с целью исследования, задаются входные свойства. **При этом наблюдаются выходные свойства**. В результате получается ***система данных***.

После определения системы данных начинается следующий этап эмпирического исследования — *обработка данных*. Его целью является определение неких *параметрически инвариантных свойств* переменных, позволяющих экономно представлять данные и, если нужно, порождать их. На этом этапе ФРИЗ может очень помочь исследователю. При этом или используются все данные для вывода нужных параметров инвариантных свойств, или сначала обрабатывается только часть данных, а остальные резервируются для последующей проверки полученных свойств.

Существует ряд параметрически инвариантных свойств, но все они имеют нечто общее. Каждое такое свойство описывает *ограничение,* наложенное на переменные исходной системы, не меняющиеся в пределах параметрического множества. Если, например, параметром является время, то любое инвариантное времени свойство описывает ограничение на переменные, не меняющиеся во времени. Разные параметрические инвариантные свойства могут служить характеристиками типов ограничений, связываемых с различными эпистемологическими уровнями, или, наоборот, могут отличаться только способом, каким представляется один и тот же тип ограничений, связанный с определенным эпистемологическим уровнем. На первых отличиях базируется различение эпистемологических уровней систем ИИ, вторые служат для представления методологических отличий на определенном эпистемологическом уровне.

После того как данные каким-то образом обработаны и определены соответствующие параметрически инвариантные свойства, им необходимо дать *интерпретацию* с учетом цели исследования, т. е. нужно посмотреть, насколько они полезны для поиска ответов на поставленные в исследовании вопросы. Если на вопросы можно ответить адекватно, то исследование успешно завершено и

исследователь может подвести итоги и **готовить заключительный отчет**. В противном случае он может попытаться обработать данные другим способом. **Этот процесс может повторяться несколько раз, причем можно искать параметрически инвариантные свойства как того же уровня, так и других эпистемологических уровней.** В конечном счете исследователь получает набор порождающих систем ИИ или систем ИИ более высокого уровня, каждая из которых с определенной точки зрения правильно представляет данные. Подобный набор взаимодополняющих систем ИИ, каждая из которых отражает определенные свойства данных, часто дает исследователю лучшее понимание проблемы, чем какая-либо одна система ИИ. Система (или системы) ИИ, полученная в результате обработки данных, иногда оказывается слишком сложной, чтобы человек мог ее охватить целиком, и, следовательно, такая система ИИ не может помочь исследователю лучше понять проблему. В подобных случаях необходимо или по крайней мере желательно *уменьшить сложность* этой системы ИИ. Таким образом, ФРИЗ должен располагать средствами упрощения систем ИИ разных типов по критериям, указанным пользователем.

После обработки данных и интерпретации полученных свойств исследователь может также захотеть собрать дополнительные данные для того, чтобы повысить свою уверенность в правильности полученных свойств, или пересмотреть их на основе новых данных. Подобный *повторный сбор данных* изменяет систему данных, но оставляет без изменений исходную систему ИИ. Однако исследователь может сделать и более решительное изменение — переопределить исходную систему ИИ. В этом случае он должен повторить весь процесс для новой исходной системы ИИ.

Вся процедура эмпирического исследования систем ИИ, согласно рис. 1, может быть теперь описана следующим образом:

1) дан объект, цель и ограничения эмпирического исследования; на объекте определяется исходная система (подробности см. на рис. 1 раз.2.7);
2) для данной исходной системы ИИ собираются данные и предствляются в удобном виде, обычно в виде массива данных;
3) данные обрабатываются с целью определения неких представляющих их параметрически инвариантных свойств;
4) полученные параметрически инвариантные свойства интерпретируются в соответствии с целью исследования и делаются окончательные выводы или исследование начинается снова с этапа 3, 2 или 1.

ФРИЗ должен уметь работать со всеми задачами, связанными с этапом обработки данных. Поэтому он должен иметь возможность: 1) вывода из заданных данных параметрически инвариантных свойств всех нужных типов; 2) сравнения выведенных свойств и исключения тех систем, свойства которых не удовлетворяют критериям пользователя; 3) упрощения систем ИИ различных типов в соответствии с определенными пользователем критериями упрощения. Эти три возможности ФРИЗ рассматриваются в данном разделе только относительно систем эпистемологического» уровня, следующего непосредственно за уровнем систем данных.

Этот уровень будем называть уровнем 2 (см. рис. 3 раз. 1.3). На этом уровне параметрически инвариантные свойства представляют собой непосредственные описания (разного рода) общего ограничения, связанного с используемыми переменными. **Системы ИИ, содержащие подобные описания, будем называть *порождающими.*** Целью данного раздела является определение порождающих систем ИИ и демонстрация некоторых типов задач, в которых они используются.

Порождающие системы ИИ (как и системы ИИ более высоких эпистемологических уровней) определяются и рассматриваются в данной раьоте на языке **обобщенных представляющих систем ИИ (абстрактных переменных и параметров).** Это означает, что они представляются в терминах языка ФРИЗ , т. е. без семантики. Однако в тех случаях, когда рассматриваются приложения, порождающая система ИИ дополняется исходной системой ИИ, через которую и вводятся соответствующие семантические аспекты. Обобщенная представляющая система ИИ, входящая в обе эти системы ИИ, представляет собой интерфейс между языком ФРИЗ и объектно-ориентированным языком конкретной дисциплины. Естественно, в обеих этих системах ИИ обобщенная подобная система ИИ должна быть одной и той же.

4.2. Системы ИИ с поведением

Термин *поведение* используется в данной работе для получения характеристики общего параметрически инвариантного ограничения на переменные обобщенной представляющей системы ИИ и, может быть, на некоторые дополнительные абстрактные переменные.

Дополнительные переменные определяются на параметрическом множестве с помощью *правил сдвига (translation ru1e).* Такое правило

может быть применено или к переменной из заданной представляющей системы ИИ, или к введенной по каким-либо методологическим соображениям гипотетической переменной, обычно называемой *внутренней.* Вопросы, связанные с внутренними переменными, рассматриваются в разд. 4.10. В остальных разделах этой главы предполагается, что внутренние переменные в рассмотрение не вводятся. Так как описание параметрически инвариантного ограничения на рассматриваемые переменные может быть использовано для порождения состояний переменных при данном параметрическом множестве, системы ИИ, содержащие такие ограничения, будем называть **порождающими системами ИИ**. Поведение представляет собой одну из форм задания этого ограничения.

Для заданной обобщенной представляющей системы ИИ диапазон возможных типов параметрически инвариантных ограничений зависит от свойств, приписываемых параметрическому множеству. Если на этом множестве никаких свойств не определено (как это часто бывает для групп), то состояния переменных могут ограничивать только друг друга. Однако если параметрическое множество упорядочено, состояния переменных могут ограничиваться не только другими состояниями, но и состояниями выбранного *соседства* для каждого конкретного значения параметра. Поскольку **соседство является основой для представления параметрически инвариантного ограничения, оно само должно быть параметрически инвариантным**.

Соседство на упорядоченном параметрическом множестве обычно называется *маской* (почему, будет объяснено ниже) и **определяется через переменные, параметрическое множество и набор правил сдвига на параметрическом множестве.** Правило сдвига, скажем правило r_j,— это однозначная функция

$$r_j:\mathbf{W}\rightarrow\mathbf{W}, \qquad (1)$$

которая каждому элементу **W** ставит в соответствие другой (причем единственный) элемент **W**. Если, например, параметрическое множество полностью упорядочено (как в случаях, когда рассматривается время или одновременное пространство) и представляет собой множество последовательных целых положительных чисел, то любое правило сдвига может быть задано простым уравнением

$$r_j(\mathbf{w})=\mathbf{w}+\rho, \qquad (2)$$

где ρ — целая константа (положительная, отрицательная или нуль). При ρ=0 r_j называется *тождественным правилом сдвига.* Пусть задана обобщенная представляющая система ИИ **I**, определяемая уравнением (2 раз. 2.7). Обозначим через **V** множество переменных из **I**, а через *R*

набор правил сдвига, рассматриваемых для этих переменных. Тогда множество переменных

$$\mathbf{S}=\{s_1, s_2,..\},$$

называемых *выборочными переменными,* может быть введено с помощью уравнений

$$s_{k,w}=v_{i,rj(\mathbf{w})} \qquad (3)$$

для некоторых переменных $v_i \in V$ и правил сдвига $r_j \in R$; $s_{k,w}$ обозначено состояние выборочной переменной s_k при значении параметра **w**, а $v_{i,rj(\mathbf{w})}$ —состояние переменной v_i при значении параметра $r_j(\mathbf{w})$, т. е. при значении, полученном для заданного **w**, при применении правила сдвига r_j . Для полностью упорядоченного параметрического множества, правила сдвига которого имеют вид (2), уравнение (3) может быть переписано в более определенном виде

$$s_{k,\mathbf{w}}=v_{i,\mathbf{w}+\rho} \qquad (4)$$

Так как любое правило сдвига из набора R может быть применено к любой переменной из множества V, то множество всех возможных выборочных переменных представляется декартовым произведением $V\times R$. В действительности рассматриваются выборочные переменные, характеризуемые отношением

$$M \subseteq V\times R \qquad (5)$$

так, что всякой паре $(v_i, r_j) \in M$ соответствует одно уравнение из (3). Отношение M представляет **схему соседства на параметрическом множестве, в терминах которого определены выборочные переменные**. Как уже говорилось выше, **эта схема обычно называется маской**. Понятно, что для введения идентификаторов выборочных переменных k должна быть введена некая однозначная функция (кодирование).

$$\lambda: M\rightarrow N_{|M|}, \qquad (6)$$

где $|M|$ —это мощность множества M.

Если выборочная переменная s_k определена через переменную v_i и некоторое правило сдвига согласно уравнению (3), то множество состояний s_k, очевидно, то же самое, что и множество состояний v_i, т е V_i. Однако для удобства обозначений будем множество состояний выборочной переменной обозначать S_k; смысл любого S_k $(k \in N_{|M|})$ однозначно определяется маской в терминах одного из множеств V_i $(i \in N_n)$- Таким образом, декартово произведение

$$\mathbf{C} = S_1 \times S_2 \times \ldots \times S_{|M|}$$

представляет собой полное множество состояний выборочных переменных.

Рассмотрим сначала понятие маски и связанное с ним поведение представляющих систем ИИ для полностью упорядоченных параметрических множеств, а затем распространим его на частично

упорядоченные параметрические множества. Обозначим полностью упорядоченные параметрические множества T, а их элементы t $(t \in T)$. При этом уравнение (4) немного изменится:

$$s_{k,t}=v_{i,t+\rho} \qquad (7)$$

Для полностью упорядоченных параметрических множеств **маска может быть изображена в виде вырезки из матрицы,** представляющей декартово произведение $V\times R$. Это показано на рис. 2,*а,* на котором строки помечены идентификаторами i переменных из множества V, а столбцы —целыми константами ρ, связанными с правилами сдвига вида (2). Элементы матрицы или пусты, или представляют собой идентификаторы k выборочных переменных, приписанные парам (i, ρ) согласно (7); пустые элементы матрицы соответствуют элементам $V\times R$, не входящим в маску. В визуальном представлении становится ясно, почему мы используем термин «маска».

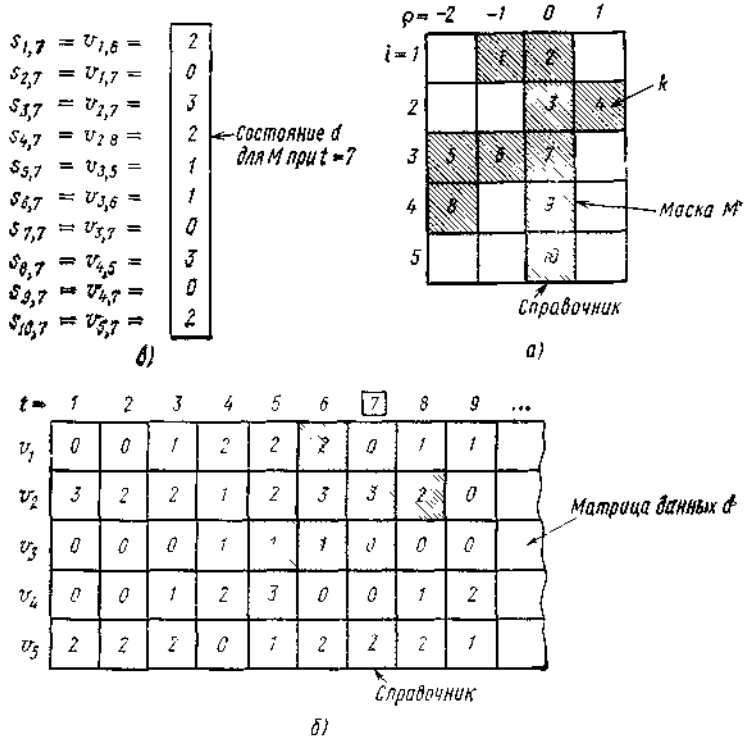


Рис 2. Пояснение понятия маски для полностью упорядоченных параметрических множеств

Часто бывает удобно разбить маску M на подмаски M_i, каждая из которых связана с одной переменной v_i из подобной системы. Формально

$$M_i = \{(\alpha, \beta) | (\alpha, \beta) \in M, \alpha = v_i\}. \quad (8)$$

В визуальном (матричном) представлении M подмаски M_i представляют собой строки.

В любой маске один столбец соответствует тождественному правилу сдвига ($\rho = 0$). Этот столбец имеет особое значение, поскольку связанные с ним выборочные переменные идентичны базовым переменным заданной представляющей системы. Будем этот столбец в масках называть *справочником.* Если маска помещена на матрицу данных таким образом, что справочник совпадает с определенным значением t, то маска выделит только некоторое подмножество элементов, а именно элементы, представляющие полное состояние выборочных переменных при данном значении t. Так, например, на рис. 2,б изображена маска (определенная на рис. 2,а), помещенная на матрицу данных d при $t=7$ (справочник маски совпадает с $t=7$). Полное состояние выборочных переменных для этого положения маски показано на рис. 2,в. Обратите внимание на то, что состояния справочника выборочных переменных $s_2, s_3, s_7, s_9, s_{10}$ в точности те же (для любого t), что и состояние базовых переменных соответственно v_1, v_2, v_3, v_4, v_5. Остальные выборочные переменные представляют собой состояния из параметрического соседства в t. Для любой маски при любом t схема соседства сохраняется. Если t — время, то переменная s_4 будет представлять будущее (относительно рассматриваемого значения t) состояние переменной v_2, а переменные s_5 и s_4 будут представлять, например, прошлые состояния переменной v_3.

Любая маска представляет определенную точку зрения, в соответствии с которой представляются ограничения на базовые переменные. Самый простой способ задания определенной маски - это перечисление всех полных состояний соответствующих выборочных переменных. В общем виде подобный перечень является подмножеством декартова произведения **C**, т. е. многомерным отношением, определенным на **C**. Это отношение определяется функцией

$$f_в: \mathbf{C} \to \{0, 1\}, \quad (9)$$

такой что $f_в(\mathbf{c}) = 1$, если состояние с входит в перечень, и $f_В(\mathbf{c}) = 0$ в'противном случае. Таким образом, **функция f_B —это типичная функция выбора. Она выбирает состояния выборочных переменных из множества всех потенциальных состоянии (из декартова произведения C).** Так как подобный выбор дает по крайней мере

некоторые сведения о поведении этих переменных, функцию f_B обычно называют *функцией поведения (behavior).* Функция, определяемая уравнением (3), задает только один из существующих типов функций поведения, разными способами описывающих ограничения на переменные. В разд. 4 3 вводятся разные функции поведения, которые рассматриваются как методологические отличия. В этом разделе мы ограничимся выбирающей функцией поведения, определяемой (9). Обратите внимание на то, что функция f_B оцределяет реально встречающиеся состояния **c**, но не определяет значение параметра при котором они имеют место. Таким образом, **эта функция является параметрически инвариантной.** Обратите также внимание на область определения f_B. Она одинакова для всех типов функций поведения и определяется через маску, которая, в свою очередь, определяется через переменные и параметры представляющей системы ИИ. Отсюда следует, что некоторая система ИИ, скажем система F_B, характеризующая параметрически инвариантное ограничение на множество переменных через функции поведения, определяется тройкой

$$\mathbf{F}_B = (\mathbf{I}, M, f_B), \quad (10)$$

где **I** —обобщенная представляющая система ИИ; M — маска, определенная на **I**; $f_В$ —функция поведения, определенная через **M** и **I**. Будем такую систему называть *системой ИИ с поведением.*

Несмотря на то, что любая система ИИ с поведением, определяемая (10), неким конкретным параметрически инвариантно описывает ограничения на переменные представляющей системы ИИ, она не содержит описания того, как использовать это ограничение **для порождения данных**. **Для разработки такого описания нужно разбить выборочные переменные на два подмножества**

1) переменные, состояния которых порождаются из ограничения; назовем их *порождаемыми переменными;*

2) переменные, состояния которых используются как условия в процессе генерации, назовем их *порождающими переменными.*

Для заданной системы ИИ с поведением одним из способов определения порожденных и порождающих переменных является определение для данной маски M двух подмасок M_g и $M_{\bar{g}}$. Будем

$$M_G = (M, M_g, M_{\bar{g}}), \quad (11)$$

где

$$M_g,\ M_{\bar{g}} \subset M,\quad M_g \cup M_{\bar{g}} = M,\quad M_g \cap M_{\bar{g}} = \emptyset,$$

называть *маской порождения,* т. е это маска M и ее разбиение на порождаемую подмаску M_g и порождающую подмаску $M_{\bar{g}}$.

По аналогии с разбиением M на M_g и $M_{\bar{g}}$ множество $M_{|M|}$ идентификаторов k выборочных переменных можно разбить на два подмножества, скажем K_g и $K_{\bar{g}}$, представляющих идентификаторы соответственно порождаемых и порождающих переменных Для удобства обозначений **кодирующая функция** (6) может быть заменена двумя функциями

$$\lambda_g : M_g \to K_g, \quad \lambda_{\bar{g}} : M_{\bar{g}} \to K_{\bar{g}}, \tag{12}$$

с помощью которых множества состояний $\mathbf{G}$ и $\bar{\mathbf{G}}$ соответственно порождаемых и порождающих переменных задаются декартовыми произведениями

$$\mathbf{G} = \bigtimes_{k \in K_g} S_k, \quad \bar{\mathbf{G}} = \bigtimes_{k \in K_{\bar{g}}} S_k. \tag{13}$$

Теперь способ представления состояния порождаемых переменных (скажем $\mathbf{g} \in \mathbf{G}$), определяемого по состоянию порождающих переменных (скажем, $\mathbf{g} \in \mathbf{G}$), можно выразить функцией

$$f_{\mathbf{GB}} : \bar{\mathbf{G}} \times \mathbf{G} \to \{0, 1\}, \tag{14}$$

где

$$f_{\mathbf{GB}}(\bar{\mathbf{g}}, \mathbf{g}) = \begin{cases} 1, \textit{если } g \textit{ может иметь место или если имеет место } \bar{\mathbf{g}} \\ 0, \textit{если } g \textit{ не может иметь место или если имеет место } \bar{\mathbf{g}} \end{cases}$$

Назовем эту функцию *порождающей функцией поведения.*

Если маску M и функцию f_B из (10) заменить соответственно на M_G и f_{GB}, то получится альтернативная система

$$F_{GB} = (\mathbf{I}, M_G, f_{GB}). \tag{15}$$

Будем называть такую систему ИИ *порождающей системой ИИ с поведением.*

Использование порождающей системы ИИ с поведением для порождения данных включает следующие два этапа:

а) для некоторого значения $t \in T$ задано состояние $\tilde{\mathbf{g}} \in \tilde{\mathbf{G}}$; для определения состояния $\mathbf{g} \in \mathbf{G}$ при том же значении используется функция f_{GB};

б) значение t заменяется на новое и повторяется этап а).

Необходимо прояснить несколько вопросов, связанных с двухэтапной процедурой порождения. Во-первых, на этапе а) неявно предполагается, что при заданном значении t состояние $\bar{\mathbf{g}}$ известно. При первом выполнении этого этапа данное состояние определяется пользователем как подходящее *начальное условие.* Однако после этого все полностью определяется самим процессом порождения, т. е. состояниями $\bar{\mathbf{g}}$ и g, связанными с предшествующим значением t. При этом предполагается, что значения t должны на этапе *(б)* изменяться в соответствии с порядком, заданным на множестве T. Таким образом, значения t заменяются или на t+1 или на t—1. В первом варианте начальное условие должно быть определено для наименьшего возможного значения t, а во втором — для наибольшего возможного значения t.

Во-вторых, из необходимости порождения данных в одном из двух порядков следует, что существует только два содержательных разбиения маски M на M_g и $M_{\bar{g}}$, каждое из которых соответствует одному из двух порядков порождения. Если данные порождаются в порядке возрастания (убывания) t, то M_g содержит ровно по одному элементу каждой подмаски M, $(i \in N_n)$, определенной в (8), элемент с наибольшим (наименьшим) значением ρ; остальные элементы M входят в $M_{\bar{g}}$. Таким образом, графически получается, что M_g — это множество самых правых элементов M (правый край этой маски) или, наоборот, множество самых левых элементов M (левый край маски).

В-третьих, предполагается, что для любого состояния $\bar{\mathbf{g}} \in \bar{\mathbf{G}}$ имеется по крайней мере одно состояние $\mathbf{g} \in \mathbf{G}$, допустимое функцией f_{GB} [т. е. $f_{\mathrm{GB}}(\mathrm{g}, \bar{\mathbf{g}}) = 1$]. Если допускается только одно состояние, то для любого начального условия данные порождаются однозначно; такие системы ИИ называются *детерминированными.* Если допускается более чем одно состояние, то порождение данных проблематично, так как порождаемое состояние не всегда однозначно определено. Для таких систем выбирающие функции поведения не подходят. Более содержательно они описываются функциями поведения других типов, рассматриваемых в разд. 4.3. Для детерминированных систем ИИ представление (14) порождающей функции поведения f_{GB} может быть заменено более простым представлением

$$f_{\mathrm{GB}} : \bar{\mathbf{G}} \to \mathbf{G}. \tag{16}$$

Пример 1. Для пояснения процесса порождения данных порождающей системой ИИ с поведением типа, определяемого уравнением (15),

положим, что подобная система ИИ состоит из упорядоченного параметрического множества $T=N_{99}$ и пяти переменных $v_1,..., v_5$, состояния которых будут определены ниже. Воспользуемся маской, заданной на рис. 2. Данные могут порождаться или в порядке возрастания, или в порядке убывания значений параметра t. Оба эти варианта показаны соответственно на рис.3 и 4.

В первом случае (рис. 3) порождаемые выборочные переменные— это переменные, соответствующие правому краю маски, т. е. s_2, s_4, s_7, s_9, s_{10}; остальные выборочные переменные являются порождающими.

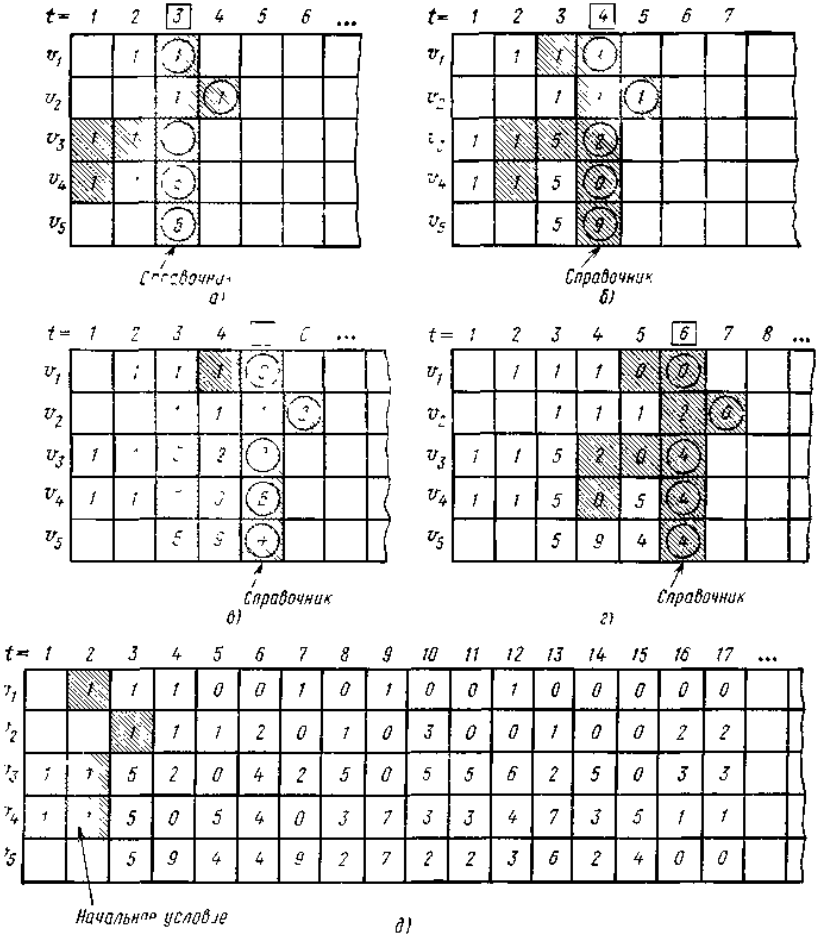


Рис. 3. Данные, порожденные в порядке возрастания значений параметра *t* (пример 1)

Порождение данных в матрице данных происходит слева направо. Пусть порождающая функция поведения f_{GB}, представленная в виде (16), определяется уравнениями

$$s_{k,t} = s_{1,t} + s_{3,t} + s_{5,t} + s_{6,t} + s_{8,t} \pmod{k}$$

при k = 2, 4, 7, 9, 10. Множества состояний порождаемых переменных определяются этими уравнениями, а множества состояний порождающих переменных — их положением в маске. Например, множество состояний порождаемой переменной s_4 — это 0, 1,2, 3, так как уравнение для s_4 берется по модулю 4; порождающая переменная s_3 имеет то же множество состояний, что и s_4, так как обе эти переменные определены через одну и ту же переменную представляющей системы ИИ (т. е. $S_3 = S_4 = V_2$).

Первой осмысленной позицией маски на матрице данных (позиция определяется положением справочника маски) является позиция t=3; позиции $t=1$ и $t=2$ смысла не имеют, так как состояния некоторых выборочных переменных для этих позиций не определены (t+ρ не входит в множество T). Начальное условие состоит из шести элементов матрицы данных: $v_{1,2}$, $v_{2,3}$, $v_{3,1}$, $v_{4,1}$, $v_{4,2}$. Пусть, например, все эти элементы равны 1. Еще пять элементов матрицы данных — $v_{1,1}$, $v_{2,1}$, $v_{2,2}$, $v_{5,1}$, $v_{5,2}$ — не могут быть порождены, а могут быть заданы пользователем, но для порождения данных эти переменные не нужны.

На рис. 3.а, *б, в, г* подробно показано порождение состояний соответственно для t=3, 4, 5, 6; кружками обведены порожденные состояния.

На рис. 3, *д* показано начальное условие и несколько больший фрагмент порожденной матрицы данных.

Если данные порождаются в порядке убывания t (см. рис. 4), то порождаемыми переменными являются переменные, представляющие левый край маски, т. е. переменные s_1, s_3, s_5, s_8, s_{10}.

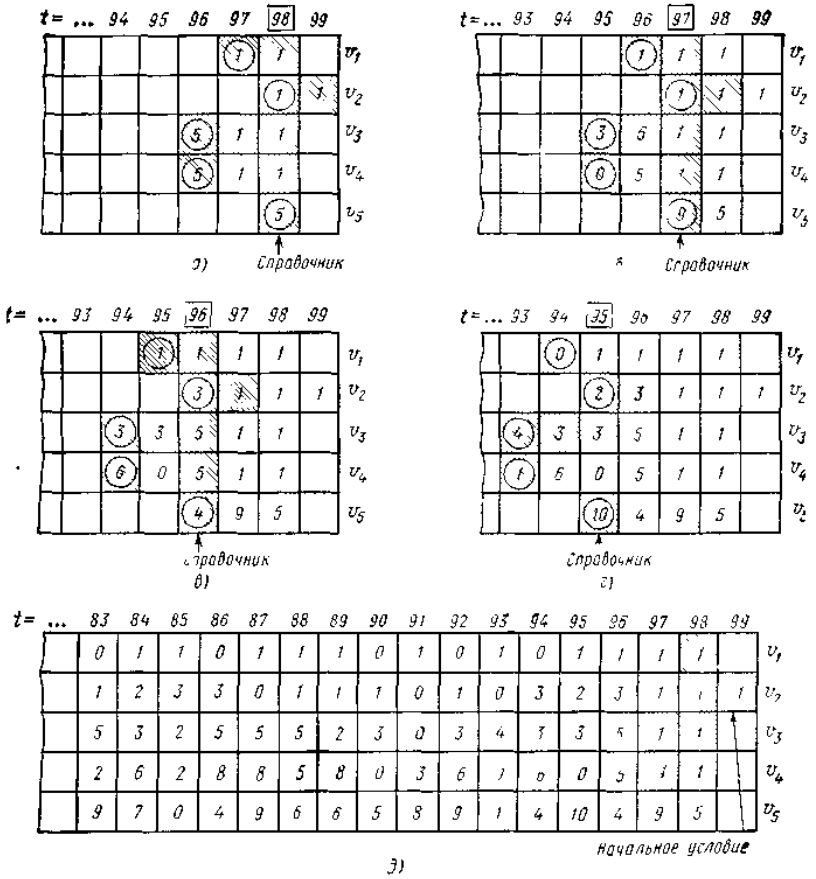


Рис. 4. Данные, порожденные в порядке убывания значений параметра *t* (пример 1)

Данные в матрице данных порождаются справа налево. Предположим теперь, что f_{GB} определяется уравнениями

$$s_{k,t} = s_{2,t} + s_{4,t} + s_{6,t} + s_{7,t} + s_{9,t} \pmod{k+1}$$

при *k=1,* 3, 5, 8, 10. Порождение данных при t=98, 97, 96, 95 подробно показано на рис. 4,а, *б, в* и *г.* На рис. 3.4,*д* показано начальное условие и несколько больший фрагмент порожденной матрицы.

4.3. Методологические отличия

Параметрически инвариантное ограничение на множество выборочных переменных может быть охарактеризовано разными способами. Простое описание, рассмотренное в разд.4.2, может ограничиться заданием функции выбора, определенной на соответствующем множестве состояний. Хотя функция выбора является, вероятно, наиболее подходящим формальным аппаратом для задания ограничений в детерминированных системах ИИ, в которых порождение данных удобно описывать с помощью функции (16), для работы с недетерминированными системами ИИ функции выбора не годятся.

Традиционно с недетерминированными системами работают методами теории вероятностей. При этом основным понятием при описании ограничений на переменные является понятие вероятностной меры. Несмотря на то, что это наиболее развитый и важнейший математический инструмент для работы с недетерминированными системами, вероятностная мера рассматривается как частный случай более общего класса мер, называемых **нечеткими мерами**.

Любая мера ставит в соответствие подмножествам заданного множества какие-либо действительные числа, характеризующие (измеряющие) количество некоего свойства, связанного с каждым подмножеством. Под множеством мы будем понимать множество всех состояний выборочных переменных и рассматривать такое свойство как степень правдоподобия того, что может иметь место любое из состояний для каждого определенного подмножества. Степень правдоподобия обычно задается действительным числом из единичного интервала; чем больше число, тем выше степень правдоподобия. Любой класс мер определяется через некие математические свойства, задаваемые набором правил вычисления, которые относятся к соответствующему классу мер. Для соотнесения этих математических свойств с общепринятыми понятиями различным классам мер были даны содержательные названия, такие, как **вероятность, возможность, правдоподобие, доверие**. Несмотря на то, что эти названия позволяют быстро сориентироваться в этих мерах, буквально их понимать не следует. Вопрос о том, подходит или нет некая мера для данного приложения (и аналогичные вопросы), должен решаться исходя из математических свойств этой меры, а не по общепринятому смыслу ее названия.

В нашем случае меры определяются на подмножествах декартового произведения **C**. Отсюда мера определяется функцией

$$\mu: \wp(\mathbf{C}) \to [0, 1], \qquad (17)$$

где $\mathcal{P}(\mathbf{C})$ —мощность множества **C**. Чтобы функция являлась мерой, она должна обладать следующими свойствами нечетких мер:

(μ1) $\mu(\emptyset) = 0;\ \mu(\mathbf{C}) = 1;$

(μ2) если $x_1 \subseteq x_2$, то $\mu(X_1) \leqslant \mu(X_2)$;

(μ3) если $X_1 \subseteq X_2 \subseteq \ldots$ или $X_1 \supseteq X_2 \supseteq \ldots$,
то $\lim\limits_{i\to\infty} \mu(X_i) = \mu \lim(X_i)$.

Требование (μ1) очевидно. Требование (μ2), обычно называемое свойством *монотонности,* не допускает, чтобы подмножество другого подмножества **C** обладало большей мерой, чем включающее подмножество. Согласно требованию (μ3), называемому *непрерывностью,* предел мер бесконечной монотонной последовательности подмножеств **C** должен совпадать с мерой предела этой последовательности. К дискретным системам, в которых **C** всегда является конечным множеством, требование непрерывности, естественно, неприменимо.

В литературе описаны самые разные классы нечетких мер, имеющих разные свойства. На рис. 5 приведена диаграмма, изображающая отношение включения для некоторых мер.

Рис. 5. Некоторые классы нечетких мер

Так, например, класс вероятностных мер входит в класс мер правдоподобия и в класс мер доверия, но не пересекается с классами мер возможности или необходимости.

Классы нечетких мер рассматриваются как методологические отличия. Они используются в порождающих системах ИИ и во всех системах более высоких эпистемологических уровней. Чтобы продемонстрировать то, как разные интеллектуальные задачи влияют на выбор определенного класса мер, в данной работе в контексте различных интеллектуальных задач рассматриваются два класса нечетких мер. **Первый — классический и хорошо разработанный класс вероятностных мер, второй — это класс возможностных мер.** Необходимо отметить, что возможные меры приложимы только к конечным множествам и к некоторым частным случаям бесконечных множеств; в общем случае эти меры не удовлетворяют требованию непрерывности. Таким образом, они наверняка применимы к дискретным, но не к непрерывным системам ИИ.

Предполагается, что читатель знаком с основами теории вероятностей, центральным понятием которой является вероятностная мера. Из теории вероятностей хорошо известно, что любая вероятностная мера, скажем мера *p,* однозначно определяется функцией распределения

$$f_B: \mathbf{C} \to [0, 1], \tag{18}$$

которая должна удовлетворять соответствующим требованиям согласно формуле

$$p(X) = \sum_{\mathbf{c}\in\lambda} f_B(\mathbf{c}), \tag{19}$$

где $X \in \mathcal{P}(\mathbf{C})$. Мы применили в обозначении функции распределения тот же индекс *B,* что и в описании функции поведения, определенной в (9), так как она также будет использоваться в качестве функции поведения. Функции (9) и (18), по существу, играют ту же самую роль при определении системы ИИ с поведением (10), хотя они методологически отличны и ни одна из них не является частным случаем другой. Поэтому мы и обозначим их одинаково. То, какая из двух функций используется в каждом конкретном случае, должно следовать из методологических отличий конкретной постановки задачи.

Мера возможности — это функция

$$\pi: \mathcal{P}(\mathbf{C}) \to [0, 1], \tag{20}$$

удовлетворяющая следующим требованиям:

(π1) $\pi(\emptyset) = 0;\quad \pi(\mathbf{C}) = 1;$

(π2) $\pi\left(\bigcup_i X_i\right) = \max_i \pi(X_i).$

Очевидно, что из (π 2) следует монотонность нечетких мер. Как уже отмечалось выше, π не всегда удовлетворяет требованию не-

прерывности и, следовательно, не подходит для систем ИИ с непрерывными переменными.

Хорошо известно, что мера возможности π однозначно определяется функцией распределения возможностей f_B, имеющей вид (18) и определяемой формулой

$$\pi(X) = \max_{\mathbf{c}\in X} f_B(\mathbf{c}). \qquad (21)$$

По тем же соображениям, что и для функций распределения вероятностей, мы снова использовали обозначение f_B.

Обратите внимание на то, что **функция выбора (9) представляет собой частный случай функции распределения возможностей, но не функции распределения вероятностей.** Это функция распределения возможностей, у которой степень возможности $f_B(\mathbf{c})$ для любого $\mathbf{c} \in \mathbf{C}$ равна или 0, или 1. Этот частный случай обычно называют *четкой функцией распределения возможностей* (рис. 5). При этом порождающая функция поведения f_{GB} имеет вид

$$f_{GB}: \bar{\mathbf{G}} \times \mathbf{G} \to [0,\ 1]: \qquad (22)$$

где $f_{GB}(\bar{\mathbf{g}}, \mathbf{g})$—условная вероятность или соответственно условная возможность при условии $\bar{\mathbf{g}}$. Чтобы подчеркнуть, что f_{GB} задает условные вероятности или условные возможности, для обозначения вероятности (или возможности) **g** при заданном **g** вместо $f_{GB}(\bar{\mathbf{g}}, \mathbf{g})$ используется стандартное обозначение $f_{GB}(\mathbf{g}|\ \bar{\mathbf{g}})$.

Функцию выбора (4) можно рассматривать как частный (четкий) случай возможностной интерпретации функции (22), но не как частный случай ее вероятностной интерпретации. Однако для детерминированных систем ИИ порождающая функция поведения в виде (16) представляет методологический интерес и, следовательно, имеет смысл методологически отличать эту форму от возможностной. Применение или неприменение функции (16) —это вопрос, относящийся к реализации ФРИЗ, но не к его архитектуре. С точки зрения пользователя ФРИЗ достаточно, чтобы различались только вероятностная и возможностная интерпретации и, может быть, некоторые другие полезные классы нечетких мер.

До сих пор мы рассматривали только нейтральные системы ИИ с поведением (базовые и порождающие). Для описания их направленных аналогов необходимо разбить соответствующее множество выборочных переменных на два подмножества:

1) выборочные переменные, определяемые средой, т. е. входные переменные [переменные v_i, для которых $u(i)=0$];

2) остальные выборочные переменные, связанные с рассматриваемой маской.

Эти два подмножества выборочных переменных можно определить, разбив заданную маску M на две подмаски. Пусть подмаска M_e определяет выборочные переменные, задаваемые средой, а подмаска $M_{\bar{e}}$ —остальные. Тогда тройка

$$\widehat{M} = (M, M_e, M_{\bar{e}}), \qquad (23)$$

для которой справедливо, что

$$M_e,\ M_{\bar{e}} \subset M,$$
$$M_e \cup M_{\bar{e}} = M,$$
$$M_e \cup M_{\bar{e}} = \emptyset,$$

определяет маску направленной системы ИИ с поведением.

Согласно разбиения $M_e M$ и на $M_{\bar{e}}$-множество $N_{|M|}$. идентификаторов выборочных перепенных., определяемых M, разобьется на подмножества K_e и $K_{\bar{e}}$. Кодирующая функция (6) будет заменена соответственно на две функции

$$\lambda_e : M_e \to K_e,$$
$$\lambda_{\bar{e}} : M_{\bar{e}} \to K_{\bar{e}}, \qquad (24)$$

и будут определены следующие два множества состояний:

$$\mathbf{E} = \bigtimes_{k\in K_e} S_k,$$
$$\bar{\mathbf{E}} = \bigtimes_{k\in K_{\bar{e}}} S_k, \qquad (25)$$

необходимые для направленных систем ИИ. Функция поведения направленных систем ИИ имеет вид

$$\widehat{f}_B : \mathbf{E} \times \bar{\mathbf{E}} \to [0,\ 1], \qquad (26)$$

где $f_B(\mathbf{e}, \bar{\mathbf{e}})$ —это условная вероятность или условная возможность (или какая-то другая мера) и, следовательно, вместо записи $f_B(\mathbf{e}, \bar{\mathbf{e}})$ можно использовать стандартную форму $f_B(\bar{\mathbf{e}}\ |\mathbf{e})$. Теперь можно определить *направленную систему с поведением* как тройку

$$\widehat{\mathbf{F}}_B = (\widehat{\mathbf{I}}, \widehat{M}, \widehat{f}_B). \qquad (27)$$

Порождающая функция поведения для направленных систем ИИ может быть введена с помощью разбиения $M_{\bar{e}}$ - на два подмножества

M_g и $M_{\bar{g}}$, соответствующих порождаемым и порождающим переменным. Делается это точно так же, как было описано для *M*. Таким образом, порождающая маска для направленных систем ИИ задается четверкой

$$\hat{M}_G = (M,\ M_e;\ M_g,\ M_{\bar{g}}),$$

где $\{M_e, M_g, M_{\bar{g}}\}$ — это разбиение *M*. Снова определяются кодирующие функции (12), но $\{M_g, M_{\bar{g}}\}$ рассматривается теперь как разбиение $M_{\bar{e}}$. Множества **G** и $\bar{\mathbf{G}}$ определяются формулами (13). Теперь

$$\hat{f}_{GB} : \mathbf{E} \times \bar{\mathbf{G}} \times \mathbf{G} \to [0,\ 1], \qquad (28)$$

где f_{GB} (**e**, **g**, $\bar{\mathbf{g}}$)—это условная вероятность или возможность (или какая-то другая мера), и, следовательно, в соответствии с традицией ее можно записать в виде f_{GB} (**g**|**e**, $\bar{\mathbf{g}}$). Для детерминированных систем ИИ f_{GB} можно переписать в более удобном виде

$$\hat{f}_{GB} : \mathbf{E} \times \bar{\mathbf{G}} \to \mathbf{G}, \qquad (29)$$

который представляет собой направленный аналог порождающей функции поведения, определенной (16). Если предположить, что смысл (методологическое отличие) f_{GB} определен, то направленная порождающая система ИИ с поведением определяется тройкой

$$\hat{\mathbf{F}}_{GB} = (\hat{\mathbf{I}},\ \hat{M}_G, \hat{f}_{GB}).\ \text{А} \qquad (30)$$

На рис. 6 показано разбиение маски на три подмаски M_e, M_g, $M_{\bar{g}}$ (и соответствующее разбиение идентификаторов выборочных переменных) в предположении, что v_1 и v_2 — это входные переменные.

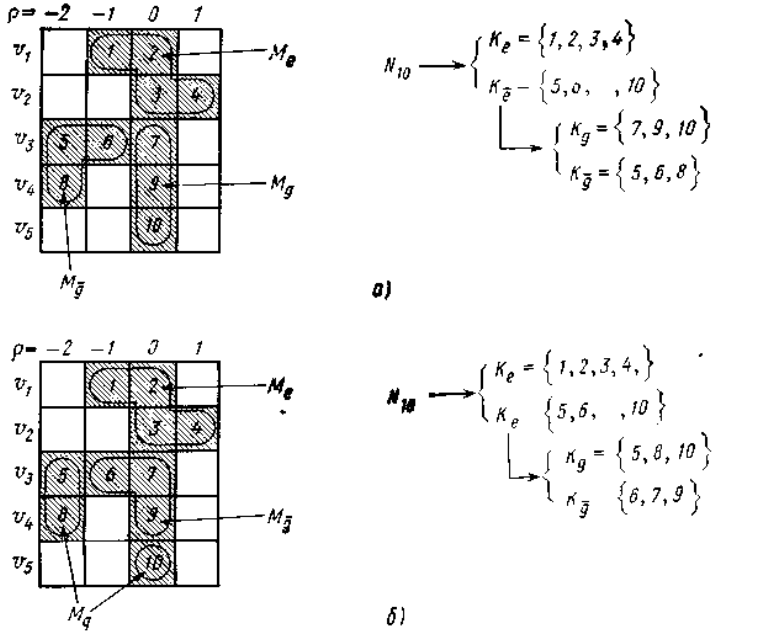


Рис. 6. Разбиения маски для направленной представляющей системы ИИ с полностью упорядоченным параметрическим множеством и **u**=(0, 0, 1, 1) для обоих возможных порядков порождения данных.

На рис.6,а и *б* показаны два варианта, соответствующие порождению данных в порядке возрастания и убывания значений параметра.

4.4. От систем данных к системам ИИ с поведением

Важный класс интеллектуальных задач, который будем называть *индуктивным моделированием систем ИИ,* может быть (в контексте ФРИЗ) описан в первом приближении как множество задач, связанных с подъемом по эпистемологической иерархии систем ИИ. Все задачи этого класса характеризуются следующим общим описанием: дано конкретная система ИИ, скажем система *x,* определенного эпистемологического уровня;
множество всех конкретных систем ИИ некоего более высокого эпистемологического уровня, совместимых с системой *x* (т. е. основанных

на той же представляющей системе ИИ, с теми же методологическими отличиями, скажем множество Y;
набор соответствующих требований Q относительно неких свойств систем ИИ из множества Y, причем одним из этих требований является требование, чтобы данная система x была аппроксимирована как можно точнее системой ИИ более высокого уровня и требуется определить Y_Q — подмножество Y, такое, чтобы любая система ИИ из Y_Q удовлетворяла всем требованиям, определенным в наборе Q.
Для того чтобы продемонстрировать в данном разделе задачи определения систем ИИ с поведением, представляющих заданную систему данных и обладающих некими подходящими дополнительными свойствами, будем считать, что x — это система данных с номинальными переменными (Номинальные переменные — это переменные с номинальной шкалой, т. е. шкалой, допускающей только взаимооднозначные преобразования.), Y — множество всех систем ИИ с поведением с вероятностными или возможностными функциями поведения, совместимыми с x, а набор Q состоит из:
1) подмножества Y_r множества Y, определенного или пользователем, или ФРИЗ (как выбор по умолчанию);
2) требования, чтобы несогласованность между соответствующими переменными заданной системы данных и системы ИИ с поведением из Y_Q была как можно меньшей;
3) требование, чтобы степень неопределенности при порождении данных системой ИИ с поведением из подмножества Y_Q была как можно меньшей;
4) требованием, чтобы система ИИ из подмножества Y_Q была как можно более простой;
5) предпочтения требования 2 требованиям 3 и 4.
В этой общей формулировке требование 1 сводится к определению множества допустимых масок. Если параметрическое множество не упорядочено, то понятие параметрического соседства не определено, и, следовательно, существует только одна осмысленная маска. Эта маска, определяемая тождественным правилом сдвига; она называется *маской без памяти.* Поскольку в этом случае имеется только одна приемлемая маска, задача оказывается довольно тривиальной [требования 3, 4 и 5 просто неприменимы]. Эта задача сводится к определению для заданных данных функции распределения вероятностей или возможностей, удовлетворяющих требованию 2. Она решается полным перебором данных с помощью маски без памяти (в данном случае порядок выбора не важен) и определения для каждого состояния выборочных переменных **c** (в данном случае они совпадают с основными переменными) числа $N(\mathbf{c})$ их появлений в данных. Числа $N(\mathbf{c})$ для всех

$\mathbf{c} \in \mathbf{C}$ обычно называются *частотами* состояний **c**. Они используются для вычисления по некоторым правилам соответствующих функций вероятностей или возможностей.
Вычислять распределение вероятности или возможности по частотам можно разными способами. Выбор способа зависит от того, какой смысл придает пользователь этим вероятностям или возможностям. Так, например, если вероятности рассматриваются как характеристики данных, то обычно вычисляются *относительные частоты,* т. е. отношения $N(\mathbf{c})$ к общему числу имеющихся выборок из данных по используемой маске. Отсюда

$$f_B(\mathbf{c}) = \frac{N(\mathbf{c})}{\sum_{\alpha \in \mathbf{C}} N(\alpha)}. \quad (31)$$

Если, однако, вероятности рассматриваются как оценки частот по уже имеющимся результатам наблюдения, то они вычисляются по формуле

$$f_B(\mathbf{c}) = (N(\mathbf{c}) + 1) \Big/ \left(\sum_{\alpha \in \mathbf{C}} N(\alpha) + |\mathbf{C}| \right). \quad (32)$$

Поскольку распределения возможностей менее ограничены, чем их вероятностные аналоги (например, к ним не надо добавлять 1), существует еще больше возможных правил для вычисления их по частотам $N(\mathbf{c})$. Естественный способ вычислений распределения возможностей, который можно считать аналогом формулы (31)—это считать значение возможности равной отношению частоты $N(\mathbf{c})$ к максимальной зафиксированной частоте, т. е.

$$f_B(\mathbf{c}) = N(\mathbf{c}) / \max_{\alpha \in \mathbf{C}} N(\alpha). \quad (33)$$

По другой формуле распределение возможности вычисляются по соответствующим вероятностям. Пусть $f_в(\mathbf{c})$ и $f'_в(\mathbf{c})$ —это соответственно возможность и вероятность состояния с ($\mathbf{c} \in \mathbf{C}$). Тогда

$$f_B(\mathbf{c}) = \sum_{\alpha \in \mathbf{C}} \min [f'_B(\mathbf{c}),\ f'_B(\alpha)]. \quad (34)$$

По этой формуле распределение возможности выражается через верхние границы значений вероятностей .
Для иллюстрации применения формул (31) — (34) в табл. 1 приведены распределения вероятностей и возможностей для конкретного распределения частот.

Таблица 1.

Сравнение разных распределений вероятностей и возможностей для одних и тех же частот $N(\mathbf{c})$

					f_β(c) вычисляется по формуле			
c_1	c_2	c_3	$N(\mathbf{c})$	(3.31)	(3.32)	(3.33)	(3.34) и (3.31)	(3.34) и (3.32)
0	0	0	20	0.20	0.194	0.4	0.7	0.722
0	0	1	0	0.00	0.009	0.0	0.0	0.072
0	1	0	10	0.10	0.102	0.2	0.5	0.538
0	1	1	5	0.05	0.056	0.1	0.3	0.354
1	0	0	0	0.00	0.009	0.0	0.0	0.072
1	0	1	50	0.50	0.472	1.0	1.0	1.000
1	1	0	10	0.10	0.102	0.2	0.5	0.538
1	1	1	5	0.05	0.056	0.1	0.3	0.354

Для нечетких данных простым подсчетом вычислить $N(\mathbf{c})$ нельзя, так как полное состояние **c** с некоторой степенью достоверности $d_{c,w}$ наблюдается при любом конкретном значении параметра **w**. Степень достоверности (для четких данных она равна либо 0, либо 1) определяется функцией

$$a: [0, 1]^2 \to [0, 1],$$

агрегирующей отдельные степени достоверности $d_{c_i, j_i, w}$, связанные с компонентами **c**. Функция *a,* называемая *агрегирующей функцией,* должна обладать по крайней мере следующими свойствами:

(a1) *a непрерывна;*

(a2) *a симметрична,* т. е. $a(x, y)=a(y, x)$;

(a3) *a ассоциативна* с точки зрения композиции, т. е. $a(x, a(y, z))=a(a(x, y),z)$;

(a4) *a монотонно неубывающая* функция, т. е. если $y>z$, то $a(x, y)\geq a(x, z)$.

Понятно, что класс функций, удовлетворяющих этим требованиям, бесконечен, но в задачи этой работы не входит изучение этого класса. Простыми примерами агрегирующих функций являются функции $a(x, y)=xy$ и $a(x, y)=\min(x, y)$.

После вычисления агрегированных степеней достоверности для всех элементов данных ($d_{c,w}$ идентифицируется значением параметра **w** и состоянием $\mathbf{c} \in \mathbf{C}$) эти значения можно суммировать для каждого **c**. В результате будут получены числа, аналогичные частотам $N(\mathbf{c})$. Целесообразно назвать их *псевдочастотами* (поскольку они не обязательно являются целыми числами) и обозначить $N(\mathbf{c})$, как и настоящие частоты. Таким образом, для нечетких данных $N(\mathbf{c})$ вычисляется по формуле

$$N(\mathbf{c}) = \sum_{\mathbf{w}} d_{c, w},$$

в которой суммирование осуществляется по всем содержательным выборкам данных, причем каждая выборка соответствует определенному значению параметра **w**, однозначно определяющему положение справочника используемой маски на параметрическом множестве. После вычисления псевдочастот их можно использовать вместо частот при вычислении распределений вероятностей или возможностей, применяя соответственно формулы (31) или (33) или какие-либо другие подходящие правила вычисления.

Из этого небольшого обзора видно, что хотя значения $N(\mathbf{c})$ для конкретных данных и заданной маски и определяются однозначно, существует бесконечно много способов вычисления распределения вероятностей или возможностей состояний ($\mathbf{c} \in \mathbf{C}$) по частотам или псевдочастотам. Поэтому важно, чтобы ФРИЗ разрешал пользователю самому определить то, как следует использовать частоты или псевдочастоты. Если выбор не определен, ФРИЗ должен предоставлять меню возможностей, предпочтительно тех, которые наиболее часто используются. Если пользователь не высказал своего предпочтения, ФРИЗ должен применить некий стандартный способ вычисления, например (31) или (33) соответственно для вероятностей и возможностей, и некую стандартную операцию агрегирования для нечетких данных (например, операцию взятия минимума или произведение). Наиболее типичные варианты демонстрируются в этой работе на разнообразных примерах.

Помимо реализации разных вариантов вычисления распределения вероятностей или возможностей, необходимо также позволить пользователю включать в вычисление дополнительную информацию, связанную с ограничениями на переменные. Будем эту информацию, не входящую в собственно данные, называть *дополнительной (back ground).* Она может принимать самые разные формы, и некоторые из них будут продемонстрированы в примерах.

Пример 2. Данные по 12 переменным были собраны в 1976 г. для группы из 200 государственных служащих, работающих в одной организации. Эти данные были использованы в исследовании, целью которого было вскрытие возможных несправедливостей. В данном примере рассматриваются только пять из 12 переменных; они приводятся в табл. 2.

Таблица 2.

Определение переменных из примера 2

Свойство	Переменная	Состояния переменной
Год рождения	v_1	1: 1930 или раньше 2: 1931–1945 3: 1946 или позже
Пол	v_2	1: мужской 2: женский
Год поступления на службу	v_3	1: 1960 или раньше 2: 1961–1970 3: 1971 или позже
Суммарная оценка работы	v_4	1: 20% или менее 2: 21%–40% 3: 41%–60% 4: 61%–80% 5: 81% или более
Средняя почасовая оплата		1. $4.99 или менее 2: $5.00–$9.99 3 $10.00–$14.99 4. $15.00 или более

Параметрическое множество (группа из 200 служащих) не упорядочено, и, следовательно, можно использовать только маску без памяти.

Пользователь принял решение описать ограничение на переменные с помощью функции распределения возможностей и выбрал формулу (33) для вычисления степеней возможностей по зафиксированным частотам отдельных состояний. Однако кроме этого он захотел использовать имеющуюся дополнительную информацию об ограничениях на переменные, обусловленные правовыми и другими нормами. Сделано это было с помощью выделения трех типов состояний:

а) наблюдаемые состояния; степени возможности этих состояний вычисляются по формуле (3);

б) состояния, которые невозможны в силу дополнительной информации (запрещены законом или другими нормативами); понятно, что для этих состояний f_B (**c**)=0;

в) состояния, которые не наблюдались, но в принципе возможны; целесообразно приписать этим состояниям ненулевую степень возможности, меньшую, чем минимальная степень возможности, вычисленная для наблюдаемых состояний, т. е. выбрать некое значение из интервала

$$0 < f_B(\mathbf{c}) < \min_{\alpha} f_B(\alpha),$$

где α — это индексы наблюдаемых состояний; в данном исследовании было принято решение воспользоваться значением

$$f_B(\mathbf{c}) = \frac{1}{2} \min_{\alpha} f_B(\alpha).$$

Все возможные согласно дополнительной информации состояния с перечислены в табл. 3, в которой также приведены их частоты *N (***c***)* и степени возможности f_B (**c**).

Таблица 3.

Возможностная функция пвведения из примера 2

c	v_1	v_2	v_3	v_4	v_5	$f_B(c)$	N(c)	c	v_1	v_2	v_3	v_4	v_5	$f_B(c)$	N(c)
1	1	1	1	2	1	0.111	2	40	2	2	2	4	4	0.111	2
2	1	1	1	3	4	0.056	0	41	2	2	2	5	4	0.056	0
3	1	1	2	2	1	0.111	2	42	2	2	3	2	2	0.056	0
4	1	1	2	4	1	0.111	2	43	2	2	3	2	3	0.056	0
5	1	1	3	3	1	0.056	0	44	2	2	3	2	4	0.111	2
6	1	1	3	4	1	0.111	2	45	2	2	3	4	1	0.056	0
7	1	2	1	1	2	0.056	0	46	2	2	3	4	3	0.056	0
8	1	2	1	1	4	0.111	2	47	2	2	3	4	4	0.333	6
9	1	2	1	2	2	0.167	3	48	2	2	3	5	4	0.222	4
10	1	2	1	2	4	0.889	16	49	2	1	3	4	1	0.056	0
11	1	2	1	3	2	0.056	0	50	3	2	1	1	4	0.056	0
12	1	2	1	3	4	0.222	4	51	3	2	1	2	2	0.056	0
13	1	2	1	4	1	0.056	0	52	3	2	1	2	4	0.167	3
14	1	2	1	4	4	0.778	14	53	3	2	1	3	4	0.167	3
15	1	2	1	5	4	0.222	4	54	3	2	1	4	2	0.056	0
16	1	2	2	3	4	0.056	0	55	3	2	1	4	4	0.333	6
17	1	2	2	4	4	0.167	3	56	3	2	1	5	4	0.056	0
18	1	2	2	5	4	0.056	0	57	3	2	2	1	4	0.111	2
19	1	2	3	2	4	0.056	0	58	3	2	2	2	2	0.222	4
20	1	2	3	3	1	0.056	0	59	3	2	2	2	4	0.667	12
21	1	2	3	3	4	0.056	0	60	3	2	2	3	2	0.056	0
22	1	2	3	4	4	0.056	0	61	3	2	2	3	4	0.333	6
23	1	2	3	5	4	0.056	0	62	3	2	2	4	2	0.222	4
24	2	1	3	2	1	0.056	0	63	3	2	2	4	3	0.056	0
25	2	1	3	4	1	0.056	0	64	3	2	2	4	4	0.167	3
26	2	1	3	4	2	0.111	2	65	3	2	2	5	4	0.111	2
27	2	2	1	1	2	0.056	0	66	3	2	3	1	1	0.056	0
28	2	2	1	1	4	0.222	4	67	3	2	3	1	2	0.111	2
29	2	2	1	2	1	0.111	2	68	3	2	3	1	4	0.111	2
30	2	2	1	2	4	1.000	18	69	3	2	3	2	1	0.167	3
31	2	2	1	3	4	0.167	3	70	3	2	3	2	2	0.167	3
32	2	2	1	4	2	0.056	0	71	3	2	3	2	4	0.556	10
33	2	2	1	4	4	0.667	12	72	3	2	3	3	2	0.222	4
34	2	2	1	5	4	0.167	3	73	3	2	3	3	4	0.167	3
35	2	2	2	1	2	0.056	0	74	3	2	3	4	2	0.056	0
36	2	2	2	2	2	0.056	0	75	3	2	3	4	4	0.500	9
37	2	2	2	2	4	0.222	4	76	3	2	3	5	1	0.056	0
38	2	2	2	3	2	0.056	0	77	3	2	3	5	4	0.167	3
39	2	2	2	3	4	0.056	0								

Понятно, что $N(\mathbf{c})\neq 0$ только для тех состояний, которые наблюдались. Предположим теперь, что параметрическое множество полностью упорядочено. В этом случае из одной и той же системы данных можно получить множество систем ИИ с поведением, отличающимся масками. Если для заданных данных они определены достаточно корректно, то они одинаково хорошо отвечают требованию согласованности. Точнее, выражение «достаточно корректно»

означает, что функция поведения хорошо согласуется с данными (и, возможно, с некоторой дополнительной информацией) с точки зрения маски и типа выбранных ограничений.

Как уже объяснялось выше, для маски без памяти функцию поведения, хорошо согласующуюся с данными и дополнительной информацией, можно получить из частот состояний (т. е. соответствующих выборочных переменных) для данных, отображенных с помощью рассматриваемой маски. Всякая маска представляет собой некоторое окно, через которое отбираются рассматриваемые данные из матрицы данных (или из массива более высокого порядка). При движении этого окна вдоль всей матрицы данных частоты состояний соответствующих выборочных переменных определяются подсчетом того, как часто наблюдается каждое состояние. Если все выборочные позиции перебираются, то направление движения маски по матрице данных не имеет значения, однако удобнее осуществлять это движение в соответствии с установленным на параметрическом множестве порядком (слева направо или наоборот).

Для конкретных целей одни маски могут подходить лучше, чем другие, но никакая маска не является правильной или неправильной. Эта мысль очень хорошо выражена Дж. Кейсом в книге «В *эту игру могут играть только двое» (Only Two Can Play This Game, Bantam Books, N. Y.,* 1974, p. 99; первый раз эта книга вышла в издательстве *Julian Press, N.Y.,* 1972; Джеймс Кейс — это псевдоним Дж. Сп. Брауна):

«Вы можете смотреть на мир как угодно, выбрать любое окно. Даже не обязательно, чтобы это было всегда одно и то же окно. При этом, естественно, мир, то, что вы видите и что не видите, и угол, под которым вы смотрите, зависит от того, каким окном вы пользуетесь. Но может ли быть окно правильным или неправильным? Окно есть окно ... очень полезно попытаться заглянуть и в другое окно, если то, что вы видите в вашем, выглядит бессмысленным или недостаточным. Если вид из окна вас устраивает, то, естественно, менять окно не нужно. Кроме того, при смене окна вам необходимо некоторое время, чтобы настроиться на то, что вы видите.»

Если рассматриваемая маска представляет собой один столбец (маска без памяти), то выборки по всем значениям параметра являются полными. Однако, если маска состоит из более чем одного столбца, то некоторые выборки в начале и конце параметрического множества (левый и правый края матрицы данных) окажутся неполными (см. рис. 3 и 4). Точнее, число неполных выборок для каждого края матрицы данных равно числу столбцов в маске минус 1. Число столбцов в маске M будем называть *глубиной маски* и обозначать ΔM. Тогда

$$\Delta M = 1 + \max \rho - \min \rho, \qquad (35)$$

где операторы max и min применены ко всем целым $(v_i, t+\rho) \in M$. Так, например, для маски, определенной на рис. 2, ΔM=4, для масок без памяти ΔM=1.

Есть по крайней мере два соображения, по которым применение масок с большой глубиной в общем случае нежелательно. Во-первых, если маска используется для порождения данных, как это было показано в разд. 4.2, то чем больше ее глубина, тем большее требуется начальное условие. Это, вообще говоря, не желательно. Во-вторых, если маска используется для выборки данных, то число неполных выборок равно 2 (ΔM—1). Это означает, что с ростом глубины маски все меньше имеющихся данных используется для определения функции поведения. Следовательно, с увеличением глубины маски сужается эмпирическая основа, на которой строится функция поведения. Это, разумеется, также нежелательно. Оба эти соображения, а также практические соображения, связанные со сложностью вычислений, приводят к тому, что глубина маски обычно выбирается не очень большой. Таким образом, представляется целесообразным определить ограниченность глубины маски как требование 1 для рассматриваемого типа задач.

Для заданной системы данных, дополнительной информации (если она доступна), маски и описания ограничений на выборочные переменные функция поведения однозначно определяется описанной выше процедурой выборки. Эта однозначность является прямым следствием требования соответствия. Для каждой конкретной задачи система данных и дополнительная информация фиксированы. Предположим, что тип ограничения как методологическое отличие также фиксирован. Тогда системы ИИ с поведением, удовлетворяющие требованию соответствия (т. е. претенденты для решающего множества Y_Q), однозначно определяются (и отличаются одна от другой) своими масками. Сведение множества Y к подмножеству Y_r в соответствии с требованием 1 может быть представлено как ограничение на множество допустимых масок. Как уже говорилось выше, желательно ограничить глубину маски. Это можно сделать, определив *наибольшую допустимую маску,* скажем маску **M** как декартово произведение

$$\mathbf{M}=V\times R$$

где $R=\{(t+\rho)|\rho_1 \le \rho \le \rho_2\}$.

Подобная маска может быть представлена в виде полной матрицы с n строками и $1+\rho_2-\rho_1\ (=\Delta M)$ столбцами. Будем называть ее **M**-*матрицей.* Если пользователь задает только Δ**M**, но не конкретные значения ρ_1 и ρ_2, то ФРИЗ выбирает для них некие стандартные значения, например $\rho_2 = 0$, а $\rho_1 = 1-\Delta\mathbf{M}$.

При заданной наибольшей допустимой маске **M** все ее *содержательные подмаски* образуют ограниченное множество Y_r систем с

поведением. Термин «содержательная подмаска» характеризует подмаски **M**, удовлетворяющие следующим требованиям:

(ml) в подмаску входит по крайней мере один элемент из каждой подмаски $\mathbf{M}_i$, определенной уравнением (8) (т. е. один элемент из каждой строки **M**-матрицы);

(m2) в подмаску должен быть включен по крайней мере один элемент с правилом сдвига $t+\rho_2$ (крайний правый элемент из **M**-маски).

Требование ml необходимо для покрытия заданной системы данных, т. е. для того, чтобы гарантировать, что любая базовая переменная из заданной системы данных была бы включена в любую из систем ИИ с поведением из ограниченного множества Y_r. Требование m2 препятствует дублированию эквивалентных подмасок, т. е. подмасок, преобразуемых одна в другую только с помощью добавления константы к правилу сдвига $t+\rho$ (сдвиг ряда в **M**-маске).

Можно легко получить формулу для числа $N(n, \Delta\mathbf{M})$ содержательных подмасок наибольшей допустимой маски, где n — число базовых переменных, а Δ**M** — глубина маски **M**:

$$N(n, \Delta\mathbf{M}) = (2^{\Delta\mathbf{M}} - 1)^n - (2^{\Delta\mathbf{M}-1} - 1)^n. \quad (36)$$

Первый член выражения (36) задает число подмасок **M**, удовлетворяющих условию ml, а второй член — число масок, нарушающих условие m2. В табл. 4 приведены значения $N(n, \Delta\mathbf{M})$ при n, $\Delta\mathbf{M} \le 10$.

Таблица 4.

Число содержательных масок $N(n, \Delta\mathbf{M})$, вычисленное по формуле (36)

n \ ΔM	1	2	3	4	5	6	7	8	9	10
1	1	2	4	8	16	32	64	128	256	512
2	1	8	40	176	736	3,008	12,160	48,896	196,096	785,408
3	1	26	316	3,032	26,416	220,256	1.8×10^6	1.5×10^7	1.2×10^8	$\sim10^9$
4	1	80	2,320	48,224	872,896	1.5×10^7	$\sim2.4\times10^8$	$\sim10^9$	$\sim10^{11}$	$\sim10^{12}$
5	1	242	16,564	742,568	$\sim2.7\times10^7$	$\sim9.6\times10^8$	$\sim10^{10}$	$\sim10^{12}$	$\sim10^{13}$	$\sim10^{15}$
6	1	728	116,920	$\sim1.1\times10^7$	$\sim8.8\times10^8$	$\sim10^{11}$	$\sim10^{12}$	$\sim10^{14}$	$\sim10^{16}$	$\sim10^{18}$
7	1	2,186	821,356	$\sim1.7\times10^8$	$\sim10^{10}$	$\sim10^{12}$	$\sim10^{15}$	$\sim10^{17}$	$\sim10^{19}$	$\sim10^{21}$
8	1	6,560	$\sim5.8\times10^6$	$\sim10^9$	$\sim10^{12}$	$\sim10^{14}$	$\sim10^{17}$	$\sim10^{19}$	$\sim10^{21}$	$\sim10^{24}$
9	1	19,682	$\sim4\times10^7$	$\sim10^{11}$	$\sim10^{13}$	$\sim10^{16}$	$\sim10^{19}$	$\sim10^{21}$	$\sim10^{24}$	$\sim10^{27}$
10	1	59,048	$\sim2.8\times10^8$	$\sim10^{12}$	$\sim10^{15}$	$\sim10^{18}$	$\sim10^{21}$	$\sim10^{24}$	$\sim10^{27}$	$\sim10^{30}$

Эта таблица разделена на три области, для которых размеры наибольшей допустимой маски представляются: а) легко поддающимися вычислительной обработке (левая верхняя область);

б) в принципе поддающимися обработке, что потребует длительной работы мощного компьютера (средняя область);

в) не поддающимися вычислительной обработке (правая нижняя область). Разумеется, эти области показаны только для наиболее типичного случая. Они определяются, по крайней мере, до некоторой степени тем, какими вычислительными мощностями располагает исследователь. Так, например, если он имеет в своем распоряжении мощную систему параллельных вычислений, то область случаев, поддающихся вычислительной обработке, может быть расширена почти вдвое.

Если число содержательных масок оказывается слишком велико, чтобы поддаваться вычислительной обработке, ФРИЗ должен предложить пользователю меню имеющихся дополнительных ограничений, накладываемых на наибольшую допустимую маску. Такими ограничениями могут быть, например, следующие:

фиксация множества порождаемых выборочных переменных;

фиксация числа выборочных переменных;

фиксация верхней границы числа выборочных переменных;

ограничение, при котором рассматриваются только маски без пропусков (примером пропуска является элемент, идентифицируемый координатами i=4, ρ = — 1 в маске, изображенной на рис. 2,а).

Подобные ограничения или их комбинации существенно сокращают множество Y_r, и таким образом, увеличивают размер наибольших допустимых масок, поддающихся вычислительной обработке.

Несмотря на то, что эти ограничения играют важную роль при уменьшении вычислительной сложности, особенно в тех случаях, когда нежелательно изменять наибольшую допустимую маску, любое такое ограничение деформирует исходную задачу. Даже в том случае, если это ограничение подтверждается некими контекстными соображениями, его следует использовать для уменьшения сложности вычислений только в самом крайней случае.

В дальнейшем будут рассматриваться обобщенные задачи, для которых ограниченное множество Y_r состоит из систем ИИ с поведением — по одной системе для каждой содержательной подмаски выбранной наибольшей допустимой маски (предполагается, что она находится в области масок, поддающихся вычислительной обработке). Как уже объяснялось выше, любая такая система определяется таким образом, что она хорошо согласуется с заданной системой данных и дополнительной информацией по маске и принятому типу описания ограничений. Таким образом, требование согласованности имеет более высокий приоритет, чем остальные требования, как это и требуется в п. 4) формулировки задачи. Теперь остается только применить условия 3) и 4) обычно называемые *условием детерминированности и условием*

простоты, для вывода подмножества решений Y_Q ограниченного множества Y_r.

Несмотря на то, что в формулировках конкретных типов интеллектуальных задач возникают и дополнительные требования, условия детерминированности и простоты имеют всеобщее значение. Поэтому обычно они не опускаются. Часто сначала определяется множество решений, удовлетворяющих этим условиям (и, разумеется, условию согласованности), а затем входящие в это множество системы ИИ с поведением изучаются исследователем. Он может использовать их в качестве вспомогательного представления базовых переменных. Однако, если необходимо дальнейшее сокращение множества решений, исследователь производит их оценку и сравнение согласно некоторым вспомогательным критериям. Эти критерии могут определяться как контекстом, так и вкусами исследователя.

4.5. Меры нечеткости

Ясно, что степень недетерминированности должна измеряться обобщенной нечеткостью, сопутствующей порождению данных. А значит, она должна быть определена через порождающие функции поведения f_{GB} и f_{GB} для нейтральных и направленных систем ИИ с поведением (см. соответственно уравнения (22) и (28)). Если эти функции представляют собой функции распределения вероятностей, то мера обобщенной нечеткости хорошо известна — это *шенноновская энтропия,* введенная. К. Шенноном в 1948 г.

Обозначим через P множество всех распределений вероятностей, которые могут быть определены на конечных множествах альтернативных (взаимно исключающих) выходов. Тогда *вероятностная мера нечеткости* — это функция

$$H{:}P\rightarrow[0,\infty],$$

обладающая некоторыми свойствами. Следующие свойства являются необходимыми свойствами любой содержательной меры нечеткости :

(Н1) *симметричность* — нечеткость инвариантна относительно перестановки вероятностей;

(Н2) *расширяемость* — нечеткость не меняется при добавлении к рассматриваемому множеству выходов выходов с нулевой вероятностью;

(Н3) *квазиаддитивность* — нечеткость совместного распределения вероятностей не больше суммы нечеткостей соответствующих безусловных распределений его компонентов;

(Н4) *аддитвность* — для распределений вероятностей любых двух независимых множеств выходов нечеткость совместного

распределения вероятностей равна сумме нечеткостей отдельных распределений вероятностей;
(H5) *непрерывность* — нечеткость — это непрерывная функция на всех своих аргументах.
Известно, что только функции вида

$$H\ (f\ (x)\ |\ x \in X) = -a \sum_{x \in X} f\ (x) \log_b f\ (x)$$

обладают свойствами (H1)— (H5);
$(f(x)|x \in X) \in P$
— это распределение вероятностей для определенного конечного множества X альтернативных выходов x, где a — произвольная положительная константа, а b — произвольное основание логарифмов.
Если потребовать выполнения еще и *нормализующего свойства*

$$H(0.5;\ 0.5) = 1$$

(нечеткость двух равновероятных выходов равна 1), то мера нечеткости определяется однозначно:

$$H\ (f\ (x)\ |\ x \in X) = -\sum_{x \in X} f\ (x) \log_2 f\ (x). \quad (37)$$

Обычно функцию (37) называют *шенноновской энтропией.* **Она измеряет нечеткость в единицах, называемых *битами.* Смысл этих единиц ясен, так как любое целое значение нечеткости, измеренное в них, скажем значение *и,* эквивалентно предсказанию значений истинности *и* суждений или *и* двоичных цифр при условии, что они равновероятны.**
Если предположить, что любое конечное множество X рассматриваемых альтернативных выходных значений характеризуется определенным распределением вероятностей, то удобнее упростить обозначения и писать $H(X)$ вместо $H(f(x)\ |x \in X)$.
Легко видеть, что

$$0 \leqslant H\ (X) \leqslant \log_2 |H|. \quad (38)$$

Нижняя граница $H(X)=0$ достигается в том случае, когда вероятности всех выходных значений, за исключением одного, равны 0; верхняя граница достигается тогда, когда вероятности всех событий одинаковы, т. е. равны 1/|X|. Отношение энтропии к ее верхней границе

$$\mathbf{H}\ (X) = H\ (X)/\log_2 |X| \quad (39)$$

называется **нормализованной энтропией**; понятно, что

$$0 \leq \mathbf{H}(X) \leq 1. \quad (40)$$

В нашем случае множествами выходов являются множества **C**, **G**, $\overline{\mathbf{G}}$, **E**, а распределения вероятностей представляются функциями поведения $f_B, f_{GB}, \hat{f}_B, \hat{f}_{GB}$, определяемыми соответственно формулами

(18), (22), (26), (28). Для упрощения записи опустим индексы B и GB, а также знак $\frown$. Таким образом,

$$f(\mathbf{c}),\ f(\mathbf{g}|\overline{\mathbf{g}}),\ f(\overline{\mathbf{e}},\ \mathbf{e}),\ f(\mathbf{g}|\mathbf{e},\ \overline{\mathbf{g}})$$

обозначают вероятности, определяемые соответственно формулами (18), (22), (26), (28); смысл любого из этих обозначений однозначно определяется заключенными в скобки аргументами. Кроме того, определим безусловные вероятности

$$f(\overline{\mathbf{g}}) = \sum_{\mathbf{c} \succ \overline{\mathbf{g}}} f(\mathbf{c}), \quad (41)$$

где $\mathbf{c} \succ \overline{\mathbf{g}}$ указывает на то, что $\overline{\mathbf{g}}$ является подмножеством состояния **c** (подсостоянием **c**); формально, если

$$\mathbf{c}_i' = (\mathbf{c}_k\ |\ k \in N_{|\mathbf{c}|}),$$
$$\overline{\mathbf{g}} = (\overline{\mathbf{g}}_j\ |\ j \in Z,\ Z \subset N_{|\mathbf{c}|}),$$

то $\overline{\mathbf{g}} < \mathbf{c}$ тогда и только тогда, когда $\overline{g}_j = c_j$ для всех $j \in Z$. Для направленных систем ИИ безусловные вероятности вычисляются по немного измененной формуле

$$f(\overline{\mathbf{g}}\ |\ \mathbf{e}) = \sum_{\overline{\mathbf{e}} \succ \overline{\mathbf{g}}} f(\overline{\mathbf{e}}\ |\ \mathbf{e}). \quad (42)$$

Условные вероятности, характеризующие процесс порождения данных, связаны с основными (совместными) и безусловными вероятностями следующим образом:

$$f(\mathbf{g}\ |\ \overline{\mathbf{g}}) = f(\mathbf{c})/f(\overline{\mathbf{g}}); \quad (43)$$

$$f(\mathbf{g}\ |\ \mathbf{e},\ \overline{\mathbf{g}}) = f(\overline{\mathbf{e}}\ |\ \mathbf{e})/f(\overline{\mathbf{g}}\ |\ \mathbf{e}). \quad (44)$$

Первая формула описывает эту связь для нейтральных, а вторая — для направленных систем ИИ.
При заданной порождающей маске для нейтральной системы ИИ, через которую определяются множества состояний **G**, $\overline{\mathbf{G}}$ генерируемых и генерирующих выборочных переменных, *порождающая нечеткость* $H(\mathbf{G}|\overline{\mathbf{G}})$ определяется как средняя нечеткость, базирующаяся на вероятностях $f(\mathbf{g}|\ \overline{\mathbf{g}})$, взвешенных вероятностями $f(\overline{\mathbf{g}})$ порождающих условий:

$$H\ (\mathbf{G}\ |\ \overline{\mathbf{G}}) = -\sum_{\overline{\mathbf{g}} \in \overline{\mathbf{G}}} f\ (\overline{\mathbf{g}}) \sum_{\mathbf{g} \in \mathbf{G}} f\ (\mathbf{g}\ |\ \overline{\mathbf{g}}) \log_2 f\ (\mathbf{g}\ |\ \overline{\mathbf{g}}). \quad (45)$$

Это значение определяет *степень недетерминированности* данной нейтральной порождающей системы ИИ с поведением.

Для направленных систем порождающая нечеткость H(**G**|**E**× $\bar{\mathbf{G}}$)) вычисляется по формуле

(46)

которую можно непосредственно применять в том случае, когда можно и имеет смысл определять вероятности $f(\mathbf{e}|\,\bar{\mathbf{g}})$, т. е. когда направленная система ИИ получена из нейтральной. Если мы не располагаем вероятностями состояний элементов множества **E** или эти вероятности несущественны, тогда в качестве базовых вероятностей берутся вероятности $f(\bar{\mathbf{e}}\,|\mathbf{e})$ [аналог вероятностей *f*(**c**) для нейтральных систем ИИ], исходя из которых вычисляются остальные необходимые вероятности. В этом случае нечеткость H(**G**|**E**× $\bar{\mathbf{G}}$) вычисляется по формуле

$$H(\mathbf{G}\mid \mathbf{E}\times\bar{\mathbf{G}}) = -\frac{1}{|\mathbf{E}|}\sum_{\mathbf{e}\in\mathbf{E}}\sum_{\bar{\mathbf{g}}\in\bar{\mathbf{G}}} f(\bar{\mathbf{g}}\mid \mathbf{e})\sum_{\mathbf{g}\in\mathbf{G}} f(\mathbf{g}\mid \mathbf{e},\mathbf{g})\log_2(\mathbf{g}\mid \mathbf{e},\mathbf{g})$$

(47)

где вероятности $f(\bar{\mathbf{g}}\,|\mathbf{e})$ и $f(\mathbf{g}|\mathbf{e},\,\bar{\mathbf{g}})$ вычисляются по заданным вероятностям $f(\bar{\mathbf{e}}\,|\mathbf{e})$ согласно формулам (42) и (44).
Формулы (45), (46) и (47) можно заменить другими, более удобными для вычисления. Например, уравнение (45) можно модифицировать следующим образом:

$$\begin{aligned}
H(\mathbf{G}\mid\bar{\mathbf{G}}) &= -\sum_{\bar{\mathbf{g}}} f(\mathbf{g}\mid\bar{\mathbf{g}})\log_2 f(\mathbf{g}\mid\bar{\mathbf{g}})\\
&= -\sum_{\bar{\mathbf{g}}}\sum_{\mathbf{g}} f(\bar{\mathbf{g}})\, f(\mathbf{g}\mid\bar{\mathbf{g}})\log_2 f(\mathbf{g}\mid\bar{\mathbf{g}})\\
&= -\sum_{\bar{\mathbf{g}}}\sum_{\mathbf{g}} f(\mathbf{c})\,[\log_2 f(\mathbf{c}) - \log_2 f(\bar{\mathbf{g}})]\\
&= H(\mathbf{C}) + \sum_{\bar{\mathbf{g}}}\sum_{\mathbf{g}} f(\mathbf{c})\log_2 f(\bar{\mathbf{g}})\\
&= H(\mathbf{C}) + \sum_{\bar{\mathbf{g}}}\log_2 f(\bar{\mathbf{g}})\sum_{\mathbf{g}} f(\mathbf{c})\\
&= H(\mathbf{C}) + \sum_{\bar{\mathbf{g}}} f(\bar{\mathbf{g}})\log_2 f(\bar{\mathbf{g}})\\
&= H(\mathbf{C}) - H(\bar{\mathbf{G}})
\end{aligned}$$

Таким образом, H(**G**|$\bar{\mathbf{G}}$) можно вычислить, не используя условные вероятности, по формуле

$$H(\mathbf{G}|\bar{\mathbf{G}}) = H(\mathbf{C}) - H(\bar{\mathbf{G}}). \quad (48)$$

Точно так же уравнения (46) и (47) можно заменить соответственно уравнениями

$$H(\mathbf{G}\mid\mathbf{E}\times\bar{\mathbf{G}}) = H(\mathbf{C}) - H(\mathbf{E}\times\bar{\mathbf{G}}), \quad (49)$$

$$H(\mathbf{G}\mid\mathbf{E}\times\bar{\mathbf{G}}) = \frac{1}{|\mathbf{E}|}\left[\sum_{\mathbf{e}\in\mathbf{E}} H(\bar{\mathbf{E}}\mid\mathbf{e}) - \sum_{\mathbf{e}\in\mathbf{E}} H(\bar{\mathbf{G}}\mid\mathbf{e})\right]. \quad (50)$$

Максимальное значение порождающей нечеткости любого типа равно $\log_2|\mathbf{G}|$; следовательно, нормализованная порождающая нечеткость получается делением порождающей нечеткости на ее максимальное значение. Например,

$$\mathbf{H}(\mathbf{G}|\bar{\mathbf{G}}) = H(\mathbf{G}|\bar{\mathbf{G}})/\log_2|\mathbf{G}|.$$

Пример 3. На рис. *7,а* показана вероятностная функция поведения *f* (**c**) для четырех выборочных переменных s_1, s_2, s_3, s_4, каждая с двумя состояниями — 0 и 1.

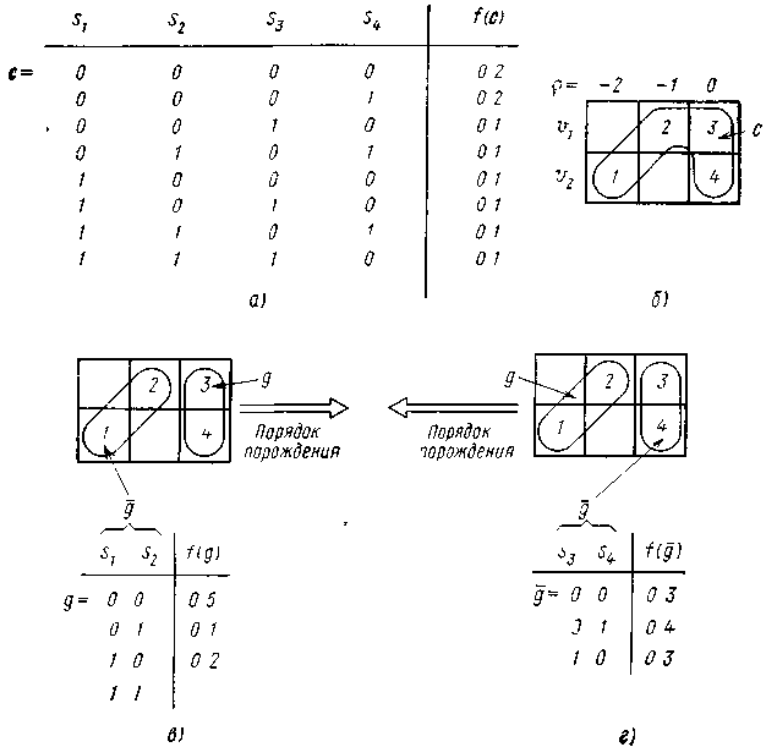


Рис. 7. К примеру 3 вероятностная нейтральная система ИИ

Состояния с нулевой вероятностью в таблице не приводятся. Выборочные переменные определены через две базовые переменные v_1, v_2 с помощью маски, изображенной на рис.7,б. Поскольку выборочные переменные s_2, s_3 суть сдвиги одной и той же базовой переменной v_1, распределения вероятностей их состояний должны быть одинаковы; они и в самом деле одинаковы; оба имеют вероятности 0.7 и 0.3 соответственно для состояний 0 и 1. Аналогично переменные s_1, s_4 (сдвиги v_2) имеют одинаковое распределение вероятностей: 0.6 и 0,4 соответственно для состояний 0 и 1. Следовательно, для данной маски приведенная функция распределения вероятностей является корректной функцией поведения.

Если данная система ИИ интерпретируется как нейтральная, ее порождающая нечеткость $H(\mathbf{G}|\bar{\mathbf{G}})$ может быть вычислена по формуле (48). Для первого члена формулы мы имеем

$$H(\mathbf{C}) = -2\times 0.2\log_2 0.2 - 6\times 0.1\log_2 0.1 = \\ = 0.9288 + 1.9932 = 2.922.$$

Значение второго члена зависит от порядка порождения и от соответствующей маски. На рис. 7,*в* и *г* показаны два возможных порядка порождения. Для порождения слева направо имеем

$$H(\bar{\mathbf{G}}) = -0.5\log_2 0.5 - 0.1\log_2 0.1 - \\ -2\times 0.2\log_2 0.2 = 0.5 + 0.3322 + 0.9288 = 1.761,$$
$$H(\mathbf{G}|\bar{\mathbf{G}}) = H(\mathbf{C}) - H(\bar{\mathbf{G}}) = 2.922 - 1.761 = 1.161.$$

Для другого порядка порождения (рис. 7,г) мы получим

$$H(\bar{\mathbf{G}}) = -2\times 0.3\log_2 0.3 - 0.4\log_2 0.4 = 1.0422 + 0.5288 = 1.571,$$
$$H(\mathbf{G}|\bar{\mathbf{G}}) = 2.922 - 1.571 = 1.351.$$

Следовательно, нам можно выбрать один из двух порядков порождения; первый порядок предпочтительнее, так как имеет более низкую порождаемую нечеткость. Поскольку в данном примере $\log_2|\mathbf{G}|=2$, то нормализованные значения вычислительных порождающих нечеткостей получаются делением их на два

В некоторых случаях применим только один порядок порождения. Если, например, параметром является время, то в каждом случае имеет смысл только один из порядков, определяемый целью использования системы ИИ с поведением. **Если она используется для *предсказания,* то состояния должны порождаться в порядке возрастания времени (слева направо); если же она используется для *ретроспекции,* то состояния должны порождаться в порядке убывания времени.** В данном примере, если параметром является время, то оказывается легче предсказывать будущие состояния системы ИИ, чем определять прошлые.

Предположим теперь, что v_1 интерпретируется как входная переменная и что по функции поведения на рис. 7,а определена соответствующая направленная система ИИ. Теперь для вычисления порождающей нечеткости можно воспользоваться формулой (49). Нечеткость $H(\mathbf{C})$ уже была вычислена раньше; $H(\mathbf{E}\times\bar{\mathbf{G}})$ зависит от порядка порождения. В любом случае множество **E** представляется состояниями переменных s_2, s_3; $\bar{\mathbf{G}}$ представляется или состояниями s_1 (в порядке возрастания параметра) или состояниями s_4 (в порядке убывания параметра). В первом случае нечеткость $H(\mathbf{E}\times\bar{\mathbf{G}})$ связана с переменными s_1, s_2, s_3:

$$H(\mathbf{E}\times\bar{\mathbf{G}}) = -0.4\log_2 0.4 - 6\times 0.1\log_2 0.1 = \\ = 0.5288 + 1.9932 = 2.522;$$

$$H(\mathbf{G}|\mathbf{E}\times\bar{\mathbf{G}} = H(\mathbf{C}) - H(\mathbf{E}\times\bar{\mathbf{G}}) = 0.922 - 2.522 = 0.4.$$

Во втором случае она представляет нечеткость переменных s_2, s_3, s_4:

$$H(\mathbf{E}\times\bar{\mathbf{G}}) = -0.3\log_2 0.3 - 3\times 0.2\log_2 0.2 - 0.1\log_2 0.1 = \\ = 0.5211 + 1.3932 = 2.2465;$$

$$H(\mathbf{G}|\mathbf{E}\times\bar{\mathbf{G}}) = 2.922 - 2.2465 = 0.6755.$$

Таким образом, снова оказывается, что предсказывать будущие состояния легче, чем определять прошлые.

Предположим теперь, что мы не располагаем никакой информацией относительно входной переменной v_1 или что эта информация несущественна (например, в том случае, когда v_1 контролируется пользователем). Тогда все вычисления должны проводиться для приведенных на рис. 8,а условных вероятностей $f(\bar{\mathbf{e}}\,|\mathbf{e})$. Как показано на этом рисунке, список вероятностей разбит на четыре части, соответствующие разным состояниям **e**. Нечеткости для каждого состояния также приведены на рис. 8,а. Здесь же показано разбиение маски на $\{M_e,\ M_{\bar{e}}\}$.

Ситуация при порождении состояний слева направо, включая значения $H(\bar{\mathbf{G}}\,|\mathbf{e})$ для каждого состояния **e**, показана на рис. 8, в.

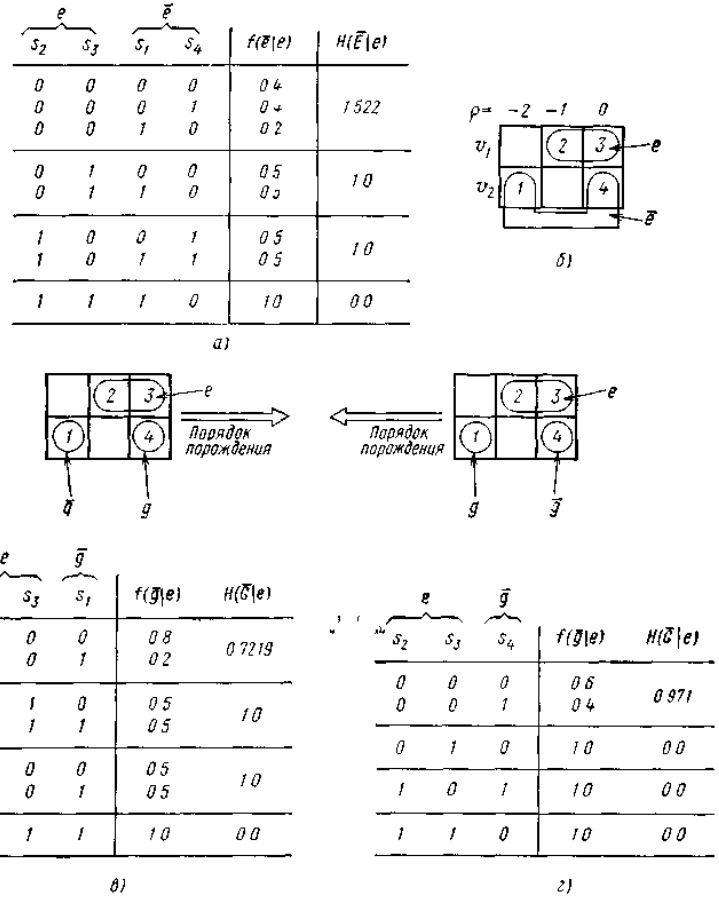


Рис. 8 К примеру 4 вероятностная направленная система ИИ

Из формулы (50) получим

$$H(\mathbf{G}|\mathbf{E}\times\bar{\mathbf{G}}) = 1/4(3.522 - 2.7219) = 0.200025.$$

Другой порядок порождения изображен на рис. 8,г. Для него имеем

$$H(\mathbf{G}|\mathbf{E}\times\bar{\mathbf{G}}) = 1/4(3.522 - 0.971) = 0{,}63775.$$

Рассмотрим теперь порождающую нечеткость в системах ИИ, описываемых с помощью функций распределения возможностей. Пусть П—это множество всех распределений возможностей, имеющих по крайней мере одно ненулевое значение, которые можно определить

на конечных множествах альтернативных выходов (распределения возможностей только с нулевыми значениями для наших целей не подходят) Тогда возможностная мера нечеткости представляет собой функцию

$$U{:}\Pi\to[0,\infty], \qquad (51)$$

обладающую определенными свойствами. Прежде чем рассматривать эти свойства, нужно сначала ввести некоторые понятия, связанные с распределениями возможностей

1. Распределение возможностей, скажем

$$\mathbf{f} = (\varphi_i \mid i \in N_{|\mathbf{X}|}) \in \Pi, \qquad (52)$$

определенное на конечном множестве X альтернативных выходов x называется *нормализованным распределением возможностей* тогда и только тогда, когда

$$\max_i \varphi_i = 1;$$

понятно, что $\varphi_i = f(x)$ для некоторого взаимооднозначного соответствия между $N_{|X|}$ и X.

2 Пусть для любого распределения и возможностей **f**, например для распределения, определенного в (52), и для любого действительного $l \in [0, 1]$

$$c \quad \Pi\times[0,\ 1]\to\mathscr{P}(\mathbf{N}) \qquad (53)$$

такая функция, что

$$c(\mathbf{f}, l) = \{i \in N_{|\mathbf{X}|} \mid \varphi_i \geqslant l\}. \qquad (54)$$

Эта функция называется *функцией l-уровня,* а множество *c(***f***, l)* — множеством *l*-уровня от **f**

3. Для заданного распределения возможностей (52) назовем

$$L_f = \{l \mid (\exists i \in N_{|\mathbf{X}|})(\varphi_i = l) \text{ или } l = 0\} \qquad (55)$$

уровневым множеством для *f.* Обозначим через

$$L_f = \{l_1, l_2, \ldots l_q\}$$

уровневое множество для **f,** где $l_1 = 0$, $q = |L_f|$, причем из $i<j$ следует, что $l_i<l_j$. Пусть для удобства

$$l_f = \max_i \varphi_i.$$

Понятно, что $l_f = l_q \in L_f$. Кроме того, $l_f=1$ тогда и только тогда, когда **f** является нормализованным распределением возможностей.

4. Для любого m $\in$ **N** пусть

$$^1\mathbf{f} = (^1\varphi_i \mid i \in N_m) \in \Pi,$$

$$^2\mathbf{f} = (^2\varphi_i \mid i \in N_m) \in \Pi,$$

два распределения возможностей Тогда ^{1}**f** называется субраспределением ^{2}**f** тогда и только тогда, когда для любого $i \in N_m$

$$\max_i {}^1\varphi_i = \max_i {}^2\varphi_i \quad \text{и} \quad {}^1\varphi_i \leqslant {}^2\varphi_i.$$

Пусть ^{1}**f**$\leq^2$**f** означает, что ^{1}**f** является субраспределением ^{2}**f** . Понятно, что отношение «^{1}**f** субраспределение ^{2}**f**» представляет собой частичное упорядочение, определенное на любом множестве распределений возможностей с числом элементов, равным *m.* Обозначим это множество $^m\Pi$. Далее, ($^m\Pi$, $\leq$) —это решетка с объединением и перечислением, определяемыми соответственно как

$$^1\mathbf{f} \vee {}^2\mathbf{f} = (\max[{}^1\varphi_i,\ {}^2\varphi_i] \mid i \in N_m),$$

$$^1\mathbf{f} \wedge {}^2\mathbf{f} = (\min[{}^1\varphi_i,\ {}^2\varphi_i] \mid i \in N_m)$$

для любых ^{1}**f,** ^{2}**f** $\in {}^m\Pi$.

Теперь, располагая определенными понятиями, связанными с распределениями возможностей, мы можем вернуться к обсуждению главного вопроса — **о мере возможностной нечеткости**. Хотелось бы, чтобы возможностные аналоги свойств (H1) — (H5), которыми обладает шенноновская энтропия, также выполнялись бы и для возможностной меры нечеткости. Возможностные аналоги этих свойств можно сформулировать точно так же, как (H1) — (H5), за тем только исключением, что слово «вероятность» нужно везде заменить на слово «возможность». Функция вида (51), удовлетворяющая этим свойствам, известна. Ее можно представить в виде

$$U(\mathbf{f}) = \frac{1}{l_f} \sum_{k=1}^{q-1} (l_{k+1} - l_k) \log_2 \mid c(\mathbf{f}, l_{k+1}) \mid, \qquad (56)$$

или в более простом виде

$$U(\mathbf{f}) = \frac{1}{l_f} \int_0^{l_f} \log_2 \mid c(\mathbf{f}, l) \mid \mathrm{d}l. \qquad (57)$$

Эта функция называется *U-нечеткостью.* Помимо возможностных аналогов свойств (H1) — (H5), *U*-нечеткость обладает некоторыми другими полезными свойствами. Важнейшим из них является *монотонность:* для любых ^{1}**f,** ^{2}**f** $\in {}^m\Pi$ *(*$m \in N$*),* если ^{1}**f**$\leq^2$**f**, то $U(^1\mathbf{f}) \leq U(^2\mathbf{f})$.

Пример 4. Вычислим *U*-нечеткости для следующих распределений нечеткостей:

$$^1\mathbf{f}(0.1,\ 0,\ 0.5,\ 0.8,\ 0.8,\ 0.8,\ 0.1,\ 0.7,\ 0.8),$$

$$^2\mathbf{f}(0.3,\ 0.2,\ 0.9,\ 1,\ 1,\ 1,\ 0.9,\ 0.8,\ 1).$$

Вычислим соответствующие уровневые множества:

$$L_{x_f} = \{0,\ 0.1,\ 0.5,\ 0.7,\ 0.8\},$$
$$L_{x_f} = \{0,\ 0.2,\ 0.3,\ 0.8,\ 0.9,\ 1\}.$$

Используя уравнения (56) или (57), получаем:

$$U(^1f) = \frac{1}{0,8}(0.1\log_2 8 + 0,4\log_2 6 + 0,2\log_2 5 + 0,1\log_2 4) =$$
$$= 1.25\,(0,3 + 1,034 + 0,464 + 0.2) = 2.4975;$$
$$U(^2f) = 0.2\log_2 9 + 0.1\log_2 8 + 0.5\log_2 7 + 0.1\log_2 6 +$$
$$+ 0.1\log_2 4 = 0.634 + 0.3 + 1.404 + 0.258 + 0.2 = 2.796.$$

Как и в случае с распределениями вероятностей, целесообразно допустить, что любое рассматриваемое конечное множество альтернативных выходов характеризуется определенным (уникальным) распределением возможностей. Затем для того, чтобы обратить внимание на множество, для которого вычисляется нечеткость, будем вместо записи *U(f (x) |x∈ X)* использовать сокращенную запись *U (X)*. Легко показать, что

$$0 \leqslant U(X) \leqslant \log_2 |X|, \quad (58)$$

аналогично (38). Нижняя граница достигается в том случае, когда распределение возможности всех выходов, за исключением одного, равны 0; верхняя граница достигается, когда возможности всех выходов равны между собой (и не равны нулю).

Отношение

$$\mathbf{U}(X) = U(X)/\log_2 |X| \quad (59)$$

— это *нормализованная U-нечеткость,* для которой

$$0 \leqslant \mathbf{U}(X) \leqslant 1. \quad (60)$$

Известно также, что условная *U*-нечеткость *U(Y|X)* может быть вычислена без использования условных возможностей по формуле

$$U(X|Y) = U(X \times Y) - U(Y), \quad (61)$$

что является точным аналогом шенноновской энтропии. Эта условная *U*-нечеткость нормализуется делением на $\log_2|X|$.

Подобно случаю с распределениями вероятностей, обозначим через

$$f(\mathbf{c}), \quad f(\mathbf{g} \mid \bar{\mathbf{g}}), \quad f(\bar{\mathbf{e}} \mid \mathbf{e}), \quad f(\mathbf{g} \mid \mathbf{e},\ \bar{\mathbf{g}})$$

возможности, определенные соответственно уравнениями (18), (22), (26), (28). Кроме того, безусловные возможности для нейтральных систем ИИ равны

$$f(\bar{\mathbf{g}}) = \max_{\mathbf{c} \succ \bar{\mathbf{g}}} f(\bar{\mathbf{e}} \mid \mathbf{e}), \quad (62)$$

а для направленных систем ИИ

$$f(\bar{\mathbf{g}} \mid \mathbf{e}) = \max_{\bar{\mathbf{e}} \succ \bar{\mathbf{g}}} f(\bar{\mathbf{e}} \mid \mathbf{e}). \quad (63)$$

Определение условных возможностей через совместные и безусловные возможности является спорным вопросом теории возможностей. К счастью, можно избежать использования условных возможностей, если применить формулу (61). Возможпостные аналоги ключевых формул (48), (49), (50) будут иметь соответственно следующий вид:

$$U(\mathbf{G} \mid \bar{\mathbf{G}}) = U(\mathbf{C}) - U(\bar{\mathbf{G}}); \quad (3.64)$$

$$U(\mathbf{G} \mid \mathbf{E} \times \bar{\mathbf{G}}) = U(\mathbf{C}) - U(\mathbf{E} \times \bar{\mathbf{G}}); \quad (3.65)$$

$$U(\mathbf{G} \mid \mathbf{E} \times \bar{\mathbf{G}}) = \frac{1}{|\mathbf{E}|}\left[\sum_{\mathbf{e} \in \mathbf{E}} U(\bar{\mathbf{E}} \mid \mathbf{e}) - \sum_{\mathbf{e} \in \mathbf{E}} U(\bar{\mathbf{G}} \mid \mathbf{e})\right]. \quad (66)$$

При использовании этих формул не нужно вычислять условные возможности. Условные возможности $f(\bar{\mathbf{e}}\,|\mathbf{e})$, необходимые для вычисления $U(\bar{\mathbf{E}}\,|\mathbf{e})$, заданы (или определяются непосредственно из данных), а не вычисляются из совместных или безусловных возможностей; возможности $f(\bar{\mathbf{g}}\,|\mathbf{e})$ — это маргиналы от $f(\bar{\mathbf{e}}\,|\mathbf{e})$, вычисленные по формуле (63).

Пример 5. Рассмотрим ситуацию, представленную на рис. 9, где $f(\mathbf{c})$ —это возможность состояния **c**.

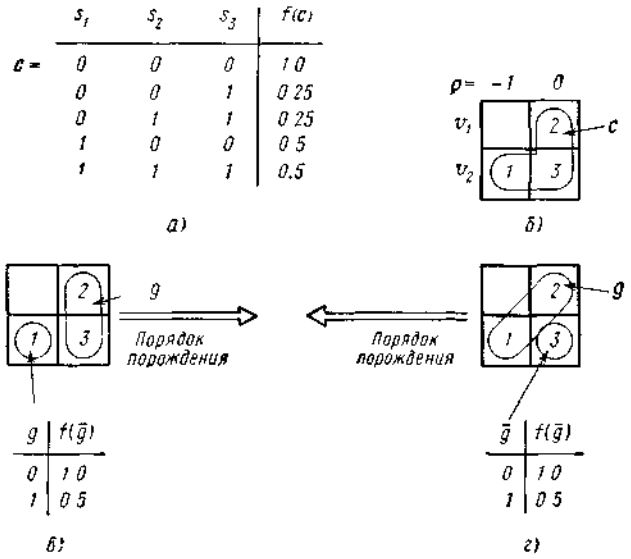


Рис. 9. К примеру 5 возможностная нейтральная система

Функция f(**c**) одинакова для обоих порядков порождения. Из формулы (64) имеем

$$U(\mathbf{C}) = 0.25\log_2 5 + 0.25\log_2 3 + 0.5\log_2 1 = 0.58 + 0.396 = 0.976;$$

$$U(\bar{\mathbf{G}}) = 0.5\log_2 2 + 0\,5\log_2 1 = 0.5;$$

$$U(\mathbf{G}|\bar{\mathbf{G}}) = U(\mathbf{C}) - U(\bar{\mathbf{G}}) = 0.976 - 0.5 = 0.476.$$

Будем теперь рассматривать переменную v_i как входную, а следующие возможности $f(\bar{\mathbf{e}}\,|\mathbf{e})$ заданными:

s_1	s_2	s_3	$f(\bar{\mathbf{e}}\,\|\,\mathbf{e})$
0	0	0	1,0
0	0	1	0,25
1	0	0	0,5
0	1	1	0,5
1	1	1	1,0

При этом $U(\mathbf{G}|\mathbf{E}\times\bar{\mathbf{G}})$ вычисляется по формуле (66). Тогда при порядке порождения слева направо и наоборот имеем соответственно

s_1	s_2	$f(\bar{\mathbf{g}}\,\|\,\mathbf{e})$
0	0	1,0
1	0	0,5
0	1	0,5
1	1	1,0

s_2	s_3	$f(\bar{\mathbf{g}}\,\|\,\mathbf{e})$
0	0	1,0
0	1	0,25
1	0	0,0
1	1	1,0

Для первого порядка порождения получаем

$$U(\mathbf{G}|\mathbf{E}\times\bar{\mathbf{G}}) = 1/2(1.146 - 1) = 0.073;$$

для второго порядка порождения

$$U(\mathbf{G}|\mathbf{E}\times\bar{\mathbf{G}}) = 1/2(1.146 - 0.25) = 0.448.$$

Хорошо известно, что средняя нечеткость, связанная с конечным набором альтернативных выходов и оцениваемая как шенноновская энтропия или как U-нечеткость, может также рассматриваться как средний объем информации, получаемый в результате эксперимента с этим набором. **Однако, если мера нечеткости принимается в качестве меры информации, фактически оцениваются только синтаксические аспекты информации, но не ее семантические или прагматические аспекты.**

4.6. Определение систем ИИ с поведением

Располагая теперь мерами нечеткости, через которые выражается степень детерминированности, вернемся в данном разделе к типу интеллектуальных задач, введенному в разд. 4.4: **дана система данных D с полностью упорядоченным параметрическим множеством и с наибольшей допустимой маской M, совместимой с D; требуется определить все системы ИИ с поведением, удовлетворяющие требованиям согласованности, детерминированности и простоты, причем требование согласованности более приоритетно, чем остальные два.**

Как уже говорилось выше, любая наибольшая допустимая маска **M** содержит набор корректных масок, каждая из которых является подмножеством **M**. Для каждой маски может быть определена функция поведения (определенного выбранного типа), хорошо согласующаяся с данными, с помощью разреженной выборки данных. Однако на практике достаточно провести выборку только для маски **M**. Функции поведения для ее подмасок могут быть получены вычислением подходящих проекций функции поведения соответствующей маски **M**.

Для заданной функции f_B , определенной через полные состояния неких выборочных переменных, любая из ее проекций также является функцией поведения, соответствующей f_B в смысле субсостояний, основанных на определенном подмножестве выборочных переменных. Пусть $s_k(k \in N_{|M|})$—выборочные переменные, через которые определяются состояния f_B ; M — маска, через которую выбираются значения выборочных переменных. Пусть $[f_B \downarrow Z]$—проекция f_B , где подмножество множества $N_{|M|}$ идентификаторов выборочных переменных, т. е. $Z \subset N$. Тогда

$$[f_B \downarrow Z] : \underset{k \in Z}{\times} S_k \to [0, 1], \quad (67)$$

так что

$$[f_B \downarrow Z](\mathbf{x}) = a(\{f(\mathbf{c}) \mid \mathbf{c} \succ \mathbf{x}\}), \quad (68)$$

где a — некая агрегирующая функция, определяемая характером функции f_B . Например,

$$\sim [f_B \downarrow Z](\mathbf{x}) = \sum_{\mathbf{c} \succ \mathbf{x}} f_B(\mathbf{c}), \quad (69)$$

где f_B—распределение вероятностей; Δ—для распределений возможностей,

$$[f_B \downarrow Z](\mathbf{x}) = \max_{\mathbf{c} \succ \mathbf{x}} f_B(\mathbf{c}). \quad (70)$$

Будем в контексте любой конкретной задачи через ${}^1f_B$ обозначать функцию поведения для наибольшей приемлемой маски **M**. Через ${}^if_B$ (i = 2, 3, ...) будем обозначать функции поведения для ее различных осмысленных подмасок ${}^i\mathbf{M}$, каждая из которых связана с множеством ${}^iZ \subset N_{|M|}$ идентификаторов выборочных переменных.

За исключением очень небольших наборов данных, с точки зрения вычислений проще определять функции поведения с помощью проекций, а не через выборки данных. Чем больше объем данных, тем больше вычислительного времени экономится, если вместо выборок данных использовать проекции. Таким образом, лучше производить выборку только однажды для наибольшей приемлемой маски, а затем определять функции поведения для всех содержательных подмасок как соответствующие проекции.

Пример 6. Определим проекцию вероятностной функции поведения, приведенной на рис. 7,а и возможностной функции поведения, приведенной на рис. 9,а, для Z={1, 2}. Применив формулу (69) для вероятностной функции, получим:

	s_1	s_2	$[f \downarrow \{1, 2_i\}](\mathbf{x})$
$\mathbf{x}=$	0	0	0.5 (=0.2+0.2+1)
	0	1	0.1
	1	0	0.2 (=0.1+0.1)
	1	1	0.2 (=0.1+0.1)

Для возможностной функции по формуле (70) имеем

	s_1	s_2	$[f \downarrow \{1,2\}](\mathbf{x})$
$\mathbf{x}=$	0	0	1.0
	0	1	0.25
	1	0	0.5
	1	1	0.5

Для заданной системы данных **D** и наибольшей допустимой маски **M** требование соответствия приводит к ограниченному множеству

$$Y_r = \{{}^i\mathbf{F}_B = (\mathbf{I}, {}^i\mathbf{M}, {}^if) \mid i = 1, 2, \ldots, (N(n, \Delta\mathbf{M})\},$$

содержащему по одной системе с поведением для каждой осмысленной маски ${}^i\mathbf{M} \subseteq \mathbf{M}$; пусть для удобства ${}^1\mathbf{M} = \mathbf{M}$. **Следующим шагом решения рассматриваемой задачи должно быть вычисление степеней недетерминированности и сложности для всех систем из множества** Y_r.

Как было показано в разд. 4.5, *степень недетерминированности* задается соответствующей мерой *порождающей нечеткости,* определяемой для вероятностных систем ИИ шенноновской энтропией, а для возможностных систем ИИ *U*-нечеткостью. Для определения порождающей нечеткости требуется, чтобы был определен порядок порождения (и соответствующее разбиение любой маски). Если допускается несколько порядков порождения, то для каждой маски мы берем порядок с наименьшей порождающей нечеткостью.

Что касается *меры сложности,* то тут возможно много вариантов . Возьмем для примера простую, но содержательную меру — размер (мощность) маски.

Пусть ${}^iq_u$ (i=l, 2, ...)—значения соответствующих порождающих нечеткостей для систем с поведением ${}^i\mathbf{F}_B$ из ограниченного множества Y_r. Поскольку любая система $\mathbf{F}_B$ однозначно идентифицируется своей маской *M,* мощность которой $|{}^i\mathbf{M}|$ задает ее сложность, статус системы ${}^i\mathbf{F}_B$ в смысле порождающей нечеткости и сложности удобно описывать парой $(|{}^i\mathbf{M}|, {}^iq_u)$. Теперь рассматриваемую задачу можно обсуждать в терминах масок ${}^iM$, а не соответствующих систем с поведением ${}^i\mathbf{F}_B$. Численное упорядочение масок ${}^iM$, идентифицирующих системы

с

из Y_r по их мощности, задает *упорядочение сложности* $\overset{c}{\leq}$ на множестве Y_r. Численное упорядочение значений ${}^iq_u$ определяет *упорядочение по нечеткости* $\overset{u}{\leq}$ на множестве Y_r. В то время, как упорядочение по сложности полностью определяется самими масками, упорядочение по нечеткости может быть определено только после оценки масок. Для любого множества порождающих масок мы можем определить частичное упорядочение ${}^iM_G \leq {}^jM_G$ тогда и только тогда, когда

$$ {}^i\mathbf{g} = {}^j\mathbf{g} \text{ и } {}^i\bar{\mathbf{g}} \prec {}^j\bar{\mathbf{g}}. \quad (71) $$

(или ${}^i\mathbf{e} \prec {}^j\mathbf{e}$ для направленных систем), которое мы будем называть *упорядочением подмасок.* Это упорядочение часто оказывается полезным при разработке различных эвристических процедур поиска на множестве систем Y_r.

Пример упорядоченности по сложности и упорядоченности подмасок для наибольшей допустимой маски **M** при n=3 и $\Delta\mathbf{M}$ = 2 приведен на рис. 10.

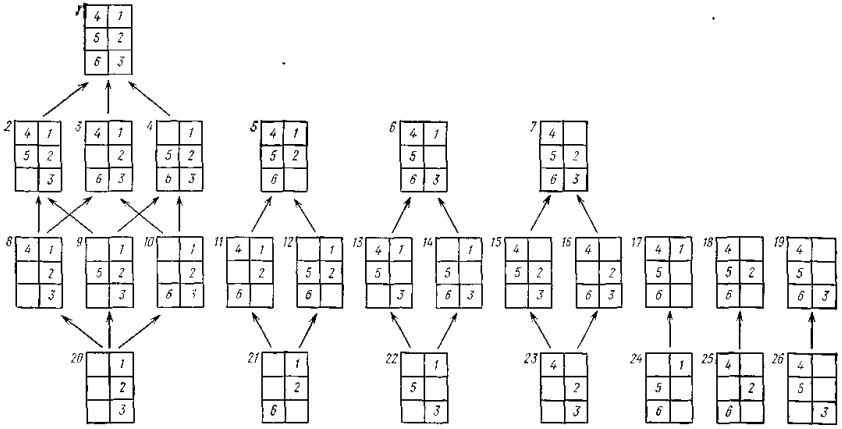


Рис.10. Содержательные маски для n=3 и $\Delta\mathbf{M}$=2, классифицированные согласно упорядоченности по сложности и упорядоченности подмасок

При этом предполагается, что данные порождаются слева направо. Все содержательные подмаски изображаются своими матрицами и помечены в левом верхнем углу своими идентификаторами *i*. По

сложности они разбиты на четыре группы. Маски с одинаковой сложностью расположены на одном уровне. Например, маски с идентификаторами 2—7 образуют группу со сложностью 5, маски 8—19 — другую группу со сложностью 4 и т. д. С точки зрения упорядоченности по сложности любая маска некоторого уровня является непосредственным преемником любой маски ближайшего более высокого уровня и непосредственным предшественником любой маски ближайшего более низкого уровня. На рис. 10 стрелками показано упорядочение по подмаскам. Из этого примера видно, что упорядочение по сложности —это связное квазиупорядочение (рефлексивное и транзитивное отношение, определенное для любой пары систем).

Упорядочение по подмаскам — это частичное упорядочение, но решетки оно не образует. Однако оно представляет собой набор решеток по одной для каждого множества порождаемых выборочных переменных (в нашем примере это крайние правые элементы масок). Упорядочение по нечеткости связное, но из-за того, что несколько разных систем могут иметь одинаковую порождающую нечеткость, это отношение не является антисимметричным. Следовательно, в общем случае это связное квазиупорядочение, которое в некоторых частных случаях оказывается полным упорядочением.

Таким образом, на множестве Y_r определены два связных квазиупорядочения — <u>по сложности и по нечеткости</u>. Было бы желательно объединить их неким подходящим образом. Поскольку для рассматриваемого типа задач требуется, чтобы *u* сложность *u* порождающая нечеткость систем во множестве решений Y_O, была минимизирована, соответствующее *объединенное упорядочение* $\overset{*}{\leq}$ определяется следующим образом:

${}^i\mathbf{F}_\text{B} \overset{*}{\leq} {}^j\mathbf{F}_\text{B}$ тогда и только тогда, когда

$$ |{}^iM| \overset{c}{\leq} |{}^jM| \text{ и } {}^iq_u \overset{u}{\leq} {}^jq_u, \quad (72) $$

где ${}^i\mathbf{F}_\text{B}, {}^j\mathbf{F}_\text{B} \in Y_r$. Это упорядочение не является связным, поскольку пары ${}^i\mathbf{F}_\text{B}, {}^j\mathbf{F}_\text{B}$, для которых $|{}^iM| < |{}^jM|$ и ${}^iq_u > {}^jq_u$ или $|{}^iM| > |{}^jM|$ и ${}^iq_u < {}^jq_u$ (подобные пары, разумеется, могут существовать), несравнимы. Оно также неантисимметрично, так как не исключена возможность того, что

$$ |{}^iM| = |{}^jM| \text{ и } {}^iq_u = {}^jq_u $$

для некоторых $i \neq j$. Следовательно, объединенное упорядочение— это общего вида квазиупорядочение (рефлексивное и транзитивное отношение) на Y_r.

Теперь множество решений Y_Q можно определить как множество всех систем из Y_r, которые или эквивалентны, или несравнимы относительно объединенного упорядочения (72). Две системы из Y_r, скажем системы ${}^iF_B$ и ${}^jF_B$, несравнимы в смысле объединенного упорядочения, если выполнено одно из следующих условий:

(а) ${}^iF_B$ более сложна и более детерминирована, чем ${}^jF_B$ или ${}^iF_B$ менее сложна и менее детерминирована, чем ${}^jF_B$. Формально

$$Y_Q = \{{}^i\mathbf{F}_B \in Y_r \mid (\forall {}^j\mathbf{F}_B \in Y_r)({}^i\mathbf{F}_B \overset{*}{\leq} {}^i\mathbf{F}_B \Rightarrow {}^i\mathbf{F}_B \overset{*}{\leq} {}^j\mathbf{F}_B)\}. \quad (73)$$

Системы из множества решений Y_Q будем называть *подходящими системами ИИ с поведением* для рассматриваемого типа задач.

Пример 7. Чтобы пояснить различные вопросы, изучаемые в данном разделе, рассмотрим этологическую систему данных, описанную в примере 1 р.2.8.(см. также рис. 2 р.2.8). Определим все подходящие в смысле (73) системы с поведением для этой системы данных в предположении, что пользователь хочет получить описания вероятностных систем с поведением и использовать их для предсказания.

Предположим сначала, что $\Delta\mathbf{M}$=2. Тогда имеется восемь содержательных масок, которые вместе с их упорядочением подмасок и указанием трех уровней сложности изображены на рис. 11,а.

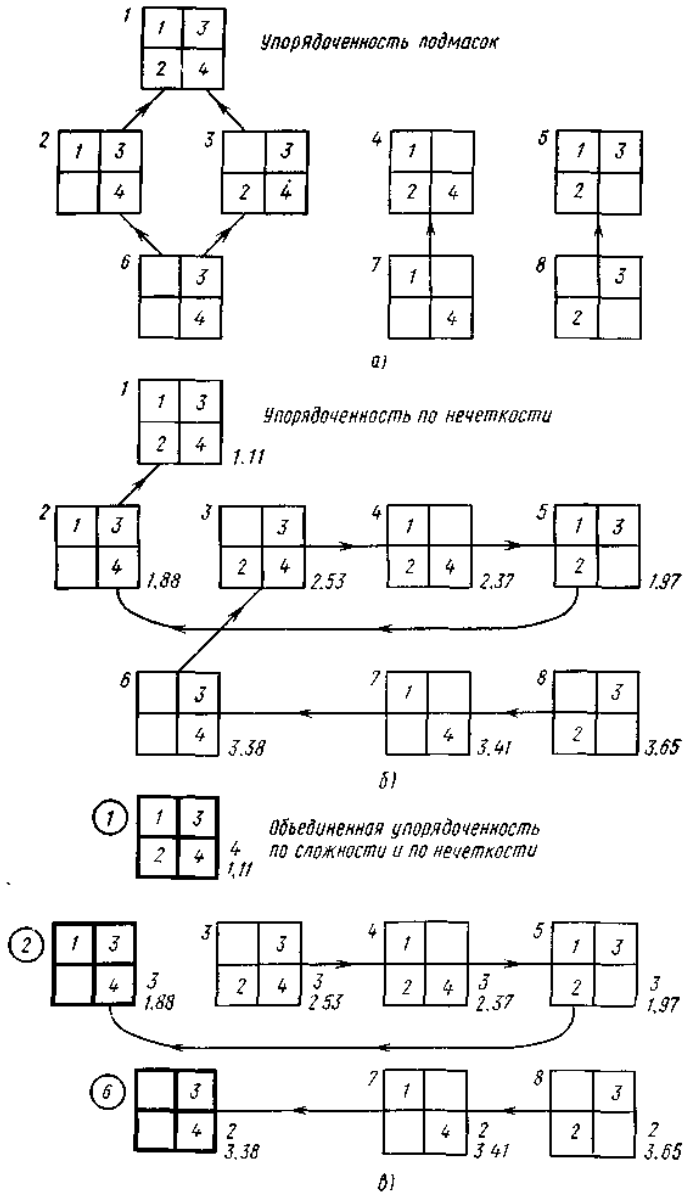


Рис. 11. К примеру 7

После выполнения исчерпывающей выборки для наибольшей приемлемой маски ${}^1\mathrm{M} = \mathbf{M}$ по определенной пользователем формуле по частотам $N(\mathbf{c})$ вычисляются вероятности $f_в(\mathbf{c})$, а порождающая нечеткость вычисляется или по формуле (45) или по более подходящей формуле (48). Если для вычисления вероятностей используется формула (31), то порождающая нечеткость равна 1.11. Затем для остальных семи содержательных масок по формуле (69) определяются соответствующие проекции и вычисляются их порождающие нечеткости. Результаты этих вычислений показаны на рис. 11,б (в правом нижнем углу масок). На рис. 11,б также изображено упорядочение масок по нечеткости. В этом примере упорядочение является полным, поскольку значения нечеткости у всех разные. Объединенное упорядочение по сложности и нечеткости (72) изображено на рис. 11,в. Как мы видим, минимальными с точки зрения объединенного упорядочения являются маски с идентификаторами 1, 2, 6. Следовательно, $V_Q = \{{}^1\mathbf{F}_B, {}^2\mathbf{F}_в, {}^6\mathbf{F}_в\}$.

Предположим теперь, что $\Delta\mathbf{M}=3$. Тогда согласно формуле (36) имеется 40 содержательных масок. После их обработки, аналогичной обработке для случая $\Delta\mathbf{M}=2$, мы получим пять подходящих систем с поведением, маски которых, значения сложности и порождающие нечеткости приведены на рис. 12,а.

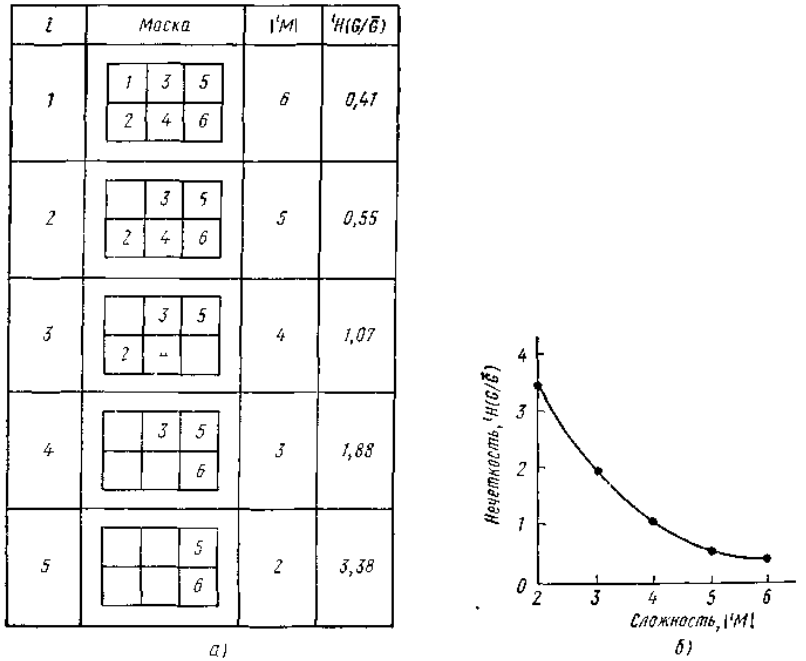


Рис. 12. Подходящие системы с поведением из примера 7

Оставшиеся 35 масок хуже с точки зрения их сложности, как и с точки зрения четкости, и, следовательно, их вовсе не нужно рассматривать. Рис. 12,а — это типичный пример ответа ФРИЗ на запрос пользователя. При соответствующих запросах могут также выдаваться различные дополнительные характеристики множества решений, такие, как график зависимости нечеткости от сложности, изображенный на рис. 12,б.

Описанный здесь поиск подходящих систем ИИ с поведением может быть реализован самыми разными способами. Основной принцип заключается в том, что содержательные маски получаются с помощью некоторого алгоритма из наибольшей приемлемой маски в порядке уменьшающейся сложности. Среди масок одинаковой сложности выбираются только маски с минимальной порождающей нечеткостью. При этом если значение этой минимальной нечеткости меньше или равно значению нечеткости для предшествующего уровня сложности, то все ранее принятые системы отбрасываются. В результате

применения этой процедуры у нас остаются только подходящие системы ИИ.

Важно понимать, что задачи данной категории представляют собой тему со многими вариациями. Например, можно многими разными способами вычислять вероятности или возможности, можно использовать разные определения сложности, можно выдвинуть дополнительные условия, такие, как задание наибольшей приемлемой нечетности. При определении объединенного упорядочения пользователь может взвесить предпочтения для масок с меньшей глубиной, сложностью и нечеткостью (а также и дополнительные требования) и т. д. ФРИЗ должен быть спроектирован так, чтобы в нем имелся набор стандартных вариантов для всех ключевых категорий интеллектуальных задач. Среди них должен быть и вариант, определяющий выбор варианта по умолчанию, но вместе с ФРИЗ должен предоставлять пользователю максимально возможную свободу определения его собственных вариантов интеллектуальной задачи.

4.7. Системы с изменяющимися состояниями

Фундаментальным понятием искусственного интеллекта является понятие «различия»: т. е. или по существу два объекта различимы или один объект изменяется во времени.

Предположим, что задана система данных с полностью упорядоченным параметрическим множеством. Как было показано в разд. 4.2—4.6, эта система данных может быть описана параметрически инвариантно через множество подходящих систем ИИ с поведением, согласующихся с системой данных и удовлетворяющих требованиям, выдвинутым пользователем. Несмотря на то, что системы ИИ с поведением совершенно адекватно описывают полное ограничение на исследуемые выборочные переменные, существует и другая форма представления этого ограничения, часто представляющаяся пользователю более подходящей. Эту форму будем называть *отношением изменения состояния ({state-transition relation)* или сокращенно ST-отношением. Это отношение определяется не на отдельных состояниях, а на последовательных парах состояний; порождающие системы ИИ, в которых используется эта формула представления состояний, будем называть *системами ИИ с изменяющимися состояниями или ST-системами.*

Для ST-систем маски выборочные переменные, множества состояний выборочных переменных и их декартово произведение **C** определяются точно так же, как и для систем ИИ с поведением, за

исключением двух отличий: (1) к ST-системам неприменимо разделение выборочных переменных на порождаемые и порождающие, и (2) содержательные маски в ST-системах имеют дополнительные ограничения (это будет пояснено ниже). Аналогами функций поведения в ST-системах являются функции *изменения состояния* (или ST-функции). Для нейтральных систем ИИ они определены на $\mathbf{C}^2 = \mathbf{C}\times\mathbf{C}$, а не на **C**, а для направленных систем на $\mathbf{E}^2\times\ \overline{\mathbf{E}}^{\,2}$, а не на **E**×**E.**

Для нейтральных систем аналогами функций поведения определяем формулами (18), (22), (16), являются следующие ST-функции:

$$f_S : \mathbf{C}^2 \rightarrow [0, 1], \qquad (74)$$

где $f(\mathbf{c}, \mathbf{c}')$ —это вероятность, возможность или некая другая характеристика состояния c', следующего непосредственно за состоянием c (согласно выбранному порядку порождения);

$$f_{GS} : \mathbf{C}^2 \rightarrow [0,\ 1], \qquad (75)$$

где $f_{GS}(\mathbf{c}, \mathbf{c}')$—условная вероятность или возможность того, что при текущем состоянии **c** следующим состоянием будет состояние **c'**; будем поэтому использовать общепринятую запись $f_{GS}(\mathbf{c}'|\mathbf{c})$*:*

$$f_{GS}: \mathbf{C} \rightarrow \mathbf{C}, \qquad (76)$$

где $f_{GS}(\mathbf{c}) = \mathbf{c}'$, т. е. следующее состояние однозначно определяется текущим состоянием **c**; функция специального вида (76) применима, разумеется, только к детерминированным системам. Будем f_{GS} называть *порождающими* ST-*функциями.*

Аналогами нейтральных систем ИИ с поведением (10) и (15) являются соответственно *ST-система*

$$\mathbf{F}_s = (\mathbf{I}, M, f_S) \qquad (77)$$

и *порождающая ST-система*

$$\mathbf{F}_{GS} = (\mathbf{I}, M_G, f_{GS}), \qquad (78)$$

где **I**, M и M_G имеют тот же смысл, что в системах ИИ с поведением. Для заданных системы данных и маски ST-функция f_s, хорошо согласующаяся с системой данных и маской, может быть определена с помощью полной выборки данных аналогично тому, как это делалось для функции поведения f_B. Единственное отличие состоит в том, что в результате выборки получаются частоты $N(\mathbf{c}, \mathbf{c}')$ пар последовательных состояний, а не частоты $N(\mathbf{c})$ отдельных состояний. Пара $(\mathbf{c}, \mathbf{c}') \in \mathbf{C}^2$ называется *переходом* из состояния **c** в другое состояние **c'** согласно объявленному на параметрическом множестве порядку порождения. Одним из важнейших свойств ST-функций является то, что переходы в некоторое состояние должны находиться в равновесии с переходами из этого состояния. Если используются вероятности, то для любого состояния $\mathbf{x} \in \mathbf{C}$ имеем

$$\sum_{\mathbf{c}\in\mathbf{C}} f_S(\mathbf{c}, \mathbf{x}) = f_B(\mathbf{x}),$$

$$\sum_{\mathbf{c}'\in\mathbf{C}} f_S(\mathbf{x}, \mathbf{c}') = f_B(\mathbf{x}), \quad (79)$$

и, следовательно,

$$\sum_{\mathbf{c}\in\mathbf{C}} f(\mathbf{c}, \mathbf{x}) = \sum_{\mathbf{c}'\in\mathbf{C}} f_S(\mathbf{x}, \mathbf{c}'), \quad (80)$$

что и определяет *равновесие переходов.* Если используются возможности, то уравнение (80) будет иметь вид

$$\max_{\mathbf{c}\in\mathbf{C}} f_S(\mathbf{c}, \mathbf{x}) = \max_{\mathbf{c}'\in\mathbf{C}} f_S(\mathbf{x}, \mathbf{c}'). \quad (81)$$

Состояния **c, c'** могут рассматриваться как состояния, определяемые двумя взаимосвязанными масками *M, M′.* Маски связаны между собой простым сдвигом в соответствии со следующими правилами сдвига:

$(v_i, \rho) \in M$ тогда и только тогда, когда $(v_i, \rho+1) \in M'$ (82)

если данные порождаются в порядке возрастания параметра, или

$(v_i, \rho) \in M$ тогда и только тогда, когда $(v_i, \rho-1) \in M'$, (83)

если данные порождаются в обратном порядке. Маски *M, M′* используются вместе для описания пар состояний **c, c'**.

Чтобы избежать противоречий и неполноты при порождении данных, содержательные маски в ST-системах должны удовлетворять следующему требованию (в дополнение к требованиям для масок в системах ИИ с поведением): для заданной маски *M,* если $(v_i, \rho_1) \in M$ и $(v_i, \rho_2) \in M$ и $\rho_1 < \rho_2$, то $(v_i, \rho) \in M$ для всех целых ρ , таких, что $\rho_1 \le \rho \le \rho_2$.

Это значит, что маски в ST-системах не должны содержать «пробелы», подобные элементам $(v_2, -1)$ на рис. 2. Маски, удовлетворяющие этому дополнительному требованию, будем называть *компактными масками.*

Для обоснования этого требования предположим, что маска *M* ST-системы некомпактна. Тогда существует по крайней мере одна пара элементов из маски *M,* скажем пара $(v_i, \rho_1) \in M$ и $(v_i, \rho_2) \in M$, такая, что $\rho_1 < \rho_2$,

$$(v_i, \rho_1+1) \notin M, \quad (v_i, \rho_2+1) \notin M, \quad (84)$$

и $\rho_2 \ge \rho$, для всех $(v_i, \rho) \in M$. Из (82) имеем

$$(v_i, \rho_1+1) \in M' \text{ и } (v_i, \rho_2+1) \in M'. \quad (85)$$

Обозначим выборочные переменные, базирующиеся на этих элементах *M′,* через s_1 и s_2. Состояния s_1, s_2 являются компонентами **c'**. Тем самым

они должны быть или определены для каждого значения параметра по состоянию **c**, или порождаться в соответствии с распределением вероятностей или возможностей $f_{GS}(\mathbf{c'}|\mathbf{c})$ для каждого конкретного **c**. Однако ни один из этих вариантов для s_1 невозможен. Из-за (84) оно не может быть определено по состоянию **c**, не может быть и корректно порождено при любом значении параметра *t,* так как

$$s_{1,t} = s_{2,\,t+\rho_1-\rho_2},$$

и, таким образом, s_1 определяется состоянием s_2 при значении (параметра t—(ρ_2—ρ_1). Нет никакой гарантии, что порожденное состояние s_1 будет соответствовать этому ранее определенному состоянию. С другой стороны, если состояние s_1 порождается, то **c'** становится неполным, так как ранее определенное состояние неизвестно (т. е. оно не является компонентом **c**). Следовательно, при любом значении t состояние s_1 не может быть ни определено из **c**, ни порождено с помощью порождающей ST-функции.

Иллюстрацией к доказательству того, что маски с «пробелами» недопустимы в ST-системах, служит рис. 13; компонента следующего состояния **c'** при значении параметра *t(r)* не может быть ни определен из состояния **c** в момент времени t (рис. 13,в), ни порожден с помощью ST-функции, так как уже порожден при значении параметра t—3 (рис. 13,а).

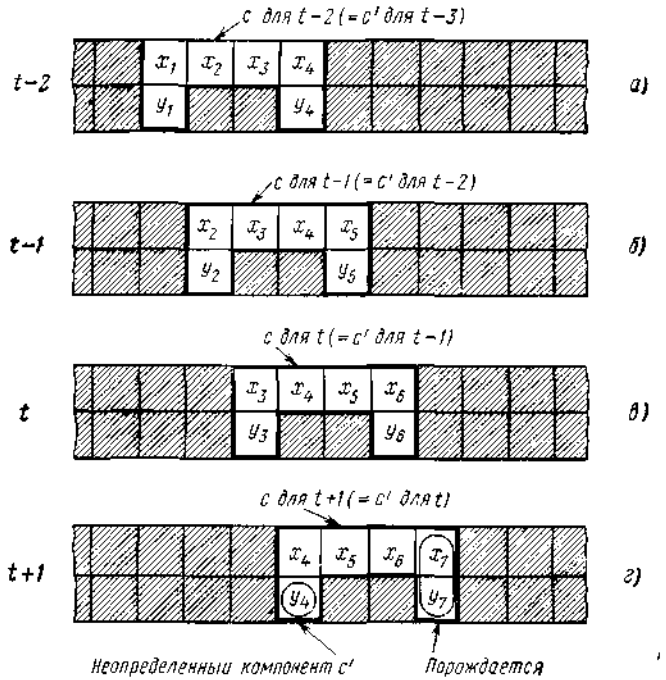


Рис.13. Пример несогласованности или неполноты ST-систем с некомпактными масками

Удобно представлять ST-функции (74) и (75) в виде квадратных матриц, строки и столбцы которых связаны соответственно с **c** и **c'**. Элементами этих матриц являются значения соответственно f_S**(c, c')** или f_{GS}**(c'| c)** /GS(C'|C).

Пример 8. Одним из подходов к оценке производительности вычислительной техники является постоянный контроль за аппаратным обеспечением. Значение этого подхода будет возрастать по мере возрастания сложности оцениваемых вычислительных систем. При контроле аппаратного обеспечения наблюдаются определенные ключевые переменные, обычно описывающие состояние отдельных компонентов вычислительной системы. Делается это в течение определенного времени обслуживания системы пользователями с помощью аппаратных мониторов. Данные обрабатываются аппаратным монитором и анализируются с целью обнаружения узких мест в системе и поиска способов повышения производительности, которая также должна быть каким-либо образом определена.

Обычно в состав аппаратных мониторов входят счетчики, которые в процессе сбора данных или считают число происшедших событий (режим счета), или измеряют длительность событий (временной режим). Это значит, что обычно аппаратный монитор предоставляет исследователю не фактические данные, а обобщенные. Например, монитор определяет, что центральный процессор (ЦП) вычислительной системы был загружен в течение 43% времени наблюдения, что канал был занят в 15% всех наблюдений, но не дает фактической последовательности событий, которые бы можно было затем обрабатывать и анализировать.

При этом часто теряется важная информация, способствующая лучшему пониманию вопросов, связанных с производительностью компьютера. Например, совершенно выпадают из анализа динамические аспекты работы компьютера.

Согласно концепции ФРИЗ все наблюдения должны быть зафиксированы, а затем обработаны любым подходящим способом (см. рис. 1). В данном примере по времени было сделано 409 610 наблюдений для четырех переменных v_1, v_2, v_3, v_4. Каждая переменная имела два состояния 0 и 1, характеризующие состояние конкретного компонента аппаратного обеспечения: 0 означает, что компонента была неактивна во время наблюдения, а 1 — что активна. Переменная v_1 описывает работу ЦП, а остальные три переменных — работу трех важных каналов связи системы. Для получения вероятностной ST-функции из этого огромного набора данных, превышающего 1,6 млн. бит, с помощью маски без памяти была сделана выборка для двух последовательных состояний. Это дало 15 состояний (см. табл. 5,а) и 113 переходов.

Таблица 5.

ST-функция для исследования по оценке производительности вычислительной системы (пример 8)

а)

c	v_1	v_2	v_3	v_4	c	v_1	v_2	v_3	v_4
1	1	0	0	0	9	0	1	0	1
2	1	0	0	1	10	1	0	1	1
3	1	1	0	1	11	1	1	1	0
4	1	1	0	0	12	1	1	1	1
5	0	0	0	0	13	0	1	1	0
6	1	0	1	0	14	0	0	1	0
7	0	1	0	0	15	0	0	1	1
8	0	0	0	1					

б)

c = \ c' =	1	2	3	4	5	6	7-15	f_B(c)
1	0.844	0.064	0.004	0.057	0.028	0.002	0.001	0.458
2	0.173	0.757	0.049	0.011	0.003	~0	0.007	0.175
3	0.022	0.093	0.725	0.155	~0	0	0.005	0.092
4	0.109	0.008	0.059	0.816	0.001	~0	0.007	0.242
5	0.755	0.056	0.002	0.036	0.146	0.001	0.004	0.016
6	0.103	0.007	0	~0	~0	0.811	0.079	0.008
7–15	0.050	0.141	0.054	0.170	0.002	0.063	0.520	0.009

$f_{GS}(c'|c)$

Состояния 7—15 появляются очень редко: вероятность того, что система находится в одном из этих состояний 0.009. Если для упрощения ST-функции, объединить эти состояния в одно (см. разд. 9), что удобно для исследователя, получится матричное представление порождающей ST-функции, приведенное в табл. 5,б. Элементами матрицы являются условные вероятности f_{GB}(**c'|c**). Обозначение ~0 используется для вероятностей, которыми можно пренебречь; через 0 обозначены переходы, которые вообще не наблюдались. В матрице подчеркнуты элементы, соответствующие переходу из состояния снов'а в это состояние. В табл. 5,б также приведен вектор-столбец значений f_B (**c**) функции поведения f_B при той же маске (маска без памяти), которая обычно и является результатом контроля работы аппаратного обеспечения. Попятно, что

$$f_S(\mathbf{c}, \mathbf{c}') = f_{GS}(\mathbf{c}'|\mathbf{c}) \cdot f_B(\mathbf{c}). \quad (86)$$

Поэтому для дальнейшей обработки можно вычислить и значения f_S(**c, c'**). Хотя определение функции f_{GS} или другого подходящего представления собранных данных для соответствующих исходных

систем, определенных на компьютерных комплексах, проводится в рамках ФРИЗ, интерпретация полученных результатов должна проводиться специалистами по оценке производительности вычислительных систем.

В некоторых случаях предпочтительнее представлять ST-функции в виде диаграмм. Такая диаграмма представляет собой набор узлов, по одному для каждого состояния наблюденных выборочных переменных, и ориентированных связей между узлами, соответствующих реально существующим переходам. Узлы на диаграмме должны быть помечены соответствующими идентификаторами состояния **c**, а связи помечены значениями f_S(**c, c'**) или f_{GS}(**c'**| **c,**); в последнем случае желательно также пометить узлы значениями f_B (**c**) так, чтобы значения f_S(**c, c'**) можно было вычислить при необходимости из уравнения (86).

Пример.9. С помощью ST-систем часто бывает удобно эффективно описывать правовые ограничения в различных законодательных учреждениях, таких, как местное и федеральное правительства. На диаграмме (рис. 14) приведена четкая возможностная ST-функция, описывающая законодательную систему США с точки зрения ограничений на выполнение пожеланий граждан (сокращенно П) в новый закон.

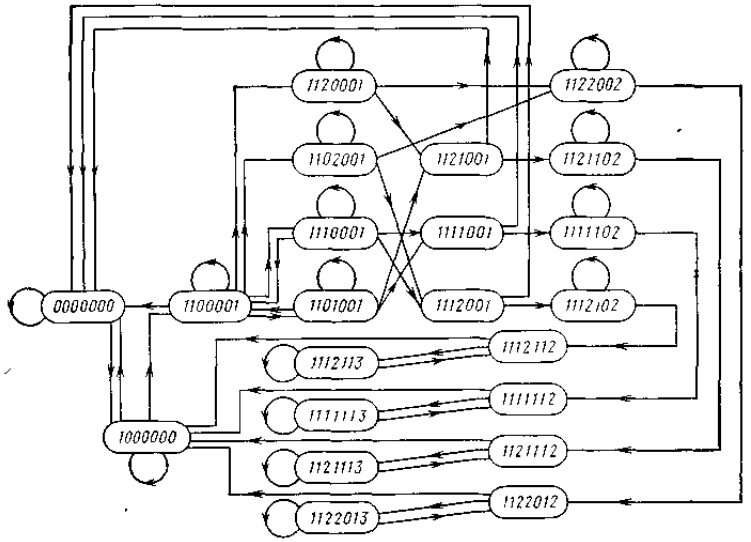


Рис. 14. ST-диаграмма правовой системы США

Данная ST-система основывается на маске без памяти, и ее представляющая система состоит из семи базовых переменных и их множеств состояний:
v_1 — политическая притягательность (0 — низкая; 1 — высокая);
v_2 — П разделяет поручитель граждан в конгрессе (0 — нет; 1-да);
v_3— статус П в палате представителей (0-—отклонено или не рассматривалось; 1 — прошло простым большинством; 2 — прошло большинством в 2/3 голосов);
v_4— статус П в сенате (0 — отклонено или не рассматривалось; 1 — прошло простым большинством; 2 — прошло большинством в две трети голосов);
v_5 — утверждение П президентом (0 — нет; 1 — да);
v_6 — одобрение П судебной властью (0 — нет; 1—да);
v_7 — правовой статус П (0 — отсутствует; 1—законопроект; 2 — законодательный акт; 3 — закон).
Параметром является время, определенное неявно, а по изменению состояний переменных. В узлах диаграммы записаны состояния переменных в порядке $v_1, v_2,...,v_7$. Ни узлы, ни связи никак не помечены, так как в данном случае естественной представляется дихотомия всех состояний переходов на возможные (т. е. допускаемые существующими законами) и неразрешенные (т. е. законами не допускаемые). Поэтому на диаграмме изображены только узлы и связи, имеющие степень возможности 1, а остальные узлы и связи, имеющие степень возможности 0, опущены.
Диаграмма на рис. 14 описывает только правовые ограничения. На ней не показаны затруднения, которые возможны при осуществлении любого из переходов. Данные относительно предыдущих значений П могут быть использованы для представления этих трудностей в виде степеней возможности в диапазоне [0, 1]. Для конкретного П и определенных политических и других обстоятельств степени возможности отдельных переходов могут быть субъективно оценены экспертом или группой экспертов.
Любую ST-систему легко можно преобразовать в изоморфную систему ИИ с поведением. Чтобы показать, как это делается, возьмем произвольную ST-систему

$$\mathbf{F}_S=(\mathbf{I},M,f_S),$$

где M компактная маска. Предположим, что любое следующее состояние соответствует большему значению параметра, чем предшествующее.
Рассмотрим теперь систему ИИ с поведением

$$\mathbf{F}_B=(\mathbf{I},M^+, f_B).$$

где M^+ определяется через M следующим образом:

$$(v_i,\ \rho)\in M^+,\ \text{если}\ (v_i,\ \rho)\in M\ \text{или}\ (v_i,\ \rho-1)\in M. \quad (87)$$

Тогда для любого набора данных из общей представляющей системы ИИ **I** все выборки данных, дающие одну пару состояний для маски M, скажем пару (**c, c'**), дают и одно и то же состояние для маски M^+, скажем состояние $\mathbf{c}^+$. Если данные полностью выбираются с помощью обеих масок, то частоты **c, c'** и $\mathbf{c}^+$ должны быть одинаковы.
Следовательно,

$$f_S(\mathbf{c},\ \mathbf{c}')=f_B(\mathbf{c}^+),$$

где состояние $\mathbf{c}^+$ состоит из **c** и порождаемой части **c'**, назовем ее $\mathbf{c}'_g$.
Таким образом, функция поведения f_B эквивалентна ST-функции f_S при однозначном соответствии

$$\gamma:\ \mathbf{c}^2\rightarrow\mathbf{C}^+, \quad (88)$$

где $\gamma(\mathbf{c}, \mathbf{c}') = \mathbf{c}^+$ тогда и только тогда, когда $\mathbf{c}^+=\mathbf{c}, \mathbf{c}'$g.
Маску M^+ (87) будем называть *расширенной маской М.* Она определена в предположении, что состояния порождаются в порядке возрастания параметра. При обратном порядке альтернативная расширенная маска, скажем маска ^+M, определяется несколько иначе:

$$(v_i,\ \rho)\in {}^+M,\ \text{если}\ (v_i,\ \rho)\in M\ \text{или}\ (v_i,\ \rho-1)\in M. \quad (89)$$

Можно аналогично тому, как это было показано для M^+, показать, что система с поведением

$$\mathbf{F}_B=(\mathbf{I}, {}^+M, f_B)$$

изоморфна ST-системе, определенной для той же представляющей системы **I** и маски M.
Соответствие между масками M, M^+ и масками M, ^+M показано соответственно на рис. 15,а и *б.*

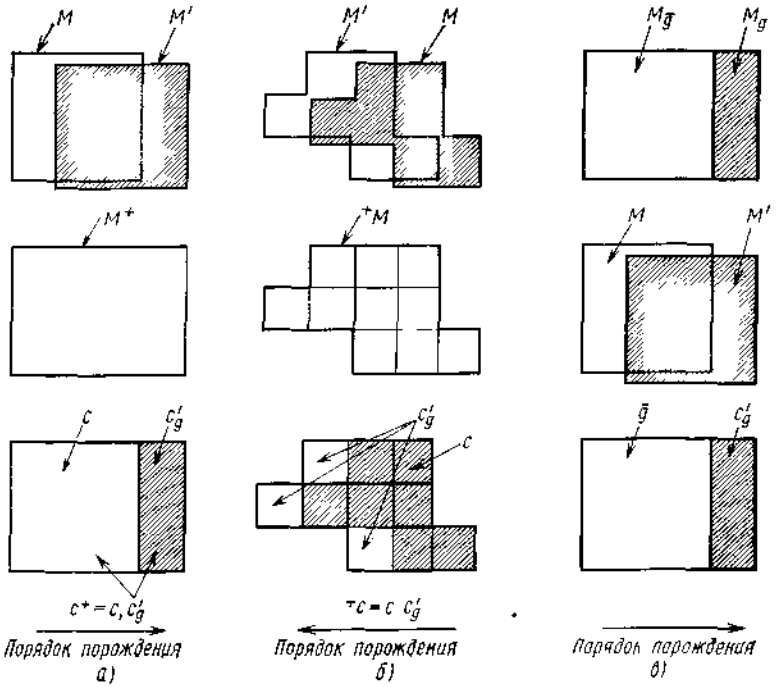


Рис.15. Изоморфизм между системами с поведением и ST-системами

На рисунке также показано однозначное соответствие (88) и его аналог для маски ^{+}M и $^{+}\mathbf{c} \in {}^{+}\mathbf{C}$.

Для заданной системы с поведением

$$\mathbf{F}_B = (\mathbf{I},\ \mathbf{M}_{\bar{g}},\ f_S),$$

изоморфная ST-система при той же представляющей системе ИИ **I** существует только тогда, когда M — компактная маска и $|M_i| \geq 2$ для любой подмаски M_i. Если эти условия выполнены, то понятно, что ST-система

$$\mathbf{F}_S = (\mathbf{I},\ M_{\bar{g}},\ f_S),$$

где $M_{\bar{g}}$ - порождаемая часть (согласно определенному порядку порождения), которая изоморфна при соответствующем однозначном соответствии между множествами состояний **C** (основанном на M) и $\bar{\mathbf{G}} \times \bar{\mathbf{G}}$ (основанном на $M_{\bar{g}}$). Данное преобразование из системы с

поведением в изоморфную ST -систему для одного из порядков порождения показано на рис. 15,в.

Для направленных систем ИИ изменяющими состояние аналогами функций поведения определяемыми уравнениями (26), (28), (29), будут соответственно следующие ST-функции:

$$\hat{f}_S : \mathbf{E}^2 \times \bar{\mathbf{E}}^2 \to [0,\ 1], \quad (90)$$

где **E** имеет тот же смысл, что и для систем ИИ с поведением, и $\hat{f}_S(\bar{\mathbf{e}},\ \bar{\mathbf{e}}' \mid \mathbf{e},\ \mathbf{e}')$ — условная вероятность или возможность, смысл которой однозначно определяется ее общепринятым обозначением

$$\hat{f}_{GS} : \mathbf{E}^2 \times \bar{\mathbf{E}}^2 - [0,\ 1], \quad (91)$$

где $\hat{f}_{GS}(\bar{\mathbf{e}}' \mid \mathbf{e},\ \mathbf{e}',\ \bar{\mathbf{e}})$ — порождающие условные вероятности или возможности;

$$\hat{f}_{GS} : \mathbf{E}^2 \times \bar{\mathbf{E}} \to \bar{\mathbf{E}}, \quad (92)$$

где $f_{GS}(\mathbf{e},\ \bar{\mathbf{e}}',\ \bar{\mathbf{e}}) = \bar{\mathbf{e}}'$.

Изменяющими состояния аналогами для направленных систем ИИ с поведением (27) и (30) будут соответственно *направленная ST-система*

$$\hat{\mathbf{F}}_S = (\hat{\mathbf{I}},\ \hat{\mathbf{M}},\ \hat{f}_{GS}), \quad (93)$$

и направленная порождающая ST-система

$$\hat{\mathbf{F}}_S = (\hat{\mathbf{I}},\ \hat{\mathbf{M}}_G,\ \hat{f}_{GS}), \quad (94)$$

где $\hat{\mathbf{I}}$ и $\hat{\mathbf{M}}_{\mathrm{g}}$ определяются так же, как и для систем ИИ с поведением.

Функции (90) и (91) удобно представлять в виде массивов квадратных матриц (трехмерных массивов) по одной матрице для каждого условия **e, e'**. Удобны также диаграммы, подобные диаграммам для нейтральных ST-систем. Для направленных систем связи на диаграммах помечаются не только значениями соответствующей ST-функции, но и условиями **e, e'**. Функции (92) можно представлять в виде матриц, строки которых соответствуют состояниям **e**, столбцы — состояниям **e'**, а элементами являются соответствующие состояния **e'**. Эти функции представляются также в виде диаграмм, таблиц и в некоторых случаях с помощью алгебраических формул.

Пример 10. На рис. 16 приведена простая направленная ST-система (без интерпретации).

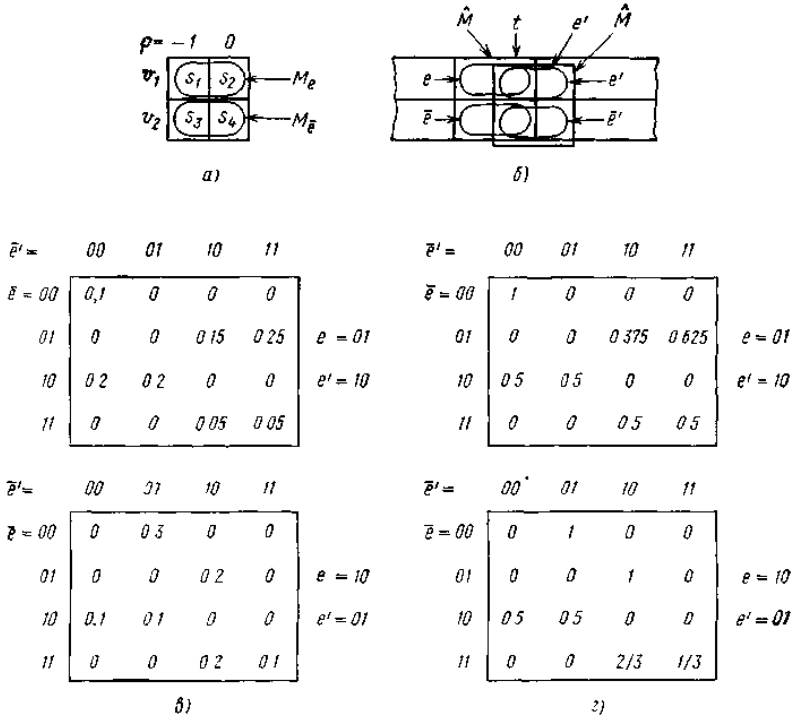


Рис. 16. Трехмерный массив, представляющий направленную ST-систему (пример 10)

Ее представляющая система состоит из входной переменной v_1 и выходной переменной v_2, каждая имеет два состояния 0 и 1. Маска системы M показана на рис. 16,а, а на рис. 16,б приведены соответствующие компоненты состояний, порожденные двумя ее последовательными положениями на матрице данных. На рис. 16,*в* и *г* показаны соответственно функции f_s и f_{GS} в виде трехмерных массивов. В данном примере массив состоит из двух матриц.

Пример 11. Рассмотрим детерминированную направленную ST-систему, описывающую метаболизм бактерий определенного класса с биохимической точки зрения. Бактерии рассматриваются как сложные мультиэнзимные совокупности, а метаболизм как множество всех возможных последовательностей биохимических реакций в бактериях. В этом примере используется маска без памяти. На рис. 17 приведена диаграмма ST-функции в форме (92).

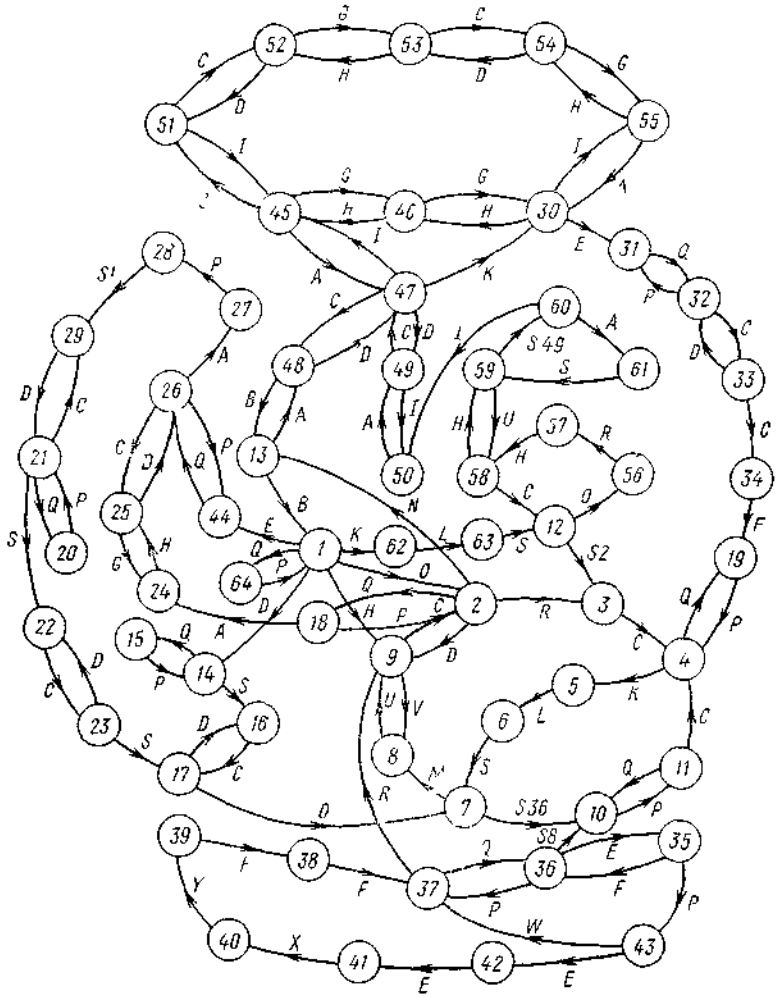


Рис. 17. ST функция, описывающая метаболизм класса бактерий

Состояния выходных переменных (на рис. 17. они помечены числами) представляют множество всех субстратов, производимых соответствующими химическими реакциями. Состояния входных переменных (они обозначены буквами) представляют коэнзимы, участвующие в химических реакциях. Коэнзимы, помеченные *S1, S2, S8, S36, S49,* эквивалентны субстратам, полученным соответственно в состояниях 1, 2, 8, 36, 49.

Пример 12. При применении оральных противозачаточных средств используются четыре гормона: стимулирующий фолликулы, гормон FSH, лютенизирующий гормон LH, экстроген E и прогестерон P. Несмотря на то, что содержание этих гормонов в женской крови может быть измерено с высокой точностью, нет необходимости использовать высокоточные данные, если предметом исследования является контроль рождаемости, основанный на применении эстрогена в виде противозачаточных пилюль. Для каждого гормона может быть определен критический пороговый уровень содержания и имеет значение только то, превышает фактический уровень этот порог или нет. Таким образом, для описания каждого гормона достаточно переменной, имеющей всего два состояния. Пусть переменная находится в состоянии 0, если уровень содержания гормона ниже порогового, и в состоянии 1 в противном случае. Параметром является время, определяемое неявно по изменениям состояний переменных. Кроме этих четырех переменных, рассматриваемых как входные, в систему входит выходная переменная, описывающая влияние противозачаточных пилюль. Пусть эта переменная находится в состоянии 0, если пилюли не используются, и в состоянии 1 в противном случае.

На рис. 18,а и *б* приведены соответственно матрицы данных для нормального менструального периода с использованием противозачаточных пилюль и без их использования.

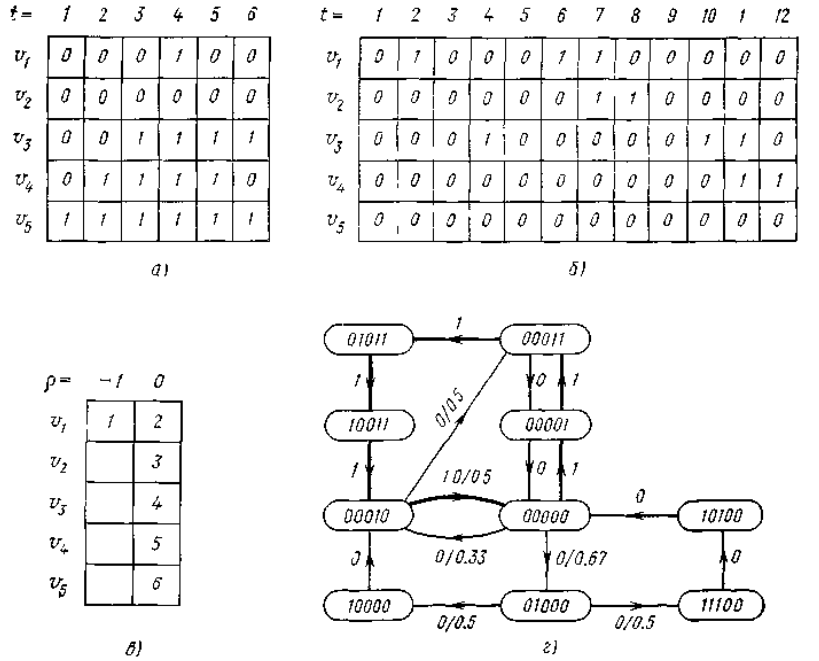


Рис. 18. Система контроля рождаемости (пример 12)

Переменные v_1— v_5 имеют следующий смысл: v_1— FSH; v_2 — LH, v_3 — E; v_4 — P; v_5 — противозачаточные пилюли. Пусть для выборки из матриц данных используется маска, изображенная на рис. 18,*в*. В данном примере матрицы периодические (за последним столбцом каждой матрицы следует ее первый столбец). Выбрав с помощью этой маски данные из матриц и вычислив вероятности, получим диаграмму, изображенную на рис. 18,г. Она представляет порождающую ST-функцию в виде (91). Узлы помечены состояниями выборочных переменных v_1— v_5 в их естественном порядке. Связи помечены состояниями c' (входной переменной $v_5=s_6$, принадлежащей следующему состоянию) и, если оно не единственное, также вероятностями соответствующих переходов. Предполагается, что $e=e'$, поскольку данные для других возможностей отсутствуют. Как можно видеть, система является детерминированной при условии, что противозачаточные пилюли принимаются в соответствии с тем, как это показано жирными стрелками на диаграмме. Эта единственная последовательность переходов минует состояния, в которых может произойти оплодотворение.

Пример 13. Для пояснения смысла пространственной инвариантности рассмотрим два примера мозаик. Первая представляет собой черно-белую шахматную доску (рис. 19,а).

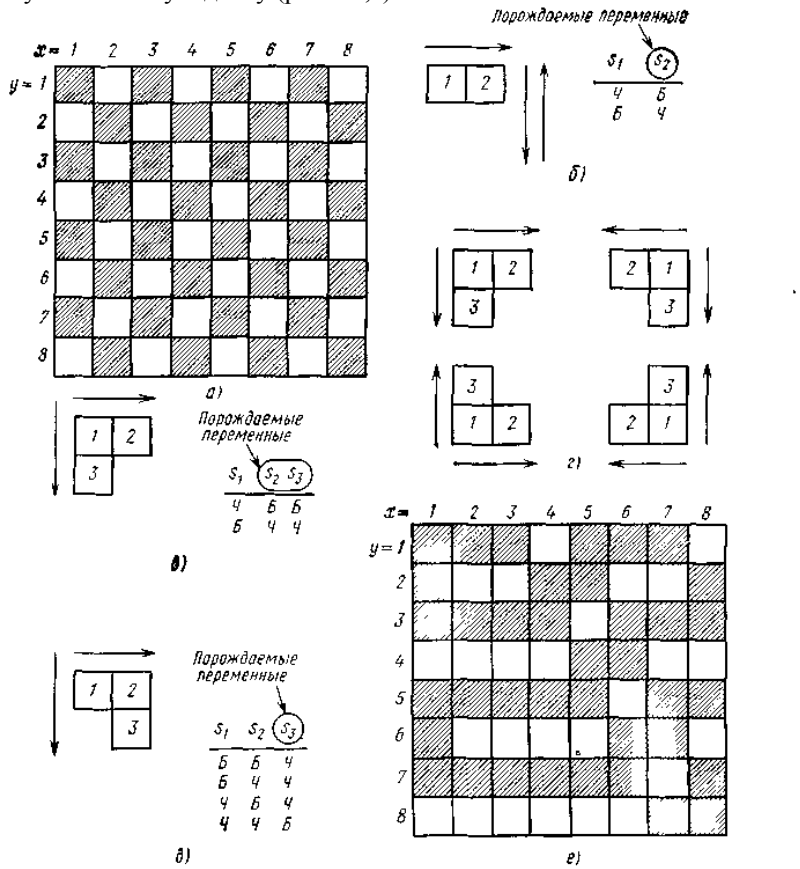


Рис. 19 Пространственная инвариантность (пример 13)

Его параметрическое множество состоит из 64 клеток пространственной решетки. Оно описывается координатами *x* и *y*. Несмотря на то, что каждая из координат полностью упорядочена, параметрическое множество упорядочено только частично. На параметрическом множестве определена одна переменная, скажем переменная *v*. Она имеет два состояния: Ч (для черного цвета) и Б (для белого). Обозначим через $v_{x,y}$ состояние *v* для клетки с координатами *x*, *y*. Тогда выборочные переменные определяются уравнением

$$s_{k,\,x,\,y} = v_{x+\alpha,\,y+\beta},$$

где $s_{k,x,y}$ — состояние выборочной переменной s_k для клетки со значениями координат *x, y*. Для порождения состояний на всем параметрическом множестве достаточно двух выборочных переменных, например

$$s_{1,\,x,\,y} = v_{x,\,y},$$

$$s_{2,\,x,\,y} = v_{x+1,\,y}.$$

На рис 19,б приведены изображение соответствующей маски с обозначением возможных порядков порождения (в предположении, что порождается переменная s_2), а также функция поведения в виде (16), полученная из данных с помощью выборки. Несмотря на то, что значение s_2 однозначно определяется по s_1, в качестве начального условия необходимо знать весь столбец, соответствующий $x=1$. Для сокращения начального условия до одной клетки ($x=y=1$) используем дополнительную выборочную переменную, определяемую уравнением

$$s_{3,\,x,\,y} = v_{x,\,y+1}.$$

На рис. 3.19,*в* изображены соответствующая маска, порядки порождения и функция поведения. В данном случае порождение должно начинаться с верхнего левого угла пространственной решетки. Для трех элементов, изображенных на рис. 19,г, имеется четыре аналогичных маски. Для каждой маски предполагается, что переменная s_1 является справочной переменной и в то же время порождающей. Каждая маска связана с определенными порядками порождения и требует особого начального условия (эти условия соответствуют четырем углам шахматной доски).

В качестве второго примера предположим, что при тех же параметрическом множестве и переменной имеются маска и порождающая функция [снова в форме (16)], определенные так, как это показано на рис. 19,в. Тогда рис. 19,е порождается этой функцией поведения при условии, что в качестве начальных условий заданы рисунки первого столбца и первой строки. Таким образом, мы можем породить двухмерный пространственный рисунок из двух одномерных, что в нашем примере оказывается эквивалентным.

4.8. Порождающие системы ИИ

...взаимоотношения между воспринимаемыми объектами являются, по крайней мере до некоторой степени, созданиями разума, который затем приписывает их внешнему миру. Таким образом, их можно рассматривать как «рабочие гипотезы» или, употребляя более точное слово, модели того, как организован реальный мир.

Р. Розен

Термин «порождающая система ИИ» используется в этой работе в качестве общего наименования для всех систем уровня 2 в эпистемологической иерархии систем ИИ. В этих системах обобщенное параметрически независимое ограничение на рассматриваемые переменные описывается с разных сторон. В предшествующих разделах данной главы были введены четыре типа порождающих систем ИИ, причем система любого типа может быть или нейтральной, или направленной.

системы с поведением

базовые: уравнение (10)—нейтральные;
уравнение (27) — направленные;
порождающие: уравнение (15)—нейтральные;
уравнение (30)—направленные;

ST-системы

базовые: уравнение (77) —нейтральные,
уравнение (93)—направленные;
порождающие: уравнение (78) — нейтральные;
уравнение (94)—направленные.

Если в некотором контексте отличия между этими четырьмя типами систем ИИ несущественны, то удобнее любую из них называть просто нейтральной или направленной и обозначать ее просто **F** или $\widehat{\mathbf{F}}$.

Как было показано в разд. 4.7, любая ST-система может быть преобразована в изоморфную систему ИИ с поведением, в то время как обратное преобразование возможно только при определенном типе масок. Следовательно, системы ИИ с поведением более общие, чем ST-системы.

Два основных недостатка ST-систем очевидны: ограниченность из-за использования только компактных масок и собственная избыточность, проистекающая из наложения текущего и следующего состояний. Это свойство всех масок, за исключением тех, в которых для любой базовой переменной определено только одно правило перехода. Из-за этих недостатков методологические средства ФРИЗ, предназначенные для порождающих систем ИИ, реализованы только на основе систем с поведением. Если пользователю нужна ST-система, то его задача решается в терминах изоморфных систем ИИ с поведением и при дополнительных ограничениях, налагаемых на маски в ST-системах. После решения задачи результирующие системы ИИ с поведением преобразуются в изоформные ST-системы и выдаются пользователю в нужной форме.

ST-системы, когда они применимы, представляют для пользователей определенные преимущества. По-видимому, ST-функции понятнее человеку, чем аналогичные функции поведения. Происходит это, вероятно, потому, что для ST-функций порождающие и порождаемые

состояния выбираются из одного и того же множества состояний, а для функций поведения — из двух разных множеств состояний. Совпадение порождающего и порождаемого множеств состояний также делает возможным использование удобных диаграмм как на рис. 14, 17, 18.

Для порождающих систем ИИ выделены различные методологические отличия. Это отличия, выделенные для систем более низких уровней, и некоторые новые. Среди первых наиболее существенными являются:

упорядоченность параметрического множества, что позволяет ввести важное понятие маски;

упорядоченность множеств состояний, что играет существенную роль в упрощении процедур для порождающих систем ИИ (см, разд. 4.9) и при работе с неполностью определенными наборами данных;

отличие четких и нечетких каналов наблюдения, дающих соответственно четкие или нечеткие данные и требующих применения различных методов обработки данных;

отличие между нейтральными и направленными системами ИИ, с которыми следует обращаться по-разному.

Методологическими отличиями, относящимися к порождающим системам ИИ, но не к системам данных и исходным системам ИИ, являются:

детерминированность и недетерминированность систем;

для недетерминированных систем различаются типы нечетких мер, характеризующих параметрически инвариантное ограничение на рассматриваемые переменные, в частности меры вероятности и возможности;

по используемой маске различаются порождающие системы без памяти и системы, зависящие от прошлого.

Эти методологические отличия характеризуют и системы более высоких эпистемологических уровней.

4.9. Упрощение порождающих систем ИИ

На некотором этапе обработки заданной системы данных часто желательно бывает упростить соответствующие этой системе данных порождающие системы ИИ. В некоторых случаях упрощения требует пользователь, для которого существующие порождающие системы ИИ оказываются слишком сложными для понимания. В других случаях упрощение требуется из-за предполагаемого использования

порождающих систем ИИ или по разным методологическим соображениям.

Существует два основных метода одновременного упрощения систем данных и соответствующих порождающих систем ИИ:

1) упрощение за счет исключения некоторых переменных из соответствующей подобной системы;

2) упрощение за счет определения классов эквивалентности состояний некоторых переменных.

Пусть множество переменных порождающей системы V состоит из n переменных и любое подмножество V, за исключением пустого множества, представляет содержательное упрощение первого рода. Следовательно, имеется 2^n—2 нетривиальных упрощения первого рода. Они частично упорядочены по отношению «подмножество». Если для удобства включить исходное множество V и пустое множество, то множество упрощений с частичным упорядочением образует решетку. Назовем эту решетку *решеткой переменных* или *V-решеткой* и обозначим $\mathscr{L}_V$. Понятно, что V-решетка может быть описана или как

$$\mathscr{L}_V = (\mathscr{P}(V), \subseteq),$$

или как

$$\mathscr{L}_V = (\mathscr{P}(V), \cap, \cup).$$

Обозначим через f_B функцию поведения заданной системы ИИ с поведением с переменными, составляющими множество V. При упрощении этой системы с помощью сокращения множества V до подмножества V' новая (упрощенная) функция поведения f_B определяется проекцией

$$f'_B(\beta) = [f_B \downarrow V'](\beta), \quad (95)$$

определенной уравнением (69).

Сокращения второго рода сводятся к уменьшению числа состояний, выделяемых для отдельных переменных. Одним из способов их описания является определение функции

$$\sigma_{i,j} : V_i \to V'_i, \quad (96)$$

где V_i — заданное множество состояний (переменной v_i); V'_i— упрощенное (сокращенное) множество состояний той же переменной; $\sigma_{i,j}(x)$ — новое состояние, присвоенное исходному состоянию x, а j— это идентификаторы, с помощью которых различаются разные функции вида (96), примененные к множеству состояний одной и той же переменной. Если $\sigma_{i,j}(x) = \sigma_{i,j}(y)$, то состояния x и y из V_i при упрощении оказываются неразличимыми. Функция (96) должна быть гомоморфна относительно всех математических свойств исходного множества V_i, которые считаются существенными с точки зрения

рассматриваемой задачи. Будем функцию (96), являющуюся гомоморфизмом в описанном выше смысле, называть *упрощающей функцией.*

Любая упрощающая функция индуцирует разбиение на множестве V_i. Используя стандартное обозначение будем это разбиение обозначать $V_i / \sigma_{i,j}$. Любое разбиение $V_i / \sigma_{i,j}$ состоит из групп состояний V_i, которые неотличимы при данном упрощении. Будем такое разбиение (которое сохраняет существенные свойства V_i) называть *разрешающей формой.* Разрешающие формы, определенные на каком-то множестве состояний V_i, могут быть упорядочены с помощью обычного отношения уточнения, определенного на разбиениях данного множества. Хорошо известно, что такое отношение уточнения является отношением частичного порядка и образует решетку. Для двух заданных разбиений, скажем X и Y, определенных на одном и том же множестве, будем говорить, что X является *уточненным разбиением* Y тогда и только тогда, когда для любой группы x из X существует группа y из Y, такая, что $x \subseteq y$. Если X, уточняющее разбиение Y, то Y называется *укрупненным разбиением X.* Решетку разрешающих форм, определенных на множестве состояний V_i, будем называть *разрешающей решеткой* и обозначать $\mathscr{L}_{V_i}$. Любая разрешающая решетка множества состояний V_i может быть определена или в виде

$$\mathscr{L}_{V_i} = (\{V_i/\sigma_{i,j}\}, \leqslant),$$

или в виде

$$\mathscr{L}_{V_i} = (\{V_i/\sigma_{i,j}\}, \times, +),$$

где × и + обозначают соответственно произведение и сумму разбиений. Если рассматриваемое множество состояний не обладает математическими свойствами, которые должны быть сохранены, то в качестве разрешающей формы приемлемо любое разбиение. В этом случае разрешающая решетка содержит все разбиения, которые могут быть определены на этом множестве состояний. Если множество состояний состоит из m состояний, то число разрешающих форм в решетке, скажем решетке Λ_m, определяется формулой

$$\Lambda_m = \sum_{i=0}^{m-1} \binom{m-1}{i} \Lambda_i, \quad \Lambda_0 = 1. \quad (97)$$

Ниже показано огромное число разрешающих форм даже для небольшого числа состояний:

m	2	3	4	5	6	7	8	9	10
Λ_m	2	5	15	52	203	877	4 140	21 147	115 975

Поскольку наименьшая уточненная разрешающая форма (все состояния в одном блоке) смысла не имеет, а наибольшее уточнение не дает упрощения, то число осмысленных упрощений равно Λ_m—2.
Если множество состояний полностью упорядочено и требуется сохранить эту упорядоченность при упрощениях, то число разрешающих форм существенно меньше числа, задаваемого формулой (97). Пусть $x_1, x_2, \ldots, x_m$ — это состояния и $x_k<x_{k+1}$ (k=1, ..., m—1). Тогда для любого $k \leq m$—1 x_k и x_{k+1} или объединяются в одну группу или нет. Только эти решения определяют конкретное разбиение. Таким образом, для m состояний принимается m—1 бинарное решение. Следовательно, для полностью упорядоченных множеств состояний

$$\Lambda_m=2^{m-1}. \qquad (98)$$

Очевидно, что эта решетка для m состояний изоморфна булевской решетке для упорядочения подмножеств любого множества из m—1 элемента. В следующей таблице приводятся значения Λ_m, вычисленные по формуле (98); в этом случае число разрешающих форм существенно меньше, чем в случае неупорядоченных множеств состояний:

m	2	3	4	5	6	7	8	9	10
Λ_m	2	4	8	16	32	64	128	256	512

Число содержательных упрощений снова равно Λ_m —2.

Пример 14. Пусть переменная, описывающая образование человека, имеет следующие состояния:
e— начальное образование;
h — полная средняя школа;
c — колледж;
g— ученая степень.
Понятно, что неравенство $e<h<c<g$ является естественным упорядочением этих состояний, и, следовательно, существует восемь разрешающих форм, решетка которых изображена на рис. 20,а в форме диаграммы Хассе.

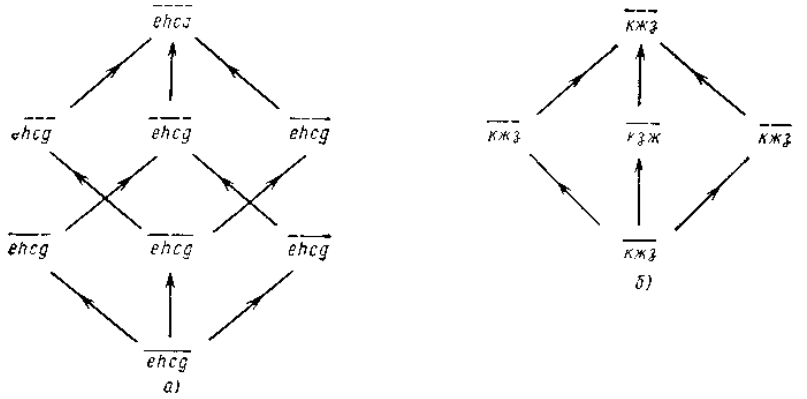


Рис. 20. Решетки разрешающих форм для полностью упорядоченного (а) и неупорядоченного *(б)* множеств (пример 14)

Группам в отдельных разрешающих формах (они показаны черточками над соответствующими буквами) можно дать соответствующие названия, например
eg — колледж или ученая степень;
kc — полная средняя школа или колледж;
eh— не выше средней школы;
ehc — любое образование, кроме ученой степени;
heg — образование выше начального.
Стрелка на диаграмме указывают направление уточнения разбиений. Для упрощения исходной системы нужно двигаться в направлении, обратном стрелкам.
Для сравнения рассмотрим переменную, состояниями которой являются цвета светофора: красный, желтый, зеленый. Поскольку они не упорядочены, все разбиения множества состояний приемлемы в качестве разрешающих форм. Диаграмма Хассе для этой решетки приведена на рис..20,б. Буквы *к, ж, з* означают соответственно красный, желтый и зеленый цвета.
Каждый элемент *V*-решетки определяет конкретный выбор переменных из исходной представляющей системы. Для любой выбранной переменной ее разрешающая решетка состоит из всех возможных разрешающих форм. Если выбрано несколько переменных, то любая разрешающая форма для одной переменной может быть объединена с любой разрешающей формой другой переменной. Все эти комбинации можно включить в одну решетку, представляющую выбранный набор переменных. Будем называть ее *объединенной*

разрешающей решеткой. Математически она представляет собой произведение отдельных разрешающих решеток. Оно определяется следующим образом:

пусть $X_1, X_2, \ldots, X_n$ — множества элементов отдельных разрешающих решеток выбранных переменных, а $\mathbf{X}$ — множество элементов соответствующей разрешающей решетки. Тогда

$$\mathbf{X} = X_1 \times X_2 \times \ldots \times X_n,$$

и для двух заданных n-ок

$$(x_1, x_2, \ldots, x_n)\,(y_1, y_2, \ldots, y_n) \in \mathbf{X}$$

мы определяем, что

$$(x_1, x_2, \ldots, x_n) \leqslant (y_1, y_2, \ldots, y_n)$$

тогда и только тогда, когда $x_j \leq y_j$ для всех разрешающих решеток (j=1, 2, ..., n). Понятно, что общее число элементов объединенной разрешающей решетки равно произведению числа элементов отдельных разрешающих решеток, т. е.

$$|\mathbf{X}| = \prod_{j=1}^{n} |X_j|,$$

однако только некоторые из них являются содержательными упрощениями. В частности, любая комбинация, в которую входит наименьшая уточненная разрешающая форма (разбиение на одну группу) одной из разрешающих решеток, является бессмысленной. Комбинация всех наиболее уточненных разрешающих форм также не представляет упрощения. Следовательно, общее число элементов объединенной решетки, представляющих содержательные упрощения, $|X_S|$ определяется по формуле

$$|X_S| = \prod_{j=1}^{n} (|X_j| - 1) - 1. \quad (99)$$

В частном случае, когда все отдельные решетки одинаковы и каждая состоит из Λ_m разрешающих форм, мы получаем

$$|X_S| = (\Lambda_m - 1)^n - 1. \quad (100)$$

Более того, если все разрешающие решетки построены на полностью упорядоченном множестве с m состояниями, то ;

$$|X_S| = (2^{m-1} - 1)^n - 1. \text{ A} \quad (101)$$

Пример 15. Предположим, что для упрощения выбраны две переменные, каждая с тремя состояниями 0, 1, 2. Предположим, также, что эти состояния полностью упорядочены: 0<1<2. Тогда отдельные

разрешающие решетки $(X_j, \leq)$ одинаковы и состоят из четырех разрешающих форм, как это показано на рис. 21.

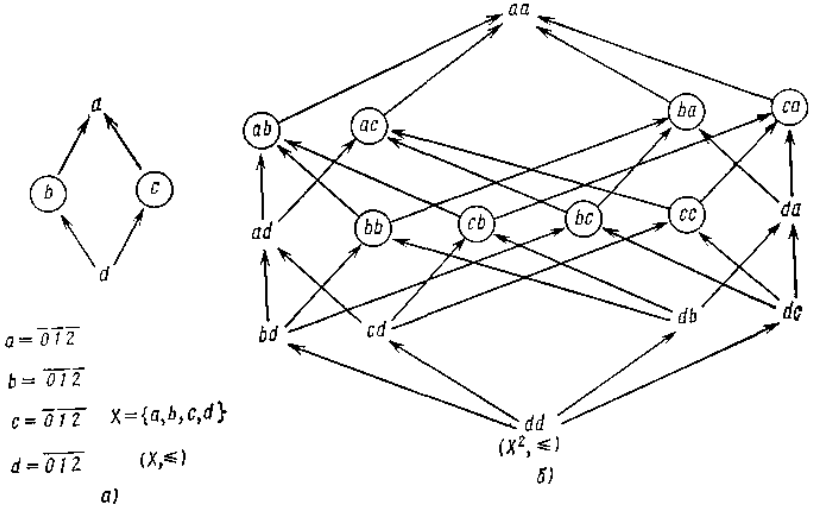


Рис. 21. Разрешающая $(x, \leq)$ (а) и объединенная разрешающая (б) $(x^2, \leq)$, решетка из примера 15

Разрешающие формы для осмысленных упрощений обведены на диаграмме Хассе кружками. Объединенная разрешающая решетка $(\mathbf{X}^2, \leq)$ приведена на рис 21,б. Она содержит 16 разрешающих форм, но осмысленными являются только, обведенные кружками.

Теперь предположим, что исходная система, которая должна быть упрощена, содержит n переменных $v_1, v_2, \ldots, v_n$, которым соответствуют множества $X_u X_2, \ldots, X_n$ разрешающих форм. Тогда общее число осмысленных упрощений (в которые входит и исключение переменных) $N(X_1, X_2, \ldots, X_n)$ определяется формулой

$$N(X_1, X_2, \ldots, X_n) = \prod_{j=1}^{n} |X_j| - 2. \quad (102)$$

Эта формула получена исходя из того, что одноблочное разбиение множества состояний (например, разбиение *d,* изображенное на рис. 21) может рассматриваться как исключение соответствующей переменной. Тогда осмысленным уточнением является любой элемент разрешающей решетки, за исключением наименее и наиболее уточненных объединенных разрешающих форм (на рис. 21, это *aa* и *dd).* Если все переменные имеют одно и то же множество

разрешающих форм, скажем множество **X**, то формула (102) значительно упрощается и принимает следующий вид:

$$N(X_1,\ X_2,\ \ldots,\ X_n) = |\mathbf{X}|^n - 2. \quad (103)$$

Если же, кроме того, множества состояний переменных полностью упорядочены и каждое состоит из m состояний, то

$$N(X_1,\ X_2,\ \ldots,\ X_n) = 2^{n(m-1)} - 2. \quad (104)$$

Если принято решение о том, какие упрощения данной системы с поведением следует выполнить, то функцию поведения упрощенной системы следует определять через функцию поведения данной системы. Если исключаются некоторые переменные, то сначала вычисляется проекция (95). Если требуются дальнейшие упрощения с помощью укрупнения разрешающих форм некоторых из оставшихся переменных, то должны быть сделаны изменения, аналогичные проекции. Сначала, как требуется, делается укрупнение разрешающих форм. Это дает группы состояний, неразличимых для новых разрешающих форм. Каждый блок заменяется одним состоянием. Вероятность (или возможность) этого состояния равна сумме вероятностей (или наибольшей возможности) всех состояний, входящих в группу.

Пример 16. Предположим, что требуется упростить вероятностную функцию поведения (см. табл. 6,а), исключив переменную v_1 и применим следующую упрощающую функцию к множествам состояний $V_2=V_3$ переменных v_2 и v_3:

$V_2 = I_3$	$I_2' = V_3'$
0	1
1	1
2	2

Таблица 6.

Упрощение функции поведения (пример 16)

а)

	v_1	v_2	v_3	$f_B(c)$
c =	0	0	0	0 20
	0	1	1	0.05
	0	2	2	0 04
	1	1	1	0.09
	1	1	2	0 06
	2	1	0	0 12
	2	1	1	0 10
	2	1	2	0 07
	2	2	0	0.15
	2	2	1	0 04
	2	2	2	0.08

б)

	v_2	v_3	$[f_B \downarrow \{v_2, v_3\}](\mathbf{x})$	
x =	0	0	0.20	Блок
	1	0	0.12	
	1	1	0 24	
	1	2	0.13	
	2	0	0.15	Блок
	2	1	0.04	
	2	2	0 12	

в)

	v_2	v_3	$f'_B(y)$
y =	1	1	0.56
	1	2	0 13
	2	1	0 19
	2	2	0.12

Сначала мы используем уравнение (69) для вычисления проекции $[f_в \downarrow \{v_2, v_3\}]$ (см. табл. 6,б). Затем мы идентифицируем блоки состояний, неразличимых после применения упрощающей функции (эти блоки показаны в табл. 6, б). И, наконец, мы суммируем вероятности всех состояний в каждой из групп и получаем упрощенную функцию поведения, представленную в табл. 3.6,в.

Упрощение систем ИИ представляет собой важный тип интеллектуальных задач. В общих чертах он может быть охарактеризован как процесс уменьшения определенной неким образом сложности системы ИИ некоторого эпистемологического уровня, в то же время сохраняющий возможно больший объем информации, содержащейся в системе ИИ. Все интеллектуальные задачи этого класса подходят под следующее общее описание:

конкретная система ИИ **x** определенного эпистемологического уровня;

множество систем ИИ того же типа Y_x, рассматриваемых как содержательные упрощения **x**;

набор требований Q, рассматриваемых как свойства систем ИИ из множества Y_x, определяющих подмножество Y_Q множества Y_x, такой, что любая система ИИ из Y_Q удовлетворяет всем требованиям из Q.

Для демонстрации класса интеллектуальных задач упрощения порождающих систем ИИ положим, что **x** — система ИИ с поведением, Y_x — множество всех содержательных упрощений **x**, базирующихся на том же множестве переменных, что и **x** (т. е. системах ИИ с поведением, базирующихся на всех содержательных объединенных разрешающих формах, выводимых из x без исключения переменных), а также положим, что Q состоит из двух требований:

1) чтобы системы ИИ из Y_Q были как можно проще;
2) чтобы степень порождающей нечеткости систем ИИ из Y_Q была как можно меньше.

Будем называть требование 1 *требованием простоты,* а требование 2 *требованием четкости.*

Чтобы конкретизировать требование простоты для систем ИИ с поведением следует задать определенную меру сложности. ФРИЗ должен давать пользователю возможность самому определять меру сложности, в том числе и по умолчанию. Пусть, например, сложность системы ИИ с поведением оценивается числом реальных состояний системы ИИ, т. е. числом состояний, имеющих ненулевые вероятности или возможности. Это очень простая мера, но, возможно, наиболее содержательная. В этом смысле эта мера является наиболее подходящим вариантом для меры по умолчанию. Будем использовать обозначение

$$f_{\text{В}} = |\{c| f_{\text{В}}(\mathbf{c})>0\}| \qquad (105)$$

для данной меры сложности системы с поведением $\mathbf{F}_B$, где $f_{\text{В}}$ — функция поведения системы.

Что касается требования четкости, то оно выражается или через вероятностную нечеткость $H(\mathbf{G}|\ \bar{\mathbf{G}}\ )$, определяем уравнением (45), или возможностную нечеткость $U(\mathbf{G}|\mathbf{G})$, определяем уравнением (64).

Пусть ${}^k q_u$ и $|{}^k f_{\text{в}}|$ $(k=1, 2, ...)$—соответственно значения порождающих нечеткостей и сложностей систем с поведением ${}^k\mathbf{F}_B$ из множества Y_x. Верхние индексы k можно интерпретировать как идентификаторы соответствующих объединенных разрешающих форм отдельных систем ${}^k\mathbf{F}_B$.

Численное упорядочение значений ${}^k q_u$ и $|{}^k f_{\text{в}}|$ задает упорядочение по нечеткости $\overset{u}{\leq}$ и упорядочение по сложности $\overset{c}{\leq}$ на множестве Y_x. В общем случае эти два порядка предпочтений противоречат друг другу. Очевидно, что и упорядочение по нечеткости и упорядочение по сложности рефлексивны, транзитивны и связны, но не антисимметричны. Следовательно, они представляют собой связные квазиупорядочения. Объединенный порядок предпочтений ${}^*\leq$ определяется следующим образом:

${}^i\mathbf{F}_B \overset{*}{\leqslant} {}^k\mathbf{F}_B$ тогда и только тогда, когда

$$ {}^i q_u \overset{u}{\leqslant} {}^k q_u \quad \text{и} \quad |\,{}^i f_B\,| \overset{c}{\leqslant} |\,{}^k f_B\,|\,, \qquad (106)$$

где ${}^i\mathbf{F}_B,\ {}^k\mathbf{F}_B \in Y_x$ — обобщенное квазиупорядочение (рефлексивное и транзитивное отношение) на Y_x.

Теперь можно определить множество решений $Y_Q \subset Y_x$, называемое *множеством допустимых упрощений* **x**, как множество всех систем из Y_x, таких, что они или эквивалентны, или несравнимы с точки зрения объединенного упорядочения (106). Формально

$$Y_Q = \{{}^i\mathbf{F}_B \in Y_x \mid (\forall {}^k\mathbf{F}_B \in Y_x)\,({}^k\mathbf{F}_B \overset{*}{\leqslant} {}^i\mathbf{F}_B \Rightarrow {}^i\mathbf{F}_B \leqslant {}^k\mathbf{F}_B)\}. \qquad (107)$$

После того как множество Y_Q допустимых упрощений заданной порождающей системы определено, пользователь может использовать все системы из Y_Q в качестве взаимодополняющих упрощений исходной системы, может выбрать одну самую подходящую или может воспользоваться дополнительными критериями для сокращения этого множества.

Задачи типа упрощения, как они описаны в данном разделе, имеют, разумеется, много разных вариантов, прежде всего из-за различных определений сложности и дополнительных требований. Интересно сравнить задачу этого типа с задачей определения подходящих систем с поведением для заданной системы данных, которая подробно рассматривалась в разд. 4.4 и 4.6. Задачи этих двух типов имеют определенное сходство, которое можно использовать при реализации ФРИЗ, т. е. некоторые процедуры, разработанные для одного типа задач, могут быть легко приспособлены для другого типа. Если разрешается упрощение заданной системы ИИ с поведением с помощью исключения переменных, то множество Y_x становится больше и некоторые введенные понятия должны быть соответствующим образом обобщены, однако основные результаты остаются неизменными.

Пример 17. Рассмотрим вероятностную систему ИИ с поведением, функция поведения которой и маска изображены на рис. 22,а, *б.*

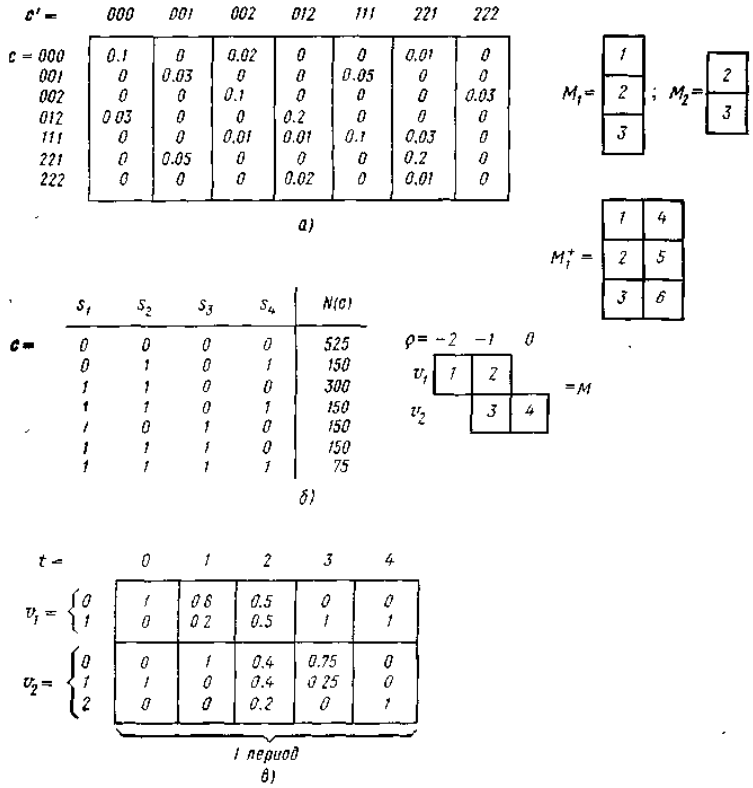


Рис. 22. Упрощение системы ИИ с поведением (пример 17)

Представляющая система ИИ очевидна. Предполагается, что параметрическое множество и все множества состояний полностью упорядочены. Воспользуемся этой системой, чтобы продемонстрировать задачу определения всех допустимых упрощений без исключения выборочных переменных.

Сначала мы должны определить множество Y_x всех допустимых упрощений. Поскольку имеются две базовые переменные, каждая с полностью упорядоченным множеством состояний, в которые входят по три состояния, то имеется восемь содержательных объединенных разрешающих форм. Каждая из форм, изображенных на рис. 21, представляет одно содержательное упрощение. Каждое уже определенное упрощение характеризуется значениями сложности и порождающей нечеткости. Эти значения приведены на рис. 22,в, на котором показано результирующее объединенное упорядочение по

сложности и нечеткости; на рис. 22,в используются те же метки, что и на рис. 21,б. Для сравнения приводятся и значения исходной системы, помеченные как *aa*.

При изучении диаграммы на рис. 22,в выделим три допустимых упрощения данной системы, основанные на разрешающих формах *ac, bc* и *cc*. Последние две системы одинаковы как по сложности, так и по нечеткости, и, следовательно, в данном примере упорядочение не является антисимметричным. Любопытно отметить, что упрощения для *ab* и *ba* повышают порождающую нечеткость, но при этом не снижают сложности заданной системы ИИ.

Функции поведения двух из трех допустимых упрощений с разрешающими формами *ac* и *cc* показаны на рис. 22,г и 22*д*. Мы не приводим подробности определения этих функций и вычисления значений на рис. 22, а предлагаем читателю самому провести по крайней мере некоторые вычисления.

4.10. Исследование и при проектирование систем ИИ

Для большинства задач, связанных с проектированием и анализом систем, характерно создание и работа с моделями реальных и теоретических явлений... Самое трудное — это создать такую структуру, такой контекст, в котором любые два решения задачи, независимо от того, насколько равными способами они были получены, можно было бы точно, объективно и всесторонне сравнить между собой.

А. У. Уэймор

Интеллектуальные задачи могут возникать в двух основных контекстах: при исследовании и при проектировании систем ИИ.

Задачей *исследования систем ИИ* является накопление знаний о различных наборах переменных и параметров, определенных с конкретными целями на существующих объектах.

Задачей *проектирования систем ИИ* является использование накопленных знаний для создания новых объектов, для которых на специфицированные переменные наложены определенные ограничения.

Несмотря на то, что интеллектуальные задачи как при исследовании систем ИИ, так и при их проектировании существуют на любом эпистемологическом уровне иерархии систем ИИ, в этом разделе мы

ограничимся общим рассмотрением интеллектуальных задач, связанных с исходными системами ИИ, системами данных и порождающими системами ИИ. Интеллектуальные задачи для систем более высоких уровней рассматриваются в гл. 5, 6.

Рассмотрим сначала некоторые вопросы, связанные с проектированием систем ИИ. Наиболее важной чертой проектирования систем ИИ является то, что параметрически инвариантное ограничение на не которые конкретные переменные определяется пользователем. Совершенно иначе обстоит дело с исследованием систем ИИ, где это ограничение неизвестно, и задача состоит в том, чтобы адекватно охарактеризовать с учетом конкретной цели исследования.

Ограничение при проектировании систем ИИ определяется или явно на языке конкретной порождающей, обычно направленной системы ИИ, или неявно на языке системы данных. В первом случае задача проектирования сводится к определению набора структурированных систем ИИ, удовлетворяющих заданным требованиям. Эти вопросы рассматриваются в гл. 5. Во втором случае необходимо определить некие порождающие системы ИИ, адекватно описывающие ограничения, содержащиеся в данных. Эта задача соответствует классу задач, рассматриваемых в разд. 4.4 и 4.6 в контексте исследования систем, однако в случае проектирования систем ИИ система данных по определению содержит всю информацию о способе, каким накладываются ограничения на переменные.

При проектировании системы ИИ функция данных часто определяется неявно через описание их свойств, а не явно в виде матрицы или массива данных. Допустим, например, что имеется простая направленная система ИИ с одной входной переменной, множество состояний которой состоит из 26 латинских букв и пробела, и с одной выходной переменной с двумя состояниями 0 и 1. Входная переменная определяется последовательностью букв и пробелов просматриваемого английского текста. Требуется, чтобы выходная переменная при определенных условиях была равна 1, например, при условии, что последнее слово просматриваемого текста кончалось на ING, и 0 в противном случае. **Задача состоит в том, чтобы преобразовать это неявное определение системы данных в некую порождающую систему ИИ, которая бы для любого английского текста порождала бы (детерминированным образом) требуемые состояния выходной переменной. Методы решения задач подобного типа хорошо разработаны в рамках теории конечных автоматов.** Соответствующий набор таких методов должен быть включен в состав ФРИЗ.

При сравнении исследования систем ИИ и их проектирования на уровне систем данных и порождающих систем ИИ, необходимо отличать два класса систем данных, встречающихся при исследовании систем ИИ. К первому классу относятся системы данных, в которых переменные не имеют смысла вне параметрического множества, на котором они определены. Примерами таких систем являются:

музыкальное сочинение, рассматриваемое как система данных, переменные которой, очевидно, не имеют смысла вне временного множества, соответствующего всему сочинению;

любая система данных с пространственным параметром, в которой пространственное множество не может быть расширено, например система пространственных данных по акустике концертного зала или система, определенная для земного шара, в котором параметрическим множеством являются значения широт и долгот для всего земного шара;

любая система данных, определенная на всей группе определенного типа, например все сочинения какого-либо композитора, все служащие определенного нанимателя и др.

Системы данных такого типа содержат всю информацию об ограничениях на их переменные. Тем самым они методологически подобны системам данных, определяемым при проектировании систем ИИ. Будем такие системы называть *полными системами данных.*

Второй класс систем ИИ, который при исследовании, по-видимому, встречается чаще, составляют системы ИИ, в которых переменные не ограничены тем параметрическим множеством, для которого имеются данные. Можно говорить о том, что практически все системы ИИ, параметром которых является время, относятся к этому классу (пример с музыкальным сочинением — редкое исключение). Примеры полных систем данных для параметров других типов, хотя и встречаются чаще, также не являются типичными.

Существует два основных метода исследования систем ИИ. При одном подходящие порождающие системы ИИ (или системы ИИ более высоких уровней), базирующиеся на определенных требованиях, выводятся из заданной системы данных. Для наиболее типичных требований этот процесс рассматривается в разд. 4.4 и 4.6. Этот метод обычно называется *методом открытия.* При другом методе гипотетическая порождающая система ИИ (или система ИИ более высокого уровня) постулируется, а затем ее правильность проверяется сравнением порождаемых ею (при соответствующих начальных условиях) данных с эмпирическими данными. Если система ИИ не проходит проверки, основанной на некоем конкретном критерии правильности (критерии совпадения), то она отвергается и посту-

лируется новая система. Этот подход к исследованию систем ИИ обычно называется *методом постулирования.*

Понятно, что при использовании метода открытия любая порождающая система ИИ, полученная непосредственно из системы данных, является неким экономным представлением каких-то аспектов системы данных. То, какие именно аспекты представляются порождающей системой ИИ, зависит от ее маски и характера функции поведения или ST-функции. Если порождающая система ИИ детерминирована, то это экономное описание всей системы данных своего рода «стенографическое» описание.

Если система данных является полной, то метод открытия сводится к нахождению моделей ее данных. Обнаруженные модели данных могут затем использоваться для разных целей. Если система ИИ неполная, то необходимо различать две проблемы, связанные с найденными моделями (т. е. с полученными подходящими системами данных): *объяснение* данных в рамках заданного параметрического множества, вывод данных вне пределов параметрического множества, т. е. *предсказание, восстановление (retrodiction)* или *обобщение* данных. Разница между этими двумя проблемами, которые часто в литературе смешиваются, хорошо описана в одной из статей Г. Саймона: «Открытие закона означает только нахождение модели данных; будет ли модель выполняться для новых данных, наблюдаемых впоследствии, решается в процессе проверки закона, а не при его открытии... **Открытие начинается с конкретных фактов и ведет к общим законам, которые каким-то образом выводятся из этих фактов**; **процессы проверки открытий эволюционируют от законов к предсказаниям конкретных фактов на основе этих законов.** Тот факт, что некий процесс выявляет модель из конечных наборов данных, ничего не говорит о предсказывающей силе такой модели для новых наблюдений. **Двигаясь от обнаружения моделей к предсказанию, мы движемся от теории процессов открытия к теории процессов проверки законов**. Для объяснения того, почему определенные нами на основе наблюдений модели часто дают правильные предсказания (если это так), необходимо обратиться к **проблеме индукции** и, возможно, выдвинуть некие гипотезы относительно единообразия природы. Однако теория процессов открытия в этой гипотезе не нуждается. Эта теория утверждает, что данные образуются согласно неким моделям, и скорее показывает, как определяется модель, если она существует. Это вопрос не описания или психологии, а нормативный и логический**. Отделяя задачу обнаружения модели от задачи предсказания, мы можем построить подлинную нормативную теорию открытия — логику открытия.»**

Если метод открытия используется для неполных систем данных, то порождающие системы ИИ (или системы ИИ более высоких уровней) определяются на столько для объяснения имеющихся данных, сколько для расширения данных за пределы заданного параметрического множества, что делает возможным предсказание, восстановление и обобщение данных. **Этот процесс требует применения индуктивного рассуждения некоего типа. Следовательно, ФРИЗ должен располагать пакетом хорошо обоснованных методологических средств для индуктивных рассуждений.** Соответствующие вопросы обсуждаются в гл. 5.

Задачи определения подходящих порождающих систем ИИ рассматривались в разд. 4.4 и 4.6 в неявном предположении, что выборочные переменные определены только через переменные, включенные в заданную систему данных, т. е. через *наблюдаемые переменные.* Это ограничение не обязательно и может в некоторых случаях затруднить получение достаточно простых порождающих систем ИИ с незначительной порождающей нечеткостью или вовсе без нечеткости. Эти задачи могут быть обобщены, если разрешить пользователю постулировать гипотетические состояния некоторых дополнительных переменных, не входящих в число наблюдаемых переменных. Такие переменные обычно называются *внутренними переменными,* а их состояния — *внутренними состояниями.*

Несмотря на то, что гипотетические внутренние состояния могут вводиться из самых разных соображений, обычно они вводятся для усиления зависимости между порождающей нечеткостью и сложностью подходящих порождающих систем ИИ. При введнии внутренних состояний требуется, чтобы для заданных данных была определена модель порождения этих состояний. В то же время эти переменные должны способствовать уменьшению общей порождающей нечеткости. Подобное определение моделей возможно только для полных систем данных и изучается в рамках **теории конечных автоматов, детерминированных и вероятностных**.

Понятия внутренних переменных и состояний весьма существенны при проектировании систем ИИ. Введение внутренних состояний в процессе проектирования систем ИИ сводится к соответствующему переопределению накладываемых ограничений. После их введения на абстрактном уровне внутренние переменные и их состояния могут быть конкретизированы любым подходящим способом. Однако при исследовании систем ИИ использование внутренних переменных довольно проблематично, поскольку они не несут семантической нагрузки и нельзя, как при проектировании систем ИИ, конкретизировать их подходящим образом.

Таким образом, **проектирование систем ИИ всегда представляет собой процесс подъема по эпистемологической иерархии систем.** Он начинается с определения или порождающей системы ИИ, или системы данных и набора требований относительно структуры систем. Задача определения подходящих порождающих систем ИИ по заданной системе данных принадлежит к тому же классу задач, что и задачи, обсуждаемые в разд. 4.4 и 4.6, с той лишь разницей, что допускается использование внутренних переменных. Исследование систем ИИ осуществляется с помощью:

подъема по иерархии посредством обнаружения систем ИИ более высоких уровней, для которых системы ИИ более низких уровней обладают определенными свойствами, и, если система данных неполная, соответствующих индуктивных выводов (метод открытия);

постулирования порождающих систем ИИ или систем ИИ более высокого уровня и отбрасывания тех из них, которые не удовлетворяют проверке на соответствие между эмпирическими и порожденными данными (метод постулирования);

любой комбинации метода открытия и метода постулирования, например подъема по иерархии до определенного уровня и постулирования систем ИИ на более высоком уровне.

Наибольшее внимание уделяется нами задачам, связанным с методом открытия. Объясняется это двумя соображениями. Метод открытия, при котором системы ИИ вводятся в порядке возрастания их концептуальной сложности, очень удобен для объяснения и формулирования всей концептуальной схемы ФРИЗ. Второе соображение заключается в том, что метод открытия еще недостаточно полно описан в литературе, методы постулирования и проектирования систем освещены вполне удовлетворительно.

В заключение рассмотрим некоторые замечания.

1. Понятия маски и выборочных переменных, центральные понятия для порождающих систем, введены А. Свободой в начале 1960-х гг. для проектирования и классификации переключательных схем , а затем использованы при создании сложной и нетрадиционной методологии работы с переключательными схемами

2. Литература по теории вероятностей обширна. Классическая книга А. Н. Колмогорова до сих пор остается наилучшим аксиоматическим исследованием теории вероятностей. Хороший сравнительный обзор различных аксиоматических оснований и интерпретаций теории вероятностей дается в книге Т. Файна .

3. Формула (32) выведена Р. Кристенсеном исходя из принципа максимума энтропии, являющегося одним из принципов индуктивного рассуждения, рассматриваемого в гл. 5. Он также получил обобщенную формулу для оценочных вероятностей, учитывающую любую дополнительную информацию. Формула (34) получена Дюбуа и Прадом. Ведутся работы по нахождению всех подходящих функций преобразования распределений частот в распределения возможностей.

4. Идея возможностнои меры, которая является основой теории возможностей, была выдвинута Л. Заде в 1978 г. Возмомностные меры образуют лишь малое подмножество множества нечетких мер, введенного М. Суджено в 1977 г.. Это множество не пересекается с множеством вероятностных мер, которые также образуют класс нечетких мер. Пури и Ралеску показали, что возможностные меры могут быть определены только на конечных множествах и некоторых специальных классах бесконечных множеств.

5. Как упоминалось в разд. 4. 5, понятие условной возможности является опорным вопросом в теории возможностей. Противоречие возникает из отношения понятий невзаимодействия (noninteraction) и независимости. Ясно, что в теории вероятностей эти понятия эквивалентны. Пусть $x \in X$ и $y \in Y$, где X и Y — конечные множества событий. Вероятностное невзаимодействие определяется как $p(x, y)=p(x)\cdot p(y)$ для всех $x \in X$ и $y \in Y$, где $p(x, y)$ —совместная вероятность событий x и y. Вероятностная независимость определяется как $p(x|y) =p(x)$ и $p(y|x)=p(y)$, где $p(x|y)$ —условная вероятность x при заданном y, а $p(y|x)$ —условная вероятность y при заданном x. Поскольку совместная вероятность событий $p(x, y)$ определена как $p(x, y)=p(x|y)\cdot p(y)=p(y|x)\cdot p(x)$, то множества событий X и Y независимы тогда и только тогда, когда они не взаимодействуют. В теории возможностей вовсе не очевидно, что невзаимодействие и независимость являются эквивалентными понятиями. В литературе на этот счет высказываются две точки зрения. Два конечных множества событий X и Y с распределениями возможностей соответственно $f(x)$ и $f(y)$ ($x \in X$, $y \in Y$) называются невзаимодействующими тогда и только тогда, когда

$$f(x, y) = \min [f(x), f(y)] \quad (108)$$

для всех $x \in X$ и $y \in Y$. Эти множества называются независимыми тогда и только тогда, когда

$$f(x|y)=f(x), \quad (109)$$

$$f(y|x)=f(y) \quad (110)$$

для всех $x \in X$ и $y \in Y$, где $f(x|y)$ и $f(y|x)$ —условные возможности соответственно x при заданном y и y при заданном x. Утверждается, что уравнения

$f(x, y) = \min [f(y), f(x|y)]$, (3.111) $f(x, y) = \min [f(x), f(y|x)]$ (112)

должны выполняться для любых двух множеств событий, характеризующихся распределениями возможностей $f(x), f(y)$. Тогда ясно, что как из (109) и (111), так и из (110) и (112) следует (108). Таким образом, из независимости возможностных событий следует их невзаимодействие. Однако обратное неверно. В самом деле, из (108) и (111) мы получаем

$$f(x \mid y) = \begin{cases} f(x), & \text{если } f(x) < f(y), \\ [f(x), 1], & \text{если } f(x) \geqslant f(y) \end{cases} \quad (113)$$

(и аналогичное выражение для другой условной возможности). Следовательно, из невзаимодействия вовсе не следует независимость. В ряде работ смысл условной возможности понимается совершенно иначе. В этих работах «нормализованные» условные возможности определены таким образом, чтобы по аналогии с теорией вероятностей возможностное невзаимодействие было эквивалентно возможностной независимости. Это следует из формулы

$$f(x \mid y) = \begin{cases} f(x, y), & \text{если } f(x) \leqslant f(y), \\ f(x, y)\ \dfrac{f(x)}{f(y)}, & \text{если } f(x) > f(y), \end{cases} \quad (114)$$

где $f(x)/f(y)$—нормализующий коэффициент.

6. Шенноновская энтропия, являющаяся естественной мерой нечетности и информации для событий, описываемых вероятностными распределениями, занимает главенствующее место в литературе с тех пор, как она была предложена К. Шенноном в 1948 г. Сначала она была введена для анализа систем связи, но ее значение и применения вышли далеко за пределы ее первоначального назначения. Эффективность введения шенноновской энтропии для общесистемной методологии, вероятно, одного из важнейших ее применений, была оценена существенно позже других ее применений. Сначала единственным сторонником такого применения шенноновской энтропии был Р. Эшби. Несмотря на то, что первые идеи в этой области были опубликованы еще в 1956 г. , он вернулся к данной теме, получив значительно более сильные результаты через 10 лет.
Литература по теории информации, базирующейся на шенноновской энтропии, очень разнообразна.

7. Общепризнано, что впервые мера информации и нечеткости была введена Р. Хартли в 1928 г.. Он определил информацию, необходимую для описания элемента конечного множества из n элементов как

двоичный логарифм n. Этой мере часто приписывается одна из двух вероятностных интерпретаций. Согласно первой она рассматривается как специальная мера, различающая только нулевые и ненулевые вероятности, т. е. мера, нечувствительная к реальным значениям вероятностей. При второй интерпретации эта мера рассматривается как шенноновская энтропия при условии, что все элементы множества имеют равные вероятности. Подобные попытки классифицировать меру Хартли как шенноновскую энтропию некорректны, поскольку **энтропия логически независима от вероятностных условий**. На самом деле она несопоставима с шенноновской энтропией, поскольку предполагает выполнение такого свойства, как монотонность (чем больше множество, тем больше его информация по Хартли), что неприменимо к распределениям вероятностей (их нельзя упорядочить аналогичным образом).
Информация по Хартли представляет собой частный случай возможностной информации, выражающейся U-нечеткостью из уравнения (56) или (57), для четких распределений возможностей (с возможностями, равными только 0 или 1). Также доказывается, что U-нечеткость удовлетворяет возможностным аналогам всех аксиом шенноновской энтропии (они перечислены в разд. 4.5) и, кроме того, обобщенному свойству монотонности (чем больше распределение возможностей, тем больше его U-нечеткость). Противоречие, связанное с условными возможностями (см. замечание 5), снимается требованием того, чтобы класс неинтерактивных множеств выходов, относящийся к классу независимых множеств выходов, удовлетворял требованию аддитивности. Условная U-нечеткость (см. уравнение (64)) определяется без использования противоречивого понятия условной возможности

8. Некоторые методы, возникшие в теории конечных автоматов (детерминированных и вероятностных), предназначены для частных случаев задач вывода подходящих порождающих систем из полных систем данных. Эти методы, как правило, применимы только к направленным системам и используют внутренние состояния. Поэтому данные методы подходят для проектирования систем, их использование при исследовании систем весьма ограничено. Один конкретный результат из теории автоматов заслуживает особого внимания как пример превосходного взаимодействия между пользователем и ФРИЗ при решении задачи. При этом подходе описание переходов из состояния в состояние для детерминированного конечного автомата с внутренними состояниями строится с помощью вопросов к пользователю относительно системы данных, причем алгоритм построен так, что пользователь отвечает только, возможна

или нет некоторая последовательность входов-выходов. После завершения процесса ответов на вопросы (за конечное число шагов, если данные полны) ST-система определяется алгоритмически в соответствии с полученными ответами. Таким образом, при этом подходе не требуется, чтобы пользователь сам полностью определял свою задачу, алгоритм помогает ему сделать это.

9. Для систем с непрерывными переменными и параметрами параметрически инвариантные ограничения на переменные описываются в общем случае дифференциальными уравнениями с постоянными коэффициентами. Понятно, что в данном случае выборочными переменными являются производные, определенные через базовые переменные и правила сдвига. **Производная самого высокого порядка в дифференциальном уравнении является аналогом глубины маски.** Решения дифференциальных уравнений при различных начальных условиях имеют смысл порождаемых данных. **Эмпирические данные представляют собой непрерывные функции на параметрическом множестве.**

Пусть, например, функция $v(t)=sinf$ представляет эмпирические данные для некоторой области t, где v — переменная, а t — время. Тогда $\dot{v}(t)=cost$ и $\ddot{v}(t) = = —\sin t$. Следовательно, дифференциальное уравнение

$$\ddot{v}(t)+v(t)=0$$

является инвариантным относительно времени характеристикой переменной v, так как она одинакова для любого значения t. При решении этого дифференциального уравнения при соответствующих начальных условиях получаются исходные функции.

10. Понятие энтропии первоначально было предложено Больцманом в виде

$$H(f(x) \mid x \in [a, b]) = -\int_a^b f(x) \log f(x)\, dx,$$

где f— плотность распределения вероятностей, определенная для непрерывной переменной $x \in [a, b]$. Несмотря на то, что по форме она похожа на шенноновскую энтропию, энтропия Больцмана не является аналогом шенноновской энтропии для непрерывных переменных, как этого можно было бы ожидать. На самом деле, энтропия Больцмана не является предельным случаем шенноновской энтропии и, соответственно, не является мерой нечеткости и информации. Однако в виде

$$H_B\left(\frac{f(x)}{g(x) \mid x \in [a, b]}\right) = \int_a^b f(x) \log f(x)\, dx$$

она становится аналогом шенноновской кросс-энтропии

$$H_S\left(\frac{f(x)}{g(x) \mid x \in x}\right) = \sum_{x \in X} f(x) \frac{\log f(x)}{\log g(x)},$$

где X — конечное множество состояний переменной x, а f, g — функции распределения вероятностей, определенные на X.

5. Структурированные системы ИИ

5.1. Целое и части

*При установке порядка появились имена. Поскольку возникли имена, нужно знать предел их употребления. Знание предела позволяет избавиться от опасности. (*Лао Цзы)

Определение порождающей системы ИИ (или множества подходящих порождающих систем ИИ), рассматриваемое в гл. 4, это теоретически только первый этап исследования систем ИИ. При введении более высоких эпистемологических уровней возникают новые проблемы. Данная глава посвящена задачам, связанным со структурированными системами ИИ.

Структурированная система ИИ, если говорить в общих чертах, представляет собой набор исходных систем ИИ, систем данных или порождающих систем ИИ, **имеющих общее параметрическое множество**. Системы ИИ, образующие структурированную систему ИИ, будем называть ее *элементами.* Некоторые переменные у них могут быть общими. Общие переменные будем называть *связывающими переменными.* Они представляют взаимодействия между элементами. Естественно называть эти три типа систем ИИ *структурированными исходными системами, структурированными системами данных* и *структурированными порождающими системами.* Для некоторых задач удобно также выделить более частные типы структурированных систем ИИ, например *структурированные представляющие системы ИИ, структурированные системыИИ с поведением или структурированные ST-системы.*

Для заданной структурированной системы ИИ одного из этих типов существует связанная с ней система ИИ, определяемая *всеми* переменными, входящими в ее элементы. Эта система ИИ (предполагается, что она того же типа, что и элементы структурированной системы ИИ) рассматривается как некая *полная система ИИ,* т. е. система ИИ,

представляющая в виде некоторого целого все входящие переменные. С этой точки зрения элементы любой структурированной системы ИИ интерпретируются как *подсистемы ИИ* соответствующей полной системы ИИ, а полная система ИИ — как *суперсистема ИИ* этих элементов. При этом структурированные системы ИИ становятся, по существу, представлениями полных систем ИИ в виде различных подсистем ИИ.

Статус системы ИИ как полной системы ИИ или подсистемы ИИ не является, разумеется, абсолютным. Например, некая система ИИ с поведением в одном контексте может рассматриваться как элемент структурированной системы ИИ (и, следовательно, как подсистема ИИ полной системы ИИ с поведением), а в другом — может рассматриваться как полная система ИИ, подсистемы которой образуют структурированную систему ИИ. Любая исходная система ИИ, система данных или порождающая система ИИ существует как бы в двух «лицах». В одном контексте она имеет статус подсистемы ИИ, а в другом — статус суперсистемы ИИ. Можно, таким образом, говорить не только 0 том, что «часть — это амплуа целого» (как предлагает Р. Гленвилл), но и о том, что целое—это ампула части. Подобная двойственность дает возможность представить любую полную систему ИИ как *иерархию структурированных систем ИИ,* т. е. как структурированную систему ИИ, элементами которой являются структурированные системы ИИ, элементами которой также являются... и т. д. вплоть до элементов, состоящих из отдельных переменных.

На заданном уровне иерархии конкретная система ИИ может рассматриваться и как внешняя (для систем расположенных ниже ее), и как внутренняя (для систем, расположенных выше); таким образом, **статус (т. е. отличительный знак)** данной системы меняется при переходе через ее уровень вверх или вниз. Выбор более высокого или более низкого уровня рассмотрения связан с тем, интерпретируется данная система как автономная или управляемая (ограниченная).

Зачем нужно представлять полную систему ИИ как совокупность ее подсистем ИИ? Причин несколько. **Одна из них связана с наблюдением или измерением.** Если в параметры входит время, то часто бывает технически невозможно или, по крайней мере, неразумно одновременно наблюдать (измерять) все переменные, имеющие отношения к цели исследования. В этом случае можно собрать данные только частично, для наибольшего возможного подмножества переменных. В других случаях исследователь вынужден использовать чужие данные, собранные различными организациями или

последователями для собственных нужд и покрывающие только часть переменных, необходимых ему для работы.

Другой причиной структурирования систем ИИ является сложность, которая в свою очередь, связана, с обозримостью рассматриваемой системы ИИ. Одной из характеристик системы ИИ является объем памяти компьютера, необходимыи для хранения системы ИИ. Рассмотрим, например, n переменных с k состояниями каждая. При работе с полной системой ИИ этих переменных для запоминания ее состояния нужно располагать nk^n ячейками памяти, если хранить в ячейке по одному из k состояний. Кроме того, если использовать структурированную систему ИИ, состоящую из всех подсистем ИИ с двумя переменными, то для той же цели понадобится только $k^2n(n—1)$ ячеек памяти. С ростом значений k и n это число, как показано на рис. 1 для k =10, растет значительно медленнее, чем nk^n.

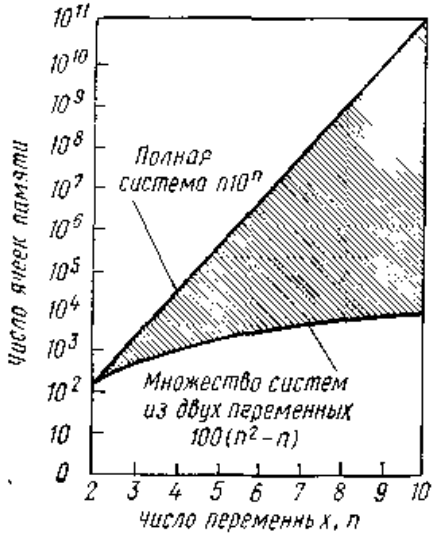


Рис. 1. Емкость памяти ЭВМ, необходимая для представления полной системы и соответ ствующей структурированной системы ($k=10$)

Если же структурированная система ИИ содержит только некоторые из двух переменных систем ИИ, это сравнение будет еще более разительным. Несмотря на то, что при малых n и k структурированные системы ИИ могут потребовать большего объема памяти, чем соответствующие полные системы ИИ, ясно, что в основном для

важных практических случаев, особенно при больших *k* и *n,* их требования к памяти существенно ниже.
Другой аспект возможности обработки систем связан с числом систем ИИ, которые должны быть рассмотрены в некоторых задачах Для сравнения чисел полных и структурированных систем ИИ определенного типа снова рассмотрим *n* переменных с *k* состояниями. Кроме того, будем отличать, является ли любое состояние системы ИИ возможным. Тогда имеется 2^{k^n} возможных полных систем ИИ, *n(n—1)* 2^{k^2-1} возможных структурированных систем ИИ, состоящих только из бинарных (состоящих из двух переменных) подсистем ИИ, и *n* 2^{k^2} возможных структурированных подсистем ИИ, состоящих только из *n* бинарных подсистем ИИ. Несмотря на то, что все эти числа достаточно велики, чтобы можно было бы говорить о полном переборе вариантов даже для небольших *n* и *k,* число структурированных систем ИИ (в обоих случаях) растет заметно медленнее, чем число возможных полных систем ИИ. Например, при *n*=10 и *k = 2* структурированных систем ИИ, содержащих все бинарные подсистемы ИИ, 720, а возможных полных систем ИИ 10^{308} (т. е. находится за пределом Бремерманна, рассматриваемом в разд. 7.4). Таким образом, в общем случае легче осуществлять поиск на множестве всех возможных структурированных систем ИИ определенного типа, чем на множестве всех возможных полных систем ИИ, хотя и в том, и другом случае часто бывают неизбежны некоторые ограничения.
Имеется много соображений в пользу применения структурированных систем ИИ. Некоторые из них связаны с обозримостью процесса проектирования. Основные из этих соображений уже обсуждались. Другие связаны с наличием ограниченного набора подходящих **готовых элементов (модулей),** с эффективностью реализации, а также с различными вопросами надежности, проверяемости и ремонтопригодности проектируемой системы ИИ.
Практические соображения, связанные с обозримостью задачи, эффективностью, ремонтопригодностью и т. п., не единственные соображения, по которым желательно использовать структурированные системы ИИ. В исследовании систем структурированные системы ИИ имеют более фундаментальное значение. Соответствующим образом обоснованная структурированная система ИИ дает исследователям сведения, не содержащиеся, по крайней мере явно, в соответствующей полной системе ИИ. Эти дополнительные сведения могут помочь ответить на определенные вопросы, возникающие в процессе исследования, помочь лучше разобраться в задаче.

Со структурированными системами ИИ связана одна из самых спорных философских проблем — **проблема взаимоотношения между *целым* и *частями.*** Эта проблема рассматривается не только в древнегреческой философии, но и в значительно более древней китайской философии.
Нет проблемы более важной для понимания природы существования, знания, ценностей или логики, чем проблема природы целого и его частей и их взаимоотношений.
Совершенно ясно, когда мы говорим «часть», то имеем в виду «часть целого», а под «целым» подразумеваем «целое, состоящее из частей». В этом смысле нет частей, не являющихся частями целого, и нет целого, не состоящего из частей. Целое и части взаимосвязаны; **каждое понятие зависит от того, что представляет собой другое и в то же время одно не есть другое.** Часть целого не есть целое, а целое, состоящее из частей не является ни одной из своих частей.
Однако проблемы в понимании того, как соотносятся друг с другом целое и части, приводят к появлению теорий, по-видимому, отрицающих или, по крайней мере, модифицирующих первоначально ясные понятия Некоторые трудности возникают также из-за того, что существуют разные типы целого и разные отношения часть-целое. Гоген и Варела предлагают четыре альтернативных критерия оценки целостности системы:
«Интересно посмотреть, как можно оценить степень целостности системы. Всегда, разумеется, можно нечто выделить и назвать «системой», но это нечто не всегда окажется тождественным понятием «цельная система», «естественное единство», «связанный объект» или «понятие». Что же делает одни системы более связными, более естественными, более «цельными», чем другие? ... **Согласно одному подходу**, **полнота — это способность к соответствующему отображению существенных *новых свойств*... С другой точки зрения полнота измеряется степенью сложности сокращения системы.. . Третий подход состоит в том, что система считается настолько полной, насколько ее части *взаимосвязаны,* т. е. насколько трудно найти относительно независимые подсистемы. .. Согласно чертвертому подходу система тем полнее, чем она *сложнее,* т. е. чем труднее свести ее к описаниям взаимосвязей компонентов более низкого уровня.»**
Философская контроверза «часть-целое» нашла свое отражение в противопоставлении двух научных методологий — *редукционизма* и *холизма* (**от греческого *holos,* что значит целый**). Редукционизм опирается на следующий тезис: свойства целого объяснимы через свойства составляющих его элементов. Холизм же отрицает этот тезис

и утверждает, что нельзя без потерь анализировать целое с точки зрения его частей. Это утверждение часто формулируется в виде известного высказывания «целое больше суммы составляющих его частей».
В схеме ФРИЗ дихотомия целого и частей выражается двойственной ролью исходных систем ИИ, систем данных и порождающих систем ИИ, являющихся одновременно и суперсистемами ИИ и подсистемами ИИ. Различные вопросы, связанные с взаимоотношением целого и частей, которые часто бывают окружены некой таинственностью, могут быть на самом деле четко сформулированы в виде интеллектуальных задач и изучаться соответствующим образом. При этом две методологические доктрины оказываются взаимодополняющими:
«В большинстве исследований холизм и редукционизм занимают полярные позиции. Это, вероятно, является следствием исторически сложившегося размежевания между эмпирическими науками, но большей части редукциональными и аналитическими, и европейскими школами философии и общественных наук, пытающимися нащупать динамику общностей.
Обе эти позиции вполне допустимы на определенном уровне описания и, по существу, дополняют друг друга. С одной стороны, можно спуститься на более низкий уровень и изучать свойства компонент, не принимая во внимание их системные взаимосвязи. С другой стороны, можно, не обращая внимания на структуру компонент, исследовать их поведение только с точки зрения их вклада в поведение большей единицы. Оба направления анализа всегда, вероятно, явно или неявно существуют, поскольку для наблюдателя оба эти уровня описаний взаимосвязаны. Невозможно представить себе компоненты, если нет системы, из которой они абстрагированы, и нет целого без составляющих его частей...
Эти уровни описания по большей части не представляются явно как взаимодополняющие в основном из-за того, что в большинстве областей науки существует различие между описываемой методологией и практикой. Позиция редукционалиста достаточно сильна, однако анализ системы не может быть начат без знания степени связности исследуемой системы; аналитик индуитивно должен представлять себе, что он имеет дело с целостным явлением. Несмотря на то, что официально наука стоит на позиции редукционализма, *на практике* используются оба подхода. Нельзя быть чистым холистом или редукционалистом: эти точки зрения на системы являются взаимодополняющими... **Редукционализм занимается более низким уровнем, а холизм — более высоким**. В любом достаточно полном

описании они переплетены, и каждый подход имеет свои достоинства и недостатки.»
Следует тонко балансировать между частями и целым. Нельзя находиться ни в одной из крайностей. Это балансирование должно продолжаться бесконечно.
Со структурированными системами ИИ связаны некоторые наиболее важные типы интеллектуальных задач. Это типы задач, имеющие в основном операционные формулировки на языке ФРИЗ, и связанные с вопросами взаимоотношений между целым и частями. Некоторые из них относятся к исследованию, а некоторые к проектированию систем ИИ: одни возникли из практики, другие имеют теоретическое значение или затрагивают определенные философские вопросы. В этой главе определяются структурированные системы ИИ различных типов и рассматриваются некоторые связанные с ними ключевые интеллектуальные задачи.

5.2. Системы, подсистемы, суперсистемы ИИ

В гл. 1 и 4 были определены различные системы ИИ трех эпистемологических уровней. Для двух заданных систем ИИ одного из этих типов часто бывает нужно определить, соотносятся ли эти системы как часть и целое Однако для того, чтобы это можно было сделать, в схеме ФРИЗ необходимо определить некий конкретный смысл отношения часть-целое, что должно достаточно хорошо отражать общепринятое понимание этого отношения. Это, в свою очередь, означает, что в формализме ФРИЗ на отношение часть-целое должны быть наложены некоторые условия, с помощью которых общепринятое понимание адекватно описывалось бы на языке ФРИЗ.
Одной очевидной особенностью отношения часть-целое является то, что при рассмотрении целое и часть сопоставлены, т. е. они являются понятиями одного типа. Отсюда следует требование, чтобы системы ИИ, связанные отношением часть-целое, также были совместимы. Ясно, что для того, чтобы системы ИИ были совместимы, необходимо, чтобы они были одного типа. Кроме того, здравый смысл подсказывает, что эти системы ИИ должны быть определены на одном и том же полном параметрическом множестве. Совместимость систем ИИ является необходимым условием того, что системы ИИ связаны отношением часть-целое, но недостаточным. Если есть две совместные системы ИИ, скажем системы *x* и *y,* то в соответствии со здравым смыслом *x* воспринимается как часть *y* только тогда, когда *x* полностью включается в *y* неким соответствующим образом, определяемым типом этих систем.

Требования *совместимости* и *включенности*, видимо, адекватно описывают самую суть отношения часть-целое. Чтобы как можно более общим образом определить смысл этого отношения никаких добавочных требований ненужно. Остается, разумеется,определить отношение часть-целое для исходных систем ИИ, систем данных и порождающих систем ИИ так, чтобы оба этих требования выполнялись.

Введем сначала соответствующую терминологию и обозначения. Пусть система ИИ x рассматривается как часть системы ИИ y. Будем x называть *подсистемой ИИ y, а y — суперсистемой ИИ x.* Формально будем обозначать, что x является подсистемой ИИ y (а y — суперсистемой ИИ для x), следующим образом: $x \prec y$.

Пусть теперь ${}^x\mathbf{S}$ и ${}^y\mathbf{S}$ — исходные системы ИИ. Для определения отношения «подсистема ИИ» (и обратного отношения «суперсистема ИИ») необходимо выполнить условие совместимости исходных систем ИИ. Это значит, что они должны быть одного методологического типа (т. е. иметь одни и те же методологические отличия) и должны быть определены для одних и тех же параметров, как и для соответствующих баз.

Требование включенности для исходных систем ИИ выражается в виде нескольких отношений включения: ${}^x\mathbf{S}$ рассматривается как *исходная подсистема ИИ* ${}^y\mathbf{S}$ (предполагается, что ${}^x\mathbf{S}$ и ${}^y\mathbf{S}$ — сравнимые исходные системы ИИ) тогда и только тогда, когда множества переменных (и обобщенных, и конкретных) и множество свойств системы ИИ ${}^x\mathbf{S}$ являются подмножествами соответствующих множеств системы ИИ ${}^y\mathbf{S}$ и, соответственно, множества состояний и проявлений свойств, а также множества наблюдений и каналы конкретизации системы ИИ ${}^x\mathbf{S}$ являются подмножествами соответствующих систем ИИ ${}^y\mathbf{S}$. Данный набор отношений включения, которые должны выполняться, чтобы выполнилось отношение «подсистема ИИ», удобно представить через отношения одного индексного множества. Элементы этого множества **идентифицируют отдельные сущности разных множеств (переменные, свойства, каналы)**, причем предполагается, что соответствующие друг другу обобщенные переменные, конкретные переменные и свойства помечаются одним и тем же элементом индексного множества (так же, как в формальном определении исходных систем ИИ). Пусть переменные, свойства и другие характеристики систем ИИ ${}^x\mathbf{S}$, ${}^yS$ **помечены (идентифицированы)** соответственно индексными множествами ${}^xJ$, ${}^yJ$. Тогда отношение ${}^x\boldsymbol{S}$ — подсистема ИИ ${}^y\boldsymbol{S}$ «полностью» описывается отношением включения

$${}^xJ \subseteq {}^yJ$$

для их индексных множеств. Обычно считается, что

$${}^yJ \subseteq N_n.$$

Пример 1. Пусть ^{1}S— исходная система из примера 1 р. 2.7 (состояние деловой древесины). Тогда ${}^1J=N_7$. Пусть ${}^2\mathbf{S}$ — исходная система, определенная как подсистема ^{1}S (${}^2\mathrm{S} \prec {}^1\mathrm{S}$) с помощью индексного множества ${}^2J=\{1, 2, 3, 7\}$. Тогда ^{2}S будет состоять из всех элементов, входящих в ^{1}S, за исключением v_i, V_i, v'_i, V'_i, a_i, A_i, o_i, e_i при i=4, 5, 6. Для направленных исходных систем ИИ отношение «подсистема ИИ» находит отражение и в соответствующих идентификаторах входов-выходов. Пусть ${}^x\mathbf{S}$, ${}^y\mathbf{S}$— направленные исходные системы ИИ, такие, что ${}^x\mathbf{S} \prec {}^y\mathbf{S}$, и пусть

$${}^x\mathbf{u} = ({}^xu(j) \mid j \in {}^xJ),$$

$${}^y\mathbf{u} = ({}^yu(j) \mid j \in {}^yJ)$$

— идентификаторы их входов-выходов. Тогда

${}^xu(j) = {}^yu(j)$ для всех $j \in {}^xJ$. Для отслеживания значений, связанных с отдельными компонентами ${}^x\mathbf{u}$ и ${}^y\mathbf{u}$, удобно считать, что для любого идентификатора входа-выхода элементы $u(j)$ упорядочены в порядке возрастания значений j.

Определенное для исходных систем ИИ отношение «подсистема ИИ» легко может быть распространено на системы данных. Понятно, что для двух сравнимых систем данных ${}^x\mathbf{D}$, ${}^y\mathbf{D}$, которым соответствуют исходные системы ${}^x\mathbf{S}$, ${}^y\mathbf{S}$, ${}^x\mathbf{D}$ является подсистемой данных ${}^y\mathbf{D}$, т. е. ${}^x\mathbf{D} \prec {}^y\mathbf{D}$, тогда и только тогда, когда ${}^x\mathbf{S} \prec {}^y\mathbf{S}$ и ${}^x\mathbf{D}$ содержит только данные, содержащиеся в ${}^y\mathbf{D}$ и относящиеся к переменным, входящим в ${}^x\mathbf{S}$. Существенно, чтобы массивы данных были помечены таким образом, чтобы для каждого элемента можно было однозначно определить, к какой конкретной переменной он относится.

Пример 2. Обозначим через

$${}^1\mathbf{D}({}^1\mathbf{S}, {}^1\mathbf{d}) \text{ и } {}^2\mathbf{D} = ({}^2\mathbf{S}, {}^2\mathbf{d})$$

систему данных ${}^1\mathbf{D}$ из примера 2 р.2.8 (блюз) и подсистему ${}^1\mathbf{D}$ исходная система которой определяется индексным множеством {1, 2} (т. е. рассматриваются только высота тона и ритм, но не гармония). Тогда ^{2}S будет содержать все компоненты ${}^1S$, за исключением v_3, V_3, v'_3, V'_3, a_3, A_3, o_3, e_3 и матрицу, изображенную на рис.3 р.2.8, но без третьей строки.

Теперь остается только определить отношение «подсистема ИИ» для двух вариантов порождающих систем ИИ — систем с поведением и ST-систем. Пусть

$${}^x\mathbf{F}_B = ({}^x\mathbf{S}, {}^xM, {}^xf_B),$$

$${}^y\mathbf{F}_B = ({}^y\mathbf{S}, {}^yM, {}^yf_B)$$

— сравнимые системы с поведением и пусть ${}^{x}J$, ${}^{y}J$— множества идентификаторов переменных, соответствующих исходных систем ^{y}S, ${}^{y}S$. Тогда ${}^{x}\mathbf{F}_{\mathrm{B}}$ является *подсистемой системы с поведением* ${}^{y}\mathbf{F}_{\mathrm{B}}$, т. е. тогда и только тогда, когда выполнены следующие три условия:

1) ${}^{x}J \subseteq {}^{y}J$, так что ${}^{y}\mathrm{S} \prec {}^{y}S$;

2) ${}^{x}M \subseteq {}^{y}M$, так что $(v_i, r_j) \in {}^{x}M$ тогда и только тогда, когда $(v_i, r_j) \in {}^{y}M$ и $j \in {}^{x}J$;

3) ${}^{x}f_B=[{}^{y}f_B\downarrow{}^{x}K]$, где ${}^{x}K$ — множество идентификаторов выборочных переменных, соответствующих ${}^{x}M$, т. е. ${}^{x}f_B$ является проекцией ${}^{y}f_B$ для выборочных переменных системы ${}^{x}\mathbf{F}_B$.

Чтобы система

${}^{x}\mathbf{F}_{\mathrm{S}}=({}^{x}\mathrm{S}, {}^{x}M, {}^{x}f_S)$

являлась ST-подсистемой сравнимой ST-системы

${}^{y}\mathbf{F}_{\mathrm{S}}=({}^{y}\mathrm{S}, {}^{y}M, {}^{y}f_S)$

необходимо, чтобы были выполнены условия 1, 2, а вместо условия 3 должно быть выполнено немного измененное условие

4) ${}^{x}f_S=[{}^{y}f_S\downarrow{}^{x}K^2]$, где $[{}^{y}f_S\downarrow{}^{x}K^2]$ $({}^{x}\mathbf{c}, \mathbf{c}')=a(\{{}^{y}f_S({}^{y}\mathbf{c}, {}^{y}\mathbf{c}')|{}^{y}\mathbf{c} \succ {}^{x}\mathbf{c}, {}^{y}\mathbf{c}' \succ {}^{y}\mathbf{c}'\})$; где a, как и в разд. 4.6, некая агрегирующая функция, определяемая типом функции ${}^{y}f_S$ (т. е. для вероятностных систем это сумма, а для возможностных систем — минимум).

Пример 3. Рассмотрим отношение подсистема-суперсистема для двух систем с поведением, изображенных на рис. 2.

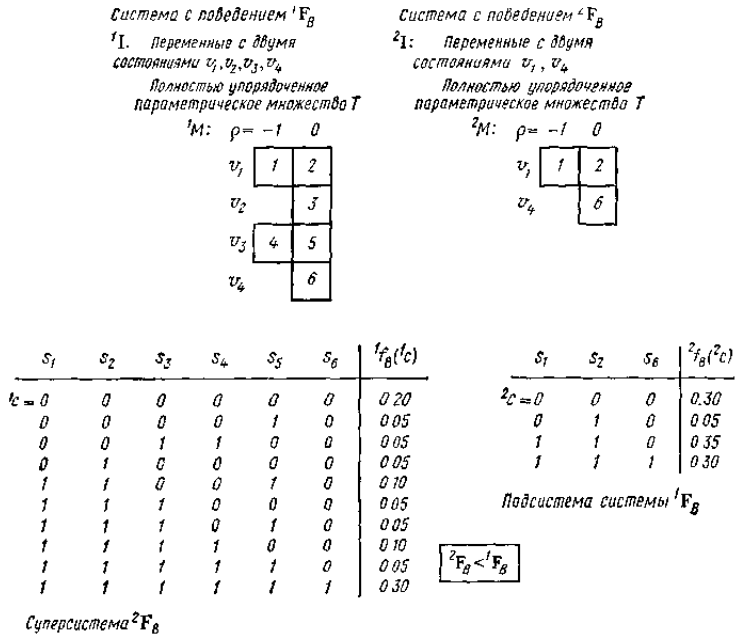


Рис. 2. Пример отношения подсистема с поведением/суперсистема с поведением(пример 3)

Обе системы определены на одном и том же полностью упорядоченном параметрическом множестве. Их представляющие системы ${}^{1}\mathbf{I}$, ${}^{2}\mathbf{I}$ содержат соответственно переменные v_1, v_2, v_3, v_4 и v_1, v_4, каждая из которых имеет два состояния. Им может быть дана некоторая интерпретация (каналы конкретизации и наблюдения), однако в данном примере эта интерпретация несущественна. Функциями поведения систем являются распределения вероятностей. Система ${}^{2}\mathbf{F}_{\mathbf{B}}$ удовлетворяет трем условиям, определяющим, что ${}^{2}\mathbf{F}_{\mathbf{B}}$ является подсистемой системы ${}^{1}\mathbf{F}_{\mathbf{B}}$, т. е. ${}^{2}\mathbf{F}_{\mathbf{B}}<{}^{1}\mathbf{F}_{\mathbf{B}}$. Поскольку эти системы вероятностные, то при проецировании $[{}^{1}f_{\mathrm{B}}\downarrow\{1, 2, 6\}]$ в качестве агрегирующей функции используется сумма. Например, значение для ${}^{2}f_{\mathrm{B}}(000)$ получается суммированием первых трех вероятностей из таблицы для ${}^{1}f_{\mathrm{B}}$.

Понятия подсистемы с поведением и ST-подсистемы достаточно легко распространить и на другие типы порождающих систем (порождающие системы с поведением, направленные системы с поведением и т. д.). Нужно только подходящим образом идентифицировать порождающие,

порождаемые и входные переменные для обеих рассматриваемых систем.

5.3. Структурированные исходные системы ИИ и структурированные системы данных

Возвращаясь к общей проблеме целого и его частей, необходимо признать, что сложность, а следовательно и богатство идей, связанных с этой проблемой, проистекает, по крайней мере, отчасти из-за того, что одна и та же система может быть разбита на части многими способами.

Как уже указывалось выше, структурированные системы ИИ — это множества исходных систем ИИ, систем данных или порождающих систем ИИ. Они нужны для объединения нескольких систем ИИ в большие. Для того чтобы такое объединение было осмысленным, необходимо, чтобы отдельные системы ИИ— элементы структурированной системы ИИ, были совместимы, т. е. были одного типа и определены на одном и том же параметрическом множестве. По существу, это то же условие *совместимости,* выполнение которого требуется для отношения «подсистема ИИ» (разд. 5.2).

Кроме условия совместимости нужно еще потребовать, чтобы никакой элемент не был подсистемой ИИ другого элемента той же структурированной системы ИИ. Выполнение этого требования позволяет избежать перемешивания уровней отдельных структурированных систем ИИ для того, чтобы они были иерархически упорядочены, как об этом говорится в разд. 5.1. Более того, в структурированной системе ИИ подсистемы ИИ, состоящие из любых элементов, полностью избыточны в том смысле, что любая содержащаяся в них информация также может быть выведена из элементов их суперсистем. Таким образом, в структурированных системах они бесполезны. Назовем это требование *требованием неизбыточности.*

Избыточные элементы часто используются в технических системах для обнаружения и коррекции ошибок. Как показано в разд. 5.4 (пример 8), требование неизбыточности никоим образом не исключает из рассмотрения системы ИИ такого типа.

Для формального определения структурированных систем ИИ предположим, что нейтральная структурированная система ИИ состоит из q элементов (нейтральных систем ИИ того же типа), удовлетворяющих требованиям совместимости и неизбыточности. Элементы

идентифицируются индексом x, где $x \in N_q$. Пусть, кроме того,

$$V=\{v_i \mid i \in N_n\} \tag{1}$$

является множеством всех переменных, входящих в элементы системы ИИ, и пусть ${}^xV$— множество переменных элементов x ($x \in N_q$). Тогда

$$V = \bigcup_{x \in N_q} {}^xV. \tag{2}$$

Будем для удобства обозначений переменные из множеств ${}^xV$ идентифицировать с помощью того же индекса i, что и переменные из полного множества V, определенного в (1). Тогда любой элемент однозначно идентифицируется множеством своих переменных ${}^xV$.

Различные типы структурированных систем ИИ будем обозначать стандартным для этого типа символом с префиксом **S**. Так, через **SS**, **SD, SF_B, SF_B** обозначены структурированные системы, элементами которых являются соответственно нейтральные исходные системы ИИ, направленные системы данных, нейтральные системы ИИ с поведением и направленные порождающие **ST**-системы ИИ. Таким образом, префикс **S** используется как оператор, показывающий, что несколько систем ИИ определенного типа объединено в большую систему ИИ.

Элементами структурированных систем ИИ простейшего типа являются нейтральные исходные системы ИИ. Системы этого типа определяются как множество

$$\mathbf{SS}=\{({}^xV, {}^x\mathbf{S}) \mid x \in N_q\}, \tag{3}$$

где ${}^x\mathbf{S}$ для каждого $x \in N_q$ — нейтральная исходная система ИИ (элемент **SS**); через ${}^xV$ обозначено множество переменных, входящих в ${}^x\mathbf{S}$; это обозначение удобно использовать как идентификатор элементов структурированной системы ИИ. Разумеется, исходные системы ${}^XS$ из (4.3) должны удовлетворять требованиям совместимости и неизбыточности, но только этим требованиям.

Если два элемента **SS**, скажем элементы, идентифицированные как x, $y \in N_q$, имеют общие переменные, т. е.

$${}^xV \cap {}^yV \neq \emptyset, \tag{4}$$

то эти элементы соединены. Будем это множество общих переменных называть *соединением* элементов x и y, а переменные из этого множества — *соединяющими переменными.* **Соединения — это важные характеристики структурированных систем ИИ, так как они определяют взаимодействия между их элементами.** Понятно, что для нейтральных структурированных систем ИИ соединения

симметричны, т. е. не зависят от порядка, в котором рассматриваются элементы. Для удобства соединение между нейтральными элементами *x* и *y* структурированной системы ИИ будем обозначать как

$$C_{x,y} = {}^{x}V \cap {}^{y}V. \quad \textbf{(5)}$$

Пример 4. Рассмотрим с точки зрения садовника структурированную исходную систему, определенную для розового куста в горшке. Эта система задается для того, чтобы определить пути повышения ежегодного урожая.

Все переменные этой системы имеют **два одинаковых параметра**: группу изучаемых розовых кустов и время. На самом деле параллельно исследуется несколько групп, причем каждая характеризуется определенными свойствами, такими, как тип почвы, удобрения и заболеваниярастения, частота срезания цветов и т. д. Необходимо, чтобы наблюдения делались для каждого члена группы раз в два дня в течение одного года; если нужно, то исследование может быть продлено еще на несколько лет.

В объекте исследования — розовом кусте в горшке — можно выделить шесть частей: почва, корни, стебель, сок растения, листья и цветы (см. рис. 3).

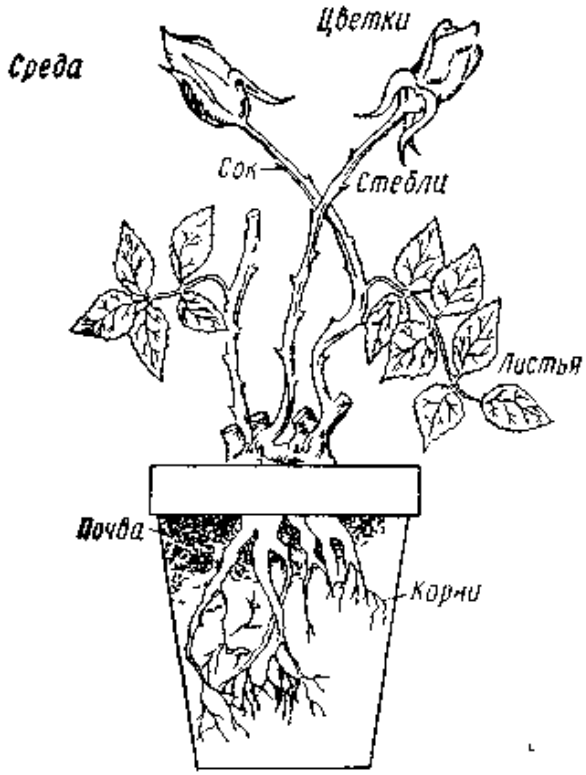


Рис. 3. Части розового куста, являющиеся элементами структурированной системы, определенной в примере 4

Исходная система определена для каждой из этих частей через следующий набор из 19 переменных (для простоты мы опускаем описание наблюдений и каналов конкретизации):

v_1 (влажность почвы) — низкая, средняя, высокая;

v_2 (способность корней поглощать влагу) — низкая, средняя, высокая;

v_3 (способность корней поглощать минеральные вещества) — низкая, средняя, высокая;

v_4 (способность стебля переносить сок) — хорошо, плохо;

v_5 (частота расположения цветов на стебле) — малая, средняя, большая;

v_6 (частота расположения листьев на стебле) — малая, средняя, большая;

v_7 (характеристики цвета сока)—плохие, средние, хорошие;

v_8 (характеристики запаха сока)—плохие, средние, хорошие;
v_9 (характеристики выработки сока) — плохие, средние, хорошие;
v_{10} (количество листьев) — небольшое, нормальное, избыточное;
v_{11} (цвет листьев) —плохой, хороший;
v_{12} (развитие листьев) —задержанное, нормальное;
v_{13} (окраска цветка) —бледная, обычная, интенсивная;
v_{14} (запах цветка) — слабый, обычный, интенсивный;
v_{15} (размер цветка) — небольшой, нормальный, огромный;
v_{16} (количество цветов)—небольшое, большое, огромное;
v_{17} (температура воздуха по Фаренгейту) — ниже 60, 60—69, 70—79, 80—89, 90° и выше;
v_{18} (осадки) —ниже нормы, норма, выше нормы;
v_{19} (среднее число солнечных часов за день) — меньше 3, 3—6, больше 6.

Шесть элементов структурированной системы определяются следующими подмножествами полного множества переменных:

$x=1$ (почва) — v_1, v_{17}, v_{18};
$x=2$ (корни) — v_1, v_2, v_3;
$x=3$ (стебли) — $v_2, v_3, v_4, v_5, v_6, v_{17}, v_{19}$;
$x=4$ (сок) — $v_2, v_3, v_4, v_7, v_8, v_9, v_{17}, v_{19}$;
$x=5$ (листья) — $v_6, v_{10}, v_{11}, v_{12}, v_{17}, v_{18}, v_{19}$;
$x=6$ (цветы) — $v_5, v_{13}, v_{14}, v_{15}, v_{16}, v_{17}, v_{18}, v_{19}$.

Соединения отдельных элементов легко получить, взяв пересечения этих множеств. Например,

$$C_{1,2}=\{v_1\},\ C_{2,5}=\emptyset,$$

$$C_{3,4}=\{v_2, v_3, v_4, v_{17}, v_{19}\} \text{ и т. д.}$$

Определим структурированную систему **SS**, элементами которой являются направленные исходные системы ИИ, как множество

$$\mathbf{S}\hat{\mathbf{S}}=\{({}^xX,\ {}^xY,\ {}^x\hat{S}) \mid x\in N_q\}, \quad (6)$$

где ${}^xX$, ${}^xY$— множества входных и выходных элементов соответственно. Понятно, что

$${}^xX \cup {}^xY={}^xV. \quad (7)$$

Если не считать выделения входных и выходных переменных, то множество (6) совершенно аналогично определенному формулой (3) множеству для нейтральных структурированных систем **SS**. Однако элементы ${}^x\hat{\mathbf{S}}$ любой направленной структурированной системы $\mathbf{S}\hat{\mathbf{S}}$ должны удовлетворять еще одному требованию, связанному с идентификаторами входов-выходов: ни одна из переменных из множества V, определенного уравнения (2), не может быть объявлена

как выходная более чем для одного элемента. Это требование обеспечивает согласованность состояний всех переменных при любом значении параметра. В самом деле, если переменная объявлена как выходная для более чем одного элемента структурированной системы ИИ, то ее состояния будут определяться (контролироваться) при любом значении параметра всеми этими элементами, что, как правило, будет приводить к несогласованности (к заданию нескольких различных состояний переменной при одном и том же значении параметра). Избежать этой несогласованности можно только тогда, когда все элементы влияют на эту переменную одинаково, что является исключительным, очень редким случаем. При этом, однако, ничто не будет потеряно, если потребовать, чтобы только один из этих элементов (любой) был объявлен управляющим этой переменной элементом. Это требование, которое должно выполняться для всех направленных структурированных систем ИИ, назовем требованием *однозначности управления.* **Классификация переменных каждого элемента направленной системы ИИ на входные и выходные и требование однозначности управления имеют важные следствия для понятия соединения элементов.** Для двух заданных элементов x, y направленной структурированной системы ИИ можно определить два *направленных соединения.* Одно из этих соединений, ведущее из x в y, обозначается $C_{x,y}$ и определяется как

$$C_{x,y}={}^xY \cap {}^yX. \quad (8)$$

Второе ведет из y в x, обозначается $\hat{C}_{x,y}$ и определяется как

$$\hat{C}_{x,y}={}^yY \cap {}^xX. \quad (9)$$

Поскольку

$${}^xY \neq {}^yY \text{ для } x\neq y$$

(из-за требования однозначности управления), понятно, что

$$C_{x,y} \neq \hat{C}_{y,x} \quad (10)$$

для разных элементов x, y.

Кроме соединений элементов направленной структурированной системы ИИ, имеются также соединения элементов со средой системы. Будем для удобства рассматривать среду как отдельный элемент с уникальной меткой $x=0$. Несмотря на то, что на самом деле среда не является элементом структурированной системы ИИ [как это следует из (6)], такой подход позволяет нам определить направленные соединения $\hat{C}_{0,x}$ и $\hat{C}_{x,0}$ $(x \in N_q)$ среды с элементами структурированной системы ИИ точно так же, как и соединения элементов.

Если переменная объявлена выходной переменной некоего элемента *x* направленной структурированной системы ИИ, то эта переменная не управляется средой (из-за требования однозначности управления) и, следовательно, не входит ни в какое соединение $\widehat{C}_{0,x}$. Однако если некая переменная не объявлена как выходная ни для какого элемента, то остается только рассматривать ее как переменную, управляемую средой. Следовательно, такая переменная должна быть включена в некое соединение $\widehat{C}_{0,x}$. Поэтому все переменные в любом ${}^{x}X$, не объявленные ни в каких элементах как выходные, образуют соединение среды с элементом *x*. Формально

$$\widehat{C}_{0,\,x} = {}^{x}X \cap (V - \bigcup_{y \in N_q} {}^{y}Y) \tag{11}$$

для любого $x \in N_q$

Для описания соединений $\widehat{C}_{x,0}$ $(x \in N_q)$ рассмотрим переменные из множества ${}^{x}Y$, не объявленные как входные ни для какого элемента направленной структурированной системы ИИ. По определению, эти переменные не входят ни в какие соединения элементов структурированной системы ИИ. Таким образом, их надо рассматривать как соединения со средой, т. е. как входящие в соединение $\widehat{C}_{x,0}$.

Остальные переменные также могут быть включены в $\widehat{C}_{x,0}$. Вопрос о том, рассматривать ли их как соединения со средой или нет, остается в компетенции пользования. Формально

$${}^{x}Y \cap (V - \bigcup_{y \in N_q} {}^{y}X) \subseteq \widehat{C}_{x,0} \subseteq {}^{x}Y \tag{12}$$

для любого $x \in N_q$

Пример 5. Определим направленную структурированную систему, состоящую из пяти исходных систем. Система предназначена для изучения деятельности высших судебных инстанций шт. Нью-Йорк (США) по делам, рассматриваемым в уголовном суде. Система описывает прохождение дела через уголовный суд и суды высших инстанций, утверждающие приговор. Эта система определяет структуру сбора и обработки данных.

Параметром в системе является время. В зависимости от конкретных целей наблюдения делаются ежемесячно, еженедельно или даже

каждый день, начиная с некоторой фиксированной даты, скажем с 1 января 2015 г. В систему включены следующие переменные (множество *V)*:

v_1 — общее число исков, поступивших в уголовный суд (в течение конкретного периода наблюдения — месяц, неделя или день);

v_2 — число исков, прекращенных в результате соглашения сторон;

v_3 — число отклоненных исков;

v_4 — число задержанных дел;

V_5 — число дел, по которым вынесены оправдательные приговоры;

v_6 — число дел, переданных на утверждение высших инстанций;

v_7 — число дел, не переданных на утверждение (это дела, в которых единственным наказанием является штраф или возмещение убытков);

v_8 — число дел, в которых нарушены условия утверждения;

v_9 — число дел, приговоры по которым отмечены судом высшей инстанции;

v_{10} — число дел, приговоры по которым отмечены учреждениями уголовного суда.

Рассматриваемая структурированная система состоит из пяти элементов (рис. 4).

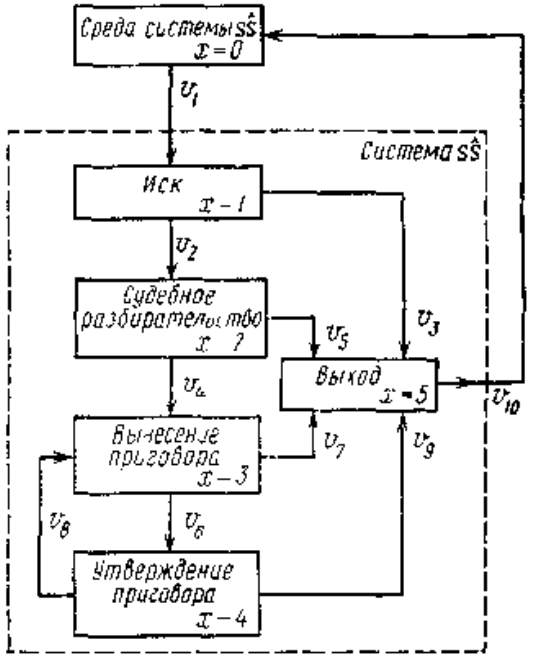


Рис . 4. Схема структурированной системы из примера 5

Их множества входных и выходных переменных приведены в табл. 1

Таблица 1.

Определение элементов направленной структурированной системы из примера 5 (эквивалентно схеме на рис 4)

x	${}^{x}X$	${}^{x}Y$
1	$\{v_1\}$	$\{v_2, v_3\}$
2	$\{v_4\}$	$\{v_4, v_5\}$
3	$\{v_4, v_8\}$	$\{v_6, v_7\}$
4	$\{v_6\}$	$\{v_8, v_9\}$
5	$\{v_3, v_5, v_7, v_9\}$	$\{v_{10}\}$

Прямоугольниками, которым даны условные названия, на схеме показаны элементы структурированной системы и ее среды. Связи между блоками представляют переменные, соединяющие элемнты (а также среду). Полное множество соединений $\hat{C}_{x,y}$ *(x, y*=0, 1, ..., *q)* представлено матрицей (табл. 2), обычно называемой *матрицей соединений.*

Таблица 2.

Матрица соединений структурированной системы, описанной в примере 5

$\hat{C}_{x,y}$	0	1	2	3	4	5
0	$\varnothing$	$\{v_1\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
1	$\varnothing$	$\varnothing$	$\{v_2\}$	$\varnothing$	$\varnothing$	$\{v_3\}$
2	$\varnothing$	$\varnothing$	$\varnothing$	$\{v_4\}$	$\varnothing$	$\{v_5\}$
3	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\{v_6\}$	$\{v_7\}$
4	$\varnothing$	$\varnothing$	$\varnothing$	$\{v_8\}$	$\varnothing$	$\{v_9\}$
5	$\{v_{10}\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$

Аналогично структурированным исходным системам определяются и *струтурированные системы данных;*

$$\mathbf{SD}=\{({}^{x}V,\ {}^{x}\mathbf{D}) \mid x\in N_q\}, \quad (13)$$

$$\mathbf{SD}=\{({}^{x}X,\ {}^{x}Y,\ {}^{x}\mathbf{D}) \mid x\in N_q\}. \quad (14)$$

Поскольку любая система данных содержит исходную систему ИИ, структурированные системы данных должны удовлетворять всем условиям, которым должны удовлетворять исходные системы ИИ (совместимости, неизбыточности, однозначности управления). Кроме того, обычно требуется, чтобы они удовлетворяли *локальной согласованности данных,* определяемой следующим образом: для всякой соединяющей переменной соответствующие ей данные должны быть одинаковыми во всех элементах, в которые входит эта переменная.

Формально, если

$$v_i\in C_{x,y} \ (\text{или } v_i\in\hat{C}_{x,y}),$$

то ${}^{x}v_{i,w} = {}^{y}v_{i,w}$ для всех $\mathbf{w} \in \mathbf{W}$, где ${}^{x}v_{i,w}$ и ${}^{y}v_{i,w}$ $(\mathbf{w}\in\mathbf{W})$ —подмножества данных, соответствующих переменной v_i в элементах x и y. Обычно предполагается, что структурированные системы данных локально согласованы. Однако если множества данных, связанных с разными элементами, собираются независимо друг от друга, например разными группами экспериментаторов, то полученные множества данных могут и не удовлетворять требованию локальной согласованности данных. Нарушение этого условия ведет к аналогичным противоречиям и на более высоких эпистемологических уровнях, поэтому необходимо на некотором этапе исследования разрешить их. Процедуры разрешения локальных противоречий до сих пор должным образом не разработаны. Представляется, однако, что в общем случае более удобно и осмысленно противоречия разрешаются на уровне структурированных порождающих систем ИИ (они вводятся в разд. 5.4), а не на уровне структурированных систем данных. Поэтому задача разрешения локальных противоречий рассматривается именно на уровне структурированных порождающих систем ИИ.

5.4. Структурированные системы ИИ с поведением

Структурированные порождающие системы ИИ определяются в столь же общем виде, как и другие типы структурированых систем ИИ. Они также должны удовлетворять условиям совместимости и

неизбыточности, которым должны удовлетворять исходные структурированные системы ИИ. Они также должны удовлетворять некоторым дополнительным требованиям относительно масок и функций поведения или ST-функций своих элементов. Тот факт, что множество выборочных переменных порождающей системы ИИ в общем случае больше множества переменных соответствующей исходной системы ИИ или системы данных, дает некоторые новые возможности для соединения порождающих систем ИИ в структурированную систему ИИ. Как было показано в разд. 4.7, любая ST-система ИИ может быть преобразована в изоморфную систему ИИ с поведением. Поэтому будет достаточно рассмотреть структурированные порождающие системы ИИ в предположении, что их элементами являются системы ИИ с поведением, т. е. без потери общности рассмотреть только структурированные системы ИИ с поведением.

Пусть нужно соединить заданное множество систем ИИ с поведением в структурированную систему ИИ. Для каждой рассматриваемой системы ИИ с поведением, идентифицированной как элемент x ($x \in N_q$) структурированной системы ИИ, обозначим через ${}^xV$ и ${}^xS$ соответственно множество переменных ее исходной системы ИИ и множество ее выборочных переменных. Пусть

$$V = \bigcup_{x \in N_q} {}^xV = \{v_i \mid i \in N_{|V|}\}; \quad (15)$$

$$S = \bigcup_{x \in N_q} {}^xS = \{s_k \mid k \in N_{|S|}\}; \quad (16)$$

Понятно, что

$$V \subseteq S \text{ и } {}^xV \subseteq {}^xS \text{ для всех } x \in N_q. \quad (17)$$

Для нейтрального варианта $\mathbf{SF}_\mathrm{B}$ структурированная система ИИ с поведением теперь может быть определена так:

$$\mathbf{SF}_B = \{({}^xS, {}^x\mathbf{F}_B) \mid x \in N_q\}. \quad (18)$$

Чтобы однозначно идентифицировать элементы x множествами ${}^xS$, предположим, что выборочные переменные из всех множеств ${}^xS$ ($x \in N_q$) идентифицируются тем же индексом k, что используется для идентификации переменных во всем множестве $\mathbf{S}$. Соединения $C_{x,y}$ элементов $x, y \in N_q$ структурированной системы ИИ $\mathbf{SF}_\mathrm{B}$ определяются теперь как пересечения множеств выборочных переменных

$$C_{x,y} = {}^xS \cap {}^yS. \quad (19)$$

В представлении (18), которое не содержит никаких ограничений на порядок порождения и на результирующее разбиение переменных из множеств ${}^xS$ ($x \in N_q$) и S на порождающие и порождаемые переменные,

обычно требуется, чтобы для всех пар элементов $x, y \in Nq$ выполнялось только одно дополнительное условие, а именно следующее уравнение:

$$[{}^xf \downarrow {}^xS \cap {}^yS] = [{}^yf_B \downarrow {}^xS \cap {}^yS]. \quad (20)$$

Это условие обеспечивает то, что проекции функций поведения ${}^xf_\mathrm{B}$, ${}^yf_\mathrm{B}$ для любой пары элементов из $\mathbf{SF}_\mathrm{B}$ равны относительно их общих переменных (соединяющих переменных). По существу, это требование сводится к требованию, чтобы переменные из разных элементов, считающихся (определяемых) одинаковыми, были действительно равны независимо от элементов, в которые они входят. Будем это требование называть *локальной согласованностью поведения.*

Как уже говорилось, системы ИИ с поведением ${}^x\mathbf{F}_\mathrm{B}$, являющиеся элементами $\mathbf{SF}_\mathrm{B}$, на практике часто определяются на локально несогласованных множествах данных и, следовательно, не удовлетворяют требованию локальной согласованности поведения. Для работы с такими структурированными системами в различных проблемных контекстах, нужно сначала разрешить эти несогласованности (см. разд. 5.11). В остальных разделах этой главы считается, что структурированные системы ИИ с поведением локально согласованы.

После определения порядка порождения для структурированной системы ИИ с поведением $\mathbf{SF}_\mathrm{B}$ множество выборочных переменных $\mathbf{S}$ и $\mathbf{SF}_\mathrm{B}$ разбивается на порождающие и порождаемые переменные, соответственно обозначаемые как $\mathrm{S}_{\bar{\mathrm{g}}}$ и $\mathrm{S_g}$. Полученная структурированная система ИИ должна удовлетворять требованию однозначности управления. В данном случае это означает, что любая переменная из S_g должна порождаться одним и только одним элементом структурированной системы ИИ. В свою очередь, это значит, что множества порождаемых переменных ${}^xS_g$ ($x \in N_g$), связанные с отдельными элементами структурированных систем ИИ, должны образовывать разбиение множества S_g.

Взяв один определенный элемент x ($x \in N_q$) структурированной системы ИИ $\mathbf{SF}_B$, рассмотрим теперь множество переменных

$$\{{}^xS \cap S_g\} - {}^xS_g.$$

Это (с глобальной точки зрения) множество порождаемых переменных, соединенных с элементом x, но не порождаемых этим элементом. Понятно, что с локальной точки зрения (отдельный элемент) это входные переменные, хотя с глобальной точки зрения (структурированная система) они являются порождаемыми переменными. Следовательно, сам элемент должен рассматриваться как направленная система ИИ, в то время как структурированная система ИИ рассматривается как нейтральная в том смысле, что переменные

множества не разбиты на входные и выходные. Это не значит, что в определении структурированных систем ИИ есть некоторая несогласованность. **Это просто следствие того, что для структурированных систем ИИ возможны два сосуществующих подхода — локальный (с точки зрения элементов систем) и глобальный (рассматривающий структурированную систему ИИ как целое). С локальной точки зрения все элементы, с которыми связан некий элемент, образуют его среду. Это нечто вроде *внутренней среды,* определенной только для структурированной системы ИИ**. С глобальной точки зрения среда (или *внешняя среда)* не выделяется. Однако нам представляется, что в данном случае предпочтительнее рассматривать структурированную систему ИИ как направленную, у которой все переменные из множества V объявлены выходными. Теперь остается обсудить только роль множества порождающих переменных ${}^{x}S_g$ для отдельных элементов структурированной системы ИИ, т. е. значение переменных из множеств

$$S_{\bar{g}} \cap {}^{x}S \text{ для всех } x \in N_q.$$

Состояния этих переменных должны быть доступны данному элементу на каждом шаге процесса порождения, как того требует функция поведения. Они могут быть доступны или изнутри, т. е. выводиться обычным образом из предыдущих состояний, так же как предыдущие состояния порождаемых и входных переменных, или снаружи, т. е. через входные переменные, представляющие соединения с другими элементами структурированной системы ИИ. Таким образом, эти переменные рассматриваются или как порождающие переменные, или как входные переменные элемента. Одну из этих альтернатив должен выбрать пользователь. В любом случае будем эти переменные обозначать ${}^{x}\mathrm{S}_{\bar{g}}$. Спецификация их действительной роли входит в определение системы ИИ с поведением, представляющей этот элемент, а также в определение направленных соединений элементов.

Итак, **порождающие системы ИИ с поведением, объединяемые в структурированную систему ИИ, обычно должны рассматриваться как направленные системы ИИ.** Поскольку такой подход возможен всегда, мы определяем

$$\widehat{\mathbf{SF}}_{GB} = \{({}^{x}S_{\bar{g}}, {}^{x}S_e, {}^{x}\mathbf{F}_{GB}) \mid x \in N_q\}, \quad (21)$$

понимая, что при этом не требуется никакого содержательного определения $\mathbf{SF}_{\mathrm{GB}}$.

Под это определение подходит и особый (вырожденный) случай, когда ни одна из переменных из V не объявляется как входная. ${}^{x}S_g$ и ${}^{x}S_e$ — это

соответственно порождаемые и входные переменные элементов x: ${}^{x}\mathrm{S}_{\bar{g}}$ — это множество переменных, введенных и рассмотренных в предыдущем параграфе. В зависимости от того, какая из альтернатив выбрана, в это множество входят входные или выходные переменные элемента. Как и в предшествующем определении структурированной исходной системы ИИ, предполагается, что переменные из всех этих трех множеств и все элементы $x \in N_q$ идентифицируются индексом b, определенным в уравнении (16). Для любого $x \in N_q$ эти три множества образуют разбиение множества ${}^{x}S$. Кроме того, для некоторого $y \in N_q$ $\{0\}$ переменные из множества ${}^{x}S_q$ входят в соединения $\widehat{C}_{x,y}$, в то время как переменные из множества ${}^{x}S_e$ входят только в соединения $\widehat{C}_{y,x}$, переменные из ${}^{x}\mathrm{S}_{\bar{g}}$ могут входить в соединения с любым направлением или вовсе не входить в соединения.

Будем структурированные системы ИИ вида (18) или (21) называть соответственно структурированными системами ИИ с поведением *основного и порождающего типа.* Понятно, что из структурированной системы ИИ с поведением основного типа может быть выведено семейство структурированных систем ИИ с поведением порождающего типа. Структурированные системы ИИ этого семейства отличаются одна от другой:

1) разбиением S на $\mathrm{S}_{\bar{g}}$ и S_g;

2) разбиением $\mathrm{S}_{\bar{g}}$ на ${}^{x}S_g$ *($x \in N_q$);*

3) использованием переменных из множеств ${}^{x}\mathrm{S}_{\bar{q}}$ *($x \in N_q$).*

Разбиение 1 определяется разными порядками порождения для предсказания или восстановления. Разбиение 2 определяется перекрытиями переменных из множества S_g с элементами структурированной системы ИИ. Говоря точнее, переменная из множества S_g, входящая в один и только один элемент, очевидно, должна порождаться этим элементом. С другой стороны, переменная, входящая в несколько элементов, может порождаться любым из этих элементов, но только одним из них. Выбор этого элемента обычно осуществляется исходя из порождающей нечеткости (чем меньше получающиеся порождающие нечеткости, тем предпочтительнее этот вариант), а также некоторыми вспомогательными критериями, определенными пользователем. Альтернативы типа 3) не влияют на порождающую нечеткость выбранной структурированной системы ИИ. Они представляют собой просто варианты формального представления. Пользователю должна быть предоставлена возможность повлиять на

выбор представления; если представление ему безразлично, то нужно использовать один из вариантов по умолчанию.

Пример 6. Рассмотрим структурированную систему основного типа (18), состоящую из двух подсистем, базирующихся на одном и том же полностью упорядоченном параметрическом множестве. Каждый из элементов состоит из двух бинарных переменных, определяемых вероятностными функциями поведения. Маски ^1M и ^2M и функции поведения ^1f_B и ^2f_B показаны соответственно на рис. 5,а и *б*.

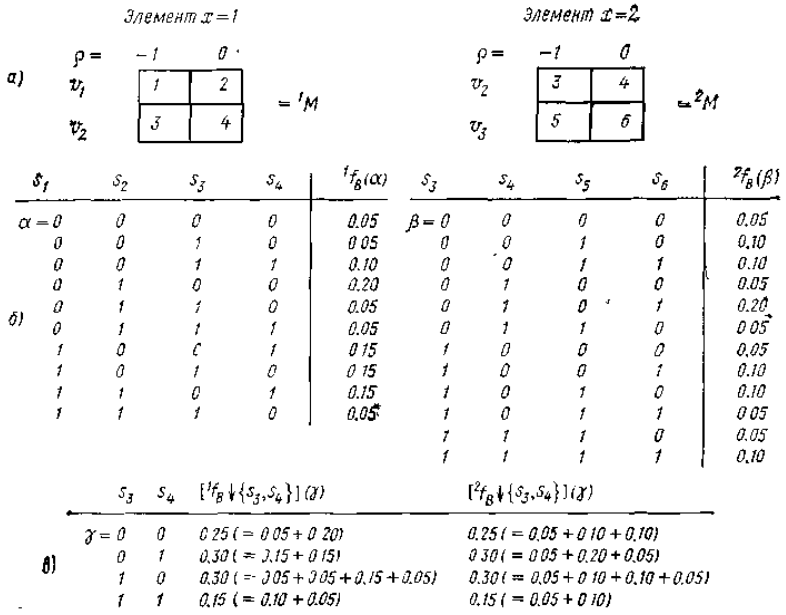


Рис. 5. К примеру 6

Ясно, что эта структурированная система удовлетворяет требованию неизбыточности. Она также локально согласована, как это показано на рис. 5,*в*. Полная маска, схема, а также две частичные маски приведены на рис. 6,а.

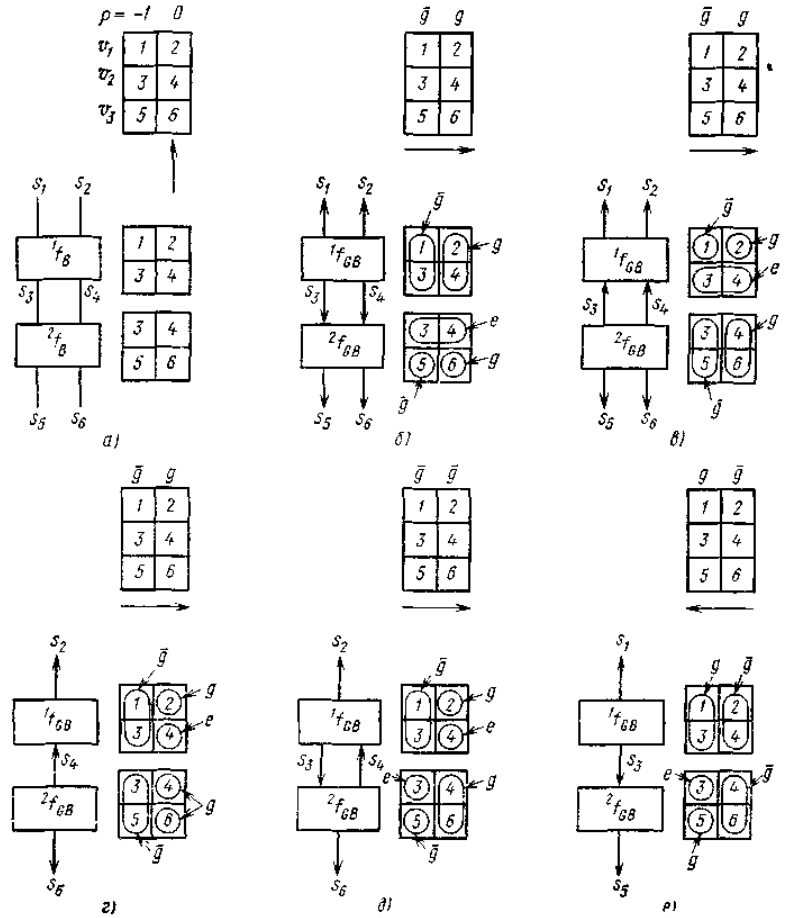


Рис. 6. Основной тип структурированной системы с поведением и некоторые полученные системы порождающего типа (пример 6)

Остальные схемы и маски на рис. 6 представляют собой характерные примеры некоторых структурированных систем порождающего типа, которые могут быть получены из этой системы основного типа.

Для вычисления общего числа существенно отличных структурированных систем порождающего типа, которые можно получить из приведенной в этом примере системы основного типа, предположим, что соединение со средой содержит все порождаемые переменные, равно как и все остальные переменные, рассматриваемые как

соединяющие переменные. Тогда в этом примере имеются 24 порождающие системы для каждого из двух порождающих порядков (предсказание, восстановление). Для целей предсказания они получаются комбинированием следующих вариантов для переменных: две возможности для переменной s_1 и две для переменной s_5 (они рассматриваются или как соединяющие, или нет); три возможности для переменной s_3 (она или не рассматривается как соединяющая, или имеет одно из двух возможных направлений); два возможных направления для переменной s_4. Если учитывать оба порядка порождения, то в данном примере существуют 48 возможных порождающих структурированных систем. Далее варианты любой системы можно получить, по-разному определяя соединения со средой. Из 48 вариантов пять, представляющих важнейшие варианты, показаны на рис. 6,б—6,е. Будем для удобства называть их системами *b , c, ..., f.* Первые четыре варианта (системы *b* — е) предназначены для предсказания, а последний вариант (система *f)* для восстановления. Структурированные системы *b* и *c* подобны в том смысле, что и в той и в другой системе выборочные переменные входят в разные соединения. Их роль отлична от роли переменных s_3, s_4. В системе *b* переменная s_4 порождается первым элементом ${}^1f_B$ (на языке функций поведения она однозначно определяется по приведенной на рис. 5,б), а переменные s_3, s_4 используются как входные переменные второго элемента. В системе *c* переменные s_3, s_4 играют противоположную роль. Для этих двух систем можно сравнить их порождающие нечеткости, связанные с переменной s_4. Они равны 0,2427 для системы *b* и 0.6754 для системы *c.* (Вычисление порождающих нечеткостей, описанные в разд. 4.5, мы предоставляем выполнить читателю в качестве упражнения.) Таким образом, система *b* является более предпочтительной, поскольку она порождает состояния переменной s_4 с существенно меньшей нечеткостью, чем система *c* (порядка 36% нечеткости системы *c*) и, следовательно, является лучшим предиктором, чем система *c.*

Система *d* похожа на систему *c* в том смысле, что они одинаковым образом порождают переменную s_4. Их отличие состоит в том, что переменные s_1, s_3, s_5 в качестве соединяющих переменных играют разные роли, а также в формальном определении первого элемента. Несмотря на эти отличия, по существу, системы *c* и *d* порождают данные одинаково. То же относится и к системе *e.* Ее единственное отличие от системы *d* состоит в том, что переменная s_3 используется как входная переменная для второго элемента, а не как порождающая переменная.

Система *f* — это одна из 24 имеющихся в данном примере восстанавливающих систем. Основной вопрос, связанный с выбором одного из этих вариантов, состоит в том, какой из двух элементов более предпочтителен для порождения переменной s_3. Порождающие нечеткости равны 0.8609 (для системы *f*) и 0.9559 (для систем, в которых s_3 порождается вторым элементом). Несмотря на то, что порождение s_3 первым элементом дает несколько меньшую нечеткость, разница значительно меньше, чем для случая предсказания.

Пример 7. Продемонстрируем гибкость, имеющуюся в определении структурированных систем ИИ. Предположим, что четыре разных изделия *a, b, c* и *d* производятся четырьмя разными подразделениями фирмы. Допустим далее, что

для произведения одной единицы продукции *a* требуется две единицы продукции *b* и три единицы продукции *c;*

для произведения одной единицы продукции *d* требуется две единицы продукции *a,* одна единица продукции *c* и четыре единицы продукции *b* .

Каждое подразделение ежедневно должно знать, сколько продукции ему необходимо произвести. Соответствующее количество продукции будем представлять переменными p_a, p_b, p_c, p_d. Их значения определяются заказом, наличием готового продукта и объемами этого продукта, необходимыми другим подразделениям для производства их собственных продуктов. Заказы будем представлять переменными o_a, o_b, o_c o_d, а объемы готовой продукции — переменными i_a, i_b, i_c, i_d. Эти объемы определяются значениями p_a, p_b, p_c, p_d .

Будем переменные, связанные с одним подразделением (имеющие одинаковые индексы), рассматривать как переменные, которые образуют элемент структурированной системы. В их число входят и переменные p_a, p_b, p_c, p_d, определяющие объемы продукции данного подразделения, которые должны быть переданы другим подразделениям. Таким образом, структурированная система состоит из четырех элементов, соединенных так, как это показано на рис. 7.

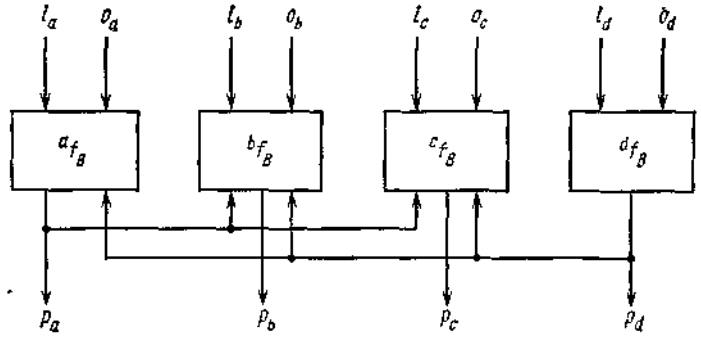


Рис. 7. Структурированная система (пример 7)

Входные переменные любого элемента, представляющие сведения об объемах заказов и готовой продукции, а также потребности других подразделений определяются средой (отделом торговли и складом), а также другими производственными подразделениями. Элементы представляют собой детерминированные направленные системы с поведением (без памяти), а параметром является время. Определим функции поведения элементов следующими простыми уравнениями:

$$^af_B : p_a = o_a - i_a + 2p_d,$$
$$^bf_B : p_b = o_b - i_b + 2p_a + 4p_d,$$
$$^cf_B, p_c = o_c - i_c + 2p_a + p_d,$$
$$^df_B : p_d = o_d - i_d.$$

Пример 8. На этом примере будет показано, что требование неизбыточности структурированных систем не противоречит тому, что в технике (в частности, вычислительной технике) для обнаружения и исправления ошибок часто используются именно избыточные системы. Одна из простейших схем корректирования ошибок состоит в том, что три системы параллельно выполняют одну п ту же работу, оперируя одними входными переменными. Подобная схема показана на рис. 8, где три последовательных двоичных сумматора оперируют переменными v_1 и v_2.

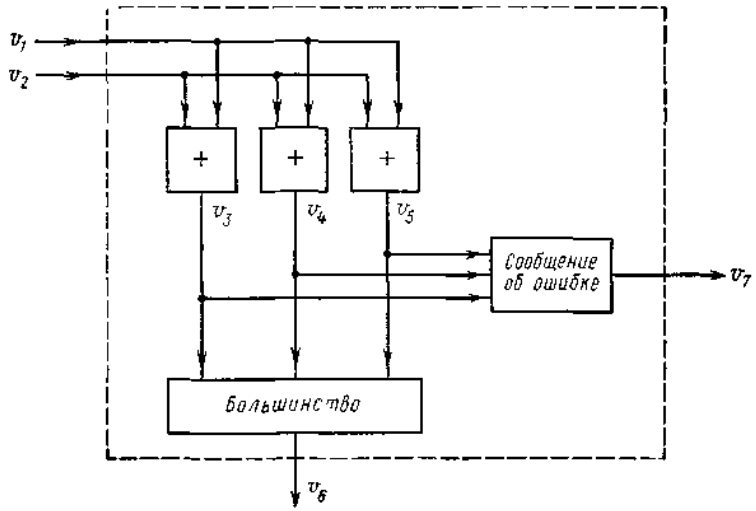


Рис . 8. Простая структурированная система для корректирования ошибок (пример 8)

При нормальной работе состояния их выходных переменных v_3, v_4, v_5 равны, однако если произошел случайный сбой или если устройство вышло из строя, то одно значение будет отличаться от двух других, чтобы распознать подобные ситуации и дать возможность системе в целом продолжить работу, состояния выходных переменных трех одинаковых устройств оцениваются двумя специальными системами. Одна из них на рис. 8 называется «большинство». Она выбирает то состояние выходной переменной (0 или 1), которое имеют по крайней мере две из трех переменных v_3, v_4, v_5, т. е. состояние выходной переменной v_6 равно тому состоянию, которое имеет большинство входных переменных. Вторая специальная подсистема, названная «сообщение об ошибке» предназначена для обнаружения любой несогласованности состояний переменных v_3, v_4, v_5 и для выдачи соответствующего сообщения об ошибке. Ее поведение может быть описано следующим утверждением: v_7=1 (сообщение об ошибке) тогда и только тогда, когда состояния переменных v_3, v_4, v_5 неодинаковы, в противном случае v_7=0 (нормальная работа). Если сообщения об ошибках появляются редко, это говорит о случайных сбоях; частое появление таких сообщений говорит о том, что одно из трех основных устройств (сумматоров) вышло из строя.

Структурированная система, изображенная на рис. 8, удовлетворяет требованию неизбыточности. Несмотря на то, что система содержит две избыточные подсистемы (два из трех сумматоров), они отличаются своими выходными переменными и, если смотреть глубже, своими внутренними переменными. Таким образом, избыточность в технике совершенно не похожа на ту избыточность, которая запрещается требованием неизбыточности. Избыточность в первом смысле означает, что одна и та же работа выполняется параллельно несколькими устройствами, которые отличаются выходными (и внутренними) переменными. Во втором понимании избыточность состоит в том, что структурированная система содержит систему, которая либо неотличима от другой системы, либо является подсистемой другой системы, входящей в эту структурированную.

Пример 9. В этом примере система из предыдущего примера, изображенная на рис. 8, используется для демонстрации уровней структурного уточнения. Рассмотрим более детально один из блоков на рис. 8 — последовательный двоичный сумматор. Его поведение обычно описывается следующими уравнениями:

$$z_t = x_t + y_t + c_{t-1} \pmod 2, \qquad \text{(а)}$$

$$c_t = (x_t + y_t + c_{t-1} + z_t)/2, \qquad \text{(б)}$$

где x_t, y_t — состояния входных переменных в момент t, равные 0 или 1 (и упорядоченные по времени в порядке возрастания значений); z_t — состояние выходной переменной (суммарный бит) в момент t; c_t (c_{t-1})—состояние внутренней переменной, представляющей цифру переноса в момент t (t—1). Согласно этому описанию двоичный сумматор представляет собой структурированную систему, схема которой изображена на рис. 9,а.

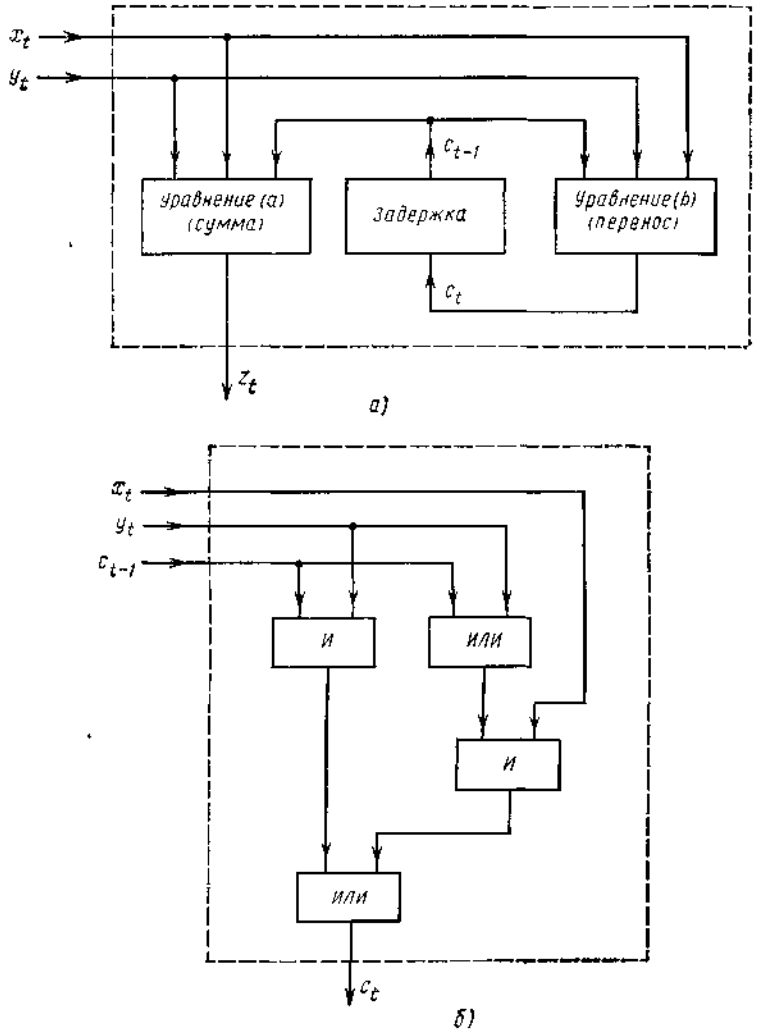


Рис. 9. Уточнение и укрупнение структуры (пример 9)

Ее элементы, описанные уравнениями (а), (б), также могут быть детализированы и представлены в виде структурированных систем. Например, элемент, представляющий уравнение (б), сам может быть представлен как структурированная система, элементами которой

являются стандартные логические функции от двух переменных (рис. 9,б).

Теперь понятен смысл парных процессов *укрупнения и уточнения структуры.* При укрупнении схемы структурированная система корректирования ошибок на рис. 8 является элементом большей структурированной системы (арифметического устройства). При дальнейшем структурном укрупнении эта структурированная система становится элементом еще большей структурированной системы (центрального процессора вычислительной машины) и т. д. до тех пор, пока не будет достигнут максимальный уровень, необходимый для конкретного исследования. Но при уточнении структуры любой элемент системы корректирования ошибок (например, сумматор) может рассматриваться как соответствующая структурированная система (например, та, что изображена на рис. 9,а), а любой элемент этой структурированной системы также может рассматриваться как структурированная система (рис. 9,б) и т. д. до тех пор, пока не будет достигнут необходимый уровень уточнения.

Два или более уровня структурного уточнения также могут быть объединены в одну систему, т. е. в структурированную систему, элементами которой являются структурированные системы, элементами которых являются структурированные системы... и т.д. Эта рекурсия, разумеется, заканчивается на тех элементах, которые не являются структурированными системами.

Будем структурированные системы, содержащие несколько уроиней уточнения, называть *многоуровневыми структурированными системами ИИ* и помечать обобщенным оператором $\mathbf{S}^k$, где k — число уровней уточнения. Например, $\mathbf{S}^2\mathbf{F}_\mathrm{B}$ — это двухуровневая структурированная система ИИ с поведением, т. е. структурированная система ИИ, элементами которой являются структурированные системы ИИ с поведением.

5.5. Задачи проектирования систем ИИ

Иерархическая схема — одна из самых важных структурных схем, используемых при проектировании сложных систем ИИ.

Структурированные системы ИИ используются при решении наиболее важных интеллектуальных задач, возникающих как при исследовании, так и при проектировании систем ИИ. В общем случае эти задачи представляют собой интеллектуальные формулировки различных вопросов, связанных с отношением между частями и целым. Проблемы

типа часть-целое, возникающие при исследовании систем ИИ, существенно отличаются от тех же проблем, возникающих при их проектировании. В данном разделе мы рассмотрим структурированные системы ИИ с точки зрения их роли при решении интеллектуальных задач проектирования. Изучению роли этих систем при исследовании систем ИИ, роли более сложной и хуже изученной, посвящены оставшиеся разделы этой главы.

Как указывалось в разд. 4.10, первым этапом проектирования является определение порождающей системы ИИ, представляющей интеллектуальное задание, которое должна выполнить данная система. **В общем случае это задание представляет собой преобразование состояний соответствующих входных переменных в состояния выходных переменных.** Таким образом, полученная порождающая система ИИ всегда является направленной. Кроме того, это обычно детерминированная система. Часто эта система не является уникальной, что показывается в следующем примере.

Пример.10. Снова рассмотрим последовательный двоичный сумматор, введенный в примере 9. **Его задачей является суммирование двух двоичных чисел, передаваемых последовательно разряд за разрядом в порядке возрастания значимости разрядов.** Обычная система ИИ с поведением, используемая для представления этой задачи, описана в примере 9. Ее маска и поведение показаны на рис. 10,а.

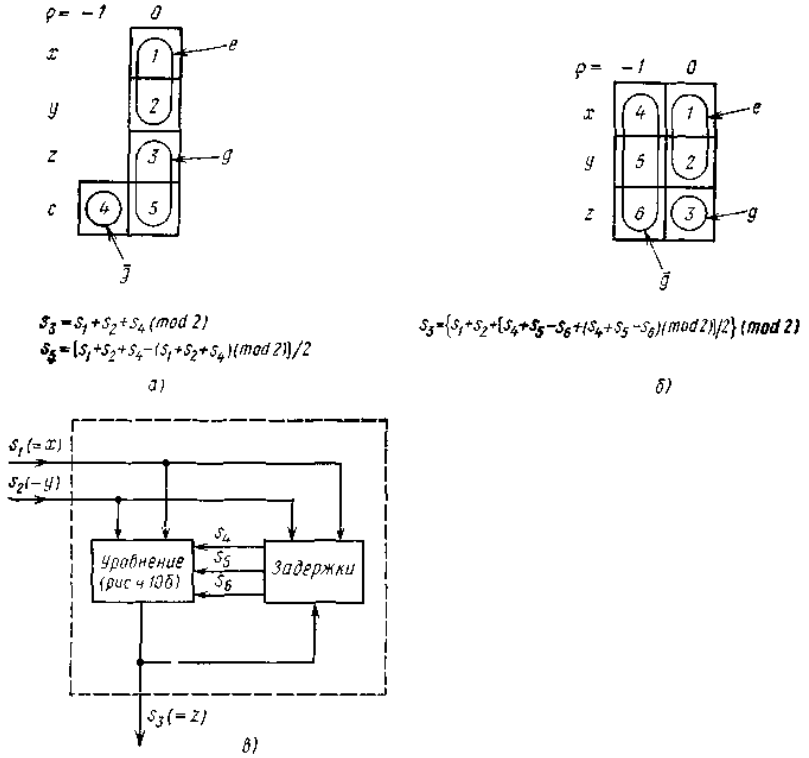


Рис. 10. Два варианта систем с поведением, представляющих последовательный двоичный сумматор (пример 10)

Помимо входных и выходных переменных *x, y, z,* полностью описывающих ее задачу, система включает еще и внутреннюю переменную *c* — переносимый разряд. На рис. 10,б показана альтернативная система с поведением (ее маска и поведение), не содержащая внутренних переменных.

Эти системы, несмотря на то, что они решают одну и ту же задачу преобразования входных переменных *x* и *y* в выходную переменную *z,* совершенно различны. Различия, хорошо видные при сравнении масок, необходимо влекут за собой различия в структурах, реализующих поведение. Так, например, если предположить, что предшествующие значения переменных (т. е. значения, определенные при ρ =—1) непосредственно доступные команде задержки, то эти две системы с поведением включают структурированные системы, изображенные

соответственно на рис. 9,а и 10,в. Эти две структурированные системы, совершенно одинаковые с точки зрения среды, представляют собой совершенно разные основы для проектирования. Выбор одной из систем может быть осуществлен пользователем сразу или процесс проектирования может быть продолжен для обеих систем, а выбор осуществлен на более позднем этапе.

Следующим после определения конкретной системы с поведением этапом проектирования является определение структурированной системы, удовлетворяющей таким требованиям:

1) она реализует функцию поведения или ST-функцию выбранной порождающей системы;

2) все ее элементы представляют собой порождающие системы с определенными (подходящими) функциями поведения или ST-фуикциями;

3) она удовлетворяет некоторым целевым критериям, определяемым как необходимые;

4) она принадлежит к определенному классу структурированных систем (т. е. удовлетворяет некоторым структурным ограничениям).

Требование 1) очевидно, поскольку предполагается, что эта структурированная система выполняет требуемое задание и это задание представляется функцией поведения или ST-функцией данной порождающей системы, то эта функция, в свою очередь, должна представляться этой структурированной системой. Требование 2) говорит об имеющихся технологических возможностях. Оно представляет собой перечень всех модулей (компонентов, кирпичиков), которые можно использовать при создании проектируемой структурированной системы. Важно быть уверенным в том, что выбранных типов элементов достаточно для реализации данной порождающей системы. Требования 3) и 4) представляют собой условия оптимизации соответственно по целям и по ограничениям. Обычно имеется множество воможных целевых критериев и ограничений. Часто они представляют собой комбинации таких факторов, как стоимость, сложность, систематичность, время реакции, надежность, тестируемость, ремонтопригодность и т. д.

Задача реализации заданной функции поведения или ST-функ-ции с помощью элементов определенных типов сводится в принципе к задаче нахождения подходящей (в смысле целевых критериев и ограничений) декомпозиции функций, соответствующих отдельным выходным переменным данной системы, на функции, представляемые элементами заданных типов.

А Одним из методов решения этой задачи, который применим только в определенных случаях, является использование формальных правил

некой алгебры, операции которой соответствуют функ-циям, представленным этими элементами. Этот метод, если считать систему детерминированной, состоит из следующих шагов:

а) Для каждой функции, соответствующей выходной переменной, определяется алгебраическое выражение. Оно может быть, например, подходящей канонической формой определенного типа. Эти алгебраические выражения задают определенный способ композиции функций проектируемой системы из функций, соответствующих элементам. Таким образом, эти выражения удовлетворяют требованиям 1) и 2) для задачи проектирования. Если, что маловероятно, отсутствуют целевые критерии и ограничения, то задача проектирования решена.

б) Полученные выражения преобразуются согласно правилам данной алгебры к виду, удовлетворяющему целевым критериям и ограничениям. В общем случае может быть несколько решений. Обычно бывает достаточно определить только одно.

Другой метод решения задачи декомпозиции, применимый только к дискретным системам ИИ, состоит в том, что декомпозиция производится независимо от алгебраических преобразований, а непосредственно с помощью действий над определениями рассматриваемых функций (их табличным, матричным или каким-то другим подходящим представлением) или с использованием соответствующих функциональных уравнений. Для демонстрации этого метода предположим для простоты, что все имеющиеся элементы имеют две входные переменные и одну выходную. На рис. 11,а показаны входные и выходные переменные проектируемой системы и одного из ее элементов.

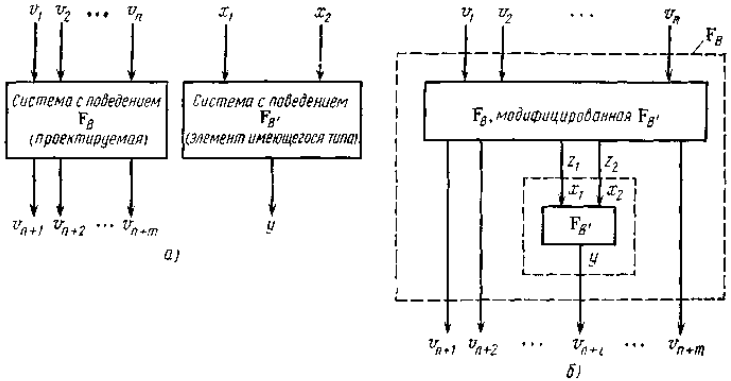


Рис. 11. Иллюстрация методов декомпозиции для проектирования систем ИИ.

Будем считать, что любая выходная переменная является функцией выходных переменных, а именно:

$$\begin{aligned} v_{n+1} &= f_{n+1}(v_1, v_2, \ldots, v_n), \\ v_{n+2} &= f_{n+2}(v_1, v_2, \ldots, v_n), \\ &\ldots\ldots\ldots \\ v_{n+m} &= f_{n+m}(v_1, v_2, \ldots, v_n) \end{aligned} \quad (22)$$

для проектируемой системы и

$$y = g(x_1, x_2) \quad (23)$$

для этого элемента. Теперь представим, что элемент включен в систему F_B таким образом, что его выходная переменная идентична одной из выходных переменных F_B, скажем переменной v_{n+i}. Это дает структурированную систему, схема которой показана на рис. 11,б. Она состоит из двух подсистем: одна — это данный элемент, а другая — модифицированная система F_B. Структурированная система содержит две переменные z_1 и z_2, не входящие в исходную систему F_B. Поскольку в этой системе переменные v_{n+i} и y рассматриваются как идентичные, то функциональное уравнение

$$f_{n+i}(v_1, v_2, \ldots. v_n) = g(x1, x2) \quad (24)$$

должно выполняться. Решение этого уравнения представляет собой две функции

$$\begin{aligned} x_1 &= h_1(v_1, v_2, \ldots, v_n), \\ x_2 &= h_2(v_1, v_2, \ldots, v_n). \end{aligned} \quad (25)$$

Чтобы эти функции были решением уравнения (24), это уравнение должно выполняться при подстановке данных функций вместо переменных x_1 и x_2. Поскольку в данном случае считается, переменные x_1, x_2 идентичны новым переменным z_1 и z_2 (см. рис. 11,б), то можно переписать уравнение (25) следующим образом:

$$\begin{aligned} z_1 &= h_1(v_1, v_2, \ldots, v_n), \\ z_2 &= h_2(v_1, v_2, \ldots, v_n). \end{aligned} \quad (26)$$

Обычно решений уравнений (24) может быть много. Следовательно, основная проблема состоит в выборе одного решения. Прежде всего, множество решений сокращается за счет решений, не удовлетворяющих заданным структурным ограничениям. Кроме того, ищутся такие решения, для которых функции h_1, h_2 зависят от как можно меньшего числа переменных $v_1, v_2, \ldots, v_n$. В самом деле, чем меньше это число, тем сильнее декомпозиция и тем легче при необходимости осуществлять дальнейшую декомпозицию функций h_1,

h_2. И, наконец, оставшиеся решения упорядочиваются относительно целевых критериев и те решения, которые оказываются худшими по всем критериям (или их подмножество), выбираются как основа для продолжения процесса декомпозиции.

Шаг декомпозиции, изображенный на рис. 11,б, должен быть повторен для всех выходных переменных v_{n+1}, v_{n+2}, ..., v_{n+m} и, если нужно, для новых переменных z_1, z_2, ..., введенных в процессе декомпозиции. Декомпозиция прекращается в тех случаях, когда все новые переменные становятся идентичны входным переменным v_1, v_2, ..., v_n. Понятно, что на каждом шаге декомпозиции нужно испытывать новые элементы и сравнивать их возможные декомпозиции.

Многошаговая декомпозиция показана на рис. 12.

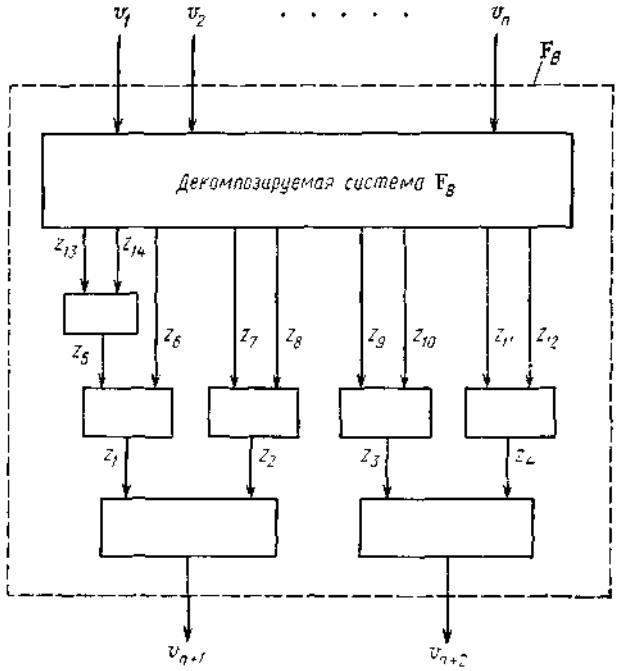
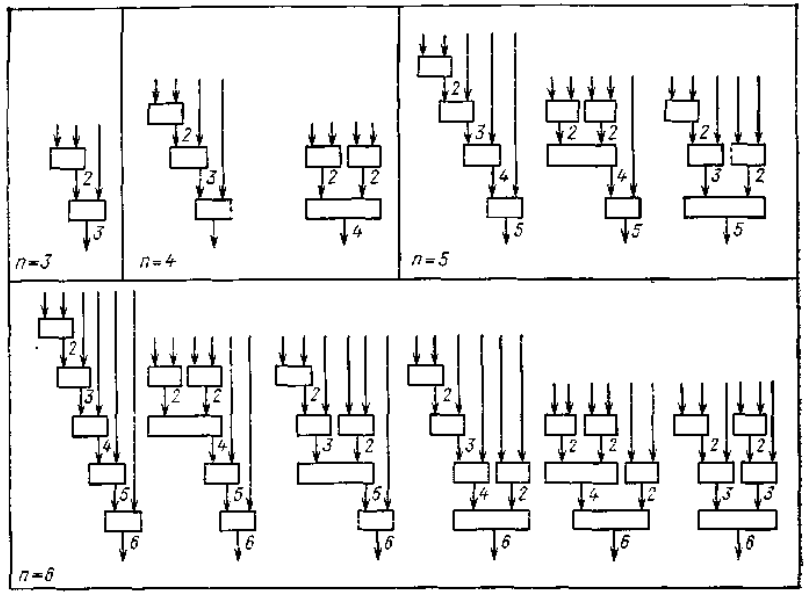


Рис. 12. Пример возможной ситуации после семи шагов декомпозиции

Для простоты используется только один тип элементов, тот, что изображен на рис. 11,а. На рис. 13 приведены эффективные типы декомпозиций системы с одной выходной переменной на элементы с двумя входными переменными и одной выходной.



Рие. 13. Примеры декомпозиций, основанных на элементах с двумя входными и одной выходной переменной

Показаны все типы декомпозиций для n=3, 4, 5, 6, для которых на каждом шаге декомпозиции новые переменные зависят от непересекающихся подмножеств входных переменных, унаследованных от предшествующей декомпозиции. Это наиболее подходящие типы декомпозиций. Числа на рис. 13 — это количество входных переменных, от которых зависят соответствующие выходные или промежуточные переменные.

Легко видеть, что вычислительная сложность задачи проектирования с ростом числа входных и выходных переменных проектируемой системы ИИ растет очень быстро. То же происходит и с ростом числа возможных типов элементов. Так, например, при n входных переменных и m выходных и одном типе элементов с двумя входными переменными и одной выходной число эффективных шагов декомпозиции, показанных на рис. 13, равно произведению $(n—1)m$, а для e типов элементов ($e \geq 2$) оно равно уже $e^{(n-1)m}$. **С каждой декомпозицией связано решение соответствующего функционального уравнения, оценка и сравнение его решений и отбор наиболее перспективных из них с точки зрения всего проекта.**

Есть два основных способа сделать обозримой задачу проектирования. Один из них — представление общего задания в виде иерархии частных заданий таким образом, чтобы на каждом уровне иерархии сложность проектирования была относительно небольшой. Подобная организация имеет некоторые дополнительные преимущества, связанные с надежностью, тестируемостью и восстанавливаемостью. **Другой способ уменьшения сложности проектирования состоит в ослаблении целевых критериев**. Вместо того чтобы требовать получения оптимального проекта, соглашаются на «хороший» или «приемлемый». **Такой подход к проектированию систем ИИ позволяет использовать эвристические методы.**
Из-за большого числа возможных типов элементов оказывается слишком сложным включить в ФРИЗ различные конкретные типы проектирования, основанные на определенных алгебрах или типах функциональных уравнений. Поэтому ФРИЗ должен служить источником информации для пользователя. Если для задачи проектирования, заданной пользователем, имеется некий определенный метод проектирования, ФРИЗ должен предоставить пользователю исчерпывающую информацию об этом методе. Средства ФРИЗ для решения проблемы проектирования на уровне структурированных систем ИИ должны разрабатываться, в рамках общего подхода к декомпозиции как оптимизации (для небольших задач) или как средства удовлетворения требований (основанного на соответствующих эвристических методах).

5.6. Задачи идентификации

Даже зная свойства частей и законы их ваимодействия, очень непросто вывести свойства целого.
В исследованиях систем важное место занимают две взаимодополняющие задачи, связанные с взаимоотношением обобщенной системы ИИ с поведением и разных множеств ее подсистем. Одна из них основывается на предположении, что система ИИ с поведением, рассматриваемая как обобщенная, уже задана. Задача состоит в определении того, какие структурированные системы ИИ, состоящие из множеств подсистем ИИ заданных обобщенных систем ИИ, подходят для реконструкции данной системы ИИ с поведением с приемлемым уровнем точности. Во втором случае структурированная система ИИ с поведением задана, и задача состоит в том, чтобы вывести свойства неизвестной обобщенной системы ИИ.

Эти задачи будем называть соответственно *задачей реконструкции и задачей идентификации.* В этом разделе рассматривается задача идентификации, а в следующем — задача реконструкции.
Задача идентификации распадается на две подзадачи. Одна состоит в определении множества всех обобщенных систем ИИ с поведением, представленных данной структурированной системой ИИ в том смысле, что функции поведения их элементов являются проекциями функции поведения любой из этих обобщенных систем ИИ. Это множество обобщенных систем ИИ будем называть *реконструктивным семейством* рассматриваемой структурированной системы ИИ. Вторая подзадача заключается в выборе из реконструктивного семейства такой обобщенной системы ИИ, которая задает в определенном смысле лучшую гипотезу относительно реальной обобщенной системы ИИ.

РЕКОНСТРУКТИВНОЕ СЕМЕЙСТВО

Рассмотрим структурированную систему ИИ с поведением **SF** вида (18) (поскольку до конца данной главы будут рассматриваться только системы ИИ с поведением, индекс *B* мы опускаем), элементы которой представлены множествами $^{x}\mathbf{S}$ выборочных переменных и функциями поведения $^{x}f\ (x \in N_q)$. Будем говорить, что система ИИ с поведением сопоставима с данной структурированной системой ИИ **SF**, если обе системы ИИ определены для одних и тех же параметров и переменных, а также используют один и тот же тип функций поведения (например, функции распределения вероятностей или возможностей). Обозначим через $\mathscr{C}_{\mathbf{SF}}$ множество функций поведения всех систем ИИ с поведением, сопоставимых с **SF**, а через $\mathscr{F}_{\mathbf{SF}}$ множество функций поведений систем ИИ с поведением из реконструктивного семейства **SF**.
$f \in \mathscr{F}_{\mathbf{SF}}$ тогда и только тогда, когда

$$[f \downarrow {}^{x}S] = {}^{x}f. \qquad (27)$$

для всех $x \in N_q$. Для вероятностных или возможностных функций поведения уравнения (27) представляются соответственно системами уравнений

$$^{x}f(^{x}\mathbf{c}) = \sum_{\mathbf{c} \succ {}^{x}\mathbf{C}} f(\mathbf{c}) \qquad (28)$$

или

$$ {}^{x}f({}^{x}\mathbf{c}) = \max_{\mathbf{c}>{}^{x}\mathbf{c}} f(\mathbf{c}), \quad (29) $$

где $x \in N_q$. В этих уравнениях определяются значения ${}^{x}f({}^{x}\mathbf{c})$, значения $f(\mathbf{c})$ определяются для всех обобщенных состояний входящих в уравнения переменных. Положим, что $\mathbf{c} \in \mathbf{C}$ и ${}^{x}\mathbf{c} \in {}^{x}\mathbf{C}$ для всех $x \in Nq$.

Чтобы определить значения $f(\mathbf{c})$, которые можно интерпретировать как вероятности или возможности состояний $\mathbf{c}$, уравнения (28) или (29) нужно дополнить требованием, чтобы $f(\mathbf{c}) \geq 0$ для всех $\mathbf{c} \in \mathbf{C}$. Несмотря на то, что $f(\mathbf{c})$ должны для вероятностных систем ИИ удовлетворять некоторым дополнительным требованиям, таким, как

$f(\mathbf{c}) \leq 1$

и

$$ \sum_{\mathbf{c} \in \mathbf{C}} f(\mathbf{c}) = 1, $$

легко увидеть, что эти дополнительные требования учитываются видом этих уравнений и требованием, чтобы $f(\mathbf{c}) \geq 0$ для всех $\mathbf{c} \in \mathbf{C}$.

Взаимоотношение между заданной структурированной системой ИИ и соответствующими обобщенными системами ИИ с поведением, описывается множеством из

$$ \sum_{x \in N_q} |\, {}^{x}\mathbf{C}\, | $$

уравнений вида (28) или (29) с $|\mathbf{C}|$ неизвестными, $f(\mathbf{c})$, удовлетворяющей неравенствам $f(\mathbf{c}) \geq 0$ для всех $\mathbf{c} \in \mathbf{C}$. Обычно некоторые из этих уравнений зависят от остальных и могут быть исключены. Если заданная структурированная система ИИ непротиворечива, то полученная система уравнений [т. е. система линейно-независимых уравнений вида (28)] имеет по крайней мере одно решение в области неотрицательных действительных чисел.

Идентификация обобщенной системы ИИ с поведением по заданной структурированной системе ИИ однозначна тогда и только тогда, когда решение ограниченной системы уравнений существует (т. е. структурированная система ИИ непротиворечива) и оно единственно. Это бывает довольно редко. Если решение не единственно, что бывает значительно чаще, то идентификация неоднозначна. По существу, это означает, что данная обобщенная система ИИ, хоть для нее и выполняются все ограничения на переменные структурированной системы ИИ, содержит некоторую дополнительную информацию. Это и есть конструктивное выражение известного утверждения науки о системах, что «целое больше суммы составляющих его частей».

Пример11. Рассмотрим структурированную систему, элементами которой являются системы с поведением без памяти, каждая из которых содержит по две из трех переменных v_1, v_2, v_3. Каждая переменная имеет одно из двух состояний 0 или 1. Вероятностные функции поведения элементов ${}^{1}f$, ${}^{2}f$, ${}^{3}f$ приведены в табл. 3.

Таблица 3.

Элементы структурированной системы из примера 11

v_1	v_2	${}^{1}f({}^{1}\mathbf{c})$	v_2	v_3	${}^{2}f({}^{2}\mathbf{c})$	v_1	v_3	${}^{3}f({}^{3}\mathbf{c})$
${}^{1}\mathbf{c}=0$ 0	0	0.4	${}^{2}\mathbf{c}=0$	0	0.4	${}^{3}\mathbf{c}=0$	0	0.4
0	1	0.3	0	1	0.2	0	1	0.3
1	0	0.2	1	0	0.1	1	0	0.1
1	1	0.1	1	1	0.3	1	1	0.2

Легко убедиться, что эта структурированная система локально согласована. Например, вероятность того, что $v_2 = 0$ равна 0.6 (и 0.4 для $v_2=1$) независимо от того, вычисляется она как проекция ${}^{1}f$ или ${}^{2}f$. Обозначим через p_0, p_1,..., p_7 (табл. 4) неизвестные вероятности состояний обобщенных систем с поведением.

Таблица 4

Обозначения, используемые в примерах 11, 12 и 14

v_1	v_2	v_3	$f(\mathbf{c})$
$\mathbf{c}=0$ 0	0	0	$f(000) = p_0$
0	0	1	$f(001) = p_1$
0	1	0	$f(010) = p_2$
0	1	1	$f(011) = p_3$
1	0	0	$f(100) = p_4$
1	0	1	$f(101) = p_5$
1	1	0	$f(110) = p_6$
1	1	1	$f(111) = p_7$

Тогда реконструктивное семейство данной структурированной системы описывается неравенствами $p_i \geq 0$ $(i \in N_{0,7})$ и 12 уравнениями вида (28)

$p_0+p_1=0.4$ (1), $p_0+p_4=0.4$ (5),
$p_0+p_2=0.4$ (9);

$p_2+p_3=0.3$ (2), $p_3+p_7=0.3$ (6),

$p_1+p_3=0.3$ (10),

$p_4+p_5=0.2$ (3), $p_1+p_5=0.2$ (7),

$p_5+p_7=0.2$ (11),

$p_6+p_7=0.1$ (4), $p_2+p_6=0.1$ (8),

$p_4+p_6=0.1$ (12).

Исследовав эти уравнения, получаем, что все неизвестные могут быть выражены через одну из них, скажем через p_0. В самом деле,

из (1): $p_1=0.4-p_0$;
из (5): $p_4=0.4-p_0$;
из (9): $p_2=0.4-p_0$;
из (2): $p_3=0.3-p_2=-0.1+p_0$;
из (3): $p_5=0.2-p_4=-0.2+p_0$;
из (6): $p_7=0.3-p_3=0,4-p_0$;
из (8): $p_6=0.1-p_2=-0.3+p_0$.

Таким образом,

$p_1=p_2=p_4=p_7=0.4-p_0$, (а)
$p_3=-0.1+p_0$, (б)
$p_5=-0.2+p_0$, (в)
$p_6=-0.3+p_0$.

Из неравенств для каждого из уравнений получаем следующие ограничения на p_0:

а) p_1 (или p_2, p_4, p_7) $\geqslant 0$ дает $p_0 \leqslant 0.4$;
б) $p_3 \geqslant 0$ дает $p_0 \geqslant 0.1$;
в) $p_5 \geqslant 0$ дает $p_0 \geqslant 0.2$;
г) $p_6 \geqslant 0$ дает $p_0 \geqslant 0.3$.

Так как должны выполняться все эти ограничения, из неравенств следует, что p_0 должно принимать значения в диапазоне

$0.3 \leq p_0 \leq 0.4$.

Если задаться значением p_0 из этого диапазона, то из уравнений а)—г) можно однозначно определить значения неизвестных. В табл. 5 показано, как таким образом определяется реконструктивное семейство для данного примера.

Таблица 5.

Реконструктивное семейство из примера 11

v_1	v_2	v_3	$p_i = f(\mathbf{c})$
$\mathbf{c}=0$	0	0	$0.3 \leqslant p_0 \leqslant 0.4$ (степень свободы)
0	0	1	$p_1 = 0.4 - p_0$
0	1	0	$p_2 = 0.4 - p_0$
0	1	1	$p_3 = -0.1 + p_0$
1	0	0	$p_4 = 0.4 - p_0$
1	0	1	$p_5 = -0.2 + p_0$
1	1	0	$p_6 = -0.3 + p_0$
1	1	1	$p_7 = 0.4 - p_0$

Пример 12. Рассмотрим структурированную систему, элементами которой являются системы с поведением без памяти с переменными v_1, v_2 и v_2, v_3 с двумя состояниями каждая. Вероятностная функция элементов ${}^1f$, ${}^2f$ определяется следующим образом:

	v_1	v_2	${}^1f({}^1\mathbf{c})$
${}^1\mathbf{c}=$	0	1	0.3
	1	0	0.5
	1	1	0.2

	v_2	v_3	${}^2f({}^2\mathbf{c})$
${}^2\mathbf{c}=$	0	0	0.1
	0	1	0.4
	1	1	0.5

При определении реконструктивного семейства этой системы будем использовать для неизвестных те же обозначения, что и в примере 11. Тогда реконструктивное семейство будет описываться неравенствами $p_i \geq (i \in N_{0,7})$ и следующими восемью уравнениями:

$p_0+p_1=0.0$, (1) $p_0+p_4=0.1$, (5)
$p_2+p_3=0.3$, (2) $p_1+p_5=0.4$, (6)
$p_4+p_5=0.5$, (3) $p_2+p_6=0.0$, (7)
$p_6+p_7=0.2$, (4) $p_3+p_7=0.5$, (8)

Из уравнений (1), (7) и неравенств имеем $p_0 = p_1 = p_2 = p_6 = 0$. Отсюда просто определить оставшиеся вероятности: $p_3 = 0.3$, $p_4 = 0.1$, $p_5 = 0.4$,

$p_7 = 0.2$. Следовательно, в данном случае идентификация однозначна. То есть данный пример — это пример одного из тех редких случаев, когда «целое равно сумме составляющих его частей».

Пример 13. Чтобы продемонстрировать более общий тип реконструктивного семейства, рассмотрим структурированную систему из трех элементов, причем каждый элемент содержит три переменных v_1, v_2, v_3. Пусть v_1 и v_2 принимают состояния из множества {0, 1}, а v_3—из множества {0, 1, 2}. Элементы представляют собой вероятностные системы с поведением без памяти. Их функции поведения ${}^1f, {}^2f, {}^3f$ приведены в табл. 6.

Таблица 6.

Элементы структурированной системы из примера 13

v_1	v_2	${}^1f({}^1\mathbf{c})$	v_2	v_3	${}^2f({}^2\mathbf{c})$	v_1	v_3	${}^3f({}^3\mathbf{c})$
${}^1\mathbf{c}=0$ 0	0	0.25	${}^2\mathbf{c}=0$ 0	0	0.17	${}^3\mathbf{c}=0$ 0	0	0.11
0	1	0.18	0	1	0.16	0	1	0.14
1	0	0.20	0	2	0.12	0	2	0.18
1	1	0.37	1	0	0.14	1	0	0.20
			1	1	0.18	1	1	0.20
			1	2	0.23	1	2	0.17

Будем использовать приведенные в табл. 7 обозначения неизвестных вероятностей состояний полной системы.

Таблица 7.

Обозначения, используемые в примере 13

v_1	v_2	v_3	$f(\mathbf{c})$
$\mathbf{c}=0$	0	0	$f(000)=p_0$
0	0	1	$f(001)=p_1$
0	0	2	$f(002)=p_2$
0	1	0	$f(010)=p_3$
0	1	1	$f(011)=p_4$
0	1	2	$f(012)=p_5$
1	0	0	$f(100)=p_6$
1	0	1	$f(101)=p_7$
1	0	2	$f(102)=p_8$
1	1	0	$f(110)=p_9$
1	1	1	$f(111)=p_{10}$
1	1	2	$f(112)=p_{11}$

Составление уравнений, описывающих реконструктивное семейство, и определение их решений при ограничениях $p_i \geq (i \in N_{0,1})$ предоставим читателю. Скажем только, что система из 16 исходных уравнений с 12 неизвестными сводится к 10 линейно-независимым уравнениям с двумя степенями свободы. Если предположить, что свободными являются неизвестные p_{10} и p_{11}, то мы, получим следующие неравенства:

$$0.04 \leqslant p_{10} \leqslant 0.18,$$

$$0.05 \leqslant p_{11} \leqslant 0.17,$$

$$0.23 \leqslant p_{10} + p_{11} \leqslant 0.34.$$

Область значений для p_{10} и p_{11}, определяемая этими уравнениями, показана на рис. 14; это выпуклое множество, натянутое на четыре точки (0.06, 0.17), (0.17, 0.17), (0.18, 0.16), (0.18, 0.05).

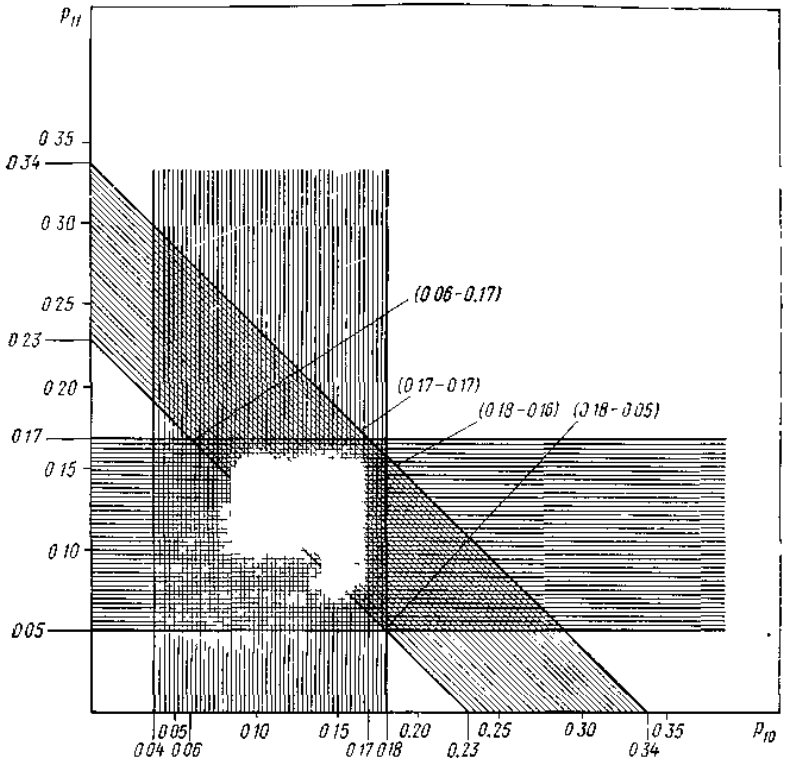


Рис. 14. Описание реконструктивного семейства из примера 13

Для любой пары значений p_{10} и p_{11} из этой области значения остальных неизвестных однозначно определяются следующим образом:

$p_0 = 0.34 - p_{10} - p_{11}$,
$p_1 = -0.04 + p_{10}$,
$p_2 = -0.05 + p_{11}$,
$p_3 = -0.023 + p_{10} + p_{11}$,
$p_4 = 0.18 - p_{10}$,
$p_5 = 0.23 - p_{11}$,
$p_6 = -0.17 + p_{10} + p_{11}$,
$p_7 = 0.20 - p_{10}$,
$p_8 = 0.17 - p_{11}$,
$p_9 = 0.37 - p_{10} - p_{11}$.

Пример 14. Рассмотрим структурированную систему, элементами которой являются возможностные системы с поведением с теми же наборами переменных, что и в примере 12. Функции поведения элементов приведены в табл. 8.

Таблица 8.

Элементы структурированной системы из примера 14

v_1	v_2	$^1f(^1c)$	v_2	v_3	$^2f(^2c)$
$c = 0$	0	0.8	$^2c = 0$	0	0.8
0	1	0.5	0	1	0.9
1	0	0.9	1	0	0.8
1	1	0.8	1	1	0.6

Как видно, они не нормализованы.

Для описания реконструктивного семейства этой структурированной системы снова воспользуемся обозначениями из табл. 4. Тогда реконструктивное семейство определяется неравенствами $p_i \geq (i \in N_{0,7})$ и следующими восемью уравнениями в виде (29):

$\max(p_2, p_3) = 0.5$, (1)
$\max(p_3, p_7) = 0.6$, (2)
$\max(p_0, p_1) = 0.8$, (3)
$\max(p_6, p_7) = 0.8$, (4)
$\max(p_0, p_4) = 0.8$, (5)
$\max(p_2, p_6) = 0.8$, (6)
$\max(p_4, p_5) = 0.9$, (7)
$\max(p_1, p_5) = 0.9$, (8).

Из уравнения (1) следует, что ни p_2, ни p_3 не могут быть больше 0. 5; следовательно, из (6) следует, что $p_6 = 0.8$, а из уравнения (2), что

$p_7 = 0.6$. Из уравнения (5) следует, что p_4 не может быть больше 0.8, а значит, из уравнения (7) следует, что $p_5 = 0.9$. Согласно этим результатам данная система может быть сведена к двум независимым подсистемам уравнений

$\max(p_2, p_3) = 0.5$,
$\max(p_0, p_1) = 0.8$,
$\max(p_0, p_4) = 0.8$,

Единственное уравнение при условии, что $p_2 \geq 0$ и $p_3 \geq 0$, имеет решение
$p_2 = 0.5$ и $0 \leq p_3 \leq 0.5$
или
$p_3 = 0.5$ и $0 \leq p_2 \leq 0.5$

Пара уравнений, при условии, что $p_2 \geq 0$, $p_3 \geq 0$ и $p_4 \geq 0$, имеет решение

$$p_0 = 0.8 \text{ и } 0 \leqslant p_1 \leqslant 0.8, \text{ а также } 0 \leqslant p_4 \leqslant 0.8$$

или

$$p_1 = p_4 = 0.8 \text{ и } 0 \leqslant p_0 \leqslant 0.8.$$

В данном реконструктивном семействе легко выделить одно максимальное и четыре минимальных распределения возможностей

Они приведены в табл. 9, где через f^{SF} и $f_{SF,i}$ $(i \in N_4)$ обозначены соответственно максимальное и минимальные распределения, а через **SF** — рассматриваемая структурированная система.

Таблица 9.

Максимальное и минимальные распределения возможностей для реконструктивного семейства из примера 14

v_1	v_2	v_3	f^{SF}	$f_{SF,1}$	$f_{SF,2}$	$f_{SF,3}$	$f_{SF,4}$
$c = 0$	0	0	0.8	0.8	0.8	0	0
0	0	1	0.8	0	0	0.8	0.8
0	1	0	0.5	0.5	0	0.5	0
0	1	1	0.5	0	0.5	0	0.5
1	0	0	0.8	0	0	0.8	0.8
1	0	1	0.9	0.9	0.9	0.9	0.9
1	1	0	0.8	0.8	0.8	0.8	0.8
1	1	1	0.6	0.6	0.6	0.6	0.6

Реконструктивное семейство состоит из этих четырех распределений, а также из всех распределений, находящихся между максимальным распределением и любым из четырех минимальных

Для возможностных структурированных систем известно, что в общем случае их реконструктивные семейства всегда имеют вид, что и в примере 14. То есть реконструктивное семейство всегда содержит одно *максимальное семейство* (предполагается, что данная структурированная система согласована) и множество решений, состоящее из возможностных распределений, находящихся между максимальным решением и одним из нескольких *минимальных решений.* Более того, возможность любого полного состояния переменных, входящих в любое минимальное решение, равна или возможности этого состояния в максимальном решении, или нулю. Данный результат, равно как и некоторые другие результаты, связанные с задачей определения реконструктивного семейства для заданной структурированной системы, получены и для возможностных, и для вероятностных систем.

КОЭФФИЦИЕНТ ИДЕНТИФИЦИРУЕМОСТИ

Часто необходимо иметь подходящую меру для оценки размера реконструктивного семейства. Если эта мера является адекватной, то ее можно использовать для оценки нечеткости, связанной с реконструкцией обобщенной системы ИИ по заданной структурированной системе ИИ, а также как степень идентифицируемости реальной структурированной системы ИИ.

Для возможностных систем ИИ размер реконструктивного семейства адекватно оценивается произведением

$$\prod_{\mathbf{c}\in\mathbf{A}} [1 + f^{\mathbf{SF}}(\mathbf{c}), \tag{30}$$

где $f^{\mathbf{SF}}$—максимальный элемент реконструктивного семейства $\mathscr{F}_{\mathbf{SF}}$, а **A** — множество всех полных состояний, для которых степень возможности в реконструктивном семействе определяется не единственным образом [т. е. множество полных состояний, для которых решение ограниченной системы уравнений вида (29) не единственное]. Обратите внимание, что это произведение всегда больше или равно 1 и что его значение пропорционально мощности множества **A** и значениям $f^{\mathbf{SF}}(\mathbf{c})$; оно равно 1 только тогда, когда множество **A** пустое (т. е. если существует единственное решение).

Если принять произведение (30) в качестве разумной оценки размера реконструктивного семейства, то естественно было бы определить

реконструктивную нечеткость $u^{\mathbf{SF}}$, связанную со структурированной системой **SF**, как логарифм этого произведения, т. е.

$$u_{\mathbf{SF}} = \log_2 \prod_{\mathbf{c}\in\mathbf{A}} [1 + f^{\mathbf{SF}}(\mathbf{c})] = \sum_{\mathbf{c}\in\mathbf{A}} \log_2 [1 + f^{\mathbf{SF}}(\mathbf{c})]. \tag{31}$$

Понятно, что

$$0 \leqslant u_{\mathbf{SF}} \leqslant |\mathbf{C}|,$$

где считается, что |**C**| —реконструктивная нечеткость всего множества $\mathscr{C}_{\mathbf{SF}}$ обобщенных систем, сопоставимых с **SF**. Можно использовать меру

$$\mathbf{I}_{\mathbf{SF}} = \frac{|\mathbf{C}| - u_{\mathbf{SF}}}{|\mathbf{C}|} = 1 - \frac{u_{\mathbf{SF}}}{|\mathbf{C}|}, \tag{32}$$

называемую *коэффициентом идентифицируемости,* в качестве разумного показателя возможности определения единственной обобщенной системы ИИ по заданной структурированной системе ИИ **SF**. Понятно, что

$$0 \leqslant \mathbf{I}_{\mathbf{SF}} \leqslant 1.$$

I = 1 только тогда, когда $|\mathscr{F}_{\mathbf{SF}}| = 1$; $\mathbf{I}_{\mathbf{SF}} = 0$ только тогда, когда

$$|\mathbf{A}| = |\mathbf{C}|, \text{ и } f^{\mathbf{SF}}(\mathbf{c}) = 1 \text{ для всех } \mathbf{c} \in \mathbf{C}.$$

Коэффициент идентифицируемости бывает полезен при решении некоторых интеллектуальных задач, особенно при сравнительных исследованиях структурированных систем ИИ. В общем случае значительно легче определить коэффициент идентифицируемости структурированной системы ИИ, чем реконструктивное семейство: для этого достаточно определить максимальное решение и состояния с единственными решениями (т. е. **C**—**A**).

Пример 15. Определим коэффициент идентифицируемости структурированной системы, определенной в примере 14. Отметим, что в этом примере |**C**|=8, а множество **A** состоит из первых пяти состояний, перечисленных в табл. 9. Используя значения $f_{\mathbf{SF}}(\mathbf{c})$ для этих состояний, получим реконструктивную нечеткость

$$u_{\mathbf{SF}} = 3\log_2 1.8 + 2\log_2 1.5 = 3.714.$$

Отсюда

$$\mathbf{I}_{\mathbf{SF}} = 1 - 3{,}714/8 = 0.536.$$

ЕДИНСТВЕННЫЙ ВЫБОР ИЗ РЕКОНСТРУКТИВНОГО СЕМЕЙСТВА

Рассмотрим теперь вторую подзадачу задачи идентификации—задачу выбора из реконструктивного семейства одной обобщенной системы ИИ как гипотезы о реальной обобщенной системе ИИ. Эта задача тривиальна, если реконструкция однозначна (т. е. если $\mathbf{I_{SF}}= 1$).
В остальных случаях (если данная структурированная система ИИ **SF** согласована и, следовательно $f_{\mathbf{SF}} \neq 0$, этот выбор совершенно произволен, если только мы не определим некий критерий и не потребуем, чтобы система ИИ, выбранная из реконструктивного семейства, наилучшим образом удовлетворяла этому критерию. В этом случае задача выбора превращается в задачу оптимизации с последующим произвольным выбором из лучших, с точки зрения данного критерия, систем ИИ. Если исполнительный критерий оптимизации обеспечивает единственность оптимизации, то из данной задачи исключается элемент произвольности.
Критерии оптимизации всегда используются исходя из неких фундаментальных соображений и, следовательно, определяются этими соображениями. **С эпистемологической точки зрения самым существенным соображением для выбора обобщенной системы ИИ является ее *максимальная независимость* со всех точек зрения, за исключением только условия для проекций (27).** Более конкретно это соображение можно сформулировать так: **для заданной структурированной системы ИИ из реконструктивного семейства следует выбирать такую обобщенную систему ИИ, которая опирается на *всю информацию,* содержащуюся в этой структурированной системе ИИ, но только на эту информацию.**
Такую обобщенную систему ИИ можно было бы назвать *несмещенной реконструкцией.* Это система, реконструированная по структурированной системе ИИ без смещений, т. е., с одной стороны, использующая всю имеющуюся информацию, а с другой, — не использующая никакой другой дополнительной информации.
Выбор несмещенной реконструкции, по существу, представляет собой *индуктивный вывод.* Он может быть описан как следующая оптимизационная задача.
По заданной структурированной системе ИИ с поведением **SF** из множества функций реконструктивного семейства $\mathcal{F}_{\mathbf{FS}}$ выбрать **SF** такую функцию поведения $f^{\mathbf{SF}}$, для которой мера нечеткости (шенноновская энтропия для вероятностных систем или *U*-нечеткость для возможностных) была бы максимальной при условии, что выполняются ограничения на проекции (27).

Для вероятностных систем ИИ эта задача оптимизации будут фигурировать под названием *«принцип максимума энтропии».*
Известно, что несмещенная реконструкция единственна и для вероятностных, и для возможностных систем. Она определяет самое слабое из возможных ограничений на переменные, соответствующие заданной структурированной системе. Для возможностных систем $f^{\mathbf{SF}}$ это максимальное распределение из реконструктивного семейства $\mathcal{F}_{\mathbf{SF}}$, или, другими словами, это распределение из множества распределений $\mathcal{F}_{\mathbf{SF}}$, которое представляет наибольшее нечеткое подмножество всех полных состояний переменных систем.
Несмотря на то, что несмещенная реконструкция эпистемологически наиболее существенна, поскольку она опирается на один- - единственный хорошо обоснованный принцип индуктивного вывода, для других целей могут лучше подойти другие реконструкции.
Совершенно естественным и важным примером этого может послужить выбор обобщенной системы ИИ, для которой минимизирована наибольшая возможная ошибка**. Под ошибкой здесь имеется в виду расстояние между распределением (вероятностным или возможностным) реконструированной обобщенной системы ИИ и распределением истинной системы ИИ.** Такого типа реконструкция — это реконструкция с *наименьшим риском.*
Конкретная формулировка полученной оптимизационной задачи зависит от того, какой тип расстояния используется. **Особую важность из всех возможных типов расстояния имеют те, что оценивают потерю информации.** Далее в этой главе, в особенности в разд. 5.7 и 5.9, мы рассмотрим такие типы расстояний.

ПРОЦЕДУРЫ СОЕДИНЕНИЯ

Одним из важнейших результатов, связанных с задачей идентификации, является то, что несмещенная реконструкция может быть определена с помощью относительно простой процедуры, не включающей решение описанной выше задачи оптимизации (задачи максимизации шенноновской энтропии или U-нечеткости при заданных ограничениях). Эта процедура, называемая *процедурой соединения,* основана на вероятностном или возможностном варианте операции соединения довольно простым образом комбинирующей функции поведения элементов заданной структурированной системы.
Рассмотрим две функции поведения

$^1f : \mathbf{A}\times\mathbf{B}\to[0, 1]$,

$^2f : \mathbf{B}\times\mathbf{C}\to[0, 1]$,

определенные на множествах состояний **A, B, C**, смысл которых мы поясним ниже. Обратите внимание, что в множество **B** входит область определения обеих функций. *Соединение ^1f и ^2f,* обозначаемое как $^1f * {}^2f$, это функция

$^1f * {}^2f : \mathbf{A}\times\mathbf{B}\times\mathbf{C}\to[0, 1]$,

свойства которой зависят от природы функций ^1f и ^2f. Если это функции распределения вероятностей, то

$$[^1f * {}^2f](\mathbf{a}, \mathbf{b}, \mathbf{c}) = \min[^1f(\mathbf{a}, \mathbf{b}), {}^2f(\mathbf{c}|\mathbf{b}), \quad (33)$$

где $^2f(\mathbf{c}|\mathbf{b})$—условная вероятность **c** при заданном **b**; если это функции распределения возможностей, то

$$[^1f * {}^2f](\mathbf{a}, \mathbf{b}, \mathbf{c}) = \min[^1f(\mathbf{a}, \mathbf{b}), {}^2f(\mathbf{b}, \mathbf{c})]. \quad (34)$$

Обратим внимание, что в отличие от (33) в (34) не используются условные возможности. Дело в том, что тут используется соотношение

$$\min[^1f(\mathbf{a}, \mathbf{b}), {}^2f(\mathbf{c}|\mathbf{b})] = \min[^1f(\mathbf{a}, \mathbf{b}), {}^2f(\mathbf{b}, \mathbf{c})],$$

которое легко может быть доказано.

Пусть операция соединения применяется к двум элементам структурированной системы ИИ с выборочными переменными из множеств ^1S и ^2S и функциями поведения ^1f и ^2f. Тогда необходимо преобразовать области определения функций ^1f и ^2f к виду **A**× **B** и **B**×**C** соответственно, где

A — множество совокупных состояний переменных, входящих только в первый элемент, т. е. переменных из множества ^1S —($^1S \cap {}^2S$).

B — множество совокупных состояний переменных, входящих в оба элемента, т. е. переменных из $^1S \cap {}^2S$;

C — множество совокупных состояний переменных, входящих только во второй элемент, т. е. переменных из множества ^2S— ($^1S \cap {}^2S$).

Для определения несмещенной реконструкции данной структурированной системы ИИ операция соединения должна быть выполнена для пар элементов этой системы. При каждом ее выполнении два элемента сливаются (объединяются) в один больший элемент новой структурированной системы ИИ. Пусть операция соединения всегда выполняется в таком порядке, чтобы результат ранее выполненных операций соединения входил в операцию соединения в качестве второго элемента. Эта процедура заканчивает свою работу тогда, когда все элементы сливаются в одну обобщенную систему ИИ.

Будем называть эту процедуру *базовой процедурой соединения.* Прежде чем ее формализовать, нужно рассмотреть два вырожденных случая, чтобы все содержательные ситуации были описаны. В первом случае все переменные из первого элемента (соответствующего ^1f) могут быть включены во второй элемент (соответствующий ^2f). Это может быть, поскольку, вообще говоря, второй элемент получен в результате выполнения неких операций соединения. В этом случае множество **A** является пустым, а ^1f принимает вырожденный вид

$^1f : \mathbf{B}\to[0, 1]$.

Во втором случае элементы не соединены. Это значит, что множество **B** является пустым, а функции поведения имеют вырожденный вид

$^1f : \mathbf{A}\to[0, 1]$,

$^2f : \mathbf{B}\to[0, 1]$.

Обратите внимание на то, что вследствие требования неизбыточности для структурированных систем ИИ (см. разд. 5.3) и договоренности о том, что результат предшествующих операций соединения всегда выступает как ^2f, множество **C** не может быть пустым. Для вероятностных систем ИИ вырожденные операции соединения определяются так:

$$[^1f * {}^2f](\mathbf{b}, \mathbf{c}) = {}^1f(\mathbf{b}) \cdot {}^2f(\mathbf{c}|\mathbf{b}) \quad (35)$$

для $\mathbf{A} = \varnothing$ и

$$[^1f * {}^2f](\mathbf{a}, \mathbf{c}) = {}^1f(\mathbf{a}) \cdot {}^2f(\mathbf{c}) \quad (36)$$

для $\mathbf{B} = \varnothing$; для возможностных систем они определяются следующим образом:

$$[^1f * {}^2f](\mathbf{b}, \mathbf{c}) = \min[^1f(\mathbf{b}), {}^2f(\mathbf{b}, \mathbf{c})], \quad (37)$$

$$[^1f * {}^2f](\mathbf{a}, \mathbf{c}) = \min[^1f(\mathbf{a}), {}^2f(\mathbf{c})]. \quad (38)$$

Через $^1f * {}^2f$ будем обозначать в зависимости от контекста как обычную, так и вырожденную операцию соединения. **Тогда базовая процедура соединения описывается следующим алгоритмом.**

Базовая процедура соединения. Дана локально согласованная структурированная система ИИ с поведением **SF** (вероятностная или возможностная) с функциями поведения ^xf $(x \in N_q)$. Для определения соединения ^xf для всех $x \in N_g$:

1) положить $k = 2$ и $f = {}^1f$;

2) произвести соответствующую группировку аргументов ^kf и f и выполнить соответствующий вариант операции соединения $^kf * f \to f$ (операция может быть вероятностной или возможностной, обычной или вырожденной);

3) если $k<q$, то $k+1 \to k$ и перейти на шаг 2;

4) стоп.

Можно доказать следующее утверждение : если базовая процедура соединения применяется к согласованной возможностной системе ИИ **SF**, то в результате всегда получается несмещенная реконструкция $f^{\mathbf{SF}}$ (максимальное для реконструктивного семейства $\mathcal{F}_{\mathbf{SF}}$ распределение возможностей). Однако если эта процедура применяется к вероятностной системе, то несмещенная (максимум энтропии) реконструкция получается только для определенного класса структурированных систем ИИ — так называемых ациклических структурированных систем ИИ, описываемых в разд. 5.7.

По полученному результату, не определяя типа данной *структурированной системы ИИ* можно определить результат применения базовой процедуры соединения f несмещенную реконструкцию или нет. Если f удовлетворяет условию для проекции

$$[\hat{f}\downarrow^{x}S] = {}^{x}f$$

для всех $x \in N_q$, то это несмещенная реконструкция, в противном случае f не соответствует данной структурированной системе ИИ и должна быть уточнена с помощью следующей *итеративной процедуры соединения.*

Итеративная процедура соединения. Дана локально согласованная структурированная система ИИ с поведением SF с вероятностными функциями поведения ${}^{1}f$ ($j \in N_{0,g-1}$). Дана также функция f, полученная с помощью базовой процедуры соединения, примененной к **SF**, и число $\Delta \in [0, 1]$; требуется с точностью до Δ определить функцию поведения несмещенной реконструкции:

1) присвоить $f = 0$, i=1 и $f_0 = f$;

2) сделать соответствующее разбиение аргументов ${}^{1}f$ и f_{i-1} и выполнить операцию соединения ${}^{1}f * f_{i-1} \to \tilde{f}_i$ для вырожденного вида (35);

3) если $i \neq 0 \pmod q$, то $i+1 \to i$, $j+1 \pmod q \to j$, и перейти на 2;

4) если $|\hat{f}_i(\mathbf{c}) - \tilde{f}_{i-q}(\mathbf{c})| > \Delta$ для какого-то $\mathbf{c} \in \mathbf{C}$, то $i+1 \to i$, $j+1 \pmod q \to j$ и перейти на 2;

5) конец.

Если после выполнения итеративной процедуры соединения $\sum_{\mathbf{c}} f_i(\mathbf{c}) = 1$, то

$$f^{\mathbf{SF}}(\mathbf{c}) - \Delta \geqslant f_i(\mathbf{c}) \leqslant f^{\mathbf{SF}}(\mathbf{c}) + \Delta$$

для всех $\mathbf{c} \in \mathbf{c}$; в противном случае данная структурированная система **SF** глобально не согласована (см. разд. 5.11) и реконструкции **SF** не существует, т. е.

$$\mathcal{F}_{\mathbf{SF}} = \emptyset,$$

и, следовательно, **SF** бессодержательна.

Пример 16. Рассмотрим структурированную систему из примера 11, определенную в табл. 3. Сначала используем для определения несмещенной реконструкции вероятностный вариант базовой процедуры соединения. В первых столбцах табл. 10 приводятся промежуточный результат ${}^{2}f*{}^{1}f$ и окончательный результат $f = {}^{3}f*({}^{2}f*{}^{1}f)$.

Таблица 10.

Последовательность функций поведения, полученных с помощью базовой и итеративной процедур соединения, пример 16

	Базовая процедура соединения		Итеративная процедура соединения				
$v_1\ v_2\ v_3$	${}^{2}f*{}^{1}f$	${}^{3}f*({}^{2}f*{}^{1}f)$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
0 0 0	0.26	0.312195	0.294647	0.302908	0.309934	0.304982	0.307696
0 0 1	0.13	0.111628	0.105353	0.099904	0.096561	0.095018	0.093371
0 1 0	0.075	0.087805	0.095379	0.088024	0.090066	0.092059	0.089834
0 1 1	0.225	0.188372	0.204621	0.210484	0.203439	0.207941	0.209672
1 0 0	0.13	0.084211	0.094444	0.097092	0.089019	0.091490	0.092304
1 0 1	0.06	0.094118	0.105556	0.100096	0.105580	0.108510	0.106629
1 1 0	0.025	0.015789	0.012977	0.011976	0.010981	0.010418	0.010166
1 1 1	0.075	0.105882	0.087023	0.089516	0.094420	0.089582	0.090328
$i=6$	$i=7$	$i=8$	$i=9$	$i=10$	$i=11$	$i=12$	$i=13$
0.309608	0.308036	0.308915	0.309511	0.309002	0.309289	0.309481	0.309315
0.092433	0.091964	0.091444	0.091148	0.090998	0.090829	0.090734	0.090655
0.090392	0.091011	0.090314	0.090489	0.090688	0.090462	0.090519	0.090584
0.207567	0.208989	0.209528	0.208852	0.209312	0.209486	0.209266	0.209416
0.090079	0.090826	0.091085	0.090388	0.090626	0.090711	0.090466	0.090563
0.108277	0.109174	0.108556	0.109086	0.109374	0.109171	0.109343	0.109237
0.009921	0.009761	0.009686	0.009612	0.009561	0.009538	0.009514	0.009498
0.091723	0.090239	0.090472	0.090914	0.090439	0.090514	0.090557	0.090502
$i=14$	$i=15$	$i=16$	$i=17$	$i=18$	$i=19$	$i=20$	$i=21$
0.309409	0.309472	0.309417	0.309448	0.309469	0.309451	0.309461	0.309458
0.090630	0.090599	0.090583	0.090565	0.090555	0.090549	0.090543	0.090540
0.090510	0.090528	0.090550	0.090525	0.090531	0.090538	0.090530	0.090532
0.209473	0.209401	0.209450	0.209469	0.209445	0.209462	0.209468	0.209460
0.090591	0.090518	0.090543	0.090552	0.090528	0.090536	0.090539	0.090531
0.109370	0.109427	0.109457	0.109435	0.109454	0.109464	0.109457	0.109463
0.009490	0.009482	0.009477	0.009476	0.009472	0.009470	0.009470	0.009469
0.090527	0.090573	0.090523	0.090531	0.090545	0.090530	0.090532	0.090537

Легко видеть, что окончательный результат f не соответствует заданной структурированной системе. Так, например, $[f\downarrow\{v_1, v_2\}]$ (0 0) =0.312195+0.111628 = 0.423823 не равно ^1f (0 0)=0.4, так что требование для проекций (27) не выполняется. Следовательно, необходимо использовать итеративную процедуру соединения. Последовательность порождаемых процедурой функций поведения сходится к несмещенной реконструкции. Она приводится в табл. 10 для $i=1,$ 2, ...,21. Результаты проверки на сходимость — шаг 4 — алгоритм процедуры соединения— приведены в табл. 11.

Таблица 11.

Проверка на сходимость — шаг 4 итеративной процедуры соединения из примера 16

			$\|f_i(\mathbf{c}) - f_{i-q}(\mathbf{c})\| \quad i = 0 \pmod q$						
c_1	c_2	c_3	$i=3$	$i=6$	$i=9$	$i=12$	$i=15$	$i=18$	$i=21$
0	0	0	0.002261	0.000326	0.000097	0.000030	0.000009	0.000003	0.000001
0	0	1	0.015067	0.004128	0.001285	0.000414	0.000135	0.000044	0.000015
0	1	0	0.002261	0.000326	0.000097	0.000030	0.000009	0.000003	0.000001
0	1	1	0.015067	0.004128	0.001285	0.000414	0.000135	0.000044	0.000015
1	0	0	0.004808	0.001060	0.000309	0.000098	0.000032	0.000010	0.000003
1	0	1	0.011462	0.002697	0.000809	0.000257	0.000084	0.000027	0.000009
1	1	0	0.004808	0.001060	0.000309	0.000098	0.000032	0.000010	0.000003
1	1	1	0.011462	0.002697	0.000809	0.000257	0.000084	0.000027	0.000009

Отсюда, если $\Delta>0.15067$, то выполнение процедуры завершится при $i = 3$; если $\Delta>0.004128$, то выполнение завершится при $i = 6$ и т. д. Так как $\Delta = 0.00002$, то процедура завершится при $i = 21$.

5.7. Задача реконструкции ИИ

Задача реконструкции может быть сформулирована следующим образом: дана система ИИ с поведением, рассматриваемая как обобщенная система ИИ; требуется определить, какие наборы ее подсистем ИИ, а каждый такой набор рассматривается как гипотетическая реконструкция, подходят для реконструкции заданной системы ИИ с заданной точностью, причем реконструкция должна производиться только по той информации, что содержится в этих подсистемах ИИ.

Укажем, прежде всего, на то, что согласно такой формулировке задачи понятие «реконструкция» становится более конкретным: при

реконструкции используется вся информация, полученная из подсистем ИИ, но только она. Это означает, что требуется, чтобы реконструкция была несмещенной в том смысле, как это определялось в разд. 5. 6, и, следовательно, можно использовать для ее получения соответствующую процедуру соединения.

В задаче идентификации несмещенная реконструкция представляет собой хорошо обоснованную гипотезу (оценку) относительно неизвестной обобщенной системы ИИ, причем гипотеза эта строится по заданной структурированной системе ИИ. Поскольку истинная обобщенная система ИИ неизвестна, то невозможно определить, насколько близка к ней эта гипотетическая система ИИ. В задаче реконструкции несмещенная реконструкция описывает реконструктивные возможности рассматриваемой гипотезы относительно заданной обобщенной системы ИИ. Чем ближе несмещенная реконструкция к истинной (заданной) системе ИИ, тем лучше гипотеза.

В общем случае *близость* двух сопоставимых систем ИИ с поведением может быть выражена через метрическое расстояние между их функциями поведения. Существует много разных типов метрических расстояний. Так, например, *класс расстояний Минковского* определяется следующей формулой:

$$\delta_p(f, f^h) = \left[\sum_{\mathbf{c}\in\mathbf{C}} | f(\mathbf{c}) - f^h(\mathbf{c}) |^p \right]^{1/p}, \quad (39)$$

где f, f^h — соответственно функция поведения заданной системы и несмещенная реконструкция по гипотезе $h,$ а $p \in \mathbf{N}$ —параметр функций расстояния. При $p=1$— это расстояние Хэмминга, при p = 2 — Евклида, при $p = \infty$ — верхняя граница расстояний.

Расстояния по Минковскому вычисляют по точечным разностям

$$|f(\mathbf{c}) - f^h(\mathbf{c})|$$

распределение вероятностейили возможностей в соответствии с формулой (39). Несмотря на то, что поточечное описание близости f и f^h полезно для многих приложений, теоретически оно недостаточно хорошо обосновано. **Более обосновано рассматривать близость как разности информации, содержащейся в *h* относительно *f* , *или,* другими словами, как потерю информации при замене *f* на *h* (на множество проекций *f*).**

Меру подобной потери информации будем называть *информационным расстоянием* и обозначать *D(f, f^h).* Для вероятностных систем она задается хорошо известной формулой

$$D(f, f^h) = \frac{1}{\log_2 |\mathbf{C}|} \sum_{\mathbf{c} \in \mathbf{C}} f(\mathbf{c}) \log_2 \frac{f(\mathbf{c})}{f^h(\mathbf{c})}, \quad (40)$$

где константа $1/\log_2|\mathbf{C}|$—нормирующий коэффициент, обеспечивающий выполнение соотношения

$$0 \leqslant D(f, f^h) \leqslant 1.$$

Поскольку, если $f^h(\mathbf{c})=0$, то $f(\mathbf{c})=0$, вероятностное информационное расстояние определено всегда. Это, однако, не метрическое расстояние, так как оно асимметрично, более того, $D(f^h, f)$ может быть не определено для некоторых f и f^h [когда $f^h(\mathbf{c})>0$ и $f(\mathbf{c}) = 0$ для некоторого $\mathbf{c} \in \mathbf{C}$].

При применении вероятностного расстояния к порождающей системе ИИ с поведением уравнение (40) приобретает следующий вид:

$$D_G(f, f^h) = \frac{1}{\log_2 |G|} \sum_{\bar{g} \in G} f(\bar{g}) \sum_{g \in G} f(g \mid \bar{g}) \log \frac{f(g \mid \bar{g})}{f^h(g \mid \bar{g})}. \quad (41)$$

Модификация уравнений (40) и (41) для направленных систем ИИ с поведением очевидна.

Для возможностных систем ИИ информационное расстояние рассчитывается по формуле

$$D(f, f^h) = \frac{1}{\log_2 |\mathbf{C}|} \int_0^1 \log_2 \frac{|c(f^h, l)|}{|c(f, l)|}, \quad (42)$$

представляющей собой аналог вероятностного информационного расстояния (40) для U-нечеткости.

Далее в этом разделе, после соответствующего описания свойств реконструктивных гипотез, будет описано применение информационных расстояний для сравнения этих гипотез.

Реконструктивная гипотеза для заданной обобщенной системы ИИ с поведением представляет собой набор ее подсистем. Если обобщенная система ИИ состоит из n переменных, то число ее подсистем, содержащих по крайней мере одну переменную равно 2^n— 1, а общее число наборов таких подсистем, содержащих не менее одной подсистемы, равно $2^{2^n-1} - 1$. С ростом n это число растет очень быстро. Однако без потери общности его можно существенно уменьшить, если рассматривать только неизбыточные наборы подсистем ИИ (см. разд. 5.3).

Для многих интеллектуальных исследований очень перспективным является другой способ сокращения числа реконструктивных гипотез. Он состоит в исключении наборов подсистем ИИ, не содержащих всех

переменных обобщенной системы ИИ. Это требование, будем называть *условием покрытия* и, формально оно выглядит так:

$$\bigcup_k {}^kS = S,$$

где ${}^hS$ — множество переменных из подсистем реконструктивной гипотезы, а S — множество переменных обобщенной системы ИИ. Это условие объясняется прежде всего необходимостью использовать в реконструктивной гипотезе информацию обо всех переменных обобщенной системы ИИ для того, чтобы реконструкция была логически возможна. Поскольку вопрос о включении или исключении выборочных переменных из обобщенной системы ИИ решается в результате анализа маски (см. разд 4.6), выполнение условия покрытия общности не нарушает.

Далее под *реконструктивной гипотезой* будут пониматься только такие наборы подсистем заданной обобщенной системы ИИ, которые удовлетворяют и требованию неизбыточности, и условию покрытия. Таким образом, **реконструктивная гипотеза — это структурированная система ИИ с поведением, сравнимая с обобщенной системой ИИ с поведением.** Однако иногда бывает нужно работать со всеми наборами подсистем ИИ, которые удовлетворяют только требованию неизбыточности. Будем такие наборы подсистем называть *обобщенными реконструктивными гипотезами.* Понятно, что для данной обобщенной системы ИИ с поведением множество ее реконструктивных гипотез является подмножеством множества ее обобщенных реконструктивных гипотез.

Любая реконструктивная гипотеза (равно как и любая обобщенная реконструктивная гипотеза) полностью описывается:

1) семейством подмножеств входящих в нее переменных,
2) функциями поведения, соответствующими отдельным подмножествам переменных.

Если опустить свойство 2, то свойство 1 определяет класс инвариантности реконструктивных гипотез, отличающихся друг от друга только функциями поведения их элементов. **Этот класс инвариантности для того, чтобы отличать его от отдельных реконструктивных гипотез класса, будем называть *структурой.*** Напомним, что **каждая отдельная реконструктивная гипотеза представляет собой конкретную структурированную систему. Таким образом, структура — это свойство структурированной системы, инвариантное относительно изменения функций поведения ее элементов.**

Для данного множества переменных, скажем множества S, множество структур, представляющих все реконструктивные гипотезы любой

обобщенной системы ИИ, определенной на S, состоит из семейств подмножеств S, удовлетворяющих условиям неизбыточности и покрытия. Будем для удобства представлять все множества переменных одной мощности, скажем мощности *n,* общим множеством структур, скажем множеством G_n, определенным на множестве N_n положительных целых чисел. Формально для любого $n \in \mathbf{N}$

$G_n = \{G_i \mid G_i \subset \mathcal{P}(N_n),\ G_i$ удовлетворяет условиям неизбыточности и покрытия$\}$.

В этом формальном определении через G_i обозначены элементы G_n, являющиеся наиболее общими структурами, рассматриваемыми при решении задачи реконструкции (некие специальные типы этих структур будут введены ниже); индекс *i* идентифицирует структуры из G_n и обычно $i \in N_{|G_n|}$. Множество G_n тривиально интерпретируется на языке любого множества переменных *S,* такого, что $|S|=n$, заданием взаимно однозначного отображения переменных из S на целые из N_n. Будем для удобства структуры из множеств G_n называть G-структурами.

Из некоторых соображений удобно расширить множество G_n до множества G^+_n всех обобщенных реконструктивных гипотез. Формально для любого $n \in \mathbf{N}$

$G^+_n = \{G_i \mid G_i \subset \mathcal{P}(N_n),$ G, удовлетворяет условию неизбыточности$\}$.

Все результаты относительно G_n могут быть легко обобщены и на множества G^+_n.

Если множество G_n для некоторого определенного *n* получает конкретную интерпретацию в контексте некой обобщенной системы ИИ с поведением с *n* переменными, то структуры в G_n представляют собой однозначные представления реконструктивных гипотез, связанных с этой обобщенной системой ИИ. Это непосредственно следует из того факта, что функции поведения, соответствующие любым подмножествам переменных, определяются однозначно как соответствующие проекции обобщенной функции поведения. Следовательно, **реконструктивные гипотезы могут изучаться в виде абстрактных структур.** Данная структура из G_n становится конкретной реконструктивной гипотезой, когда интерпретируется в контексте сравнимой с ней определенной обобщенной системы ИИ с поведением (т. е. системы ИИ с *n* переменными).

Основным вопросом задачи реконструкции является разработка эффективных вычислительных процедур, допускающих представление, оценку и сравнение реконструктивных гипотез, представленных всеми структурами для данного множества переменных. Это очень непростая задача, поскольку, как будет показано ниже в этом разделе, число структур растет очень быстро. **Для успешного решения этой задачи необходимо использовать соответствующее упорядочение и классификацию структур.**

Определим сначала естественное упорядочение структур, называемое *уточняющим упорядочением.* Пусть даны две структуры $G_i,\ G_j \in \mathcal{G}_n$ Будем называть G_i *уточнением* G_j (и, соответственно, G_j укрупнением G_i) тогда и только тогда, когда для любого $x \in G_i$ существует $y \in G_j$, такое, что $x \subseteq y$; пусть $G_i \le G_j$ означает, что G_i это уточнение G_j.

Рассмотрим две структуры $G_i,\ G_j \in \mathcal{G}_n$, такие, что $G_i \le G_j$. Тогда G_i называется *непосредственным уточнением* G_j (а G_j — *непосредственным укрупнением* G_i) тогда и только тогда, когда не существует $G_k \in \mathcal{G}_n$, такого что $G_i \le \mathcal{G}_k$ и $G_k \le G_j$. Для заданной структуры $G_i \in G_n$ *структурное соседство* определяется как множество всех непосредственных уточнений и непосредственных укрупнений G_i в $\mathcal{G}_n$.

Легко видеть, что отношение уточнения определяет частичное упорядочение. Более того, пара $(\mathcal{G}_n, \le)$ определяет решетку, что подтверждают следующие факты: 1) существует универсальная верхняя граница — множество $\{N_n\}$; 2) существует универсальная нижняя граница — множество $\{\{x\} \mid x \in N_n\}$; 3) для любой пары G_i, $G_j \in \mathcal{G}_n$ наибольшим общим уточнением является неизбыточныйэквивалент множества $\{x \cap y \mid x \in G_i,\ y \in G_j\}$; 4) для любой пары G_i, $G_j \in G_n$ наименьшим общим укрупнением является неизбыточный эквивалент множества $G_i \cap G_j$. Будем называть эти решетки *решетками уточнения* G-структур). (по одной для каждого $n \in \mathcal{N}$).

Отметим, что уточняющее упорядочение применимо и к множествам G^+_n, что дает решетки $(\mathcal{G}^+_n, \le)$.

Понятно, что решетка уточнения или некая нужная ее часть может быть получена с помощью неоднократного выполнения процедуры, порождающей все непосредственные уточнения для любой структуры из решетки. Одна такая процедура определяется следующим образом.

Уточняющая процедиоа для G-структур* (или *RG-процедура).

Заданы G-структуры $G_i = \{{}^kS \mid k \in N_q\} \in \mathcal{G}_n$. Для определения всех их непосредственных уточнений

1) положить $k=0$;
2) если $k<q$, то $k+1 \to k$, иначе перейти на шаг 5;
3) если $|{}^hS| \ge 2$, то $(G_i - \{{}^hS\}) \bigcup X \to R$, где $X = \{x \mid x \subset {}^kS,\ |x| = |{}^kS| - 1\}$; иначе перейти на шаг 2;

4) $R \to Q$, где Q — неизбыточный аналог R, записать Q в качестве непосредственного уточнения G_i, перейти на 2;
5) конец.

Обратите внимание на то, что условие $|^kS| \geq 2$ из шага 3 обеспечивает то, что порождаемые структуры будут удовлетворять условию покрытия; замена этого условия на условие $|^kS| \geq 1$ позволяет работать с множествами $\mathcal{G}^+_n$ обобщенных реконструктивных гипотез. Шаг 4 обеспечивает выполнение условия неизбыточности. Тот факт, что наименьшее возможное изменение делается на шаге 3, т. е. только один элемент G_i изменяется на непосредственно следующие меньшие элементы (подмножества), обспечивает то, что порождаемые структуры являются непосредственными уточнениями G,.

Пример 17. Дано $G_i = \{^1S=\{1, 2, 3\}, {}^2S=\{2, 3, 4\}, {}^3S=\{1, 4\}\}$. Сразу видно, что для всех $k \supseteq N_3 \mid {}^kS \mid \geqslant 2$, и, следовательно, RG-процедуры применимы к любому элементу. Таким образом, имеются три непосредственных уточнения G_i. Множество ^1S заменяется на множества {1, 2}, {1, 3}, {2, 3}, однако третье множество является подмножеством и будет исключено на шаге 4; это даст следующее непосредственное уточнение:

{{1, 2}, {1, 3}, {2, 3, 4}, {1, 4}}.

Аналогичная замена ^2S дает второе непосредственное уточнение:

$$\{\{1, 2, 3\}, \{2, 4\}, \{3, 4\}, \{1, 4\}\}.$$

Наконец, множество ^3S заменяется на множества {1} и {4}, которые являются избыточными и будут исключены на шаге 4; отсюда имеем третье непосредственное уточнение:

$$\{\{1, 2, 3\}, \{2, 3, 4\}\}.$$

Для сокращения вычислительной сложности порождения реконструктивных гипотез полезно разбить решетки уточнения на подходящие классы эквивалентности по уровню вычислений. Тогда эти классы эквивалентности могут быть представлены соответствующими каноническими структурами, а уточняющие процедуры разработаны для этих канонических представлений разных уровней вычислительной сложности. **Чтобы продемонстрировать этот подход, предположим, что описаны только два уровня вычислений, называемые локальным и глобальным.**

Локальный уровень вычислений представлен уже описанной RG-процедурой. Для определения *глобального уровня* вычислений определим функции

$$r_n : G_n \to \mathcal{R}_n, \quad n \in \mathbf{N},$$

где $\mathcal{R}$ — множество симметричных и рефлексивных бинарных отношений, определенных на множестве N_n (они также называются

отношениями сравнения, отношениями толерантности или неориентированными графами с циклами), а $r_n(G_i)$—бинарное отношение, выполняемое для целых a и b ($a, b \in N_n$) тогда и только тогда, когда и a, и b принадлежат по крайней мере одному из подмножеств N_n, входящих в G_i. Формально

$$r_n(G_i) = \{(a, b) \mid (\exists x \in G_i)\ (a \in x \text{ и } b \in x)\}.$$

Будем элементы $\mathcal{R}_n$ называть *графами*. Однако мы должны помнить, что эти графы не ориентированы (симметричность) и содержат циклы (рефлексивность). Некоторые примеры функций классов r_4 и r_5 показаны на рис. 15. При изображении этих графов опущены тривиальные циклы в узлах.

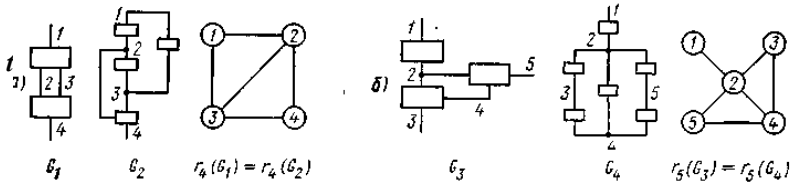


Рис. 15. Примеры функции r_n

Понятно, что функции r_n сюръективны, а при $n \geq 3$ прообразы могут состоять из более чем одного элемента. Поэтому они индуцируют следующее отношение эквивалентности на соответствующих множествах G_n G-структур: $G_i \overset{r}{=} G_j$ тогда и только тогда, когда $G_i, G_j \in G_n$ и $r_n(G_i) = r_n(G_j)$ для некоторого $n \in \mathbf{N}$. Если $G_i = G_j$, то мы говорим, что структуры G_i и G_j r-эквивалентны. Для обозначения классов эквивалентности, индуцируемых r_n на $\mathcal{G}_n$, будем использовать стандартную запись $\mathcal{G}_n/r_n$.

Для любого $n \in N_n$ множество $\mathcal{R}_n$ и отношение подмножества (или, иначе, операции объединения и пересечения множеств) определяют булеву решетку. Очевидное взаимнооднозначное соответствие между $\mathcal{G}_n/r_n$ и $\mathcal{R}_n$ индуцирует изоморфную булеву решетку на множестве G_n/r_n. Этот изоморфизм позволяет нам порождать классы эквивалентности на $\mathcal{G}_n/r_n$ с помощью соответствующих операций на графах из $\mathcal{R}_n$. При этом, однако, желательно, чтобы любой класс эквивалентности из $\mathcal{G}_n/r_n$ был представлен некоей канонической структурой. С этой целью введем для любого $n \in \mathbf{N}$ следующие подмножества $\mathcal{G}_n$:

$\mathcal{C}_n$, в которое входят те G-структуры G_i из множества $\mathcal{G}_n$, которые состоят из максимально сочетаемых классов, соответствующих графу

$r_n(G_i)$ или, но в другой терминологии, основанных на кликах этого графа. (Клика графа — его максимальный полный подграф.) Подобные структуры называются также полными покрытиями $r_n(G_i)$. Будем обозначать структуры C_i из $\mathscr{C}_n$ и называть *С-структурами.*

Примерами С-структур являются структуры G_1 и G_2, изображенные на рис. 15.

$\mathscr{P}_n$, в которое входят те G-структуры из $\mathscr{G}_n$, элементы которых состоят из пар, связанных на графе $r_n(G_i)$ узлов (обозначаемых целыми числами), или являются отдельными изолированными узлами. Будем структуры из множества $\mathscr{P}_n$ называть Р-структурами и обозначать через P_k. Примером Р-структуры является структура G_4 на рис. 15 Из этих определений и из того факта, что множество всех максимально сочетаемых классов для любого неориентированного графа единственно, следует, что любой класс эквивалентности из $\mathscr{G}_n/r_n$ содержит в точности одну С-структуру из $\mathscr{C}_n$ и одну Р-структуру из $\mathscr{P}_n$. Таким образом, *С-* и Р-структуры могут рассматриваться как два *канонических представления* классов *r*-эквивалентности G-структур. Каждый класс *r*-эквивалентности, представленный графом, описывается двумя каноническими структурами— наименее уточненной С-структурой и наиболее уточненной Р-структурой. Взаимно однозначное соответствие между $\mathscr{R}_n$ и $\mathscr{C}_n$ (или $\mathscr{P}_n$) индуцирует булеву решетку на $\mathscr{C}_n$ (или на $\mathscr{P}_n$), изоморфную естественной булевой решетке, определенной на $\mathscr{P}_n$. Например, на рис. 16 показаны решетка $(\mathscr{G}_3 \leqslant)$ и взаимно изоморфные булевы решетки, определенные на $\mathscr{R}_3$, $\mathscr{C}_3$, $\mathscr{P}_3$ и $\mathscr{G}_3/r_3$.

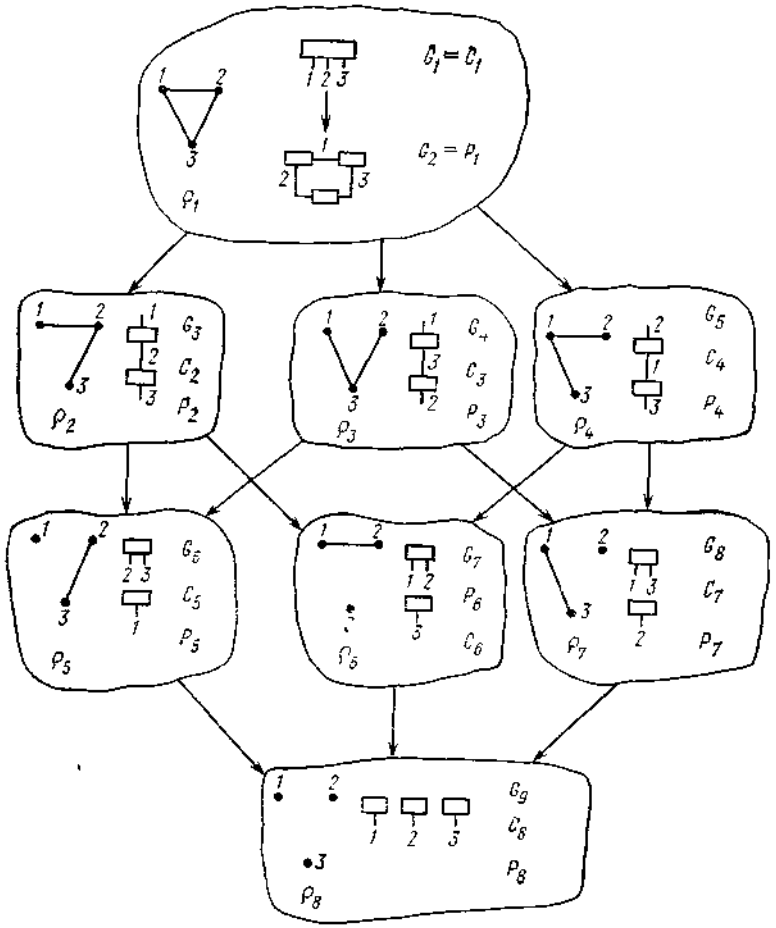


Рис. 16. Решетка $(\mathscr{G}_3, \leqslant)$ и булевы решетки, определенные на $\mathscr{R}_3$, $\mathscr{C}_3$, $\mathscr{P}_3$ и $\mathscr{G}_3/r_3$

Для описания больших решеток полезно определить отношение эквивалентности $\overset{i}{=}$ на множествах $\mathscr{G}_n$ следующим образом:

$G_i \overset{i}{=} G_j$ тогда и только тогда, когда G_i, $G_j \in \mathscr{G}_n$ и G_i, G_j изоморфны (т. е. когда одна структура может быть получена из другой только перестановкой целых чисел из N_n, напоминаем, что каждое целое число соответствует узлу). Будем $\overset{i}{=}$ называть

i-эквивалентностью и обозначать через $\mathcal{G}_n/i$ множество классов эквивалентности изоморфных структур (или перестановочных классов эквивалентности), определенных на $\mathcal{G}_n$.

Пример, демонстрирующий смысл *i*-эквивалентности, приведен на рис. 17, где обозначенные полужирным шрифтом $G_k (k \in N_5)$ показывают классы эквивалентности в $\mathcal{G}_3/i$.

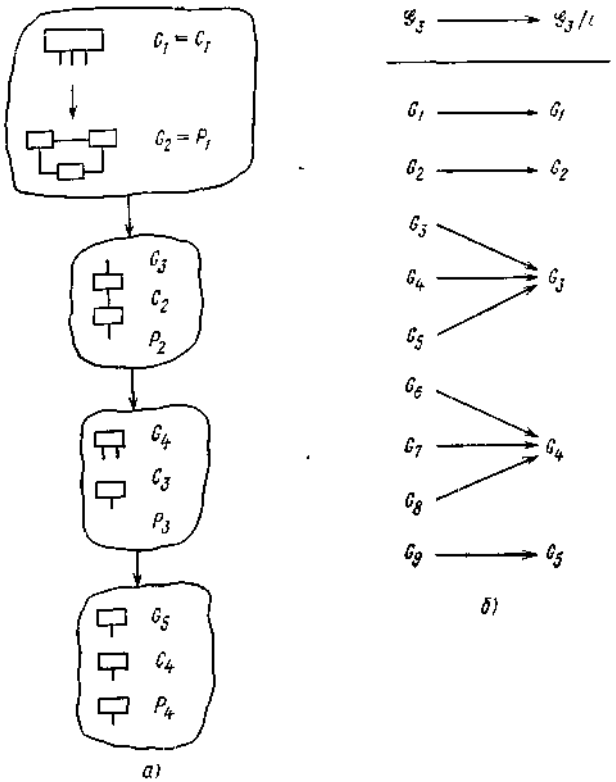


Рис. 17. Решетка $(\mathcal{G}_3/i, \leqslant)$ и гомоморфное отображение $\mathcal{G}_3 \to \mathcal{G}_3/i$

На рис. 17,а показана решетка $(\mathcal{G}_3/i, \leqslant)$. Это та же самая решетка, что и на рис. 16, но упрощенная, так как на ней не различаются изоморфные структуры. Сделано это за счет удаления меток входов в блоки на схеме и включения только одной схемы для каждого перестановочно эквивалентного класса. Структуры в каждом классе легко определяются перестановкой целых 1, 2, 3 для входов в блоки. Упрощенная решетка на рис. 17,в является гомоморфным образом

решетки, изображенной на рис. 16. Гомоморфное отображение, являющееся основой данного упрощения, показано на рис. 4.17,б.

На рис. 18 показана более сложная решетка $(\mathcal{G}_4/i, \leqslant)$.

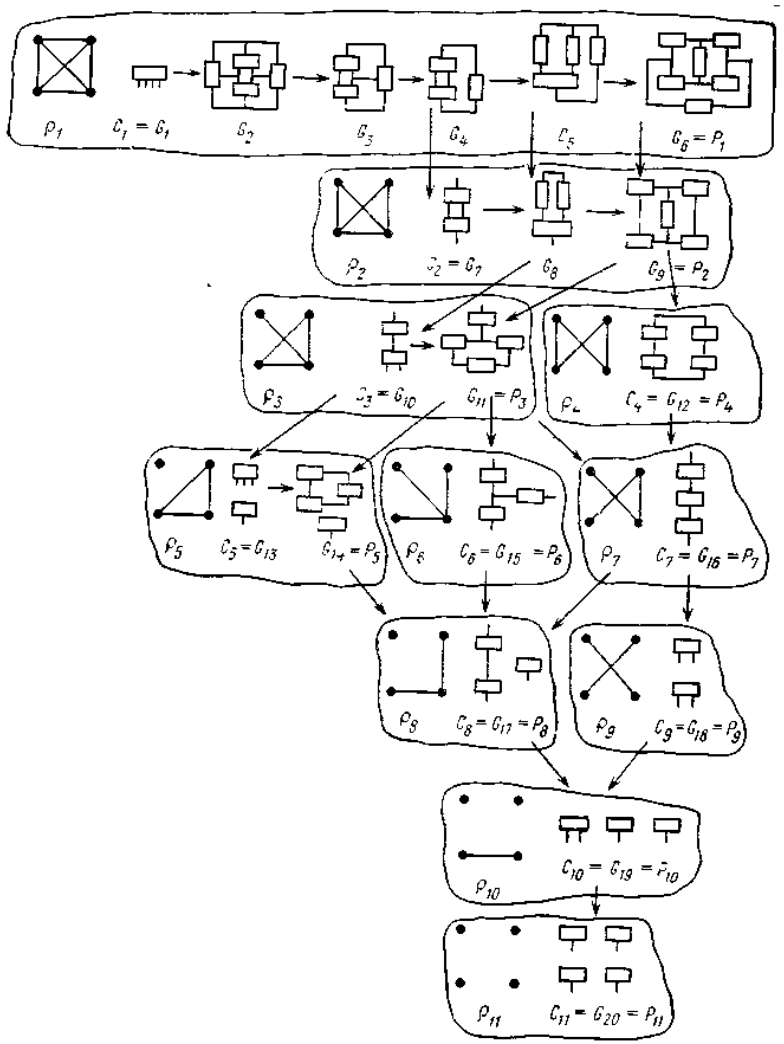


Рис. 18. Решетка $(\mathcal{G}_4/i, \leqslant)$ с указанием классов *i*-эквивалентности и канонических *G*- и P-структур

Перестановочные классы эквивалентности G-структур снова выделены жирным шрифтом, равно как С-структуры и Р-структуры, сгруппированные по классам *r*-эквивалентности. Каждому такому классу соответствует граф ρ_k и две канонические структуры C_k и P_k ($k \in N_{11}$). Для того чтобы более ярко продемонстрировать общие свойства этой решетки, на рис. 19 перестановочные классы С-структур обозначены только их идентификаторами и выделены классы *r*-эквивалентности.

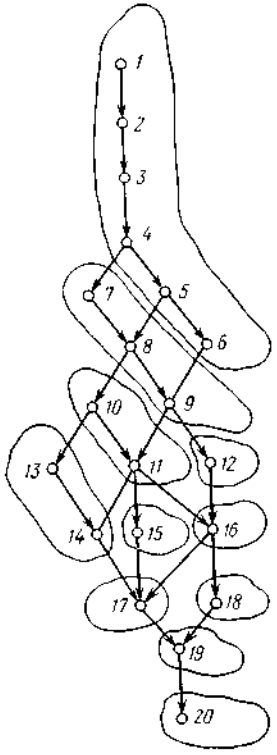


Рис. 19. Упрощенная схема решетки $(\mathcal{G}_4/i, \leqslant)$, полностью показанной на рис. 18

На рис. 20 изображена решетка $(\mathcal{G}_4/i, \leqslant)$, являющаяся подрешеткой $(\mathcal{C}_4/i, \leqslant)$.

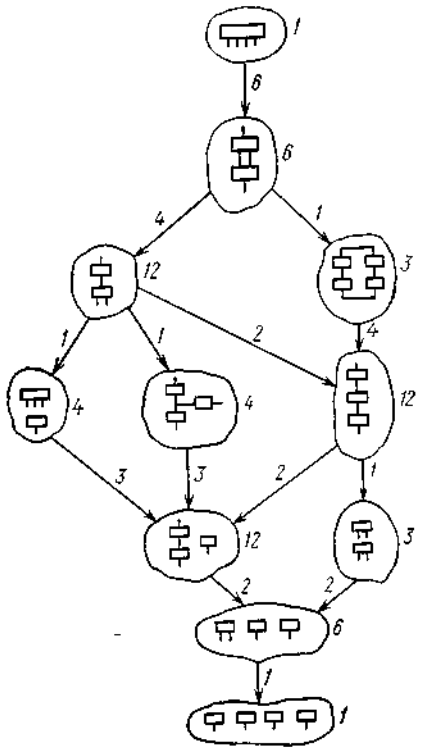


Рис. 20. Решетка $(\mathcal{C}_4/i, \leqslant)$

Число около каждой схемы указывает на число различных С-структур, входящих в перестановочный класс эквивалентности, показанный на этой схеме; числа, помещенные около дуг, указывают на число непосредственных уточнений С-структуры из данного перестановочного класса эквивалентности в другой класс. Как уже объяснялось выше, эта решетка изоморфна решеткам, определенным на $\mathcal{R}_4/i$, $\mathcal{P}_4/i$ и $(\mathcal{G}_4/i)/r$.

В то время как полные решетки $(\mathcal{G}_n, \leqslant)$ представляют основу для локального уровня вычислений в задаче реконструкции, решетки $(\mathcal{G}_n, \leqslant)$ и их изоморфизмы являются основой для глобальных вычислений. Для работы на глобальном уровне вычислений необходима процедура, порождающая для любой заданной С-структуры $C_k \in \mathcal{C}_n$ ($n \in \mathbf{N}$) все непосредственные уточнения в решетке $(\mathcal{G}_n, \leqslant)$. Ниже приводится одна такая процедура, использующая представление С-структур в виде графов.

Уточняющая процедура для С-структур (или *RC-процедура)* Дана С-структура $C_k \in \mathscr{C}_n$ и соответствующий граф $r_n(C_k)$ Нужно определить все непосредственные уточнения на множестве

1) исключить одно ребро из графа $r_n(C_k)$, скажем ребро (a, b);
2) разделить каждый элемент x из C_k, который содержит a и b на два элемента $x_a = x - \{b\}$ и $x_b = x - \{a\}$ и заменить, x из C_k на x_a и x_b ;
3) исключить все избыточные x_a и x_b, полученные на шаге 2 и записать полученный результат в качестве непосредственного уточнения C_k в решетке $(\mathscr{C}_n, \leqslant)$;
4) выполнить шаги 1—3 для всех ребер графа $r_n(C_k)$ и остановиться.

Данная процедура имеет следующие обоснования- 1) имеется взаимно однозначное соответствие между множествами $\mathscr{R}_n$ и $\mathscr{C}_n$ и, следовательно, любое изменение графа приводит к изменению , соответствующей С-структуры; 2) чем меньше число ребер графа, тем более уточненной является соответствующая С-структура;

3) поскольку никакой из циклов в вершинах нельзя исключить без нарушения условия покрытия соответствующей С-структуры, то наименьшим допустимым сокращением графа является исключение одного из ребер. Таким образом, число ребер в графе определяет число непосредственных уточнений соответствующей С-структуры.

Пример 18. Рассмотрим граф $_{\text{Pl}}$ и соответствующую С-структуру C_1 (рис. 21,а).

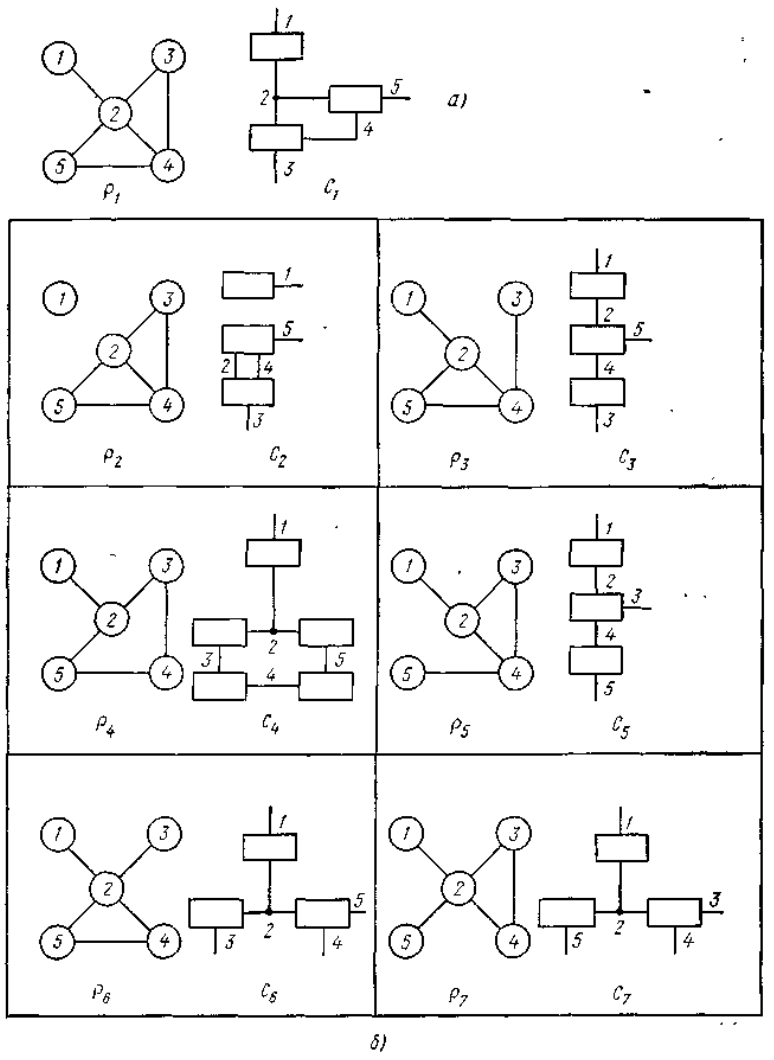


Рис. 21. Пояснение к RC-процедуре: *а* — заданный граф и соответствующая С-структура; *б*—непосредственные уточнения C_1

Этот граф имеет шесть ребер и, следовательно, шесть непосредственных уточнений этой С-структуры. Они изображены на рис. 21,б. Например, уточнение получается с помощью RC-

процедуры следующим образом: 1) из графа ρ_1 исключается ребро (4,5) и получается граф ρ_7; 2) элемент {2, 4, 5*}* из C_1 (единственный элемент C_1 содержащий и 4 и 5) разбивается на элементы {2, 5} и {2, 4}; 3) поскольку элемент (2, 4} является единственным избыточным элементом ({2, 4} $\subset$ (2 ,3 ,*4)* он исключается, а полученный результат C_7 = {{1, 2} {2, 5} , {2*, 3, 4}}* записывается как непосредственное уточнение C_1.

Так как элементы Р-структур представляют собой просто ребра соответствующих графов, то процедура уточнения Р-структур (или RP-процедура) совершенно тривиальна. Она состоит в исключении отдельных ребер из заданного графа [см. шаг 1 RC-процедуры] и интерпретации результатов как Р-структур

Полезны также процедуры, с помощью которых получаются все непосредственные укрупнения для множеств G-, С- и Р-структур. Они нужны для определения полного структурного соседства заданной структуры. Формулирование этих процедур мы предоставляем читателю в качестве упражнения (см. также рис. 10). Примеры такого соседства для трех этих типов структур приведены на рис. 22—24.

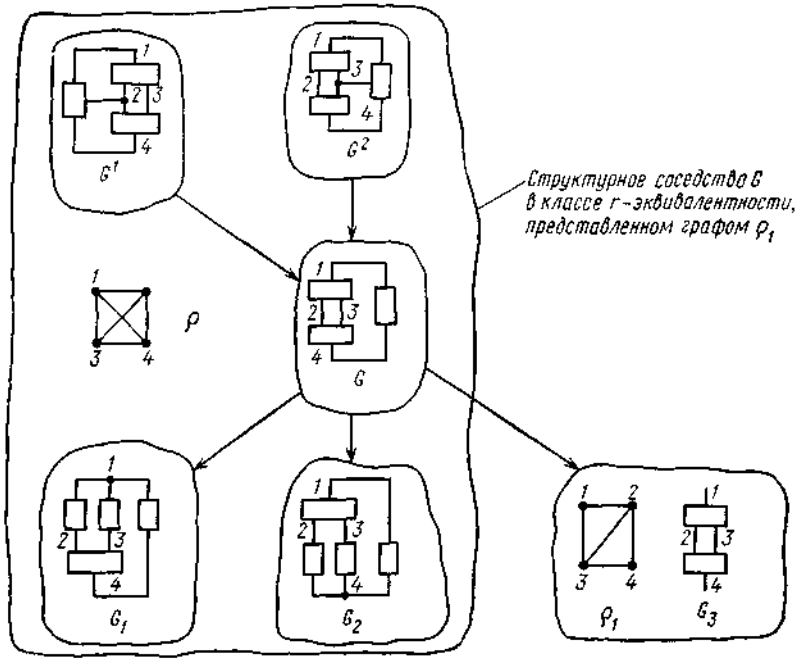


Рис. 22. Структурное соседство G-структуры *G*

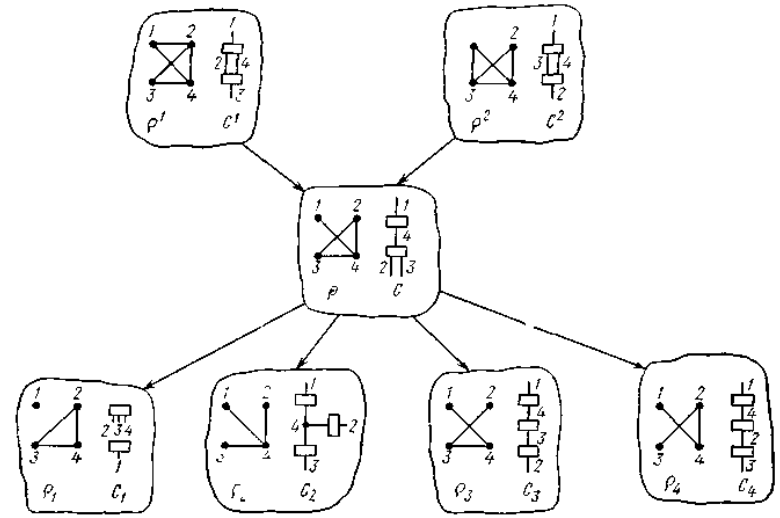


Рис. 23. Структурное соседство С-структуры С

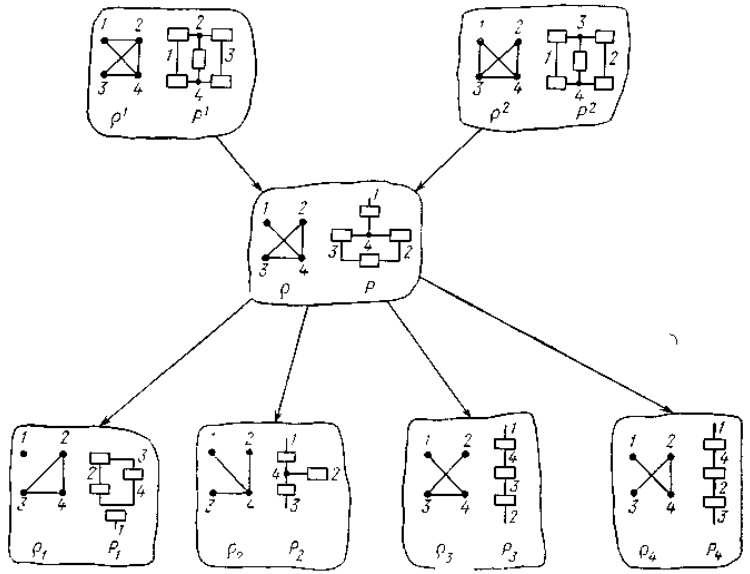


Рис. 24. Структурное соседство Р-структуры *P*

В этих примерах структуры обозначены соответственно как *G, C* и *P*. Их непосредственные уточнения помечены нижними индексами, а непосредственные укрупне-ния — верхними. На рис. 22 показано, что

в структурное соседство данной G-структуры могут входить G-структуры, входящие в другой класс *r*-эквивалентности (на рис. 22 это структура G_3). Чтобы непосредственные уточнения принадлежали тому же классу *r*-эквивалентности, можно слегка модифицировать *RG-иро*-цедуру, заменив условие $|^kS|\geq 2$ на шаге 3 условием $|^kS|>2$. При этом запрещается изменять элементы, содержащие только две вершины, и, следовательно, граф данной *r*-структуры остается неизменным.

Представление о непосредственных уточнениях (или укрупнениях) структур различных типов можно использовать для разбиения соответствующего множества структур на блоки структур, эквивалентных в смысле уровня уточнения, т. е. таких структур, которые достигаются от универсальной верхней границы $\{N_n\}$ соответствующей решетки уточнения за одинаковое число шагов уточнения. Будем называть эту эквивалентность *эквивалентностью уровня уточнения* и обозначать $\overset{i}{\equiv}$. Так, например, структуры G_1, G_2, G_3 на рис. 22 — *l*-эквивалентные G-структуры из множества $\mathscr{G}_4$; структуры, показанные на рис. *21,б* ,— *l*-эквивалентные С-структуры из множества $\mathscr{C}_5$; структуры P_1, P_2, P_3, P_4 на рис. 24 являются *l*-эквивалентными Р-структурами из множества $\mathscr{P}_4$.

Для того чтобы можно было составить представление о скорости роста числа структур этих трех типов с ростом *n,* а также числа их классов *i*- и *l*-эквивалентности, в табл. 12 приведены соответствующие данные для $n\leq 7$.

Таблица 12

Число G-структур в $\mathscr{G}_n{}^+$ и $\mathscr{G}_n$,С-структур и их классов *l*-эквивалентности и классов *i*-эквивалентности для $n\leq 7$

n	1	2	3	4	5	6	7
$\lvert\mathscr{G}_n^+\rvert$	1	4	18	166	7,579	7,828,352	2,414,682,040,996
$\lvert\mathscr{G}_n\rvert$	1	2	9	114	6,894	7,785,062	2,414,627,396,434
$\lvert\mathscr{C}_n\rvert$	1	2	8	64	1,024	32,768	2,097,152
$\lvert\mathscr{G}_n^+/i\rvert$	1	3	8	28	208		
$\lvert\mathscr{G}_n/i\rvert$	1	2	5	20	180		
$\lvert\mathscr{C}_n/i\rvert$	1	2	4	11	34	156	1,044
$\lvert\mathscr{G}_n^+/l\rvert$	1	3	7	15	31	63	127
$\lvert\mathscr{G}_n/l\rvert$	1	2	5	12	27	58	121
$\lvert\mathscr{C}_n/l\rvert$	1	2	4	7	11	16	22

Понятно, что

$$|\mathscr{C}_n| = |\mathscr{P}_n| = |\mathscr{R}_n|, \quad |\mathscr{R}_n| = 2^{n(n-1)/2},$$

где показатель $n(n-1)/2$ — общее число возможных ребер в графе, определенном на N_n. Ясно также, что $|\mathscr{G}_n/1| = n(n-1)/2+1$. Вычисление $|\mathscr{R}_n/i|$ для заданного более сложно, но эта задача уже решена в теории графов (см 11). Приведенные в табл. 12 значения $|\mathscr{G}^+{}_n|$, $|\mathscr{G}^+{}_n/i|$, $|\mathscr{G}^+{}_n/l|$, $|\mathscr{G}_n|$ $|\mathscr{G}_n/i|$ и $|\mathscr{G}_n/l|$ известны только для $n\leq 7$.

Введем упоминавшиеся уже структуры еще одного типа. Это такие структуры, для которых при работе с вероятностными системами для определения несмещенной реконструкции не нужна итеративная процедура соединения. Обычно их называют *ациклическими структурами.*

Для решения задачи реконструкции наибольший интерес представляют ациклические структуры особого типа, которые можно было бы назвать *строго ациклическими структурами* или L-структурами. G-структура $G_i \in G_n$ является L-структурой тогда и только тогда, когда не существует такой пары $(a, b) \in N^2_n$, что она входит в некий элемент G, и связана через несколько соединенных элементов. Будем множество L-структур для некоего *n* обозначать $\mathscr{L}_n$.

Определим множество $\mathscr{L}_n$ формально. Пусть дана G-структура $G_i \in \mathscr{G}_n$. Для любой пары $(a, b) \in N^2_n$ пусть

$$X_{a,b} = \{x \mid x \in G_i,\ \{a, b\} \subset x\}.$$

Тогда G_i является L-структурой $(G_i \in \mathscr{L}_n)$ тогда и только тогда, когда не существует пары $(a, b) \in N^2_n$, являющейся элементом транзитивного замыкания $r_n(G_i - X_{a,b})$.

Очевидно, например, что все структуры из множества $\mathscr{G}_3$ (рис. 16), за исключением структуры $\mathscr{G}_2 = \{\{1, 2\}, \{2, 3\}, \{3, 1\}\}$, являются L-структурами. В множестве $\mathscr{G}_4$ только три структуры не являются L-структурами. Они принадлежат к одному и тому же классу *i*-эквивалентности, который можно, например, представить С-структурой $C=\{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 1\}\}$. В самом деле, транзитивное замыкание $r_4(C - X_{1,2})$ —это N_4, и, следовательно, пара (1, 2) является ее элементом.

L-структуры отличаются тем, что для них не нужно использовать итеративную процедуру соединения независимо от порядка, в котором объединяются элементы рассматриваемой L-структуры. Ациклические структуры, не являющиеся L-структурами, этим свойством не

обладают. Для таких структур, чтобы избежать применения итеративной процедуры соединения, нужно применять базовую процедуру соединения к элементам, расположенным в определенном порядке. Определение таких порядков требует сложных вычислений и проверок, так что методологическое значение таких структур существенно уступает значению L-структур.

С помощью различных понятий, введенных в этом разделе, можно более конкретно сформулировать задачу реконструкции. Дана обобщенная система и множество заданных пользователем реконструктивных гипотез (базирующихся на множестве $\mathcal{G}_n$, $\mathcal{C}_n$, $\mathcal{P}_n$ или $\mathcal{L}_n$-структур). Решение задачи реконструкции сводится к выбору сподмножества данного множества в соответствии с некоторыми требованиями. Обычно требуется, чтобы: 1) расстояния, соответствующие выбранным гипотезам, были как можно меньше и 2) сами гипотезы были, возможно, более уточненными. Оба эти требования предполагают упорядочение множества реконструктивных гипотез. Упорядочение в соответствии с требованием 2 фиксировано — это частичное упорядочение по структурному уточнению с решеткой, свойства которой описаны выше. Упорядочение согласно требованию 1 — его можно назвать *упорядочением по расстоянию* — не фиксировано. Оно зависит от данной обобщенной системы и выбранного типа расстояния и определяется только вычислением несмещенных реконструкций и расстояний отдельных реконструктивных гипотез.

Если используется расстояние, определяемое по формуле (40) для вероятностных систем [или по формуле (42) для возможностных систем], которое является мерой количества информации, потерянной при замене обобщенной системы реконструктивной гипотезой, то существует определенное *предупорядочение по расстоянию:* информационное расстояние монотонно не убывает с увеличением уточнения реконструктивных гипотез. (В данном случае под предупорядочением понимается не научный термин, обозначающий рефлексивное и транзитивное отношение, а некое частичное упорядочение.)

Кроме того, оба варианта информационного расстояния *аддитивны* для любого пути на используемой решетке уточнения. Это значит, что

$$D(f^x, f^z) = D(f^x, f^y) + D(f^y, f^z) \quad (43)$$

для любых трех реконструктивных гипотез *x, y, z* одной обобщенной системы, таких, что $x \geq y \geq z$. **Свойства предупорядоченности и аддитивности очень полезны при решении задачи реконструкции и придают особую важность информационному расстоянию**. В

дальнейшем при обсуждении задачи реконструкции всегда будет предполагаться, что используется соответствующая версия информационного расстояния (т. е. вероятностная или возможностная, базовая или порождающая).

Сочетание упорядочения по расстоянию с упорядочением по уточнению образует объединенное упорядочение для задачи реконструкции. Теперь множество решений задачи реконструкции характеризуется с помощью этого комбинированного упорядочения следующим образом: это множество представляет собой такое подмножество реконструктивных гипотез, в которое не входят гипотезы, худшие, чем любая другая гипотеза. Слово «худшие» используется здесь в обычном смысле: гипотеза h_1 хуже гипотезы h_2 тогда и только тогда, когда или h_1 является менее уточненной и ее расстояние не меньше, чем у h_2 или у h_1, большее расстояние, чем у h_2, и в то же время она не является более уточненной, чем h_2. Элементы этого множества решений будем называть *подходящими реконструктивными гипотезами.*

Теперь видно совершенное сходство задачи реконструкции с двумя рассмотренными выше задачами — задачей определения подходящих систем с поведением (разд. 4.4 и 4.6) и задачей определения подходящих упрощений заданной системы с поведением (разд. 4.9). Если упорядочение по нечеткости и упорядочение по сложности из этих задач сопоставить соответственно с упорядочением по расстоянию и упорядочением по уточнению из задачи реконструкции, то сходство этих трех задач станет очевидным. Предоставляем читателю использовать это сходство для формального определения комбинированного упорядочения и множества решений для задачи реконструкции.

Мы видим, что задачи, связанные с подъемом по эпистемологической иерархии систем ИИ, а также с упрощением систем ИИ, образуют важнейшую категорию задач, имеющих следующие общие черты:

ДАНО:

множество X рассматриваемых систем ИИ;

множество отношений порядка $\overset{a}{\leqslant}$, $\overset{b}{\leqslant}$, $\overset{c}{\leqslant}$, ... на X.

МНОЖЕСТВО РЕШЕНИЙ:

$$X_s = \{x \in X \mid (\forall y \in X)\ (y \overset{*}{\leqslant} x \Rightarrow x \overset{*}{\leqslant} y)\},$$

где $\overset{*}{\leqslant}$ —объединенный порядок предпочтения на X, определенный для всех $x, y \in X$ следующим образом:

$x \overset{*}{\leqslant} y$ тогда и только тогда, когда $x \overset{a}{\leqslant} y$,

$$\text{и } x \overset{b}{\leqslant} y, \quad \text{и } x \overset{c}{\leqslant} y, \quad \text{и } \ldots$$

В процессе решения задачи реконструкции необходимы три набора процедур:

1) процедуры порождения всех нужных реконструктивных гипотез;

2) процедуры оценки и сравнения порожденных реконструктивных гипотез с точки зрения целей задачи реконструкции;

3) процедуры, которые на соответствующих этапах процесса решения задачи принимают решение о том, какие из порожденных реконструктивных гипотез должны быть включены в множество решений, какие использованы для дальнейшего порождения реконструктивных гипотез и должен ли процесс решения продолжаться или закончиться.

На рис. 25 показано, каким образом эти три набора процедур объединяются в единый процесс решения.

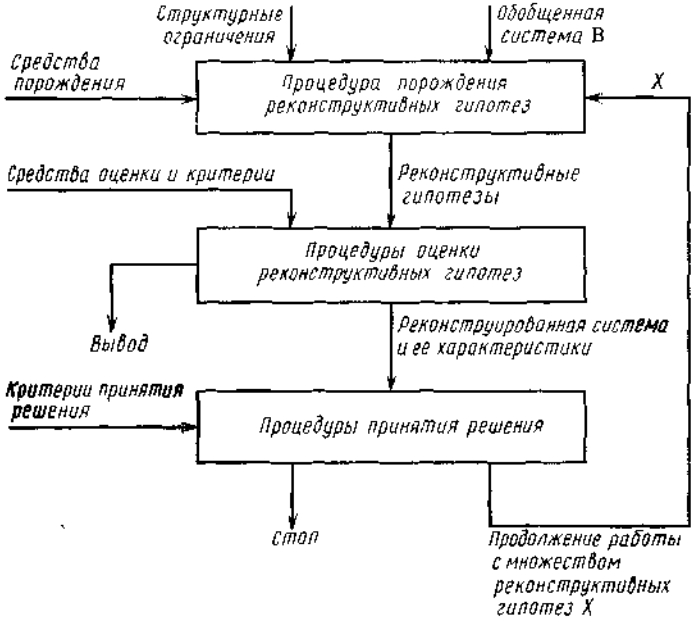


Рис. 25. Общая схема процесса решения задачи реконструкции

Ядром процесса является порождение всех подходящих реконструктивных гипотез. Удобно это делать, порождая

соответствующие структуры, которые затем интерпретируются как реконструктивные гипотезы для заданной обобщенной системы. Интерпретация эта сводится к установлению соответствия между переменными обобщенной системы и целыми числами, которыми идентифицированы узлы структур, а также к вычислению, если нужно, проекций обобщенной функции поведения. Порожденные структуры можно разными способами сократить, что уменьшает время вычислений и их цену, а также, возможно, и по другим соображениям. Например, можно выбрать подмножество соответствующих *G-, C-* или L-структур или ограничиться рассмотрением только тех уровней уточнения (классов *l*-эквивалентности), для которых потеря информации не превышает некоторого заданного пользователем значения. Таким образом, процесс порождения реконструктивных гипотез может быть ограничен либо набором рассматриваемых структур, либо ограничениями на способ порождения.

Для достижения гибкости в решении задачи реконструкции ФРИЗ должен располагать разнообразным набором способов порождения, но этот вопрос находится за пределами проблем, связанных с архитектурой ФРИЗ. Это способы порождения соответствующих уточнений (или укрупнений) имеющихся реконструктивных гипотез, примерами чему служат RG-процедура и RC-процедура (и их укрупняющие аналоги).

Как уже говорилось выше, порождение структур также может быть организовано на нескольких уровнях вычислений. Так, например, RC-процедура может быть использована на *глобальном уровне* для работы с классами *r*-эквивалентности G-структур. Модифицированная RG-процедура, в которой на шаге 3 вместо $|^kS|\geq 2$ используется условие $|^kS|>2$, которое затем на локальном уровне для порождения непосредственных уточнений некоторых важных классов *r*-эквивалентности, определенных на глобальном уровне.

Часто бывает необходимо породить уточнения или укрупнения нескольких структур одного уровня уточнения. В этих случаях нужно использовать такие процедуры, которые не порождают одинаковых структур.

Вход в процедуры для оценки реконструктивных гипотез — второй блок на рис. 25 — состоит из порожденных реконструктивных гипотез и различных способов оценки и критериев, определенных пользователем. **К ним относятся определения расстояния (а также другие необходимые характеристики, такие, как коэффициент идентифицируемости, реконструктивная нечеткость или некий уровень доверия) и принцип, на котором основывается реконструкция (несмещенная, минимаксная и т. д.). По умолчанию**

следует использовать такие хорошо теоретически обоснованные понятия, как информационное расстояние и несмещенная реконструкция. Полученные реконструктивные гипотезы нужным образом оцениваются и сравниваются. Если получены интересующие пользователя результаты, особенно относительно множества решений, то они выдаются на печать.

Процедуры принятия решений — третий блок на рис. 25 — используют информацию об оценке реконструктивных гипотез и принимают различные решения в соответствии с заданными пользователем критериями. Самые важные — это решение о том, продолжать или завершить процесс решения, и, если процесс продолжается, решение о том, какая из реконструктивных гипотез должна использоваться на следующем шаге (множество X на рис. 25).

Проиллюстрируем некоторые вопросы, связанные с задачей реконструкции и рассматриваемые в этом разделе, на нескольких примерах.

Пример 19. Рассмотрим возможностиую систему с поведением, определенную на данных, полученных путем наблюдения за четырьмя переменными, характеризующими работу вычислительного комплекса. Целью является нахождение условий, при которых загрузка ЦП (центрального процессора) оказывается высокой. Наблюдаемые значения переменных представляют загрузку ЦП и трех каналов, скажем каналов К1, К2 и КЗ. Наблюдение проводилось в течение одного часа типичной рабочей нагрузки комплекса, и загрузка каждого устройства фиксировалась с интервалом в 1 с. Таким образом, было сделано 3600 наблюдений. Если загрузка, наблюдаемая в течение некоторого интервала— в 1 с, была меньше некоторого заданного исследователем порога (определенного на основании предшествующих исследований), то она считается низкой (Н), а если выше этого порога, то высокой (В).

Для определения на этих данных системы с поведением исследователь решил использовать возможностную методологию без памяти. Из 16 возможных состояний переменных в действительности наблюдались только 6. Поскольку эти состояния наблюдались примерно с одинаковыми частотами, исследователь решил, что функция поведения должна только отличать далее наблюдаемое и ненаблюдаемое состояния. Таким образом, он объявил, что единственно возможными состояниями являются наблюдаемые ранее состояния. Эти состояния имеют возможность, равную 1, остальные состояния (т. е. которые не наблюдались) — равную 0. Полученная функция поведения f приведена в табл. 13.

Таблица 13.

Возможностные функции поведения для обобщенной системы и некоторых несмещенных реконструкций из примера 19

ЦП	К1	К2	К3	f	$f^1=f^4=f^6$	f^2	f^3	f^5	$f^7=f^8=f^9$	f^{10}
Н	Н	Н	Н	1	1	1	1	1	1	1
Н	Н	Н	В	1	1	1	1	0	1	1
Н	Н	В	Н	0	0	1	0	0	0	0
Н	Н	В	В	0	0	1	1	1	0	1
Н	В	Н	Н	0	0	0	1	0	0	0
Н	В	Н	В	0	0	0	0	1	0	0
Н	В	В	Н	0	0	0	0	0	1	0
Н	В	В	В	1	1	1	1	1	1	1
В	Н	Н	Н	0	0	1	1	1	0	1
В	Н	Н	В	0	0	1	0	1	0	0
В	Н	В	Н	1	1	1	1	1	1	1
В	Н	В	В	1	1	1	1	1	1	1
В	В	Н	Н	1	1	1	1	1	1	1
В	В	Н	В	0	0	0	0	0	1	0
В	В	В	Н	0	0	0	0	1	0	0
В	В	В	В	0	0	0	1	0	0	0

Она дает исследователю понимание важной вещи: при некоторых изменениях в организации работы компьютера загрузка ЦП может поддерживаться по высоким уровням, если три канала загружены одним из следующих способов:

К1	К2	К3
Н	В	Н
В	Н	Н
Н	В	В

Для подтверждения этой идеи исследователь решил изучить реконструктивные свойства обобщенной функции поведения f. Его интересовали только реконструктивные гипотезы без потери информации.

В этом случае переменные оказываются сильно ограниченными (возможны только 6 из 16 состояний переменных). Однако легко определить, что проекции f, соответствующие любой паре из четырех переменных (всего таких пар шесть), совершенно неограничены, т. е. любое из четырех состояний, определенных на этой паре переменных, имеет возможность, равную 1. Следовательно, эти переменные попарно независимы, и обобщенная функция поведения не может быть реконструирована только по ее двумерным проекциям.

Чтобы определить, может ли f вообще быть реконструирована по каким-либо проекциям, полезно сначала рассмотреть реконструктивные гипотезы, основанные на использовании

С-структур. С помощью R^C-процедуры мы получаем шесть реконструктивных гипотез на первом уровне уточнения. Их схемы и соответствующие графы изображены на рис. 26,а.

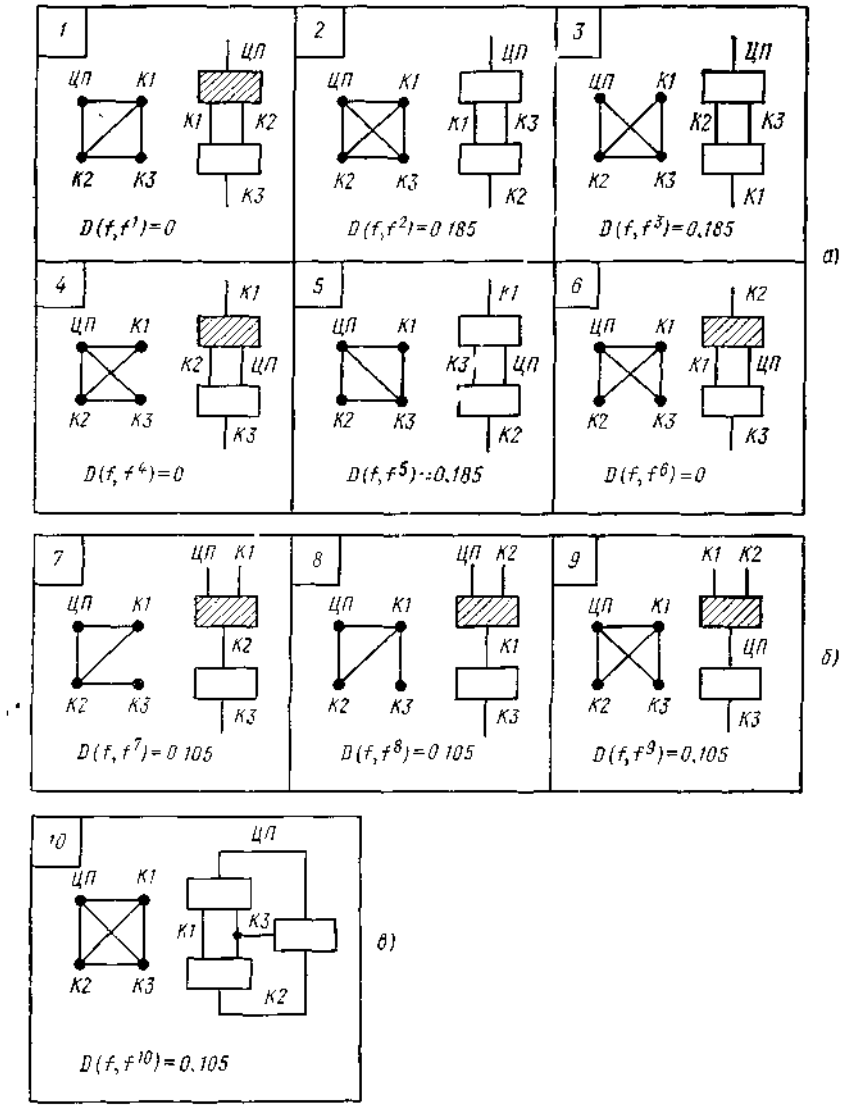


Рис. 26. Рассматриваемые в примере 19 реконструктивные гипотезы

Каждая гипотеза помечена номером в левом верхнем углу соответствующей клетки, а ее несмещенные реконструкции f^h $(h \in N_6)$ приведены в табл. 13. Гипотезы 1, 4 и 6 в точности реконструируют f и являются перспективными кандидатами на включение в множество решений. Каждая из оставшихся гипотез дает по четыре некорректных состояния обобщенной системы. Они также имеют следующее информационное расстояние, вычисленное по формуле (42):

$$(\log_2 10 - \log_2 6)/\log_2 16 = (3.32 - 2.58)/4 = 0.185.$$

Несмещенные реконструкции вычисляются с помощью возможностного варианта процедуры соединения. Для реконструктивной гипотезы 1 она показана на рис. 27.

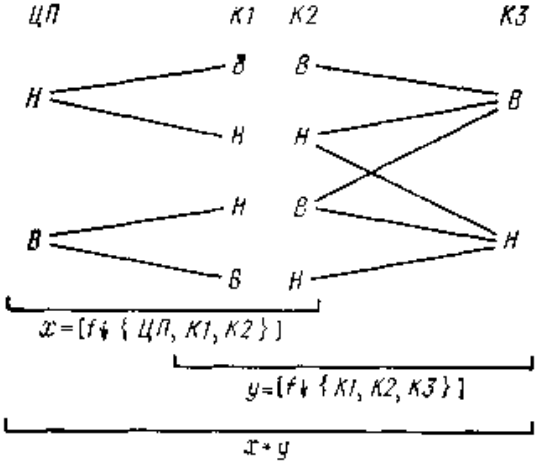


Рис. 27. Иллюстрация к процедуре соединения для реконструктивной гипотезы 1, изображенной на рис. 26

Связями на диаграмме показаны те состояния отдельных трехмерных проекций, возможности которых равны 1. Результатом процедуры соединения, которая выполняется только один раз, являютсявсе четверки из H и B, которые лежат на путях на диаграмме, соединяющих левые и правые узлы.

При изучении трех удачных реконструктивных гипотез, изображенных на рис. 26,а, видно, что все они содержат подсистему, основанную на переменных ЦП, K1 и K2. На рис. 26,а эта подсистема изображена заштрихованным прямоугольником. Таким образом, любая потенциально удачная гипотеза на следующем уровне уточнения должна содержать эту подсистему. Таких гипотез только три (они

показаны на рис. 26,б). Как следует из табл. 13, их несмещенные реконструкции равны. Происходит это из-за того, что двумерные проекции не содержат никакой информации. Данная реконструкция не является безупречной: вместо шести реконструируется восемь состояний, а их расстояния равны 0,105. Это согласно условиям задачи является неприемлемым.

Рассматривать укрупнения этих трех удачных реконструктивных гипотез нет необходимости, поскольку они явно хуже: по определению, они являются менее уточненными, а их расстояния не могут быть меньше расстояний успешных гипотез (т. е. не могут быть меньше 0). Следует, однако, рассмотреть укрупнения неудачных гипотез 2, 3 и 5 (см. рис. 26,а). Воспользовавшись рис. 18, на котором изображена соответствующая решетка G-структур, можно установить, что рассматриваемые структуры— это G-структуры, принадлежащие классу изоморфизма G_4. Из каждой подсистемы, изображенной на рис. 26,а, за исключением успешных подсистем, представленных переменными ЦП, К1 и К.2 (на рисунке они заштрихованы), выбираются две подсистемы с двумя переменными. Однако мы знаем, что эти пары подсистем являются неподходящими и что подсистемы из двух переменных информации не добавляют. Следовательно, реконструктивные гипотезы из класса изоморфизма G_4 могут быть отброшены без вычисления несмещенных реконструкций и расстояний.

Остается только рассмотреть реконструктивные гипотезы, основанные на G-стрктурах из класса изоморфизма G_3. Поскольку подсистемы, представленные переменными ЦП, К1 и К2, снова должны быть исключены из рассмотрения, то остается только одна гипотеза. Она показана на рис. 26,в, а ее функция поведения f^{10} приведена в табл. 13. Мы видим, что и эта гипотеза не подходит: ее расстояние равно 0.105, и, следовательно, она должна быть отвергнута.

Поскольку укрупнением успешных реконструктивных гипотез является только гипотеза, основанная на G_2, ее также не следует рассматривать. Таким образом, можно прийти к заключению, что множество решений состоит из изображенных на рис. 26,а реконструктивных гипотез 1, 4 и 6.

Данный результат подтверждает предположение пользователя о том, что следует обратить внимание на критическую подсистему, базирующуюся на переменных ЦП, К1 и К2. В соответствии с этим загрузка ЦП может поддерживаться на высоком уровне при любой организации вычислительного комплекса, при которой не допускается, чтобы загрузка каналов К1 и К2 была одновременно или высокой, или низкой.

Пример 20. Пример основан на данных по использованию противозачаточных средств до замужества. Состояния следующих двоичных переменных были определены на группе из 414 студенток старших курсов университета:

v_1— отношение к внебрачным связям (0-—всегда отрицательное, 1 — не всегда отрицательное);

v_2—обращение в университетскую клинику для предотвращения беременности (0 — да, 1 — нет);

v_2 — девственность (0 — да, 1 — нет).

Частоты N **(с)** отдельных состояний и соответствующая вероятностная функция поведения f приведены в табл. 14а.

Таблица 14.

Функции поведения из примера 20,а и примера 21,б

(а)					(б)				
v_1	v_2	v_3	N (с)	f (с)	s_1	s_2	s_3	s_4	f (с)
с = 0	0	0	23	0 056	с = 0	0	0	0	1/3
0	0	1	127	0.307	0	0	1	0	2/3
0	1	0	23	0.056	0	0	2	0	1/3
0	1	1	18	0 043	0	0	3	0	1/3
1	0	0	29	0 070	0	1	0	0	1
1	0	1	112	0 270	0	0	3	1	1/3
1	1	0	67	0 162	0	1	0	1	2/3
1	1	1	15	0 036	0	1	1	1	1/3
					0	1	2	1	1
					0	1	3	1	1/3
					0	0	2	2	1/3
					1	0	1	2	1
					1	0	2	2	1/3
					1	0	3	2	2/3
					1	1	3	2	1/3
					0	1	0	3	1/3
					1	1	1	3	2/3
					1	1	2	3	2/3
					1	1	3	3	1/3

Если рассматривать реконструктивные гипотезы, основанные только на С-структурах, то для получения гипотез первого уровня уточнения можно использовать RC-процедуру. Схемы гипотез, их графы и расстояния (полученные в результате выполнения этой процедуры) приведены на рис. 28.

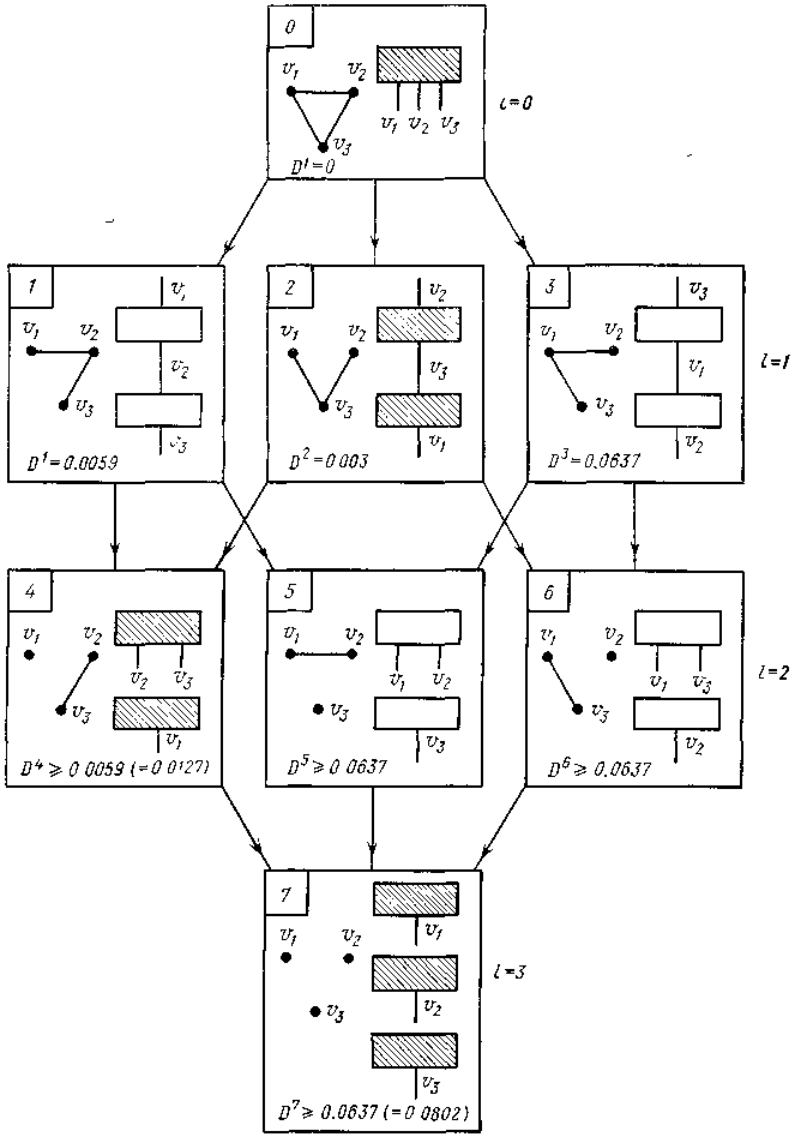


Рис. 28. Иллюстрация к задаче реконструкции, рассматриваемой в примере 20

С помощью предупорядочения по информационному расстоянию по расстояниям для первого уровня уточнения мы можем, как это показано на рисунке, определить нижние границы расстояний для всех реконструктивных гипотез второго уровня уточнения. Так, например, $D^6 \geq 0.0637$, поскольку гипотеза 6 является уточнением гипотезы 3 и $D^3 = 0.0637$. По этим нижним границам расстояний сразу определяется, что гипотеза 2 входит в множество решений.

Оценивая гипотезу 4, имеющую самую маленькую нижнюю границу среди конкурирующих гипотез второго уровня уточнения, получим, что действительное расстояние $D^4 = 0.0127$. Поскольку оно меньше любой нижней границы для других гипотез и $D^7 \geq 0.0637$, то гипотеза 4 является членом множества решений. Обратите внимание на то, что мы пришли к этому заключению не прибегая к вычислению ни конкурирующих гипотез, ни ее преемника. Если нас интересует гипотеза 7, то можно вычислить, что $D^7 = 0.0802$, и понятно, ее следует включить в множество решений, поскольку она является самой уточненной гипотезой из этой решетки уточнения.

Элементы множества решений выделены на рис. 28 заштрихованными прямоугольниками. В данном случае комбинированное отношение упорядочения упорядочивает их полностью. Как видно, переменные v_1 (отношение) и v_2 (обращения в клинику) больше зависят от переменной v_3 (девственность), чем друг от друга. Особенно сильна связь между v_2 и v_3.

Пример 21. На этом примере мы хотим продемонстрировать некоторые проблемы, возникающие при решении задачи реконструкции в тех случаях, когда заданная функция поведения зависит от памяти. Данная система представлена тремя переменными, описывающими человека (v_1 — производительность труда, v_2 — общее состояние здоровья, v_3 — стресс). Параметром является время (полностью упорядоченный параметр). Наблюдения делаются ежедневно в течение определенного периода времени. Ограничения на переменные заданы возможностной функцией поведения, приведенной в табл. 14,б. Они определены на множестве состояний следующих выборочных переменных:

$$s_{1,t} = v_{1,t} \qquad s_{2,t} = v_{2,t},$$

$$s_{3,t} = v_{3,t}, \qquad s_{4,t} = v_{3,t-1}.$$

Функция поведения получена по данным с помощью метода оценки масок, описанного в разд. 4.6. Не вдаваясь в подробности относительно предыдущих этапов исследования, сосредоточим свое внимание на задаче реконструкции данной функции поведения. Пусть стандартная формулировка этой задачи основывается на идеях несмещенной реконструкции и информационного расстояния. Предположим далее,

что требуется, чтобы реконструктивные гипотезы основывались только на С-структурах, и что максимальное приемлемое расстояние равно 0.1.

Сначала породим и оценим реконструктивные гипотезы, основанные на С-структурах первого уровня уточнения. Они показаны на рис. 29 (гипотезы 1—6), причем выборочные переменные S_k представлены их идентификаторами k ($k \in N_k$), а подмножества переменных разделяются косой чертой.

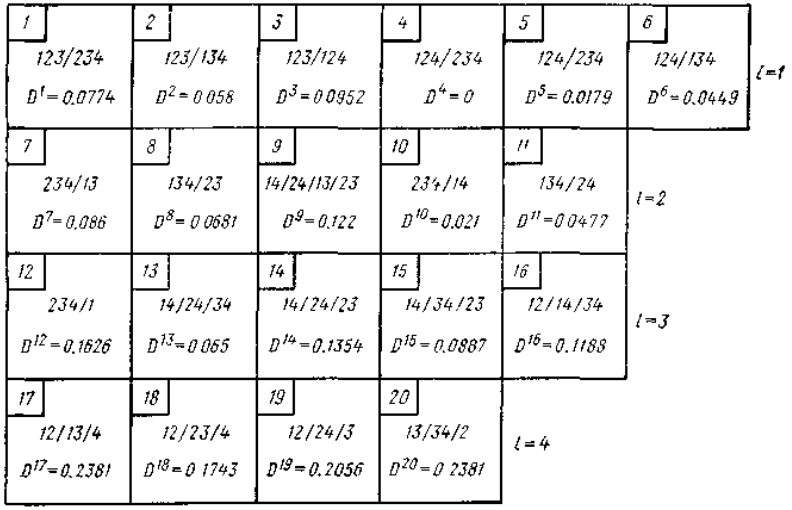


Рис. 29. Реконструктивные гипотезы, оцениваемые в примере 21

Оценка этих гипотез заключается в определении их несмещенных реконструкций (с помощью возможностного варианта процедуры соединения) и вычисления их расстояний [по формуле (42)].

Поскольку гипотеза 4 имеет наименьшее расстояние ($D^4 = 0$), то породим и оценим все ее непосредственные С-уточнения. Всего этих уточнений пять, и они помечены номерами 7—11. Наименьшее расстояние в этой группе $D^{10} = 0.021$. Из монотонности информационного расстояния следует, что расстояние любой гипотезы на первом уровне уточнения является также нижней границей расстояний для всех ее уточнений. Следовательно, единственной гипотезой первого уровня уточнения, которая потенциально может быть источником уточнений с расстояниями, меньшими или равными 0.021, является гипотеза 5, чье расстояние $D^5 = 0.0179$. Однако легко видеть, что любое непосредственное уточнение гипотезы 5 является

одновременно уточнением какой-то другой гипотезы первого уровня. Отсюда следует, что любое непосредственное уточнение гипотезы 5 либо находится среди гипотез 7—11, либо среди гипотез, нижняя граница расстояний которых больше 0.021. Следовательно лучшей на втором уровне является гипотеза 10. Ее непосредственными уточнениями являются гипотезы 12—15, среди которых гипотеза 13 имеет наименьшее расстояние.

Для того чтобы убедиться, что гипотеза 13 является лучшей на третьем уровне, необходимо оценить все остальные гипотезы этого уровня, за исключением уточняющих гипотез 1, 3, 7 и 8, нижние границы которых превышают D^{13}. Поскольку гипотеза 7 является уточнением гипотезы 1, ею можно пренебречь. Графы гипотез 1, 3 и 8 приведены на рис. 30,а. Графы гипотез, не являющихся их уточнением на третьем уровне, должны содержать ребра (1, 4), (3, 4), а также либо ребро (1, 2), либо ребро (2, 4). Имеется всего два таких графа. Они приведены на рис. 30,б.

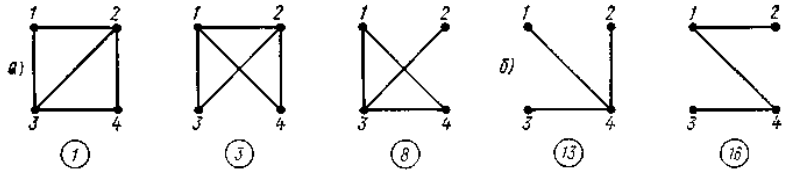


Рис. 30. Графы некоторых реконструктивных гипотез, рассматриваемые в примере 21

Первый на самом деле является графом гипотезы 13, второй представляет гипотезу 12/14/34, являющуюся единственным потенциальным конкурентом гипотезы 13.

Оценив этого потенциального конкурента (на рис. 29 он помечен номером 16), мы получим, что $D^{16} \neq 0.1188 > D^{13}$. Следовательно, гипотеза 13 является лучшей на третьем уровне уточнения.

Поскольку наименьшее расстояние на третьем уровне ($D^{13} = 0.065$) меньше, чем наибольшее допустимое расстояние, необходимо исследовать 4-й уровень. На этом уровне 15 гипотез (представленных всеми парами ребер на графе из четырех узлов), но только четыре из них не являются уточнениями гипотез 9, 12, 14 и 16, расстояния которых превышают критическое значение 0.1. Это гипотезы 12/13/4, 12/23/4, 12/24/3 и 13/34/2. Их номера и расстояния приведены на рис. 29. Поскольку все эти расстояния превышают 0.1, никакая из этих гипотез не входит в множество решений, и дальнейшие уточнения не нужны. Множество решений полностью упорядочено и состоит из

гипотез 4, 10 и 13 (а также, возможно, гипотезы 0 — обобщенной системой 1234).

Используя предупорядочение по информационному расстоянию, нам удалось решить эту задачу (причем совершенно точно), оценив только 20 из 63 возможных реконструктивных гипотез, т. е. только одну треть всех гипотез. **Для систем с большим числом переменных применение предупорядочения по информационному расстоянию еще более эффективно**

Вообще говоря, чем больше отличаются (по расстоянию) оцениваемые реконструктивные гипотезы на отдельных уровнях уточнения, тем эффективнее использование этого предупорядочения.

Часто бывает полезно посмотреть на приращения минимального расстояния, соответствующие соседним уровням уточнения. Для этого определяется расстояние для наиболее уточненной гипотезы и вычисляется среднее приращение расстояния, равное этому наибольшему расстоянию, деленному на общее число уровней уточнения. В данном примере расстояние для наиболее уточненной гипотезы 1/2/3/4 равно 0.4591, следовательно, среднее приращение расстояния равно 0.4591/6 = 0.0765. **Экстраполируя по известным значениям расстояний, можно получить график зависимости минимального расстояния D_l от уровня уточнения *l*.** Этот график для данного примера приведен на рис. 31.

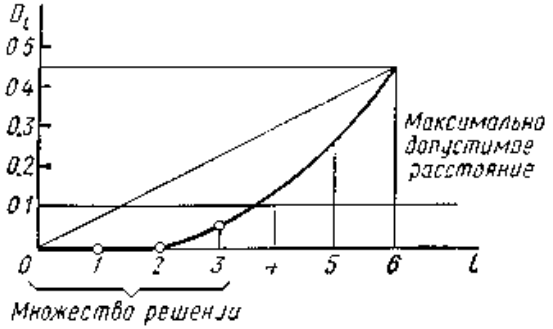


Рис. 31. Зависимость минимального расстояния *D,* от уровня уточнения *l* (пример 21)

График точен для *l=0,* 1, 2, 3, 6, приблизителен для *l*=4 (нам известно, что $0.1188 \le D_4 \le 0.1748$) и оценен для *l = 5.*

Остается только решить вопрос относительно однозначности управления для каждого элемента множества решений (разд. 5.4, пример 6). Как показано на рис. 32,а, переменные 1, 2, 3, очевидно, являются порождаемыми, а единственной порождающей переменной является переменная 4.

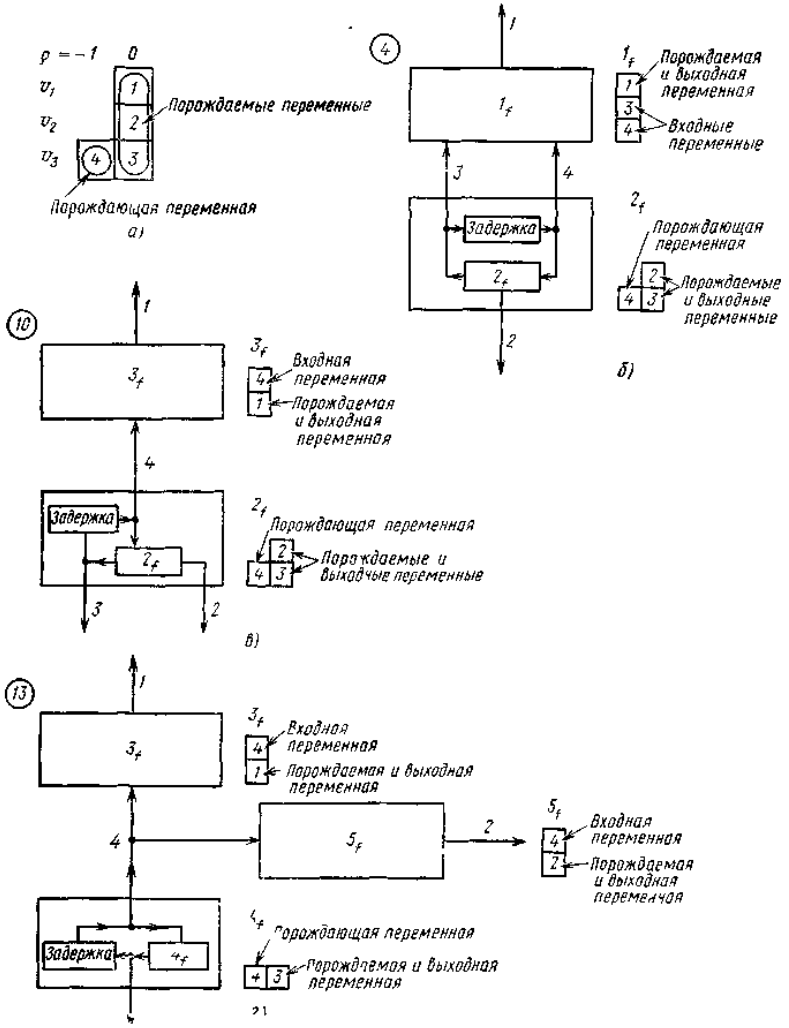


Рис. 32. Подробное изображение элементов множества решений из примера 21

Каждая порождаемая переменная должна управляться (определяться) в точности одной подсистемой реконструктивной гипотезы. Для гипотезы 134/234 ясно, что переменные 1 и 2 управляются соответственно подсистемами 134 И 234, однако переменная 3 может управляться любой из них. Решение о том, какая из подсистем должна быть выбрана для управления переменной 3, должно быть принято исходя из их порождающих нечеткостей. Для данного примера вычисленные *U*/-нечеткости равны: U(3| 1,4) =0.834 для подсистемы 134 и U(3|2,4) для подсистемы 234. Так как U(3|2,4) <U(3| 1,4), для управления переменной выбирается вторая подсистема.

Нужно также принять решение о том, как представить в подсистемах порождающую переменную 4. Здесь есть три возможности: переменная может запоминаться в одной из подсистем или в обеих. Если она запоминается только в одной подсистеме, она должна использоваться как входная переменная другой подсистемы. Однако необходимо отметить, что разница между этими возможностями скорее внешняя, чем функциональная, и, следовательно, выбор может быть произведен совершенно произвольно. Пусть в нашем примере переменная 4 будет запоминаться в подсистеме 234 и рассматриваться как входная переменная подсистемы 134.

Из принятых решений относительно ролей переменных 3 и 4 реконструктивной гипотезы 134/234 будет получена схема, изображенная на рис. 32,б. На этой схеме также приведены маски, соответствующие отдельным подсистемам, причем указаны порождаемые, порождающие и входные переменные.

Окончательные схемы остальных элементов множества решений— гипотез 14/234 и 14/24/34 — приведены соответственно на рис. 32,в и г. В обоих случаях роли выборочных переменных, как это показано на схемах, определяются единственным образом.

Важно понимать, что для зависящих от памяти систем осмысленными являются далеко не все реконструктивные гипотезы. В самом деле, понятно, что гипотеза не имеет смысла, если порождающая переменная не входит по крайней мере в одну подсистему этой гипотезы, содержащую соответствующую порождаемую или входную переменную или другую порождающую переменную (определенную на той же базовой переменной), по которой она может быть определена с помощью запоминания. Такая порождающая переменная может быть оставлена неопределенной, поскольку она не может быть ни порождена (так как она порождающая), ни определена с помощью запоминания другой переменной, которая сама определяется

некоторым особым образом. Так, например, если обобщенная система описывается маской, изображенной на рис. 32,а, то все гипотезы, для которых переменные 3 и 4 не входят по крайней мере в одну общую подсистему, являются бессмысленными. Из этого следует, что в данном случае ровно половина реконструктивных гипотез, базирующихся на С-структурах, являются бессмысленными; это те гипотезы, графы которых не содержат ребра (3, 4), т. е. гипотезы 123/124; 14/24/23, 123/4, 13/24 и т. д. Несмотря на то, что множество решений в примере 21 не содержит бессмысленных гипотез, сам процесс решения может быть упрощен за счет того, что оценивались бы только осмысленные гипотезы (не нужнс было оценивать 8 из 20 оцененных гипотез).

Если предположить, что используемое параметрическое множество полностью упорядочено, то можно следующим образом определить содержательную реконструктивную гипотезу для зависящей от памяти обобщенной системы с поведением. **Реконструктивная гипотеза *h* является *содержательной* тогда и только тогда, когда любая порождающая переменная s_h, определяемая уравнением**

$$s_{k,t} = v_{i,t+a},$$

входит по крайней мере в одну подсистему *h*, содержащую переменную s_j, которая определяется уравнением

$$s_{l,t} = v_{t,t+b},$$

где $b>a$, если переменные порождаются по возрастанию t (предсказание), и $b<a$, если переменные порождаются по убыванию t (восстановление). Понятие содержательной реконструктивной гипотезы может быть легко обобщено на зависящие от памяти системы, базирующиеся на двух и более полностью упорядоченных параметрических множествах (таких, как двух- и трехмерные декартовы пространства), однако для таких систем формализация оказывается существенно более сложной прежде всего из-за значительного роста числа возможных порядков порождения.

5.8. Анализ реконструируемости

Анализ реконструируемости — это пакет методологических инструментов, входящий в ФРИЗ и используемый при решении целого **класса интеллектуальных задач, имеющих следующее общее свойство: в них изучается взаимосвязь между обобщенными системами ИИ и различными их подсистемами.** В задачах этого класса фигурируют системы ИИ двух эпистемологических типов: порождающие системы ИИ и структурированные порождающие

системы ИИ, причем обычно они представляются в виде систем ИИ с поведением. Эти задачи естественным образом разбиваются на два подкласса в зависимости от эпистемологического уровня исходной (заданной) системы ИИ. **Задачи, в которых исходная система ИИ является порождающей структурированной, называются *задачами идентификации,* а задачи, в которых исходная система ИИ является структурированной системой,— *задачами реконструкции.***

Самые общие типы задач идентификации и реконструкции, в которых множества состояний переменных не обладают никакими особыми свойствами, сформулированы и рассмотрены соответственно в разд. 5.6 и 5.7. Основные вопросы (подзадачи), связанные с этими задачами, не зависят от конкретных методологических отличий и являются предметом общего анализа реконструируемости. Они показаны на рис 33 и приведены в следующем списке:

определение *реконструктивного семейства* для заданной системы ИИ с поведением;

определение *коэффициента идентифицируемости* (реконструктивной нечеткости) для заданной системы ИИ с поведением;

определение *несмещенной реконструкции* для заданной системы ИИ с поведением;

определение *реконструкции наименьшего риска* или, возможно, реконструкции какого-то другого типа для заданной системы ИИ с поведением;

разрешение локальных несогласованностей в заданной системе ИИ с поведением (разд. 5.11);

порождение подходящих *реконструктивных гипотез* для заданной системы ИИ с поведением;

вычисление соответствующих *проекций* заданной системы ИИ с поведением;

вычисление *расстояния* между заданной системой ИИ с поведением и системой ИИ, реконструированной по реконструктивной гипотезе;

упорядочение соответствующих реконструктивных гипотез и определение *приемлемых реконструктивных гипотез* (множества решений задачи реконструкции);

определение управления для рассматриваемых переменных.

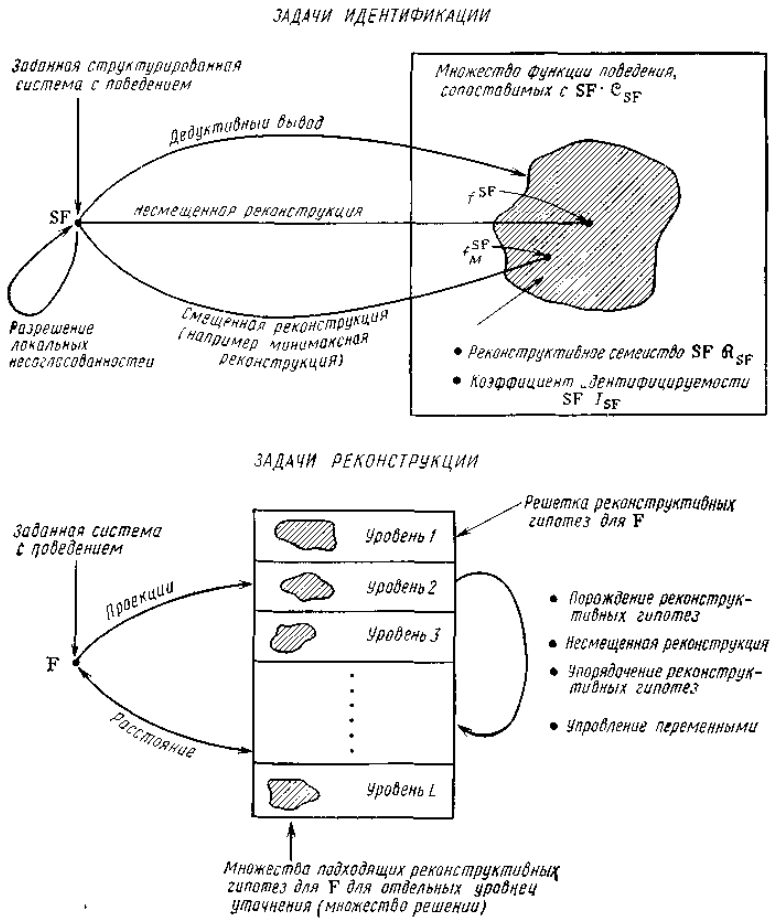


Рис. 33. Основные вопросы, связанные с анализом реконструируемости

Если посмотреть с точки зрения эпистемологической иерархии систем ИИ, то легко увидеть, что анализ реконструктивности представляет собой решение последовательностей задач, принадлежащих к четырем категориям, которые на рис. 34 изображены помеченными стрелками.

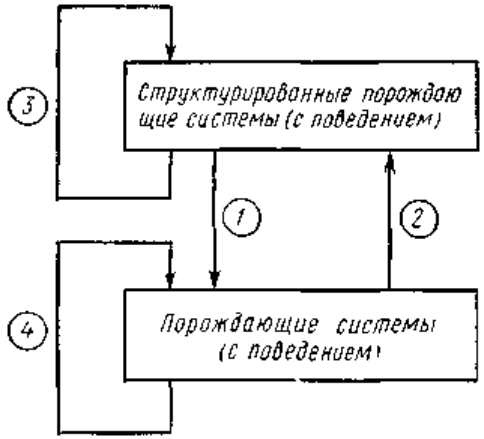


Рис. 34. Категории задач, рассматриваемые при анализе реконструируемости

Приведем список подзадач, относящихся к каждой категории:
1 — реконструктивное семейство, коэффициент идентифицируемости, несмещенная реконструкция или реконструкция наименьшего риска;
2 — проекции;
3 — разрешение локальных несогласованностей, порождение и упорядочение реконструктивных гипотез, управление переменными;
4 — расстояние.
При других методологических отличиях возникают другие типы задач идентификации и реконструкции. Например, если переменные непрерывны, то проекции обобщенной системы ИИ с поведением зависят не только от выбранных подмножеств переменных, но и от преобразований координат. Так, трехмерный объект, изображенный на рис. 35,а, может быть полностью реконструирован по трем своим двумерным (планарным) проекциям (видам), скажем по виду слева, виду спереди и виду снизу (рис. 35,б) в соответствии с декартовыми координатами, определенными на этом объекте.

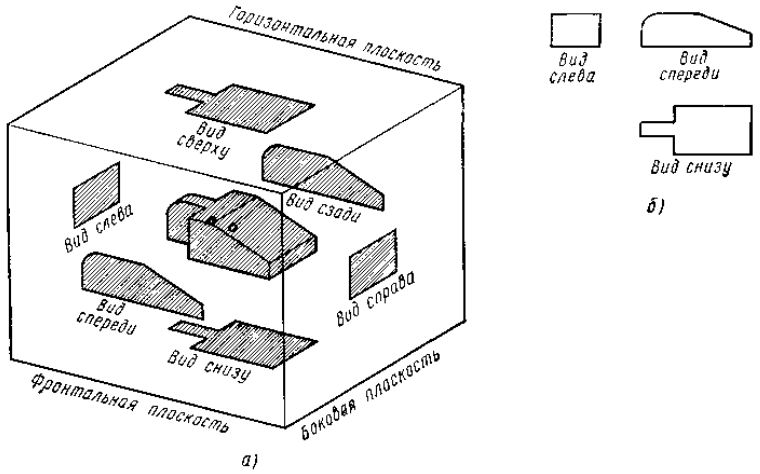


Рис. 35. трехмерное тело , реконструируемое по трем своим двумерным ортогональным проекциям.

Несмотря на то, что реконструируемость сохраняется при изменении положения начала системы координат, очевидно, что это свойство не сохраняется при вращении координат. В действительности существует континуум проекций одного объекта, соответствующий континууму вращений системы координат или, иначе, континууму вращений объекта в одной системе координат. Кроме того, дополнительные проекции могут быть получены на любую плоскость, определенную в данной системе координат. Необходимо отметить, что показанные на рис. 35 проекции являются проекциями специального типа — так называемые *ортогональные проекции,* получаемые опусканием перпендикуляра на плоскость проектирования из каждой точки объекта. Другим типом проекций являются так называемые *тени,* получаемые соединением каждой точки объекта с некоторой фиксированной точкой (называемой точкой проекции или источником света). Полученные пересечения этих прямых с плоскостью проектирования и дают тень. Понятно, что и здесь имеется континуум проекций, соответствующий континууму положений точки проектирования. Далее, пересечения объекта с различными плоскостями,— так называемые *сечения,*— также могут быть использованы в качестве двумерных представлений.
Несмотря на огромное разнообразие возможных проекций для систем с непрерывными переменными, основные понятия, связанные с отношением целого (скажем, трехмерных объектов) и частей (их

различных двумерных и одномерных проекций), такие, как реконструктивное семейство, несмещенная реконструкция, локальная согласованность, расстояние и т. д., остаются теми же самыми, даже если они принимают некие специфические формы. Например, процедура соединения, с помощью которой по двумерным ортогональным проекциям определяется несмещенная реконструкция, состоит в восстановлении для каждой из проекций бесконечных цилиндров и определении общего объема этих цилиндров (т. е. множества точек, принадлежащих их пересечению). Разумеется, существуют некоторые дополнительные проблемы, связанные с непрерывными переменными. Они прежде всего связаны с выбором соответствующих проекций. Несмотря на то, что эти проблемы имеют важное значение в таких областях, как оптика, механика, картография или томография, это тем не менее проблемы, частные, не применимые ко всем системам. Поэтому они не входят в сферу вопросов, рассматриваемых в данной работе, хотя в ФРИЗ должны быть включены соответствующие методы решения этих задач. Некоторые из этих задач рассматривабтся в дескриптивной геометрии и в системах обработки изображений.

Методологический пакет, такой, как пакет анализа реконст-руируемости, должен быть доступен как в *интерактивном,* так и в *автоматическом (пакетном) режиме.* При работе в интерактивном режиме пользователь может применять соответствующие процедуры в любом порядке и объеме, принимает решения, исходя из промежуточных результатов и собственного опыта. Например, при решении задачи реконструкции он может начать с исходной структурированной системы ИИ как возможной реконструктивной гипотезы, оценить ее и, если нужно, сравнить с непосредственными уточнениями, со всеми реконструктивными гипотезами, в ее структурном соседстве, с другими возможными гипотезами, не находящимися в соседстве, или предпринять какие-либо другие действия. При решении задачи идентификации можно сравнить несколько вариантов структурированных систем ИИ по их ко-эффициентам идентифицируемости. В зависимости от полученных результатов можно определить затем или реконструктивные семейства, или только некие типы реконструкций для некоторых из них. То, что интерактивный режим дает возможность пользователю сосредоточить внимание на конкретных вопросах и воспользоваться своим опытом и знаниями, является большим преимуществом интерактивного режима при работе с большими системами ИИ, в которых полная обработка практически невозможна из-за неприемлемых запросов на вычислительные ресурсы.

При работе в автоматическом режиме пользователю должен быть предоставлен набор последовательностей различных процедур, что позволило бы ему выбирать различные варианты решения задачи. Так, например, при решении задачи реконструкции одна из последовательностей процедур, повторяющаяся на каждом уровне уточнения, может состоять из *RC*-процедуры, процедуры соединения, вычисления расстояния и процедуры принятия решения о продолжении. В другой последовательности *RC*-процедура может быть заменена на RG-процедуру (ограниченную уточнениями из одного класса *r*-эквивалентности), за которой следуют три остальные процедуры (процедура соединения и т. д.); другие последовательности могут базироваться на укрупнении, а не на уточнении и т. п. Одна из последовательностей должна выполняться по умолчанию, например простая последовательность, основанная на RC-процедуре.

Понятие структурного соседства можно, например, непосредственно использовать для обнаружения дефектов в связях между элементами структурированной системы ИИ в тех случаях, когда непосредственное наблюдение связей невозможно. Правильно использованное, оно может оказать большую помощь и при проектировании. Его, например, можно применять для определения всего множества структурных уточнений, полностью сохраняющих данную проектируемую систему ИИ с поведением. Такие уточнения являются основой для естественного способа проектирования по частям, что делает этот процесс более управляемым. Под «естественным» здесь понимается то, что эти уточнения содержат только входные и выходные переменные, входящие в данную систему ИИ с поведением, т. е. не содержат дополнительных (или искусственных) переменных, вводимых на данном этапе. В каждом уточнении по крайней мере некоторые из заданных переменных входят в несколько подсистем ИИ и, следовательно, играют несколько разных ролей. В максимально подробных уточнениях множественность использования любой переменной достигает предела. Далее переменные должны при необходимости вводиться с помощью обычных методов декомпозиции или другого подходящего метода проектирования.

Несмотря на такое использование реконструктивного анализа для систем ИИ, необходимо подчеркнуть, что основным его назначением является исследование естественных систем. Дело в том, что отношение часть-целое в естественных системах куда более неопределенно, чем в искусственных. Так, например, любая система ИИ является также определением соответствующей обобщенной системы ИИ. Это непосредственно следует из того факта, что соединяющие переменные в любой системе ИИ являются либо

переменными, которые прямо представляют единственную систему ИИ с поведением (определенную в задаче проектирования), либо искусственными соединяющими переменными, введенными только для косвенного представления этой единственной системы ИИ. Следовательно**, реконструктивное семейство любой структурированной системы ИИ является единственным и представляет собой объединение функций поведения ее элементов. То есть обобщенная система ИИ любой структурированной системы ИИ всегда представляется несмещенной реконструкцией этой структурированной системы ИИ.**

Это взаимно однозначное соответствие между структурированными системами ИИ и связанными с ними обобщенными системами ИИ является доводом, возможно самым главным, в пользу того, что связь между сопоставимыми системами ИИ с поведением и структурированными системами ИИ при исследовании систем (т. е. при исследовании естественных систем) часто недостаточно хорошо понимается, особенно людьми с инженерной подготовкой. В самом деле, в литературе описано множество больших систем, которые, как предполагается, описывают различные естественные явления и составлены из меньших взаимосвязанных систем (подсистем). На основании полученной конкретной структурированной системы делаются выводы о свойствах обобщенной системы с помощью соединения или композиции функций поведения соответствующих элементов. Понятно, что подобные выводы основываются на предположении, что структурированные системы представляют эту обобщенную систему точно так же, как для систем ИИ. Это недо-казанное и обычно неверное предположение, которое в подобных исследованиях никогда явно не декларируется, принимается без доказательства из-за неправомерной и сбивающей с толку аналогии с искусственными системами.

Сопоставим эти наши рассуждения с очень образными рассуждениями, с помощью которых Р. Розен пришел практически к тем же выводам: «Под *анализом* здесь понимается разложение системы на семейство в некотором смысле более «простых», чем эта система, подсистем, полученных из нее, и попытки вывести свойства этой системы из свойств ее подсистем. Выделению подсистем формально соответствует процесс *абстрагирования,* при котором из первоначальной системы исключается ряд степеней свободы (т. е. потенциальных интерактивных возможностей), и их остается только ограниченное число. Этот процесс может быть реализован физически (когда специалист по молекулярной биологии выделяет из клетки примеси, создавая тем самым абстрактную клетку), а может быть чисто

формальным (когда эколог представляет популяцию реальных организмов в терминах отношения хищник-жертва). Такие абстракции должны удовлетворять следующим основным требованиям:

1) полученные подсистемы должны быть «проще» первоначальной системы, из которой они абстрагированы;
2) эти подсистемы должны быть получены «естественным» образом (т. е. с помощью известных и обоснованных процедур);
3) свойства полученных таким образом процедур должны позволять определить свойства первоначальной системы.

Очевидно, что требование 1 является решающим; ничего не получится, если выделенные системы будут столь же необозримыми, как и первоначальная система. В научных методах анализа это неявно было понято давно. Столь же важно требование 3: всякое свойство отдельных подсистем, не переносимое на свойства первоначальной системы, является *артефактом (*В данном случае, искусственно созданным фактом). Требование 2, однако, является субъективным и счязано с тем, как следует работать с первоначальной системой. Таким образом, оно не равноценно требованиям 1 и 3.

Тем не менее во многих эмпирических подходах к системному анализу наибольшее внимание уделяется именно требованию 2. Вероятно, интуитивно предполагается, что, если процедуры удовлетворяют требованию 2, то требования 1 и 3 будут выполнены автоматически. По крайней мере, считается, что из l+2 следует 3. Однако, как уже было сказано, в общем случае это совершенно неверно. В самом деле, мы знаем, что важнейшие свойства 1 и 3, которые должны быть выполнены с помощью любых средств системного анализа, должны допускать определение того, что мы рассматриваем как «естественное». Действительно, «естественность» не должна постулироваться заранее, а должна определяться в контексте рассматриваемых задач.

Приведем в пояснение простой пример. Задача трех тел в физике является сложной во вполне определенном смысле, динамические уравнения, описывающие произвольную систему из трех гравитационных масс, не могут быть проинтегрированы непосредственно. Можно попробовать подойти к решению подобной задачи путем анализа семейства более «простых» подсистем. Интуитивно кажется, что такими системами являются системы из двух тел и из одного тела. Они и в самом деле «проще» первоначальной системы, и получены из нее «естественным» образом. Тем не менее ясно, что таким образом решить задачу трех тел нельзя, поскольку декомпозиция исходной системы на отдельные подсистемы не-обратимо нарушает ее первоначальную динамику, которая нас и

интересует (**здесь мы снова сталкиваемся с неспособностью физики работать с произвольными взаимодействиями**). Таким образом, для решения задачи трех тел наши кажущиеся «естественными» декомпозиции бесполезны; если какой-то анализ и подойдет для решения задачи такого типа, то соответствующие подсистемы (т. е. подсистемы, удовлетворяющие требованию 3 будут совершенно «неестественными» с точки зрения обычного физического разбиения системы на части.»

5.9. Вычислительные эксперименты

Эксперименты на компьютере не только возможны, но и могут дать информацию, которую невозможно получить иным путем.
У. Росс Эшби

В этом разделе в качестве примера метаметодологических средств ФРИЗ описываются вычислительные эксперименты, с помощью которых можно определить некоторые важнейшие характеристики анализа реконструируемости (связанные с задачей реконструкции). Эти характеристики нужны для более глубокого понимания реконструктивного анализа, помощи пользователям ФРИЗ по применению реконструктивного анализа при общесистемных исследованиях и оценки новых принципов, таких, как принцип индуктивного вывода, рассматриваемый в разд. 5.10.
В типичном эксперименте реконструктивная гипотеза выбирается при заданном числе переменных и мощностях их множеств состояний. Затем процесс порождения данных с помощью этой гипотезы моделируется на компьютере. В большей части этих экспериментов порождается последовательность из 2000 элементов данных. В соответствии с описанными выше правилами анализ реконструируемости проводится на 10 разных сегментах этих последовательностей, содержащих 10, 20, 40, 80, 160, 320, 640, 1000, 1500 и 2000 элементов данных. Полученные для каждого сегмента результаты затем сравниваются с данной реконструктивной гипотезой.
Для заданного числа переменных и их множеств состояний порождается и анализируется соответствующее число различных последовательностей данных. Усредненные результаты этих экспериментов затем используются для определения различных характеристик. Эксперименты для простоты ограничиваются только С-структурами. Они были проведены для множеств $\mathscr{C}_3$, $\mathscr{C}_4$ и $\mathscr{C}_5$; для каждого множества были соответствующим образом представлены все уровни

уточнения. Аналогичные эксперименты были проведены для множеств состояний одинаковой мощности (2, 3, 4 и 5) для всех рассматриваемых переменных, а также для определенных смесей разной мощности. Поскольку отличия переменных, описываемых масками и средами, которые для общесистемных исследований очень важны, собственно для реконструктивного анализа значения не имеют, эксперименты проводились только для нейтральных систем ИИ без памяти.
Последовательности данных порождались с помощью генератора случайных чисел в соответствии с конкретной вероятностной структурированной системой (представленной С-структурой). Затем они **анализировались вероятностным и возможностным методами**. На самом деле **одной из целей проведения экспериментов было сравнение этих двух методов и определение их областей применения.**
На начальном этапе эксперимента было замечено, что возможностный анализ демонстрирует тенденцию к естественной классификации реконструктивных гипотез на каждом уровне уточнения на хорошие и плохие гипотезы, т. е. **на гипотезы соответственно с малыми и большими расстояниями**. Было также отмечено, что **корректные гипотезы (т. е. гипотезы, с помощью которых были порождены анализируемые данные**) часто не обладают наименьшим расстоянием, но почти всегда принадлежат к хорошему кластеру. Исходя из этих наблюдений вероятностный и возможностный анализы проводились по немного отличающимся правилам.
При вероятностном анализе любая порожденная последовательность данных анализируется на соответствующей решетке уточнения дважды с помощью двух различных процедур поиска. Согласно первой процедуре на каждом уровне уточнения уточняются только структуры с минимальным расстоянием. Согласно второй процедуре уточняются все структуры, чьи расстояния не превышают минимальное более чем на 100%. По каждой из процедур характеристики вычисляются отдельно.
При возможностном анализе структуры на каждом уровне уточнения кластеризуются на плохие и хорошие и далее уточняются только хорошие структуры. Любая последовательность данных анализируется дважды с помощью двух разных процедур кластеризации. Для описания этих процедур пусть

$$R = \{(C_i, d_i) \mid i \in N_r\}$$

обозначает множество всех С-структур C_i, оцениваемых на определенном уровне уточнения конкретного эксперимента, с их расстояниями d_i. Пусть $d_i \le d_{i+1}$ для всех $i \in N_{r-1}$ и пусть

$$G = \{C_1, C_2, \ldots, C_c\};$$
$$B = \{C_{c+1}, C_{c+2}, \ldots, C_r\}$$

— соответственно кластеры хороших и плохих структур, где $1 \leq c \leq r$ (т. е. кластер G всегда непустой, в то время как кластер B в некоторых случаях может быть пустым).

В первой процедуре кластеризации c определяется наименьшим значением i, для которого разность $d_i - d_{i-1}$ превышает среднюю разность в R для всех $i \in N_{r-1}$. То есть

$$d_i - d_{i-1} \leqslant \frac{d_r - d_1}{r}$$

дая $i \in N_c$ $(d_o=0)$ и

$$d_{c+1} - d_c > \frac{d_r - d_1}{r}.$$

Будем называть эту процедуру *кластеризацией по средней разности* или AD-кластеризацией *(average difference)*.

Во второй процедуре кластеризации c определяется значением $k \in N_r$, для которого выражение

$$\frac{1}{a_2 - a_1}\left(\sum_{i=1}^{c} | d_i - a_1 | + \sum_{i=c}^{r} | d_i - a_2 | \right)$$

достигает минимума, причем

$$a_1 = \frac{1}{c}\sum_{i=1}^{c} d_i,$$

$$a_2 = \frac{1}{r-c}\sum_{i=c+1} d_i.$$

Эта процедура основана на естественном кластеризационном требовании о том, что расстояния между кластерами должны быть велики, а расстояния внутри кластеров малы; будем называть это *кластеризацией по внутреннему и внешнему расстоянию* или IOD-кластеризацией (от англ. *Inside and Outside Distance)*.

Теперь можно целиком описать процедуру реализации одного эксперимента (см. диаграмму на рис. 36).

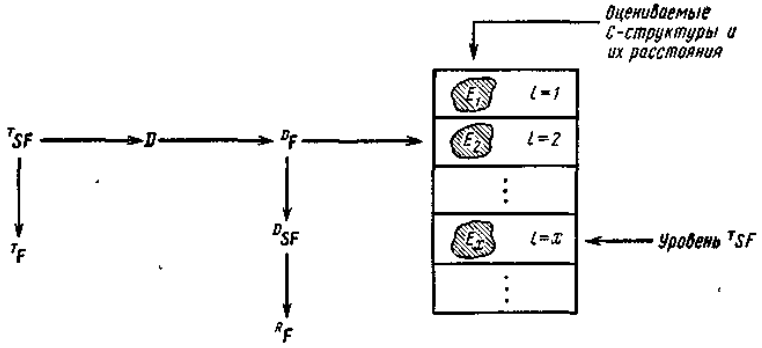


Рис. 36. Схема вычислительного эксперимента

Она начинается с выбора структурированной системы ИИ с поведением ^{T}**SF** (которая рассматривается в эксперименте как подлинная). Эта система основывается на С-структуре. Эта структуризованная система, представляющая обобщенную систему ИИ с поведением ^{T}**F** (полученную из системы ^{T}**SF** с помощью процедуры соединения), моделируется на компьютере и используется для порождения данных. После порождения данных из соответствующей системы данных D выводится обобщенная система ИИ с поведением без памяти ^{D}**F** (вероятностная или возможностная). Затем для системы ^{D}**F** проводится реконструктивный анализ в соответствии с одной из упомянутых выше процедур поиска (например, для возможностных систем ИИ, основанных на одном из двух типов кластеризации). Результатом является последовательность множеств С-структур (и их расстояний), которые оцениваются на отдельных уровнях соответствующей решетки уточнения. Это множества, скажем множества

$$E_l = \{(C_{i_l}, d_{i_l}) \mid i_l \in I_l\}$$

для $l=1, 2, ..., n(n-1)/2$, где n — число рассматриваемых переменных. Особый интерес представляет множество E_l для того же уровня уточнения, что и система ^{T}**SF**. Система ^{D}**F** также используется для определения структурированной системы ^{D}**SF**, основанной на такой же С-структуре, что и заданная структурированная система ^{T}**SF**. Эта система (^{D}**SF**) представляет обобщенную систему ИИ с поведением ^{R}**F** (реконструированную обобщенную систему ИИ).

По множеству E_l $(l \in N_{n\,(n-1)/2})$ и обобщенным системам ИИ с поведением ^{T}**F**, ^{D}**F** и ^{R}**F**, полученным в результате экспериментов одного типа (т. е. для определенного числа переменных, определенных множеств состояний, для вероятностного или возможностного

варианта и т. д.), можно определить различные характеристики результатов анализа реконструируемости. Опишем эти характеристики на нескольких примерах систем из трех переменных.
Некоторые основные характеристики для вероятностных систем (с тремя переменными) представлены на рис. 37.

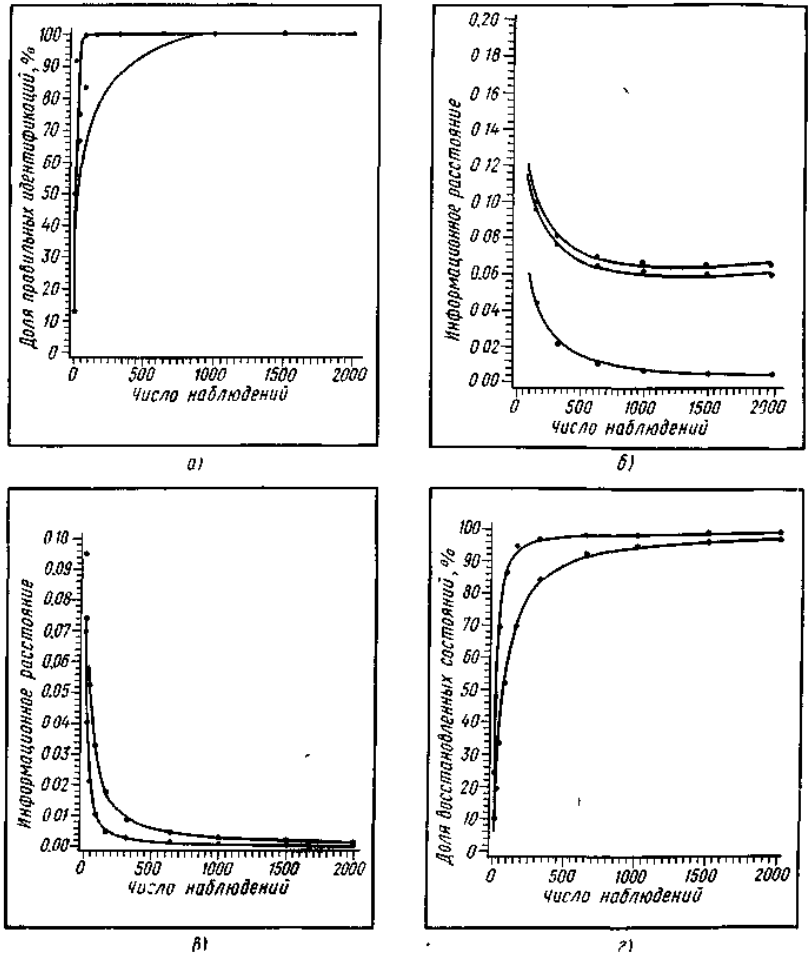


Рис. 37. Некоторые характеристики анализа реконструируемости для вероятностных систем

Они получены с использованием процедуры поиска, в которой уточняются только структуры с минимальным расстоянием.

Графики на рис. 37,а описывают влияние числа наблюдений (объем данных $|d|$) на качество анализа реконструируемости для переменных с двумя и с пятью состояниями. Качество оценивается как для тех экспериментов, для которых процедура поиска получает на соответствующем уровне уточнения корректную структуру, являющуюся структурой с наименьшим расстоянием. Как видно, 100%-ое качество достигается достаточно быстро для обоих случаев. Несмотря на то, что с ростом числа наблюдений результативность сходится к 100% во всех исследованных случаях, скорость сходимости несколько падает с ростом числа переменных. Это объясняется прежде всего высокой селективностью используемой процедуры поиска. Видно также, что переменные с пятью состояниями (верхний график) оцениваются лучше, чем переменные с двумя состояниями (нижний график). Это также общая тенденция: с ростом мощностей множеств рассматриваемых состояний улучшается и качество. Таким образом, **для любого конкретного числа переменных характеристики представленных систем с двоичными переменными могут рассматриваться как худший случай.**
На остальных графиках на рис. 37 показаны характеристики анализа реконструируемости только для двоичных переменных. На графике (рис. 37,б) показано, насколько отличается корректная структура по информационному расстоянию от других структур на том же уровне уточнения. На нижнем графике представлено расстояние для корректной структуры $D(^{D}f, {}^{R}f)$, на среднем — наименьшие расстояния для структур, конкурирующих с корректной на том же уровне уточнения, и на верхнем средние расстояния для всех структур, конкурирующих с корректной (в соответствии с процедурой поиска) на одном уровне уточнения. Несмотря на то, что на вид этих кривых влияет число переменных и мощность множеств их состояний, а также используемый тип расстояния, с ростом числа наблюдений эти расстояния всегда убывают, а расстояние для корректной структуры стремится к нулю.
На графике (рис. 37,е) сравниваются информационные расстояния между подлинной системой ${}^{T}\mathbf{F}$ и соответственно системами ${}^{D}\mathbf{F}$ и ${}^{R}\mathbf{F}$. Поскольку соответствующие пары распределений вероятностей ${}^{T}\mathbf{f}$, ${}^{D}\mathbf{f}$ и ${}^{T}\mathbf{f}$, ${}^{R}\mathbf{f}$ являются произвольными, необходима общая мера информационного расстояния. Это расстояние, назовем его G, определяется формулой

$$G({}^{1}\mathbf{f}, {}^{2}\mathbf{f}) = D\left({}^{1}\mathbf{f}, \frac{{}^{1}\mathbf{f} + {}^{2}\mathbf{f}}{2}\right) + D\left({}^{2}\mathbf{f}, \frac{{}^{1}\mathbf{f} + {}^{2}\mathbf{f}}{2}\right), \quad (44)$$

где ${}^1\mathbf{f}$ и ${}^2\mathbf{f}$ — произвольные распределения вероятностей, определенные на одном и том же конечном множестве состояний; D — специальное информационное расстояние, заданное уравнением (40); $({}^1\mathbf{f} + {}^2\mathbf{f})/2$— распределение вероятностей, полученное взятием среднего для каждой пары соответствующих вероятностей из ${}^1\mathbf{f}$ и ${}^2\mathbf{f}$. Нижний график на рис. 37,в представляет $D({}^T f, {}^R f)$, а верхний — $D({}^T f, {}^D f)$. Таким образом, реконструированная система ${}^R\mathbf{F}$ оказывается ближе к подлинной системе ${}^T\mathbf{F}$, чем система ${}^D\mathbf{F}$, опирающаяся только на доступные данные. Это довольно неожиданный результат, важность которого будет проанализирована в разд. 5.10.

На графиках (рис. 37,г) показана взаимосвязь множеств состояний с ненулевыми вероятностями для трех участвующих в вычислительном эксперименте систем с поведением ${}^T\mathbf{F}$, ${}^D\mathbf{F}$ и ${}^R\mathbf{F}$; будем эти множества состояний обозначать соответственно ${}^T X$, ${}^D X$ и ${}^R X$. На нижнем графике показана доля тех состояний ${}^T\mathbf{F}$, которые имеются в ${}^D\mathbf{F}$ (это происходит из-за недостатка данных), т. е. $({}^D X/{}^T X) \cdot 100$; на верхнем графике представлен процент состояний ${}^T\mathbf{F}$, имеющихся в

$$\frac{{}^R X}{{}^T X} \cdot 100.$$

Из этих графиков ясно видно, что

$$ {}^D X \subseteq {}^R X \subseteq {}^T X. \qquad (45)$$

Это столь же важное свойство, как и то, что получено из графиков на рис. 37,г, и оно также будет рассмотрено в разд. 5.10.

Возможностные аналоги описанных характеристик приведены на рис. 38.

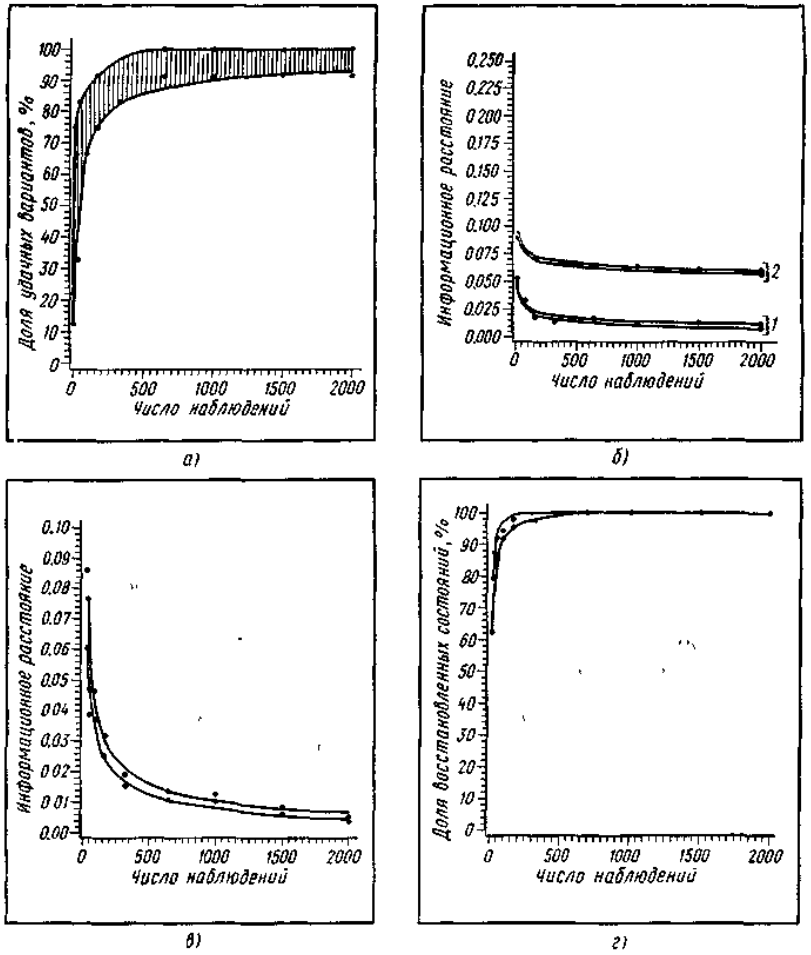


Рис. 38, Некоторые характеристики анализа реконструируемости для возможностных систем

Они основаны на IOD-кластеризации. Поскольку реконструктивный анализ возможностных систем ИИ основывается на работе с кластерами структур, а не с отдельными структурами, то соответствие между вероятностными характеристиками и их возможностными аналогами не является прямым.

На рис. 38 (а) показано качество возможностного анализа реконструируемости для различных множеств состояний (от двух до

пяти для переменной). График в данном случае имеет обобщенный вид, поскольку различия для разных множеств состояний малы и никаких особых тенденций выявить не удается. Качество представляется как доля экспериментов, в которых корректные структуры входят в кластер хороших структур. Остальные зависимости на рис. 38 построены только для двоичных переменных.

На рис. 38,г показаны верхнее и нижнее информационные расстояния для двух кластеров структур. Они существенно отличаются от своих вероятностных аналогов. Напротив, на рис. 38,в и *г* очень похожи на свои вероятностные аналоги. Возможностная версия общего информационного расстояния, использованная при построении графиков на этих рисунках, определяется формулой

$$G(^1\mathbf{f},\ ^2\mathbf{f}) = \mathbf{D}(^1\mathbf{f},\ ^1\mathbf{f} \vee ^2\mathbf{f}) + D(^2\mathbf{f},\ ^1\mathbf{f} \vee ^2\mathbf{f}), \quad (46)$$

где $^1\mathbf{f}$ и $^2\mathbf{f}$ — произвольные распределения возможностей, определенные на одном и том же конечном множестве состояний; D — специальное информационное расстояние, определяемое уравнением (42); $^1\mathbf{f} \vee ^2\mathbf{f}$ — распределение возможностей, получаемое взятием максимума для каждой пары соответствующих возможностей из $^1\mathbf{f}$ и $^2\mathbf{f}$. Все эксперименты, для которых определялись данные характеристики, проводились в предположении, что данные порождены некой структурированной системой. Целью этих экспериментов было определение того, насколько недостаток данных влияет на качество анализа реконструируемости. Несмотря на то, что эти идеализированные эксперименты достаточно ценны и представляют собой естественный первый этап анализа реконструируемости, очень желательно было бы расширить их для более общих и более реальных ситуаций.

Вычислительные эксперименты так, как они описаны в этом разделе для анализа реконструируемости, являются важнейшим инструментом для метаметодологических исследований в науке о системах ИИ. ФРИЗ должен не только предоставлять в распоряжение пользователя метод решения разных типов интеллектуальных задач, но и давать метаметодологическое описание этих методов. Характеристики анализа реконструируемости, описанные в данном разделе,— это простой пример такого метаметодологического описания.

5.10. Индуктивное рассуждение

Хотя индуктивное рассуждение неявно применялось только при решении задач идентификации и реконструкции, оно используется практически в любой задаче, связанной с применением метода от-

крытия в исследовании систем. Поэтому желательно дать обзор вопросов, связанных с индуктивными рассуждениями, и это—одна из задач данного раздела; вторая задача — введение нового принципа индуктивного вывода, основанного на некоторых особенностях анализа реконструируемости.

Понятие **индуктивного рассуждения** мы будем рассматривать не в традиционном узком смысле — **как выведение общего заключения из частных примеров**, а более широко — **как «охват всех недоказуемых случаев, в которых истинность предпосылок хотя и не определяет инстинность заключения, но показывает, что есть серьезные основания доверять истинности этого заключения»** (Философская энциклопедия).

Понятие индуктивного рассуждения в течение столетий являлось предметом философских споров, особенно после опубликования в 1739 г. Дейвидом Хьюмом классического анализа этого понятия *(David Hume, A. Treatise of Human Nature, William Collins, Glasgow,* 1962). Несмотря на то, что было выдвинуто множество доказательств, опровергающих скептическое отношение Хьюма к возможности обоснования **индуктивного вывода**, в конечном счете оказалось, что все эти доказательства имеют слабые места. В одних случаях доказательства оказывались несостоятельными из-за скрытой цикличности (т. е. **из-за попыток обоснования индукции с помощью индукции**), в других —зависели от неких метафизических допущений (типа единообразия природы).

Основной недостаток этих доказательств заключается в том, что они всерьез рассматривают тезис Хьюма как содержательную проблему. ***Альтернативный подход состоит в том, чтобы отказаться от попыток обоснования индуктивного вывода дедуктивными методами, что неявно присутствует в анализе Хьюма, и рассматривать индуктивное рассуждение как процесс правильного суждения на основе неполной информации.*** Одним из самых многообещающих способов обоснования индуктивного рассуждения при таком подходе является ***методологический прагматизм***, предложенный Решером. Согласно этому способу **ключом к обоснованию индуктивного рассуждения являются методы**; методологический прагматизм хорошо согласуется с темой нашей работы. **Дадим краткий обзор основных особенностей этого способа**.

Решер рассматривает индуктивное рассуждение как **метод «правильного суждения с помощью систематизации на основе опыта, представляющего собой оптимально правдоподобное сочетание догадки и имеющейся информации»**. Он утверждает, что **информацию о природе можно получить, только взаимодействуя с**

ней, и если мы не готовы довериться такому взаимодействию, то остается только отказаться от всего проекта исследования. Несмотря на то, что цели такого проекта являются как научными, так и практическими, окончательная проверка должна основываться на том, насколько полученные знания оказываются полезны для деятельности человека. Происходит это из-за того, **что в отличие от теоретических построений на практике для достижения целей (например, чтобы избежать гибели, травмы, болезни, страданий, огорчений и т. д.) нужно принимать неотложные решения.**

Поскольку **индуктивное рассуждение рассматривается как *метод решения задач при неполной информации,*** его обоснование — это обоснование с точки зрения практический пригодности; отсюда и название «методологический прагматизм».

Обоснование метода индуктивных рассуждений, по Решеру, разбивается **на два этапа, называемые соответственно начальным и окончательным обоснованием**. При ***начальном обосновании,*** которое является неиндуктивным, требуется показать, что метод является потенциально более пригодным по сравнению с другими сопоставимыми методами. ***Окончательное обоснование*** — это проверка **реальной эффективности метода**, т. е. проверка того, насколько его эффективность выше, чем у альтернативных методов.

Существенной стороной начального обоснования метода индуктивного распределения, по Решеру, является учет того, в какой степени полученные с помощью этого метода результаты могут быть включены в систему, основанную на предшествующем опыте. Все остальное остается таким же. **В качестве критериев для начального обоснования метода используются различные параметры *систематичности* (полнота, связанность, совместимость, простота и т. д.).** Такой подход является следствием согласованной теории истинности, подробно излагаемой в книге Решера «Когнитивная систематизация».

Решер утверждает также, что **индуктивное рассуждение «лежит в самой основе использования человеком языка для коммуникаций. Именно естественный язык служит для согласования явного расхождения между полученными утверждениями и реальными свидетельствами; это делается с помощью завуалирования фактов, причем этот процесс имеет индуктивную природу»**.

Взаимосвязанность индуктивного рассуждения и естественного языка, хорошо согласующаяся с основными идеями методологического прагматизма, отмечается и в работах некоторых современных философов. Одно из самых ясных высказываний на этот счет принадлежит Блэку. Он утверждает:

Мне представляется естественным считать, что **индукция—это система, управляемая правилами**. **Индукция как система человеческой деятельности использует соответствующую терминологию, а также определенные правила вывода суждений**. **При индуктивном рассуждении требуется соответствующим образом помечать определенные ситуации, определенными способами делать выводы и, что важно, принять определенный способ мышления, прежде чем предпринять соответствующие действия**... Грубо говоря, **индуктивные правила определяют: как говорить, как думать и как действовать в определенных рамках**... У этих правил есть **априорная и есть практическая стороны**, определяющие правильное использование этих правил; **при нашем языке и включающей его системе понятий невозможно представить себе, как можно вообще отказаться от них.** Однако это определяется опытом. Большая часть этого опыта воплощена в структуре нашего языка, не означает, что мы находимся в плену у догмы-устанавливающие правила индукции дают значительную свободу для различных суждений относительно индуктивных заключений, достоверности правил и т. п. При этом можно обратиться к прошлому опыту как основе для повышения мощности вывода или при разумных ограничениях для модификации системы индукции и ее компонентов. Однако обращение к предшествующему опыту — это, пользуясь политической терминологией, ревизионизм: для революций в способах мышления нужно искать другие пути.»

Индуктивное рассуждение хорошо формулируется на языке теории информации. Большой вклад в эту область внесли работы Кристенсена. Он определил *индуцированное суждение* как суждение, представляющее всю доступную информацию, но только ее. Несмотря на то, что индуцированные суждения получаются не обязательно с помощью дедукции из имеющейся информации, случаи, в которых применима дедукция, также рассматриваются как особые (экстремальные) случаи индуцированных суждений. **Из определения индуцированного суждения следуют два общих принципа индуктивного рассуждения:**

1) наши суждения не должны быть отражением большего объема информации, чем тот, которым мы располагаем;
2) наши суждения должны отражать всю доступную нам информацию.

Для создания определенной методологии индуктивного рассуждения, основанного на этих общих принципах, нужно принять какое-то определение **термина «информация».** В разных контекстах этот термин понимается по-разному. В контексте исследования систем,

когда **информация понимается как мера ограничений на рассматриваемые переменные**, **смысл этого термина определяется способом представления этих ограничений.** В рамках теории вероятностей два общих принципа индуктивного рассуждения становятся соответственно принципами минимума и максимума энтропии. В альтернативных теориях есть соответствующие аналоги этих принципов. Например, в теории возможностей это принципы максимума и минимума U-нечеткости.

Принцип максимума энтропии используется для оценки неизвестных вероятностей (которые нельзя определить дедуктивно) по имеющейся информации. Согласно этому принципу оцененное распределение вероятностей должно быть таким, чтобы его энтропия была максимальной при имеющихся ограничениях, т. е. ограничениях, представляющих имеющуюся информацию. Таким образом, этот принцип обеспечивает то, что не будет использована никакая информация, кроме имеющейся.

Принцип минимума энтропии применяется при формировании разрешающих форм и в связанных с этим процессом задачах. Согласно этому принципу энтропия оцениваемого распределения вероятностей, обусловленного определенной классификацией заданных событий (например, состояний рассматриваемой переменной), является минимальной основой для ограничений, накладываемых на ситуацию. Таким образом, этот принцип гарантирует, что при оценке неизвестных вероятностей используется вся имеющаяся информация, насколько это возможно при заданных ограничениях (например, при требуемом числе состояний) и по существу согласно Ватанабе, это является основным принципом распознавания образов:

Распознавание образов—это способность интеллекта, направленная на то, чтобы в ряде событий выделить некую «форму». На языке математики самым подходящим было бы такое определение: распознавание образов — это формирование, переформирование и модификация нашей системы отсчета с тем, чтобы соответствующим образом минимизировать с учетом неизбежных ограничений определенную для этой системы отсчета энтропию.

Соответствующее применение обоих методологических принципов — максимума и минимума энтропии и дает методологию индуктивного рассуждения по Кристенсену. Она называется *методологией минимакса энтропии.* Для обоснования этой методологии Кристенсен анализирует грамматические структуры и морфемы естественного языка. Он утверждает следующее:

«Так же как в физике, где измерения зависят от физической системы отсчета, **наши индуктивные суждения зависят от системы понятий, выработанной нашим опытом.** Большая часть этого опыта **воплощена в структуре нашего языка, в смысле используемых нами слов, фраз, предложений**. Искать твердый «принцип индукции», на котором основывались бы наши обобщения, столь же тщетно, как искать окончательную систему отсчета для ньютоновской механики... Любое сделанное человеком обобщение зависит не только от конкретных данных, непосредственно из которых оно получено, но и (только косвенно) от всего опыта истории эволюции языка, на котором получено это обобщение. В конечном счете **все суждения людей используются при принятии решений того или иного рода или при выборе последовательности действий. Решение зависит от оценок и суждений лица, принимающего решение, причем эти оценки относятся и к природе окружающего его мира. На систему ценностей и убеждения влияет как внешний мир, так и сама личность ..**

Предположим, что индуктивное рассуждение широко используется всеми членами некоего общества, говорящими на одном языке. Тогда, приняв первую часть определения индуцированного суждения, а именно, что оно представляет не больше информации, чем нам доступно, мы видим справедливость термодинамического закона неубывания энтропии. Из этого закона следует, что будущий опыт стремится быть представленным так же просто на существующем в настоящее время языке, как и опыт прошлый.

Принимая вторую часть определения индуцированного суждения, а именно, что оно представляет всю доступную нам информацию, и считая, что индуктивное рассуждение используется всем обществом, мы приходим к ***принципу эволюции языка: [Язык эволюционирует в направлении, ведущем к более простому описанию опыта всех членов общества, использующих этот язык].*** **Из чего следует, что прошлый опыт может быть легко выражен на языке современности**. Однако это также означает, что индуктивное рассуждение порождает представление физической действительности. Таким образом, мы показали условную обоснованность индуктивного рассуждения, условную, поскольку она зависит от применения индуктивного рассуждения в обществе. Другими словами, обоснованность индуктивного рассуждения зависит от того, поддерживается оно «живым» языком или нет. Однако, и это очень важно, такая условность находится под контролем людей, принимающих решение о том, доверять или нет этим индуцированным суждениям. Основывая свои решения на результатах индуктивных рассуждений, они оценивают

все, от чего зависит обоснование индуктивного рассуждения. В этом смысле принятие решений на основе индуктивного рассуждения, реализуемое в «живом» языке, представляет собой процесс, обосновывающий сам себя».

Рассматривая все доводы в обоснование метода минимакса энтропии как методологии индуктивного рассуждения, можно прийти к выводу, что эта методология обоснована достаточно хорошо. Будут, вероятно, разработаны и соответствующим образом обоснованы аналоги этой методологии для других классов нечетких множеств (например, методология минимакса U-нечеткости).

После этого краткого обзора некоторых важнейших вопросов индуктивного рассуждения опишем еще один принцип индуктивного вывода. **Поскольку этот принцип связан с задачей реконструкции, назовем его *реконструктивным принципом индуктивного вывода*.**

Предположим, что некое ограничение для заданной обобщенной системы ИИ с поведением было определено по эмпирическим данным с помощью индуктивных рассуждений, скажем с помощью методологии минимакса энтропии. Поскольку количество данных ограничено, это ограничение представляет собой только оценку того, как рассматриваемые переменные ограничены в действительности; если эта оценка основана на всей информации относительно действительного ограничения, содержащейся в базе данных, то она является **несмещенной**.

Теперь допустим, что действительное ограничение таково, что оно может быть реконструировано по определенному набору своих проекций. Из-за ограниченности данных оцененное ограничение может этим свойством не обладать. Тем не менее реконструктивная гипотеза, основанная на подсистемах, будет, вероятно, лучше подходить для оценки обобщенного ограничения, чем другие гипотезы на том же уровне решетки уточнения. Затем это превосходство будет проявляться на любом более низком уровне этой решетки при всех уточнениях такой удачной реконструктивной гипотезы. Теперь можно привести решающий довод. **Если корректная реконструктивная гипотеза и/или ее уточнения на различных уровнях уточнения в самом деле определяются как лучшие, то с помощью любой из них потенциально возможно реконструировать некоторые обобщенные состояния, которые переменные могут иметь, но которые не входят в имеющиеся данные и, следовательно, в ограничения для заданной обобщенной системы.** Это хорошо согласуется с масштабными экспериментами, результаты которых приведены на рис. 37,г и 38,г. Более того, поскольку с любой подсистемой ИИ связано меньшее множество состояний обобщенной системы ИИ, то в общем

случае его ограничения лучше описываются данными, чем ограничения обобщенной системы ИИ (например, больше отношение числа наблюдений к числу потенциальных состояний). То есть с помощью лучших реконструктивных гипотез можно улучшить нашу исходную оценку обобщенного ограничения. Это также подтверждается результатами экспериментов, приведенными на рис. 37,в и 38,в. Полученные результаты ясно указывают на то, что неравенство

$$G({}^{T}\mathbf{f},\ {}^{R}\mathbf{f}) < G({}^{T}\mathbf{f},\ {}^{D}\mathbf{f})$$

для конечных данных выполняется всегда независимо от объема данных. **Это значит, что обобщенная система, реконструированная на основе корректной реконструктивной гипотезы, информационно всегда ближе к истинной обобщенной системе, чем система, выведенная только из заданных данных**. Свидетельством того, что реконструированное ограничение ${}^{R}\mathbf{f}$ оценивает истинное ограничение ${}^{T}\mathbf{f}$, лучшее, чем полученное из данных (${}^{D}\mathbf{f}$), является неравенство

$$\delta_1({}^{T}\mathbf{f},\ {}^{R}\mathbf{f}) < \delta_1({}^{T}\mathbf{f},\ {}^{D}\mathbf{f}),$$

где δ_1 — расстояние Хемминга. Выполнение этого неравенства подтверждается также результатами вычислительных экспериментов. Так, например, результаты реальных экспериментов для трех переменных с пятью состояниями каждая приведена в табл. 15 соответственно для вероятностной и возможностной систем.

Таблица 15.

Пример экспериментального подтверждения реконструктивного принципа индуктивного вывода

$\lvert d\rvert$	10	20	40	80	160	320	640	1,000	1,500	2,000
					а) Вероятностная система					
$\delta_1({}^{T}\mathbf{f},\ {}^{D}\mathbf{f})$	0.0132	0.0111	0 0086	0.0063	0.0044	0.0031	0.0023	0.0018	0.0015	0.0012
$\delta_1({}^{T}\mathbf{f},\ {}^{R}\mathbf{f})$	0.0106	0.0073	0.0052	0.0036	0.0025	0.0018	0.0014	0.0011	0.0009	0.0008
$G({}^{T}\mathbf{f},\ {}^{D}\mathbf{f})$	0.0952	0.0741	0.0526	0.0328	0 0174	0.0084	0.0042	0.0025	0.0017	0.0012
$G({}^{T}\mathbf{f},\ {}^{R}\mathbf{f})$	0.0697	0.0404	0.0213	0.0104	0.0045	0.0024	0.0013	0.0008	0.0005	0.0004
					б) Возможностная система					
$\delta_1({}^{T}\mathbf{f},\ {}^{D}\mathbf{f})$	0.2779	0.2479	0.2188	0.1859	0.1508	0.1100	0.0883	0.0746	0.0627	0.0565
$\delta_1({}^{T}\mathbf{f},\ {}^{R}\mathbf{f})$	0.2645	0.1999	0.1470	0.1216	0.1021	0.0764	0.0648	0.0544	0.0457	0.0403
$G({}^{T}\mathbf{f},\ {}^{D}\mathbf{f})$	0.0999	0.0894	0.1002	0.0872	0.0776	0.0599	0.0526	0.0450	0.0379	0.0369
$G({}^{T}\mathbf{f},\ {}^{R}\mathbf{f})$	0.0972	0.0742	0.0691	0.0647	0.0568	0.0450	0.0383	0.0334	0.0260	0.0261

Для сравнения в этих таблицах приведены результаты, полученные с помощью информационного расстояния.

Будем ли мы получать реконструкцию с помощью лучшей реконструктивной гипотезы на некотором уровне уточнения как улучшенную оценку обобщенного ограничения, зависит от того, влияет ли, по нашему мнению, рассматриваемая гипотеза на некоторые фундаментальные реконструктивные свойства переменных или нет. Что может помочь исследователю принять то или иное решение? Можно предложить следующее: с помощью вычислительных экспериментов на компьютере, как это описано в разд. 5.9, исследователю следует определить некоторые реконструктивные характеристики. Эти характеристики позволят исследователю оценить конкретную ситуацию и сформировать соответствующее мнение. Такие характеристики могут быть использованы даже в соответствующих руководствах по формированию этого мнения, и, в конечном счете, могут быть разработаны некие обоснованные функции формирования мнения. Очевидно, что эти функции должны быть определены реконструктивными характеристиками типа изображенных на рис.37,а, *б* или 38,а, *б.*

Умение решать задачу реконструкции позволяет нетрадиционно подойти к процессу индуктивного рассуждения. Он осуществляется в <u>два этапа</u>. <u>На первом</u> обобщенное ограничение определяется по имеющимся данным с помощью обычных принципов индуктивного рассуждения (например, с помощью принципа минимакса энтропии).

<u>Второй этап</u> состоит из трех шагов:

1) на различных уровнях уточнения определяются лучшие реконструктивные гипотезы для обобщенной системы;
2) формируются мнения о том, насколько эти гипотезы влияют на реальные реконструктивные свойства рассматриваемых переменных; эти мнения формируются на основе соответствующих экспериментальных характеристик, руководств и конкретных функций;
3) заданное обобщенное ограничение дополняется (или заменяется) ограничениями, реконструированными с помощью лучших реконструктивных гипотез, причем с каждой связывается определенная степень доверия.

При использовании только той информации, что содержится в данных, этот двухэтапный метод позволяет включать в оцененное обобщенное ограничение некоторые характеристики (например, обобщенные состояния), которые нельзя непосредственно определить по имеющимся данным. Следовательно, такой метод позволяет нам предсказывать или восстанавливать с определенной степенью

достоверности некоторые состояния исследуемых переменных, которые не входят в момент предсказания или восстановления в данные, которыми мы располагаем.

5.11. Несогласованные структурированные системы ИИ

Можно предположить, что **<u>согласованность</u> является самым <u>фундаментальным критерием</u> для классификации структурированных систем ИИ.** Для структурированных систем ИИ согласованность тесно связана с задачей идентификации (разд. 5.6):

<u>если структурированная система ИИ с поведением согласованная, то ее реконструктивное семейство непусто, если несогласованная, то пусто</u>

В структурированных системах ИИ с поведением возможна несогласованность двух типов — **локальная и глобальная**. Структурированная система ИИ *локально несогласованная,* если она не удовлетворяет требованиям локальной согласованности, задаваемым уравнением (20), и *глобально несогласованная,* если она локально несогласована и ее реконструктивное семейство пусто.

Пример 22. Рассмотрим структурированную систему с поведением, элементы которой описываются вероятностными функциями поведения, приведенными в табл. 16а.

Таблица 16.

Пример несогласованных структурированных систем с поведением

а) Локально несогласованная структурированная система

	v_1	v_2	${}^1f({}^1\mathbf{c})$
${}^1\mathbf{c}=$	0	0	0.5
	0	1	0.2
	1	0	0.1
	1	1	0.2

	v_2	v_3	${}^1f({}^2\mathbf{c})$
${}^2\mathbf{c}=$	0	0	0.4
	0	1	0.25
	1	0	0.15
	1	1	0.2

б) Глобально несогласованная структурированная система

	v_1	v_2	${}^1f({}^1\mathbf{c})$
${}^1\mathbf{c}=$	0	1	0.7
	1	0	0.3

	v_2	v_3	${}^1f({}^2\mathbf{c})$
${}^2\mathbf{c}=$	0	1	0.3
	1	0	0.7

	v_1	v_3	${}^3f({}^3\mathbf{c})$
${}^3\mathbf{c}=$	0	0	0.4
	0	1	0.3
	1	0	0.3

Вычислив проекции этих функций относительно соединяющей переменной v_2. имеем

α	$v_2[{}^1f \downarrow \{v_2\}](\alpha)$
0	0 6
1	0,4

α	$v_2[{}^2f \downarrow \{v_2\}](\alpha)$
0	0,55
1	0,45

Эта система локально не согласована, так как

$$[{}^1f \downarrow \{v_2\}](\alpha) \neq [{}^2f \downarrow \{v_2\}](\alpha).$$

Пример 23. Понятно, что структурированная система с поведением, вероятностные функции поведения которой приведены в табл. 16 б, локально согласованная. Однако несложная проверка уравнений, определяющих реконструктивное семейство, показывает, что реконструктивное семейство является пустым. Например, требуется, чтобы вероятности состояний 001 и 011 согласно функциям ${}^1f$ / и ${}^2f$ были равны 0, и в то же время требуется, чтобы их сумма согласно функции ${}^3f$ была равна 0.3. Таким образом, эта система глобально несогласованная.

Локальные несогласованности в структурированных системах обычно возникают из-за того, что соответствующие их элементам функции поведения являются всего лишь оценками, полученными по ограниченным экспериментальным данным. Из таких несогласованностей (в отличие от глобальных) не следует, что сами исследуемые переменные являются несогласованными. Это просто отражение того факта, что информация о каждом подмножестве входящих в структурированную систему переменных неполная. **Эта неполнота (т. е. наше незнание) и порождает локальные несогласованности,** и, следовательно, можно и нужно работать с локально несогласованными структурированными системами.

Иначе обстоит дело с глобальной несогласованностью. **Глобальная несогласованность означает, что структурированная система плохо задумана; это математическая конструкция, не имеющая реального смысла.** Примеры глобальной несогласованности целого при локальной согласованности можно найти в живописи. Например, это некоторые рисунки М. Эшера (скажем, его литография «Бельведер») и многие рисунки шведского художника О. Ретерсвярда, которые называют невозможными рисунками или японской перспективой.

Понятно, что **к локально несогласованным структурированным системам неприменимы обычные логические процедуры**.

Относиться к ним можно двояко: либо вообще отбросить такие структурированные системы, поскольку они не представляют никакую обобщенную систему, либо разрешить эти локальные несогласованности с помощью такой модификации заданных **функций поведения**, чтобы новые функции поведения были согласованными и в некотором смысле как можно более близки к исходным. **Обычно близость определяется через информационное расстояние.** Полученная в результате структурированная система ИИ используется затем вместо исходной.

Пусть дана локально несогласованная структурированная система ИИ с поведением

$$\mathbf{SF} = \{({}^xS, {}^x\mathbf{F}) \mid x \in N_q\}$$

с вероятностными функциями поведения ${}^xf\ (x \in N_q)$. Тогда задачу разрешения несогласованностей можно сформулировать следующим образом.

Требуется определить функции поведения ${}^xf_c$ того же вида, что и функции ${}^xf\ (x \in N_q)$, так, чтобы функция

$$\sum_{x \in N_q} D({}^xf,\ {}^xf_c) \quad \textbf{(47)}$$

достигала минимума при ограничениях

$$[{}^xf_c \downarrow {}^xS \cap {}^yS] = [{}^yf_c \downarrow {}^xS \cap {}^yS] \quad (48)$$

для всех $x, y \in N_q$ и

$${}^xf({}^x\mathbf{c}) \neq 0 \Rightarrow {}^xf_c({}^x\mathbf{c}) \neq 0 \quad (49)$$

для всех состояний ${}^x\mathbf{c}\ (x \in N_q)$. Назовем ее *задачей оптимального разрешения локальных несогласованностей.*

Согласно уравнению (48) полученная структурированная система ИИ должна быть локально согласованная. Из утверждений (49) следует, что любое состояние, возможное при исходной формулировке, не будет отброшено при модифицированной локально согласованной формулировке; это требование делает возможным использование простого информационного расстояния D (см. уравнение (40) или (42) для функции (47).

Пример 24. Рассмотрим локально несогласованную структурированную систему с поведением, состоящую из двух элементов, чьи функции поведения приведены в табл. 16а. Решение задачи оптимального разрешения локальных несогласованностей для этих функций поведения дают функции поведения ${}^1f_c$ и ${}^2f_c$, приведенные в табл. 17.

Таблица 17.

Решение задачи оптимального разрешения локальных несогласованностей для функций поведения, приведенных в табл. 16а

v_1	v_2	${}^1f_c({}^1c)$	v_2	v_3	${}^2f_c({}^2c)$
${}^1c=0$ 0	0	0.5208	${}^2c=0$ 0	0	0.3846
0	1	0.1875	0	1	0.2404
1	0	0.1042	1	0	0.1732
1	1	0.1875	1	1	0.2018

Легко убедиться в том, что эти функции локально согласованы и что

$$D({}^1f, {}^1f_c)+D({}^2f, {}^2f_c)=0.0245.$$

ЗАМЕЧАНИЯ

1. В пользу принципа максимума энтропии можно привести по крайней мере три разных довода.

1. Распределение вероятностей с максимумом энтропии является единственным *несмещенным распределением,* т. е. **распределением, учитывающим всю имеющуюся информацию, и только ее**. Это непосредственно следуют из того, что вся имеющаяся информация (и только она) необходима для формирования ограничений для задачи оптимизации, и от выбранного распределения вероятностей требуется, чтобы оно представляло максимум нечеткости (энтропии) при выполнении множества ограничений, которым должны удовлетворять распределения вероятностей. В самом деле, **любому сокращению нечеткости соответствует равное увеличение информации**. Следовательно, **при выборе любого распределения вроятностей, за исключением распределения с максимальной энтропией, имеет место сокращение нечеткости по сравнению с ее максимальным значением, значит, неявно добавляется некая дополнительная информация.**

2. Из комбинаторных соображений было показано, что **распределение вероятностей с максимальной энтропией является *наиболее правдоподобным.*** Для данной реконструктивной гипотезы любой элемент реконструктивного семейства этой гипотезы может быть порожден по некоторому числу наборов реальных данных. Наибольшее число возможных наборов данных, являющихся взаимно сравнимыми и совместимыми с данной реконструктивной гипотезой,

требуется для обобщенного распределения вероятностей с максимальной энтропией.

3. Дж. Шор и Р. Джонсон показали, что **принцип максимума энтропии дедуктивно *выводим* из следующих *аксиом непротиворечивости* для индуктивного рассуждения:**

единственность: результат должен быть единственным;

инвариантность: результат не должен зависеть от выбора системы координат (от перестановки переменных);

системная независимость: не должно иметь значения то, как учитывается информация — по отдельности для независимых систем через маргинальные вероятности или совместно через совместные вероятности;

независимость подмножества: не должно иметь значения, исследуется независимое подмножество состояний системы на основе отдельных условных вероятностей или на основе полных системных вероятностей.

Шор и Джонсон следующим образом обосновывают выбор этих **аксиом**. **Метод вывода должен быть таким, чтобы результаты, полученные исходя из одной и той же информации, но разными способами, были непротиворечивы**. На основе этих аксиом они выдвинули следующее предположение: **для заданной информации, представляющей собой ограничения на оцениваемые вероятности, существует только одно распределение вероятностей, удовлетворяющее этим ограничениям, которое может быть определено методом, не нарушающим аксиом непротиворечивости; это единственное распределение может быть получено максимизацией энтропии (или любой другой функции имеющей те же максимумы, что и функция энтропии) при соблюдении заданных ограничений.**

Кроме этих классических доводов в пользу принципа максимума энтропии существует и довод, который основан на **свойствах искусственных структурированных систем.** Как утверждалось в конце разд. 5.8, **любой искусственной структурированной системе соответствует единственная обобщенная система, та, что представлена своей несмещенной реконструкцией.** То есть, если дана вероятностная структурированная система и известно, что эта система искусственная, то единственно возможной реконструкцией является реконструкция с максимальной энтропией. Или, другими словами, **невозможно создать реальную вероятностную структурированную систему, действительная реконструкция которой отличается от реконструкции с максимальной энтропией.** При этом предполагается, что система спроектирована для того, чтобы

функционировать в любой среде. Однако для **систем, спроектированных для функционирования в некой определенной среде, этот аргумент также верен. Только в этом случае процедура соединения охватывает не только элементы проектируемой структурированной системы, но и ее среду.**

Для возможностных систем аналогом принципа, максимума энтропии является принцип максимума U-нечеткости. Несмотря на то, что пока не показана его выводимость из соответствующих аксиом непротиворечивости, возможные аналоги других аргументов обоснованы. Известно, в частности, что возможностная процедура соединения дает максимальную нечеткость, т. е. несмещенную реконструкцию.

2. В математике G-структуры называются несократимыми гиперграфами. *Гипергpаф* определяется как семейство подмножеств заданного множества (скажем, множества N_n), удовлетворяющее требованию покрытия и не содержащее пустого множества.

3. Впервые общая задача реконструкции была поставлена У. Росс Эшби. Он ввел понятие **цилиндричности для многомерных отношений (в нашей терминологии это четкие возможностные системы)** и предложил алгоритм определения того, можно ли реконструировать данное отношение размерности *n* по его проекциям определенной размерности *(k<n)*. Если возможна такая реконструкция, то она получается пересечением расширений (цилиндров) всех *k* -мерных проекций данного отношения Легко показать, что процедура Эшби является частным случаем процедуры соединения. У. Росс Эшби (совместно с Р. Мадденом) первым сформулировал также общую задачу идентификации, описанную в разд. 5. 6. Главное, Эшби оценил всю **важность задач реконструкции и идентификации для исследования систем** До него таким задачам не придавалось особого значения. Его работы, с методологической точки зрения узкоспециальные, явились толчком для многих исследований в этой области. Первые идеи относительно более общего подхода к задаче реконструкции появились у в 70-х гг .

4. Вопрос о влиянии степени квантификации данных на результаты анализа реконструируемости вероятностных систем исследовался радом авторов, которые пришли к следующим выводам.

Если говорить об исследовании системы, описываемой вероятностными характеристиками, то несмотря на улучшение результатов при квантификации данных, в конечном счете с относительным ростом объема исходной информации это улучшение становится все менее заметным при *Q* (**числе состояний переменной**),

большем 4 или 5. С другой стороны, с ростом квантификации данных растет объем памяти ЭВМ и время счета необходимые для анализа реконструируемости. Таким образом, можно утверждать, что **тернарная квантификация лучше бинарной**, однако при Q, большем 3 или 4, резко увеличивается объем работы ЭВМ, а результаты улучшаются лишь незначительно.

Эти выводы, полученные в результате математического анализа, практически совпадают с выводами, полученными на основе вычислительных экспериментов, описанных в разд. 5.9.

6. Метасистемы ИИ

6.1. Изменение и инвариантность

Одной из основных **способностей человека**, и возможно самой существенной, является **способность распознавать отличия**. Отличие разединяет мир надвое: на «это» и «то», «среду» и «систему», на «мы» и «они» и т. д. В человеческой деятельности различение занимает одно из самых важных мест и является одним из самых важных действий в науке о системах ИИ, поскольку любое **определение системы ИИ есть различение собственно системы и ее среды**.

Отличия неразрывно связаны с намерениями. В частности, наиболее распространенным будем считать случай, когда **система ИИ определяет свои границы и пытается их поддерживать**; это, видимо, соответствует тому, что мы называем **самоосознанием**. Подобное наблюдается и у отдельных личностей (самосохранение), и у социальных групп (клубы, субкультуры, нации). В этих случаях имеет место не только отличие, но и ***индикация,* т. е. выделение одного из двух различаемых состояний как более важного («это», «я», «мы» и т. д.). В самом деле, целью различения является, как правило, именно такая индикация.**

Менее важным типом отличия является отличие, которое делается при самостоятельном определении цели: в основном при научных исследованиях, например тогда, когда какая-то дисциплина «определяет область своих интересов» или когда ученый определяет систему, которую он будет изучать.

В любом случае **установление границ системы ИИ** неизбежно связано с тем, что мы называем ***когнитивной точкой зрения; ...* в частности, оно связано с понятием ценности или пользы, а также с**

мыслительными способностями (чувствительностью, знаниями) того, кто определяет отличия. И наоборот, выделенные отличия обнаруживают уровень мыслительных способностей того, кто их сделал.

Именно таким способом биологические и социальные структуры проявляют свою связность и позволяют убедиться в том, что они обладают мыслительными способностями или что они до некоторой степени «ощущают».

Таким образом, **распознавание различий, близкие (по смыслу) к отличиям, может быть двух типов. Можно распознать либо то, что две вещи являются различными, либо то, что одна и та же вещь меняется во времени** (**в терминологии системы ИИ, с точки зрения соответствующей базы или соответствующего параметра**). Эти два понимания отличия теснейшим образом связаны и дополняют друг друга. **Первое охватывает *неизменные* (инвариантные, постоянные) свойства вещей, а второе — те свойства, которые рассматриваются как временные (варьирующиеся, изменяющиеся).**

Важность понятия *изменение,* являющегося одним из производных от понятия *отличие,* выражена в литературе многими способами. Например, древнегреческому философу Гераклиту принадлежит знаменитое высказывание о том, что

«Ничто не постоянно, кроме изменений.»

Независимо от того, разделяем мы точку зрения Гераклита или нет, хотя бы из практических соображений необходимо считать, что **некоторые характеристики среды являются неизменными**. **Если это невозможно, то невозможно также и передавать сообщения, поскольку отсутствуют идентифицируемые единицы, и на самом деле нельзя действовать осмысленным образом, поскольку ничто в среде нельзя считать истинным.**

Имеется несколько соображений, по которым можно определить инварианты в среде. Одно из таких очевидных соображений состоит в том, что изменения в среде происходят значительно медленнее, чем мы воспринимаем, думаем или действуем. Таким образом, на практике можно или пренебречь этими изменениями, или вообще их не замечать. Другое соображение заключается в том, что эти изменения происходят на таком уровне разрешения, что человек не может их наблюдать. Поэтому эти изменения, если они только не проявляются каким-то образом в диапазоне нашего восприятия, являются для нас несущественными.

Можно выделить инвариантность другого типа, которая связана с процессом изменений, а не с тем, что меняется. **В схеме ФРИЗ этот тип инвариантности нашел свое отражение в понятии порож-**

дающей системы ИИ. Ее переменные изменяются, однако способ изменения, описываемый функцией поведения системы (или ее ST-функцией), **параметрически инвариантен, т. е. постоянен (неизменен) относительно параметрического множества.**

Поиск инвариантностей составляет самую суть науки, о чем очень хорошо пишет Г. Спенсер Браун:

«**Наука занимается определением констант: это изучение неизменного.** Если я брошу бомбу из окна верхнего этажа, то она будет падать вниз со все возрастающей скоростью. Это изменение скорости — проклятие для ученого. Он не успокоится до тех пор, пока не придумает, как описать это изменение неизменным образом. В данном случае долго искать решение не нужно. Скорость этой бомбы может меняться, но неизменной остается скорость ее изменения (**называемая ускорением**). Функция 32 $фут/с^2$ —это константа, описывающая поведение не только этой бомбы, но и всех другlх бомб, сброшенных поблизости.

Мы говорим о функции 32 $фут/с^2$ как об абсолютной константе, но если вдуматься, то это не так. **Масса Земли понемногу увеличивается за счет захвата метеоритов и космической пыли. Следовательно, можно ожидать, что гравитационное ускорение будет со временем увеличиваться.** Можно считать это увеличение «константой», но нет оснований считать, что и эта «константа» будет оставаться неизменной. Наша попытка исчерпывающего описания гравитационного ускорения оказалась неудачной. Может показаться, что положение можно исправить следующим образом. Мы можем утверждать, что *g* зависит от заданных масс, расстояний и других факторов, которые называются *соответствующими.* Если соответствующие факторы определены, то мы в состоянии определить неменяющуюся константу. Однако теперь эта задача представляется чисто лингвистической: либо изменение этой константы, сделанное исходя из наблюдений и экспериментов, может быть объяснено нашим ошибочным определением соответствующих условий, при которых следует наблюдать эту константу. Другими словами, всегда существует «действительная» константа, к которой сходятся наши наблюдения; если нам даже покажется, что мы ее определили, впоследствии обнаружится, что мы нашли только некоторое ее приближение.

Это похоже на философское понятие «вещь в себе» или «реальности вне проявления». Его можно было бы назвать «константой вне аппроксимации». Подобное предположение является частью научного подхода, и для определенных целей этот подход, несомненно, удобен. Его плодотворность мы обсудим позже, а сейчас необходимо подчеркнуть, что **законы природы — это всего лишь сделанные**

нами описания таких структур, относительно которых было выяснено, что они меняются, но только очень медленно. По существу, мы не располагаем свидетельствами того, что какая-то структура вообще не меняется...

То, что мы замечаем, зависит от того, как и особенно как быстро меняемся мы сами. Например, замечаем вещи, которые меняются также медленно, как мы, или еще медленнее, но в общем случае не те, что меняются значительно быстрее. Таким образом, **чем быстрее мы меняемся, тем больше мы замечаем.**

Если мы снимем на кинопленку растение со скоростью один кадр в минуту и прокрутим эту пленку со скоростью 30 кадров в секунду, то нам покажется, что это растение ведет себя как животное. Если поместить что-то рядом с ним, то растение явно ощутит это и отреагирует. Это, безусловно, чувствующее существо. Тогда почему в обычных условиях не кажется, что оно обладает чувствительностью? Возможно, дело в том, что оно слишком медленно думает. Для существ, которые реагируют в 1800 раз быстрее нас, мы тоже выглядим как лишенные чувств растения. В самом деле, существа, двигающиеся так быстро, будут уверены в том, что мы лишены чувств, поскольку, как правило мы не будем ощущать их поведение. Их мимолетные появления не будут для нас ничего значить. Дерево не может почувствовать, что я прошел мимо, так же как я не могут почувствовать пролетевшую мимо меня пулю. Я должен ощутить определенные события, связанные с полетом пули, например простреленную руку. Точно так же, если моя прогулка была достаточно «разрушительной», то дерево ощутит определенные события, связанные с моей прогулкой, скажем сломанную ветку. Но то, что для дерева быстро, для меня медленно и скучно, а то, что происходит с обычной для меня скоростью, вообще находится за пределами мира дерева ... **Тот, кто сможет двигаться бесконечно быстро, будет способен знать все, поскольку для него все будет находиться в состоянии покоя. Он будет располагать бесконечным временем для узнавания. А если ему будет позволено и самому перемещать частицы вселенной, то он станет не только всезнающим, но и всесильным, поскольку сможет сколь угодно долго изменять положение вещей.**

Мы видели, что **наука всегда стремится описать изменение в виде неизменной формулы**. **Такая формула всегда может быть найдена, если изменение уже произошло, однако она не всегда применима к будущему. Если же изменяется само изменение, то нужна новая формула**. **Теперь мы в состоянии разделить задачу историка и задачу ученого.** Мы видели, что **ученый стремится зафиксировать**

неизменным образом меняющиеся явления, в то время как историк занимается только фиксированием изменений, которые уже произошли. Историк не занимается поиском формулы, которая была бы истинной на все времена. Если бы им когда-нибудь была обнаружена такая формула, то больше не потребовалось бы никаких новых записей, и он потерял бы работу. **Повторяется не история, а наука. Ученый начинает с того, что вглядывается в сумбур изменений и что может фиксирует в виде формул. История — это то, что остается после , того, как ученый уже отобрал свое. Таким образом, история важнее науки, поскольку она дает первоначальное понимание вещей.** Но изучать ее не обязательно. **То, что не изменяется (например, прошлое), не опасно. Оно не может нам повредить. Остерегаться следует того, что изменяется. А для того, чтобы приспособиться к изменениям и суметь их почувствовать, нужно быстро ощущать.**

Понятно, что **поиск инвариантностей, столь важный для науки, должен быть одним из главных компонентов системы ИИ**.

Некоторые виды параметрической инвариантности связаны с порождающими системами ИИ и структурированными порождающими системами ИИ. Они рассматриваются соответственно в гл. 4, 5. Однако они представляют собой только частные случаи общего понятия параметрической инвариантности, рассматриваемого в следующем разделе.

6.2. Первичные и вторичные характеристики системы ИИ

Существовать — значит быть тождественным самому себе, что, в свою очередь, значит быть идентифицированным. Система ИИ обладает тождественностью, если определены какие-то ее характеристики. **Характеристики, совокупность которых идентифицирует систему ИИ, будем называть *первичными характеристиками.* Любые другие характеристики системы ИИ, не участвующие в ее идентификации, назовем ее *вторичными характеристиками.***

Таким образом, **множество всех характеристик системы ИИ образует ее определение**. Общим свойством эпистемологической иерархии систем ИИ является то, что множество первичных характеристик определенного уровня является подмножеством множества первичных характеристик всех более высоких уровней. На рис. 1 эти включения

для нейтральных систем ИИ показаны вплоть до уровня структурированных систем ИИ с поведением.

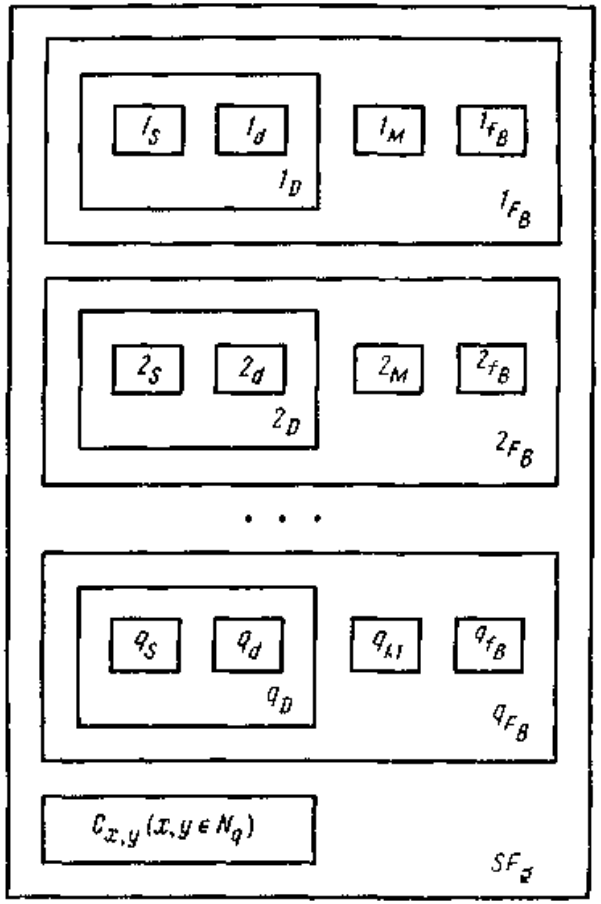


Рис. 1. Отношение включения первичных характеристик для эпистемологической иерархии систем ИИ

Аналогично можно представить эти включения для структурированных систем данных или структурированных исходных систем ИИ. Для этого нужно только исключить те характеристики, которые не входят в определения соответствующих систем ИИ (**маски, функции поведения и функции данных**); модификация этого рисунка для направленных систем ИИ тривиальна.

Необходимым условием оперирования с системой ИИ при решении интеллектуальной задачи является сохранение ее тождественности. Это значит, что **первичные характеристики (<u>но не вторичные</u>) должны оставаться неизменными.** Так, например, **заданная система данных может быть дополнена <u>функцией поведения</u>, определенной по ее данным**. Очевидно, что она является характеристикой данной системы данных. Однако, так как существует множество разных функций поведения, которые можно определить для одной и той же системы данных при разных масках и способах представления ограничений на переменные, эта функция не может быть использована для идентификации системы ИИ. Она является вторичной характеристикой системы ИИ, и, следовательно, ее можно изменять. Можно заменить одну функцию поведения другой, не изменив тождественности системы данных. Можно привести и обратный пример. Заданная система ИИ с поведением при разных начальных условиях может порождать разные наборы данных. Любой такой набор данных является вторичной характеристикой этой системы ИИ с поведением. Она остается неизменной независимо от того, какой набор данных рассматривается.

На определенном этапе процесса решения интеллектуальной задачи система ИИ обычно переопределяется в том смысле, что ее вторичные характеристики принимаются как первичные. Например, при эмпирическом исследовании система ИИ первоначально определяется как исходная, и это определение сохраняется на этапе сбора данных. После того как исследователь придет к заключению, что полученных данных достаточно, чтобы соответствующим образом описать переменные исходной системы ИИ, он может принять полученный массив данных в качестве первичной характеристики. Это означает, что он переопределил исходную систему ИИ в систему данных. **Такое переопределение представляет собой *шаг индукции,* поскольку оно является следствием индуктивного действия, основанного, например, на предположении, что любой экземпляр возможного для переменных состояния имеется среди собранных данных или может быть выведен из них. Решение о переопределении исходной системы ИИ в систему данных в этом случае отражает уверенность исследователя в том, что <u>имеющихся данных достаточно для целей исследования</u>.** Эта уверенность основывается не только на самих данных, но и на цели исследования, на представлении о том, как будут обрабатываться такие данные, на сравнении с аналогичными исследованиями, проведенными ранее, а также зависит от субъективных качеств самого исследователя (от его опыта в данной области, интуиции и т. п.). После переопределения исходной системы

ИИ в систему данных **исследование приобретает теоретический характер**. **Первой задачей такого исследования является поиск такой порождающей системы ИИ, которая адекватно представляла бы систему данных.** Любая функция поведения или ST-функция, полученная по данным при различных масках, представляет собой вторичную характеристику этой системы данных. Если исследователь достаточно уверен в какой-то из этих функций, то может принять ее за первичную характеристику и, таким образом, **переопределить систему данных в порождающую систему ИИ. Это переопределение также содержит элемент индукции, поскольку принятая функция поведения или ST-функция по природе своей параметрически инвариантна, а следовательно, определение порождающей системы ИИ выходит за пределы заданного параметрического множества и различных начальных условий. С помощью другого шага индукции порождающую систему ИИ можно аналогичным образом переопределить в структурированную порождающую систему ИИ при условии, что реконструктивные свойства этой системы проанализированы.**
Переопределяются системы ИИ и в процессе проектирования. Предположим, что задача состоит в том, что заданную систему ИИ с поведением нужно реализовать в виде структурированной системы ИИ, состоящей из элементов определенных типов и удовлетворяющей определенным требованиям (**целям, ограничен**иям). **Характеристики структурированных систем ИИ, определяемых в процессе проектирования, рассматриваются как вторичные.** Эти харакеристики не изменяют тождественности заданной системы ИИ с поведением; их исходная **система, маски и функция поведения остаются единственными первичными характеристиками в течение всего процесса проектирования.** В результате обычно получается несколько структурированных систем ИИ. Если одна из них принимается в качестве решения, то ее можно переопределить как структурированную систему ИИ с поведением; она и будет служить основой при реализации проекта.
Первичные характеристики как средства идентификации системы ИИ должны быть известны и параметрически инвариантны. К вторичным характеристикам подобные требования не предъявляются. Они могут быть или полностью неизвестны, или известны только частично и при этом не должны быть параметрически инвариантны. **Если первичные характеристики системы ИИ каким-либо образом изменяются, то, по определению, система ИИ перестает быть тождественной самой себе и появляется новая система ИИ.** С другой стороны, изменения вторичных характеристик не влияют на

тождественность системы ИИ. Так, например, объем имеющихся данных не влияет на исходную систему ИИ, в то время как конкретная система данных меняется при добавлении или исключении данных. Аналогичным образом ST-система не меняется при замене ее текущего состояния другим состоянием или реализующей ее структурированной системы ИИ на другую, также реализующую ее структурированную систему ИИ.
Параметрическая инвариантность является одним из свойств функций поведения и ST-функций. Здесь понятие инвариантности строго связано с конкретным параметрическим множеством, определенным как часть рассматриваемой исходной системы ИИ. Однако иногда бывает нужно использовать это понятие в локальном смысле— для некоторого подмножества, заданного параметрического множества. Такую инвариантность можно назвать *локальной инвариантностью* или *субинвариантностью.*
Теперь рассмотрим множество функций поведения, определенных на одной исходной системе ИИ, причем эти функции только локально инвариантны и, следовательно, не полностью характеризуют переменные (и порождают их состояния) на всем параметрическом множестве исходной системы ИИ. Таким образом, эти функции не могут рассматриваться как первичные характеристики одной системы с поведением. Однако они в принципе могут быть интегрированы в большую систему ИИ. **Для этого требуется описать процедуру замены одной функции поведения другой на параметрическом множестве. Назовем эту процедуру *процедурой замены*.**
Если некие функции поведения, которые являются параметрически инвариантными только локально, интегрируются с помощью соответствующей процедуры в одну систему ИИ, то для удобства их параметрическая инвариантность может быть распространена на все параметрическое множество. Подобное распространение не повлияет на интегрированную систему ИИ, так как процедура замены не позволяет использовать функцию поведения вне области ее локальной инвариантности. Таким образом, **интегрированную систему ИИ удобно рассматривать как множество функций поведения и процедуру замены.**
Предложенный метод интегрирования систем ИИ с поведением подходит и для других типов систем ИИ. В следующем разделе мы введем, формализуем и рассмотрим различные категории интегрированных систем ИИ.

6.3. Метасистемы ИИ

Один из способов интегрирования нескольких сопоставимых систем ИИ в большую систему ИИ состоит в образовании структурированной системы ИИ, как это описано в гл. 5. **Другой способ интегрирования систем ИИ состоит в определении соответствующей процедуры, как это предлагается в данном разделе. Интегрированные таким образом системы ИИ будем называть *метасистемами ИИ.***

В термине «метасистема» используется греческий префикс *мета.* По гречески он имеет три значения:

1. «Мета *X*» называется то, что наблюдается (имеет место) *после X,* т. е. *X* является предпосылкой мета *X.*
2. Выражение «мета *X*» показывает, что *X* меняется и служит общим названием этого изменения.
3. «Мета *X*» используется в качестве названия того, что *выше X* в том смысле, что оно более высоко организовано, имеет более высокий логический тип или рассматривается в более широком смысле.

Мы видим, что термин «метасистема» в приложении к системам ИИ, соединяющим с помощью соответствующей процедуры замены несколько систем ИИ, **включает все три смысла этого понятия**.

Понятно, что:

1) метасистема ИИ может быть определена только *после* того, как определены другие типы систем ИИ;
2) эта система ИИ описывает *изменение* — замену одной системы ИИ другой;
3) она выше отдельных систем ИИ — процедура замены делает ее чем-то большим, чем набор отдельных процедур.

Таким образом, название «метасистема» вполне обоснованно.

Метасистемы ИИ вводятся в основном для описания изменений при заданном параметрическом множестве тех системных характеристик, которые определяются как параметрически инвариантные. Такими характеристиками являются множества переменных и соответствующие множества состояний и каналов, функций поведения и ST-функций и соединения структурированных систем ИИ. Метасистемы ИИ могут быть определены через системы ИИ **любого из трех определенных ранее типов. Включенные в метасистему ИИ системы ИИ будем называть *элементами.*** Они должны быть сопоставимы в том смысле, что должны иметь **один тип базы (время, пространство, группа**).

Для обозначения метасистем ИИ будем использовать (подобно оператору **S** для структурированных систем ИИ) оператор **M** следующим образом: помещенный перед обозначением системы ИИ

определенного типа, он означает метасистему ИИ, элементами которой являются системы ИИ данного типа. Например, $\mathbf{MF}_B$, $\widehat{\mathbf{MF}}_B$ и $\mathbf{MSD}$ —это метасистемы ИИ, элементами которых являются соответственно нейтральные системы с поведением, направленные ST-системы и структурированные системы данных (нейтральные).

Для формального определения метасистем ИИ рассмотрим сначала метасистемы ИИ, элементами которых являются нейтральные системы ИИ с поведением, т. е. метасистемы ИИ $\mathbf{MF}_B$. Всякая метасистема ИИ этого типа определяется как тройка:

$$\mathbf{MF}_B = (\mathbf{W}, \mathscr{F}_B, r), \tag{1}$$

где **W** —параметрическое множество; $\mathscr{F}_B$ —множество нейтральных систем ИИ с поведением, чьи параметрические множества являются подмножествами **W** (но не обязательно точными); *r* — процедура замены, реализующая определенную функцию вида

$$r : \mathbf{W} \to \mathscr{F}_B. \tag{2}$$

Назовем функцию (2) *функцией замены*. Важно понимать, что эта функция не обязательно должна быть явно включена в метасистему ИИ. Требуется только, чтобы была задана процедура, представляющая определенную функцию вида (2), даже если невозможно или трудно определить, какую функцию она реализует. Можно определить функцию замены и явно. В этом случае процедура замены идентична функции замены или ею определяется. Этот подход будет продемонстрирован в этом разделе на нескольких примерах.

Можно легко модифицировать уравнение (1), определяющее *метасистему ИИ нейтральных систем ИИ с поведением так,* чтобы оно подходило и для других систем ИИ. Для этого нужно заменить обозначения $\mathbf{MF}_B$ и $\mathscr{F}_B$ на обозначения, представляющие другие системы ИИ. Для удобства договоримся, что множество систем ИИ определенного типа обозначается прописной рукописной буквой, соответствующей системам ИИ этого типа. Тогда, например,

$$\mathbf{MSF}_S = (\mathbf{W}, \mathscr{SF}_S, r),$$
$$\widehat{\mathbf{MD}} = (\mathbf{W}, \widehat{\mathscr{D}}, r)$$

является соответственно определением *метасистемы ИИ структурированных* (нейтральных) *ST-систем* и определением *метасистемы ИИ направленных систем данных.* Нетрудно дать определения и остальных типов метасистем ИИ.

Вообще говоря, можно определить метасистему ИИ и для множества систем ИИ разных типов. Обозначим **MX** подобный общий тип метасистем ИИ. Тогда

$$\mathbf{MX} = (\mathbf{W},\ \mathscr{X},\ r), \tag{3}$$

где $\mathscr{X}$ — произвольное множество систем ИИ, чьи параметрические множества являются подмножествами **W**; *r*— снова процедура замены, которая должна реализовывать определенную функцию замены

$$r : \mathbf{W} \to \mathscr{X}. \tag{4}$$

В общей формулировке метасистемы ИИ (4), элементами которой являются системы ИИ одного типа, могут рассматриваться как частные случаи системы ИИ (3), где

$$\mathscr{X} \in \{\mathscr{P},\ \mathscr{D},\ \mathscr{F}_B,\ \mathscr{F}_S,\ \mathscr{PP},\ \mathscr{PD},\ \mathscr{PF}_B,\ \mathscr{PF}_S\};$$

$$\hat{\mathscr{X}} \in \{\hat{\mathscr{P}},\ \hat{\mathscr{D}},\ \hat{\mathscr{F}}_B,\ \hat{\mathscr{F}}_S,\ \mathscr{\hat{P}\hat{P}},\ \mathscr{\hat{P}\hat{D}},\ \mathscr{\hat{P}\hat{F}}_B,\ \mathscr{PF}_S\}$$

соответственно для нейтральных и направленных систем ИИ. Такие метасистемы ИИ будем называть *гомогенными метасистемами ИИ.*

Процедуры замены, являющиеся первичными характеристиками метасистемы ИИ, могут быть определены различными способами. Они допускают даже случайный выбор. Единственное условие состоит в том, что процедура замены должна реализовывать определенную функцию замены общего вида (4).

Приведем несколько примеров, показывающих типичные способы определения функций замены.

Пример 1. В этом примере описывается работа светофоров на перекрестке в течение суток. Описание представляет собой гомогенную метасистему, состоящую из трех элементов, определенных как системы данных. Все три элемента содержат одни и те же переменные и множества состояний. Переменные описывают сигналы светофоров для транспортных потоков с севера на юг, с юга на север, с востока на запад, с запада на восток, обозначенные соответственно СЮ, ЮС, ВЗ и ЗВ, а также сигналы для левых поворотов с севера на восток, с юга на запад, с востока на юг, с запада на север, обозначенные соответственно СВ, ЮЗ, ВЮ и ЗС. **Параметром является время; единицей измерения времени — секунда,** соответствующие интервалы времени измеряются в секундах.

На рис. 2 приведены матрицы данных $\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3$ для трех элементов $\mathbf{D}_1, \mathbf{D}_2, \mathbf{D}_3$; их временные множества определены непосредственно соответствующими интервалами времени.

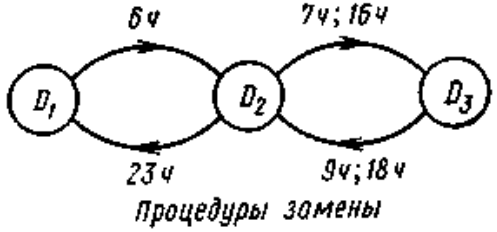


Рис. 2. Метасистема управления движением (пример 1)

Матрицы данных периодические и задаются одним периодом. Как показано в табл. 1, системы $\mathbf{D}_1$, $\mathbf{D}_2$ и $\mathbf{D}_3$ определяют управление движением ночью, днем и в часы пик.

Таблица 1.

Метасистема управления движением

Элемент $\mathbf{D}_1$: управление движением ночью

$t_i \in$	[0, 20)	[20, 30)	[30, 50)	[50, 60)
СВ-ЮЗ	з	ж	к	к
СЮ-ЮС	з	ж	к	к
ВЮ-ЗС	к	к	з	ж
ВЗ-ЗВ	к	к	з	ж

Элемент $\mathbf{D}_2$: управление движением в нормальных условиях

$t_i \in$	[0, 15)	[15, 25)	[25, 55)	[55, 65)	[65, 80)	[80, 90)	[10, 110)	[110, 120)
СВ-ЮЗ	з	ж	к	к	к	к	к	к
СЮ-ЮС	к	к	з	ж	к	к	к	к
ВЮ-ЗС	к	к	к	к	з	ж	к	к
ВЗ-ЗВ	к	к	к	к	к	к	з	ж

Элемент $\mathbf{D}_3$: управление движением в часы пик

$t_i \in$	[0, 30)	[30, 40)	[40, 50)	[50, 60)
СВ-ЮЗ	к	к	к	к
СЮ-ЮС	з	ж	к	к
ВЮ-ЗС	к	к	к	к
ВЗ-ЗВ	к	к	з	ж

Эти системы, рассматриваемые как элементы метасистемы, заменяют одна другую в определенные моменты времени в течение суток. Функцию замены в данном случае удобно представить в виде помеченной диаграммы, изображенной на рис. 2. Ее узлами являются элементы данной метасистемы, стрелка из $\mathbf{D}_i$, в $\mathbf{D}_j$ *(i, j*=1, 2, 3) показывает, что $\mathbf{D}_i$ заменяется на $\mathbf{D}_j$, а метка стрелки указывает на момент времени, в который происходит эта замена. Таким образом, данная метасистема представляет собой тройку

$$\mathbf{MD}=(T,\ \mathscr{D}=\{\mathbf{D}_1,\ \mathbf{D}_2,\ \mathbf{D}_3\},\ r),$$

где $\mathscr{D}$ и r полностью определены в табл. 1, а T состоит из 5760 определенных интервалов времени суток (420 периодов $\mathbf{d}_1$, 390 периодов $\mathbf{d}_2$ и 240 периодов $\mathbf{d}_3$).

Пример 2. Рассмотрим больного, у которого время от времени почки перестают функционировать соответствующим образом. Наблюдение за состоянием больного ведется по нескольким переменным. При необходимости работу почек берет на себя искусственная почка или устройство для гемодиализа. При работе искусственной почки контроль осуществляется по нескольким дополнительным переменным. Таким образом, при наблюдении за больным имеются две исходные системы, скажем $\mathbf{S}_1$ и $\mathbf{S}_2$. Одна из них связана с теми периодами, когда почки работают нормально, а другая — с периодами, когда больному подключают искусственную почку. Система $\mathbf{S}_1$ состоит из четырех переменных:

v_1 — вода в урине (измеряется с точностью 0,1 л в диапазоне 0—1 л);

v_2 — глюкоза в урине (измеряется с точностью 20 г в диапазоне 0—200 г);

v_3—мочевина в урине (измеряется с точностью 5 г в диапазоне 0—50 г);

v_4—содержание азота в крови (канал наблюдения дает только два состояния 1 и 0 в зависимости от того, достигает ли содержание азота 150 мг на 100 мл крови или нет).

В систему $\mathbf{S}_2$ входят все перечисленные выше переменные, а также две дополнительные:

v_5 — температура крови (измеряется с точностью 0,2° F в диапазоне 97—100° F);

v_6 — кровяное давление (измеряется с точностью 2 мм ртутного столба в диапазоне 110— 130 мм).

Значения этих переменных искусственная почка должна поддерживать в определенном узком диапазоне. Все введенные переменные наблюдаются во времени. Реально моменты времени определяются

тяжестью состояния пациента и другими факторами, которые нет необходимости рассматривать в данном примере.

Эти две исходные системы можно рассматривать как метасистему со следующей процедурой замены *r:* если v_4=1» заменить $\mathbf{S}_1$ на $\mathbf{S}_2$; если v_4=0, заменить $\mathbf{S}_2$ на $\mathbf{S}_1$. Таким образом, данная метасистема представляет собой тройку

$$\mathbf{MS}=(T,\ \mathscr{S}=\{\mathbf{S}_1,\ \mathbf{S}_2\},\ r),$$

где T — объединение временных наборов $\mathbf{S}_1$ и $\mathbf{S}_2$.

Пример 3. Рассмотрим структурированную систему, элементы которой образуют массив $n\times n$. Пусть каждый из ее элементов, часто называемых *ячейками,* соединен только с соседними ячейками из массива. Пример такого массива 5×5 приведен на рис. 3.

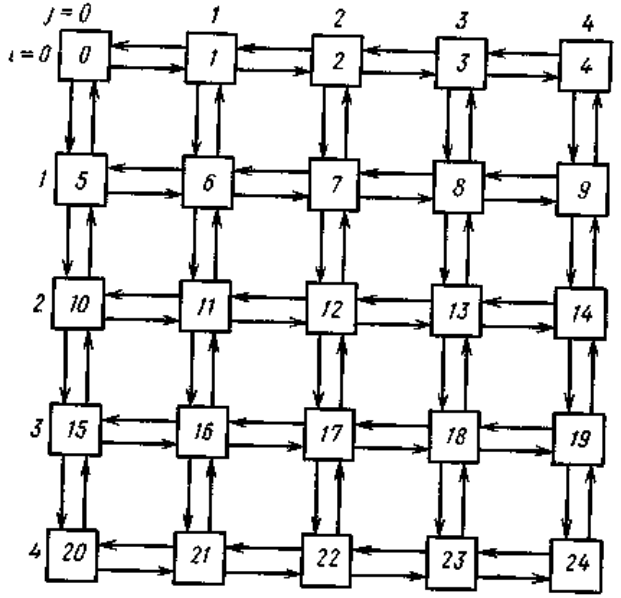


Рис. 3. Массив ячеек 5×5 (пример 3)

Ячейки массива удобно идентифицировать двумя целыми числами $i, j \in N_{0,\,n-1}$, указывающими номера соответственно строки и столбца. На рис. 3 также показано, что ячейку можно идентифицировать и одним целым числом

$$c=ni+j.$$

Будем называть *с идентификатором ячейки.*Допустим, что *внутренняя среда* ячейки (за исключением пограничных ячеек) состоит из четырех ее соседних ячеек, как это показано на рис. 4.

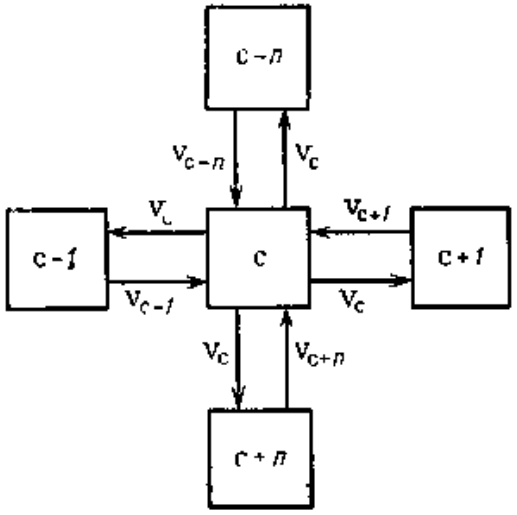


Рис.4. Внутренняя среда ячейки в массиве ячеек (пример 3)

Такая ячейка имеет четыре входные переменные v_{c-n}, v_{c-1}, v_{c+1}, v_{c+n} — по одной на каждую соседнюю ячейку и одну выходную переменную, соединенную со всеми соседними ячейками. Понятно также, как определяется внутренняя среда для пограничных ячеек, т. е. ячеек из строк 0, n—1 и столбцов 0, n— 1.

Предположим, что все ячейки из массива, скажем изображенного на рис. 3, представляют собой детерминированные направленные ST-системы, определенные на одном и том же полностью упорядоченном временном множестве T с ST-функцией

$$v'_c = f_c(v_{c-n}, v_{c-1}, v_{c+1}, v_{c+n}), \quad (5)$$

где v'_c — следующее состояние переменной v_c; $c \in N_{0,24}$; для пограничных ячеек определение этой функции должно быть уточнено, поскольку у этих ячеек отсутствуют некоторые входные переменные. Допустим далее, что каждая из переменных имеет только два состояния 0 или 1. Если $v_c=1$, будем называть ячейку *с активной* (если $v_c=0$, *пассивной).*

Для заданного массива ячеек можно определить множество структурированных систем, где каждая система описывается под-множеством ячеек этого массива. Так, например, для массива ячеек, изображенного на рис. 3, существует 2^{25} (т. е. более $3{,}3 \cdot 10^7$) структурированных систем. Иногда нужно объединить структури-

рованные системы из этого множества, назовем его множеством $\mathcal{SF}_S$, в метасистему

$$\mathbf{MS}\widehat{\mathbf{F}}_S = (T, \mathcal{SF}_S, r). \quad (6)$$

с помощью соответствующей процедуры замены. В качестве простого примера процедуры r из определения (6) можно предложить следующий: ячейка c ($c \in N_{0,24}$) включается в структурированную систему тогда и только тогда, когда или она сама является активной, или активна по крайней мере одна ячейка из ее внутреннего окружения, т. е. тогда и только тогда, когда

$$v_{c-n} + v_{c-1} + v_c + v_{c+n} \geqslant 1.$$

Для демонстрацииконкретной процедуры типа (6) воспользуемся предложенной процедурой замены и ST-функцией для массива 5×5, имеющей вид

$$v'_c = [(v_{c-5} + v_{c-1} + v_{c+1}) \pmod 2 + v_c + v_{c+5}] \pmod 2.$$

Если какие-то переменные недоступны для определенной ячейки, то они просто исключаются из этой формулы. Для каждой начальной структурированной системы данная метасистема порождает последовательность структурированных систем. Короткие фрагменты трех таких последовательностей показаны на рис. 5.

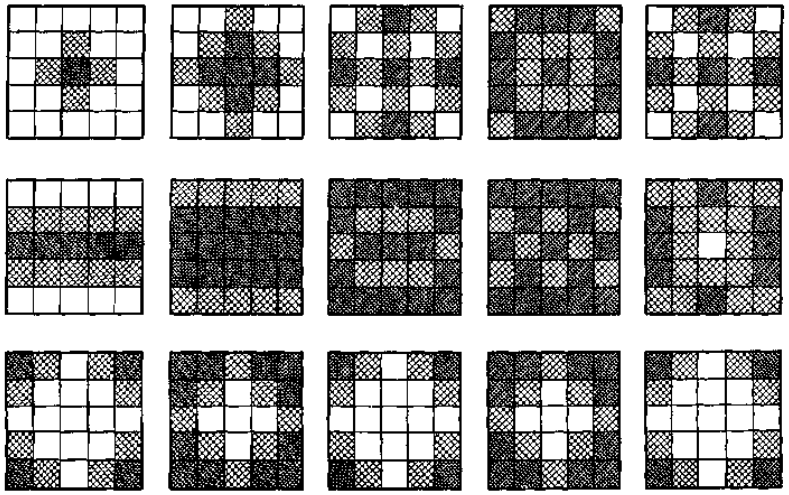


Рис. 5. Фрагменты трех последовательностей структурированных систем, которые могут быть порождены метасистемой, описанной в примере 3

Черные и серые клетки—это ячейки, входящие в структурированную систему, причем черные клетки — активные ячейки, а серые— пассивные; белыми клетками обозначены ячейки, не входящие в структурированную систему.

Разнообразие введенных в данном примере метасистем объясняется использованием различных ST-функций и процедур замены. Еще большего разнообразия можно достигнуть за счет использования разных массивов, в общем случае *k*-мерных, где $k \geq 1$. **Элементы этого класса метасистем известны в литературе как *клеточные автоматы.***

Пример 4. Рассмотрим систему, состоящую из одной переменной *v* с множеством состояний *V* и одного параметра *t,* представляющего собой индекс, определяющий позицию в строках состояний *V.* Полностью упорядоченное параметрическое множество *T*—это множество неотрицательных целых чисел. Для этой системы может быть определен класс метасистем типа

$$\mathbf{MD} = (T, \mathscr{D}, r)$$

таким образом, чтобы:

$\mathscr{D}$ представляла собой множество всех систем данных, которые могут быть образованы переменной *v* для всех возможных подмножеств N_n из множества *T* (n=1, 2, ...);

r представляло процедуру замены, в общем случае определенную следующим образом: задана система данных $\mathbf{D} \in \mathscr{D}$ с массивом данных **d**; массив **d** просматривается в порядке возрастания параметра (обычно слева направо) и заменяется на **d'** с помощью подстановки вместо каждого состояния α из **d** строки состояний *p(α),* определяемой функцией

$$p : V \to V \cup V^2 \cup \ldots \cup V^k \qquad (7)$$

для некоторого конечного *k;* **d'** определяет новую систему данных $\mathbf{D}' \in \mathscr{D}$.

Такие метасистемы называются *развивающимися* OL-системами (или системами без взаимодействий Линденмайе-ра). Пары α , *p(α)* обычно называются *продукционными правилами* и обозначаются $\alpha \to p(\alpha)$.

В качестве конкретного примера детерминированной OL-системы (или в нашей терминологии метасистемы) положим $V = N_{0,9}$ и пусть функция продукционного правила

$$p : V \to V \cup V^2$$

определяется таблицей

α	0	1	2	3	4	5	6	7	8	9
p(α)	12	93	49	61	25	87	78	34	9	9

Тогда, например, для начальной системы данных с массивом данных [0] эта метасистема порождает последовательность систем данных со следующими массивами данных:

Для недетерминированных OL-систем продукционные правила

[0]

[1 2]

[9 3 4 9]

[9 6 1 2 5 9]

[9 7 8 9 3 4 9 8 7 9]

[9 3 4 9 9 6 1 2 5 9 9 3 4 9]

[9 6 1 2 5 9 9 7 8 9 3 4 9 8 7 9 9 6 1 2 5 9]

[9 7 8 9 3 4 9 8 7 9 9 3 4 9 9 6 1 2 5 9 9 3 4 9 9 7 8 9 3 4 9 8 7 9]

определяются не функцией вида (7), а множеством пар (α , β) из декартового произведения

$V \times (V \cup V^2 \cup \ldots \cup V^k)$.

Выбор определенных продукционных правил может быть основан на условных вероятностях β при заданном α .

6.4. Структурированные системы ИИ и метасистемы ИИ

Структурированные системы ИИ и метасистемы ИИ—это две схемы интегрирования других систем ИИ (исходных систем ИИ, систем данных и порождающих систем ИИ). Это разные схемы, они независимы и ни одна из них не имеет преимущества перед другой. Более того, их можно комбинировать, т. е. применять одну к другой.

В структурированных системах ИИ интегрирование осуществляется по множествам переменных в предположении, что все они имеют одно и то же параметрическое множество. Таким образом, элементами структурированных систем ИИ являются системы ИИ с разными множествами переменных, но с одинаковыми параметрическими множествами.

В метасистемах ИИ, напротив, интегрирование систем ИИ осуществляется по параметрическим множествам независимо от того, имеют эти системы ИИ одно множество переменных или нет. Следовательно, элементами метасистем ИИ являются системы ИИ с разными локальными параметрическими инвариантами, определенными на обобщенном параметрическом множестве; они могут быть определены и для одного обобщенного множества переменных.

Как было показано в разд. 6.3, метасистемы можно использовать для интегрирования структурированных систем ИИ, которые в свою очередь, используются для интегрирования других систем ИИ. Примером тому служит класс метасистем из структурированных ST-систем (клеточных автоматов), рассмотренный примере 3. Такие системы ИИ обозначаются как системы типа $\mathbf{MS}\widehat{\mathbf{F}}_S$, поичем в этом обозначении используются оба символа интегрирования **M** и **S**

Структурированные системы ИИ можно также использовать для интегрирования метасистем ИИ, приведенных для различных множеств. В обозначении таких систем ИИ оператор **S** предшествует оператору **M**. Так, например, $\mathbf{SMF_B}$ обозначается структурированная метасистема ИИ, элементами которой (т. е. элементами метасистем ИИ объединенных в структурированную систему ИИ) являются нейтральные системы ИИ с поведением, а $\mathbf{MSF_B}$ обозначается метасистема ИИ, элементами которой являются структурированные

метасистемы с поведением. Следовательно, эти операторы некоммутативны. Системы **MSX** и **SMX** (для любого **X**) не только различны, но и не сопоставимы в смысле упорядочения в соотвтетствии с истемологической иерархией систем.

Пример 5. Рассмотрим направленную структурированную систему, состоящую из двух метасистем, элементами которых являются направленные системы с поведением. Эти метасистемы ${}^1\mathbf{M}\widehat{\mathbf{F}}_{GB}$ и ${}^2\mathbf{MF}_{\mathrm{CB}}$ соединены так, как это показано на схеме на рис 6а.

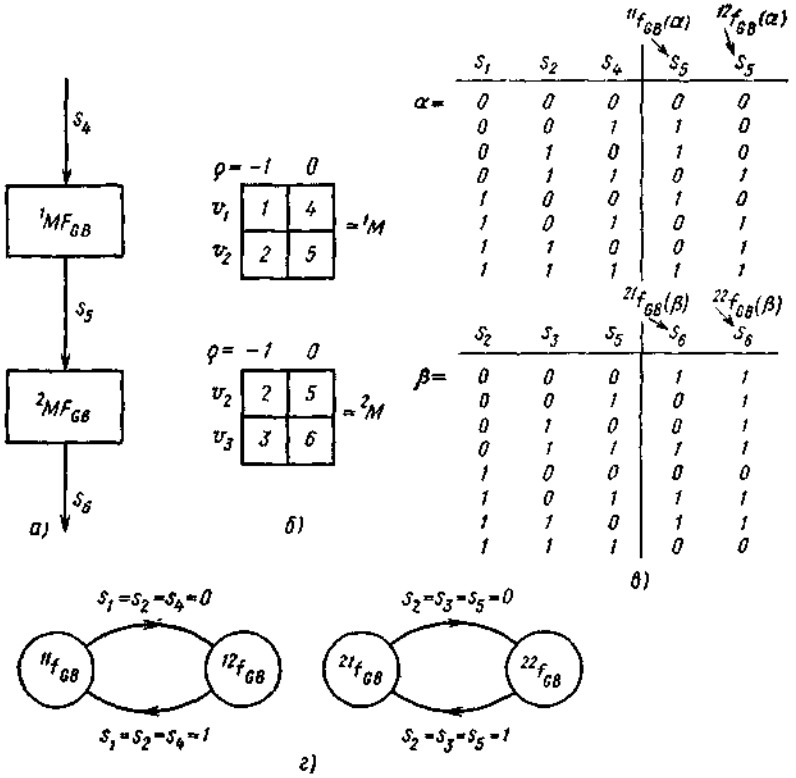


Рис. 6. Структурированная метасистема (пример 5)

Каждая метасистема состоит из двух направленных систем с поведением, переменные которых имеют два состояния 0 и 1 с одним и тем же полностью упорядоченным параметрическим множеством.

Системы с поведением для каждой метасистемы имеют одинаковую маску, но разные функции поведения.

Маски $^1\hat{M}$ и $^2\hat{M}$ приведены на рис. 6,б. Выборочные переменные, наблюдаемые в обеих системах, имеют, как это требуется для структурированных систем, одинаковые идентификаторы. Соответствующая маска используется в обеих системах, входящих в одну метасистему.

Функции поведения (порождающие функции) являются детерминированными, т. е. порождающие переменные в обеих системах с поведением являются функциями порождающих и входных выборочных переменных. Эти функции приведены на рис. 6,в. Функции поведения для первой метасистемы обозначены $^{11}f_{GB}$ и $^{12}f_{GB}$, а для второй метасистемы —$^{21}f_{GB}$, $^{22}f_{GB}$.

В обеих метасистемах используются процедуры замены одного типа: если все входные и порождающие переменные имеют состояние 0, то первая система с поведением заменяется второй; если все входные и порождающие переменные имеют состояние 1, то вторая система заменяется первой. Если этот алгоритм применить к соответствующим переменным, то будут получены процедуры ^1r и r^2, показанные на рис. 6,г.

На рис. 7 показан образец данных, порожденных этой структурированной метасистемой при начальном условии $s_1=s_2=s_3=0$ и определенной входной последовательности (последовательности состояний переменной v_1).

Рис. 7. Пример данных, порождаемых структурированной метасистемой, изображенной на рис.6 (пример 5)

На рис. 7 также указано, какая из двух функций поведения каждой из метасистем используется при соответствующем значении параметра *t*.

6.5. Многоуровневые метасистемы ИИ

Теперь мы можем определить многоуровневые метасистемы ИИ точно так же, как были определены многоуровневые структурированные системы ИИ (разд. 5.4): это метасистемы, элементами которых являются метасистемы, элементами которых... и т д . Такая рекурсия заканчивается на элементах, которые не являются метасистемами. Продолжая аналогию со структурированными системами ИИ, будем обозначать многоуровневые метасистемы ИИ обобщенным оператором, где $\mathbf{M}^k$ — число уровней метасистемы ИИ. Так, например, $\mathbf{M}^3\mathbf{D}$ обозначает трехуровневую метасистему ИИ, конечными элементами которой являются системы данных; это метасистема ИИ, элементами которой являются метасистемы ИИ, элементами которых снова являются метасистемы ИИ, элементами которых являются нейтральные системы данных. Двухуровневые метасистемы ИИ можно для удобства называть *мета-метасистемами.*

Формально *k* -уровневая метасистема ИИ определяется как тройка

$$\mathbf{M}^k\mathbf{X}=(\mathbf{W}^k,\ \mathscr{M}^{k-1}\mathscr{X},\ r^k), \quad (8)$$

где $\mathbf{W}^k$ — ее параметрическое множество; r^k — процедура замены; $\mathscr{M}^{k-1}\mathscr{X}$ — множество ее элементов (метасистем уровня k—1) чьими конечными элементами являются системы из множества $\mathscr{X}$, не являющиеся метасистемами.

Для многоуровневой метасистемы ИИ $\mathbf{M}^k\mathbf{X}$ первичной характеристикой, по которой определяется тождественность системы, является процедура замены только самого высокого уровня (r^k). Процедуры замены более низких уровней представляют собой вторичные характеристики, так как с точки зрения метасистемы ИИ $\mathbf{M}^k\mathbf{X}$ они определяют только локальную параметрическую инвариантность.

Пример 6. Рассмотрим двухуровневую метасистему (или мета-метасистему)

$$\mathbf{M}^2\mathbf{S}\hat{\mathbf{F}}_S=(T,\ \mathscr{M}\mathscr{S}\hat{\mathscr{F}}_S=\{^1\mathbf{MS}\hat{\mathbf{F}}_S,\ ^2\mathbf{MS}\hat{\mathbf{F}}_S\},\ r^2),$$

с параметрическим множеством *T;* первый элемент этой метасистемы $^1\mathbf{MS}\hat{\mathbf{F}}_S$— метасистема, описанная в примере 3 (клеточный автомат). Второй элемент (вторая метасистема) имеет вид

$$^2\mathbf{MS}\hat{\mathbf{F}}_S=(T,\ \mathscr{S}\hat{\mathscr{F}}_S,\ ^2r),$$

причем T и $\mathcal{PF}_S$ те же, что и в метасистеме $^1\mathbf{MS}\hat{\mathbf{F}}_S$, а ^2r представляет собой следующую процедуру замены: выбирается случайным образом активная ячейка, затем она делается пассивной и из структурированной системы исключаются те ячейки внутренней среды этой ячейки (включая и саму эту ячейку), которые являются пассивными и у которых все соседи пассивны.

Эти метасистемы интегрируются в мета-метасистему с помощью следующей процедуры второго уровня (или метапроцедуры) r^2: если структурированная система при значении параметра t—1 отличается от структурированной системы при значении t—2, то использовать метасистему $^1\mathbf{MS}\hat{\mathbf{F}}_S$; в противном случае использовать метасистему $^2\mathbf{MS}\hat{\mathbf{F}}_S$.

Пример 7. Рассмотрим развивающуюся OL-систему (пример 4), определенную как метасистему. Она состоит из двух метасистем

$^1\mathbf{MD}=(T, \mathcal{D}, {}^1r)$,

$^2\mathbf{MD}=(T, \mathcal{D}, {}^2r)$,

отличающихся только процедурами замены ^1r и ^2r. Их компоненты T и $\mathcal{D}$ те же, что и в примере 4, только $V=\{0, 1, 2, 3\}$. Процедуры замены ^1r, ^2r определяются следующими функциями ^{1}p и ^{2}p:

α	0	1	2	3
$^1p(\alpha)$	01	21	30	32

α	0	1	2	3
$^2p(\alpha)$	01	20	30	32

Тогда мета-метасистема $\mathbf{M^2D}$ определяется как

$\mathbf{M^2D}=(T, \{^1\mathbf{MD}, {}^2\mathbf{MD}\}, r^2)$,

причем метапроцедура r^2 определяется следующим образом: просматривается последний полученный массив данных; если не меньше половины элементов массива равно 0, то используется метасистема $^1\mathbf{MD}$ (функция ^{1}p); в противном случае —метасистема $^2\mathbf{MD}$ (функция ^2p).

Например, для начального массива данных [0] метасистема $^1\mathbf{MD}$ порождает следующую последовательность массивов:

```
[0]
[0 1]
[0 1 2 0]
[0 1 2 0 3 0 0 1]
[0 1 2 0 3 0 0 1 3 2 0 1 0 1 2 0]
[0 1 2 0 3 0 0 1 3 2 0 1 0 1 2 0 3 2 3 0 0 1 2 0 0 1 2 0 3 0 0 1]
```

Метасистема $^2\mathbf{MD}$ порождает другую последовательность массивов


```
[0]
[0 1]
[0 1 2 1]
[0 1 2 1 3 0 2 1]
[0 1 2 1 3 0 2 1 3 2 0 1 3 0 2 1]
[0 1 2 1 3 0 2 1 3 2 0 1 3 0 2 1 3 2 3 0 0 1 2 1 3 2 0 1 3 0 2 1]
. . .
```

Мета-метасистема $\mathbf{M^2D}$ порождает другую последовательность массивов (в каждом случае показано, какое продукционное правило было использовано:

```
[0], 1p
[0 1], 1p
[0 1 2 1], 2p
[0 1 2 0 3 0 2 0], 1p
[0 1 2 1 3 0 0 1 3 2 0 1 3 0 0 1], 2p
[0 1 2 0 3 0 2 0 3 2 0 1 0 1 2 0 3 2 3 0 0 1 2 0 3 2 0 1 0 1 2 0], 2p
. . .
```

Многоуровневые метасистемы ИИ можно комбинировать с многоуровневыми структурированными системами ИИ в любой последовательности. Единственное требование состоит в том, чтобы элементами метасистем ИИ или структурированных систем ИИ нижнего уровня были системы одного из трех основных типов — исходные системы ИИ, системы данных и порождающие системы ИИ. На языке ФРИЗ вполне адекватно можно описать систему $\mathbf{MSMSF_B}$, $\mathbf{M^2S^2SD}$ или $\mathbf{M^2S^2MSS}$.

6.6. Идентификация изменения

Многообразие способов, которыми можно определить метасистемы ИИ с разными типами элементов и различным числом уровней продемонстрировано в разд. 6.3—6.5 на нескольких показательных примерах. Однако задача получения соответствующего метасистемного описания исследуемых переменных по имеющимся данным является одной из наиболее сложных и наименее разработанных интеллектуальных задач.

Своим возникновением эта задача обязана одному из фундаментальных вопросов исследования систем: **должны ли ограничения на исследуемые переменные рассматриваться как параметрически**

инвариантные или только как меняющиеся в соответствии с некими параметрически инвариантными правилами изменения (процедуры замены)? Трудность состоит в том, что на этот вопрос нет однозначного ответа. Выбор той или иной точки зрения зависит не только от природы самих переменных, но и от целей исследования, от того, каким образом заданы ограничения (в виде маски, степени ограничения), являются данные полными или нет, и от других факторов, в частности связанных с контекстом исследования.
Если выбран определенный способ представления ограничений (например, вероятностные меры и наибольшая приемлемая маска) и используются обычные целевые критерии порождающей нечеткости и сложности, **то эта задача сводится к определению изменения функции поведения или ST-функции.** Другими словами, **она становится задачей определения существенных локальных ограничений на переменные на рассматриваемом параметрическом множестве.**
Если функция поведения, представляющая все параметрическое множество (глобальная функция), несильно отличается от функций поведения, соответствующих различным подмножествам параметрического множества, то нет необходимости в использовании формализма метасистем. Но если между этими функциями имеются существенные различия, то следует рассмотреть вопрос об использовании такого формализма. Однако применение этого подхода встречает некоторые трудности.
Прежде всего следует неким конструктивным образом определить, что понимается под «существенным различием функций поведения». То есть для придания понятию «разница» конкретного смысла необходимо выбрать функцию расстояния для функций поведения. Кроме того, нужно задаться каким-то пороговым значением расстояния для определения того, что это расстояние «существенно». Несмотря на то, что принятие этих решений предоставляется пользователю, ФРИЗ должен располагать неким вариантом решений, которые используются по умолчанию (при наличии соответствующего запроса).
Во-вторых, **разница (расстояние) между локальной и глобальной функциями поведения может считаться существенной только в том случае, когда локальная функция определена на достаточно большом подмножестве параметрического множества.** И снова должно быть принято решение, какой наименьший размер подмножества параметрического множества может считаться достаточным, чтобы на нем можно было бы определить содержательную локальную функцию поведения. Размер этот зависит от числа состояний переменных, от меры, с помощью которой задаются ограничения на

переменные, от используемой маски и, возможно, от каких-то других факторов.
Помимо отмеченных теоретических проблем, в задаче определения существенных локальных ограничений имеются и трудности практического характера. Они связаны прежде всего с тем, что число подмножеств параметрического множества, рассматриваемых в процессе определения существенных программных ограничений, с ростом параметрического множества растет экспоненциально. Как следствие с ростом параметрического множества лавинообразно растет число необходимых вычислений, так что эта задача становится неразрешимой даже для параметрических множеств относительно небольшого размера.
В заключение этого раздела опишем простую процедуру определения локальных ограничений. Будем называть ее *процедурой идентификации метасистемы ИИ.* Эта процедура использует предположение о том, что параметрическое множество T полностью упорядочено и что переменные описываются системой данных. Эта процедура или не определяет никакой метасистемы (если не находится существенных локальных ограничений), или определяет метасистему, состоящую из последовательности определенных на параметрическом множестве систем с поведением. Замена одной системы на другую происходит при определенных значениях параметра, которые вычисляются этой процедурой.
Даны: система данных с полностью упорядоченным параметрическим множеством $T=N_n$, маска (обычно наибольшая допустимая) и определенный способ представления ограничений на переменные (со своей мерой порождающей нечеткости). Процесс идентификации метасистемы состоит в следующем.
Шаг 1. Пусть дано целое число m, рациональное число Δ и известно, что $t=1, k=1$.
Шаг 2. Необходимо определить функцию поведения для подмножества данных, соответствующих отрезку $[t, t+m]$ параметрического множества, и вычислить ее порождающую нечеткость
Шаг 3. Затем надо увеличить k на 1; если $t+km \notin T$, то перейти на шаг 6.
Шаг 4. Определить функцию поведения для подмножества данных, соответствующего отрезку $[t, t+km]$ параметрического множества, и вычислить ее порождающую нечеткость U_k.
Шаг 5. Если $|U_k — U_{k-1}|/\max(U_k, U_{k-1}) < \Delta$, то перейти на шаг 3; иначе записать $t+(k—1)m$ в качестве аппроксимированной точки замены элементов метасистемы, $t=(k—1)m$, k присвоить значение 1 и перейти на шаг 2.

Шаг 6. Стоп.

В основе этой процедуры лежит следующее наблюдение: если данные не имеют существенных локальных ограничений, то порождающие нечеткости локальных функций поведения для отрезков [1, *t]* с ростом *t* быстро сходятся (после некоторых начальных возмущений) к значениям, лежащим в малом интервале Δ; напротив, если данные содержат существенное локальное ограничение, например, на отрезке параметрического множества $[t_1, t_2]$, то порождающие нечеткости обычно демонстрируют значительно превосходящие Δ колебания около значений параметра t_1 и t_2, а внутри отрезка $[t_1, t_2]$ снова находятся внутри небольшого интервала. Таким образом, существенные вариации порождающей нечеткости после того, как она колебалась в небольшом интервале, позволяют считать данную систему метасистемой, каждый элемент которой соответствует некоему подмножеству параметрического множества.

Чувствительность и вычислительная сложность данной процедуры существенным образом зависят от выбранных значений *m* и Δ. Для оказания помощи пользователю в выборе этих значений ФРИЗ должен располагать соответствующими характеристиками, полученными в результате вычислительных экспериментов, подобных тем, что описаны в гл. 5 для анализа реконструируемости. Можно также выполнить эту процедуру несколько раз при разных значениях *m,* Δ и усреднить полученные результаты. Как бы то ни было, эта процедура определяет возможные точки замены только приблизительно. Для более точного определения и оценки их существенности необходимо провести более общее исследование.

Пример 8. В данном примере описывается применение процедуры идентификации метасистемы при оценке работы пилотов в процессе их тренировки на летных тренажерах. Исходная система состоит из четырех переменных, описывающих соответствующие характеристики реактивного самолета; параметром является время. Не вдаваясь в технические подробности, будем каждую переменную описывать ее идентификатором, соответствующей характеристикой и множеством состояний:

s — скорость, множество состояний N_5;
a — высота, множество состояний N_7;
r — курс, множество состояний N_4;
d — расстояние, множество состояний N_4.

Каналы наблюдения и матрица данных для этих переменных приведены в табл. 2.

Таблица 5 2

Описание системы, рассматриваемой в примере 8

		Каналы наблюдения			
Характеристика:		Скорость	Высота над уровнем моря	Курс	Расстояние
Переменная		*s*	*a*	*r*	*d*
Единицы		Узлы	Футы	Градусы относительно магнитного севера	Морские мили
Интерпретация состояний	1	[150, 170)	[1700—1840)	[0—93)	[0—2,6)
	2	[170, 220)	[1840—2400)	[93—113)	[2,6—14)
	3	[220, 250)	[2400—2700)	[113—122)	[14—21)
	4	[250—270)	[2700—4900)	[122—360)	>21
	5	⩾270	[4900—6900)	—	—
	6	—	[6900—22800)	—	—
	7	—	>22800	—	—

Матрица данных

```
s 4444555555555555555555555555555555555555555555554444444333
a 7776666666666666666666666666666666666666666666666655555
r 2111111111111111111111111111111111111111111111111111222
d 444444444444444444444444444444444444433333333333333333333

s 333333333333322222222222211111111111111111111111111111111
a 555555555554444444444444444333333333333333333333333333333
r 222333333333333333333333333333333333333333333333333333333
d 333333333333333222222222222222222222222222222222222222222

s 111111111111111222223333333333333333333333333333333333333
a 322222211111111222222333444444444444444444444444444444444
r 333333333333333333344444444444444444444444444444444444444
d 222222222211111111111111111222222222222222222222222222222
```

Матрица, в которой данные представлены в порядке возрастания времени, описывает типичный (идеальный, правильный) способ захода на посадку реактивного самолета. Весь период захода на посадку разделен на 163 равных временных интервала, и для каждого определены соответствующие состояния переменных.

Матрица данных, приведенная в табл. 2, является своего рода эталоном, по которому оценивается работа тренирующихся летчиков. И для курсантов, и для инструкторов желательно, чтобы вся задача была естественным образом разбита на подзадачи так, чтобы было легче локализовать ошибки.

Процедура модификации метасистемы была применена к матрице данных, приведенной в табл. 2 для вероятностной функции поведения, маски из двух столбцов и Δ=0.1. Процедура выполнена для нескольких значений *m*. Полученные для разных значений *m* точки замены были затем усреднены. В результате получена метасистема, состоящая из трех элементов, определенных

соответственно на следующих отрезках времени: 1—70, 71—122, 123—163. На самом деле, три этих элемента являются естественными этапами выполнения посадки. Они соответствуют снижению высоты, полету по дуге с заходом на посадочный курс и собственно посадке. Для каждого элемента метасистемы была проведена оценка маски (разд. 4.6) и выполнена процедура реконструкции (разд. 5. 7) Полученные результаты приведены на рис. 8.

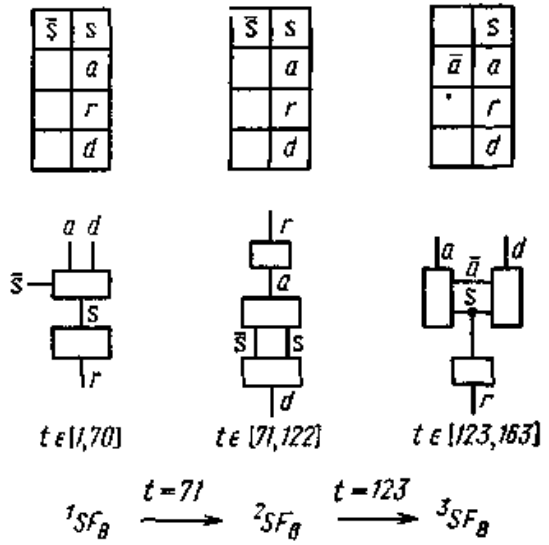


Рис. 8. Метасистема для типичного захода на посадку реактивного самолета (пример 8)

Это система типа **MSF$_B$.** ПОНЯТНО, ЧТО ПО сравнению с обобщенной системой с поведением данная система позволяет более точно оценивать работу курсантов.

7. Сложность

7.1. Сложность при решении интеллектуальных задач

Мы множество раз говорили о сложности, причем в самых разных контекстах. Упоминались и разные формы этого понятия, зависящие в первую очередь от типа системы ИИ и от интеллектуальной зада-

чи. Из этого видно, что сложность играет важную роль при решении интеллектуальных задач. По сути дела, она является столь же фундаментальным понятием науки о системах ИИ, сколь фундаментально понятие энергии в естественных науках.

Сложность многогранна. В толковом словаре написано, что сложность означает «иметь сложное качество или состояние», т. е. «иметь много различных взаимосвязанных частей, структур или элементов и, следовательно, быть трудно понимаемым полностью», или «включать множество частей, аспектов, деталей, понятий, требующих для понимания или овладения серьезного исследования или рассмотрения». **В этом общем описании отсутствует характеристика объектов, к которым применимо это понятие**. Поэтому оно потенциально применимо к любым типам объектов—материальным и абстрактным, естественным и искусственным, к творениям науки или искусства, а также к системам, задачам, методам, теориям, законам, играм, языкам, машинам, организмам, организациям и к любым другим объектам. Независимо от того, что рассматривается как сложное или простое, в общем случае **степень сложности связана с числом различаемых частей и мерой их взаимосвязанности**. Кроме того, понятие сложности имеет субъективную обусловленность, поскольку оно связано со способностью понимания или использования рассматриваемого объекта. Таким образом, то, что сложно для одного, может оказаться простым для другого.

Для ФРИЗ общепринятое понятие сложности выражается через взаимодействие исследователя с объектом исследования, результатом которого является исходная система. В этом смысле сложность не является неотъемлемым свойством исследуемого объекта, а, скорее, представляется следствием способа, которым исследователь взаимодействует с ним. Другими словами, **мы имеем дело не со сложностью объектов, а со сложностью систем, определенных на объектах.** Об этом хорошо сказано у Росса Эшби в одной из его работ : «Слово «сложность» в применении к системам имеет много смыслов, поэтому нужно пояснить, как его понимаю я. У этого слова нет очевидного или основного (доминирующего) понимания, поскольку хотя все и согласятся, что мозг сложен, а велосипед прост, для мясника мозг овцы прост, а велосипед, если его тщательно исследовать (как, скажем, единственную улику преступления), может иметь множество важных деталей.

Не прибегая к более глубокой аргументации, я в данной статье буду понимать «сложность» так, как я понимал ее в течение более чем десяти последних лет. Я буду оценивать ***степень сложности количеством информации, необходимой для описания реальной***

системы. Для нейрофизиолога мозг как сплетение волокон и бульон из энзимов сложен, и, соответственно, адекватная передача его подробного описания требует много времени. Для мясника мозг прост, так как ему нужно только отличать его примерно от тридцати других сортов «мяса», для чего он использует не более чем $\log_2 30$, т. е. 5 бит. При данном подходе сложность системы сознательно ставится в прямую *зависимость от наблюдателя;* оценить абсолютную сложность при этом невозможно; однако, на мой взгляд, единственным конструктивным способом оценки сложности является именно понимание сложности как точки зрения наблюдателя.»

У других авторов этот подход описывается иначе, но смысл остается тем же. Приведем еще две цитаты в подтверждение нашей точки зрения:

«Можно рассчитывать только на ясные модели, но не на реальную действительность и даже не на малую ее часть (П. Сьюпс).»

«Одной из задач экспериментального метода является замена сложных природных систем на простые искусственные (Г. Саймон).»

На уровне исходных систем системная сложность определяется тривиально. Она выражается только через мощности рассматриваемых множеств — множества переменных, множества параметров, множеств состояний и параметрических множеств, поскольку между этими множествами нет взаимосвязей. На более высоких эпистемологических уровнях понятие системной сложности становится более содержательным. Оно, конечно, разное для разных типов систем ИИ.

В гл. 1—6 говорилось о том, что одна и та же исходная система ИИ на различных более высоких эпистемологических уровнях может быть описана самыми разными способами. **Обычно один из критериев, по которым проводится обобщенное упорядочение на множестве таких альтернативных систем ИИ, является сложность.**

В некоторых случаях сложность желательна, т. е. ищется при заданных ограничениях система, обладающая высокой степенью сложности. Типичными примерами таких случаев являются шифрование и разработка датчиков случайных чисел. **В некоторых случаях определенная степень сложности — необходимое условие получения определенных системных свойств, которые обычно называются *выявляющимися свойствами.* Примерами таких свойств является самовоспроизведение, обучение и развитие.**

В других случаях, которые, по-видимому, более распространены при решении интеллектуальных задач, либо ищется простая система , либо делается попытка упростить уже существующую. Важность того,

чтобы системы были простыми, и важность методов их упрощения, сформулирована Г. Саймоном:

«Своему сохранению и процветанию в этом мире, который может быть и простым, и сложным, человечество обязано не мощности и скорости вычислений, а тому, что интересующие его системы представляют собой весьма частные случаи, которые поддаются анализу относительно простыми средствами, позволяющими определить их глубинную структуру. Таким образом, **нужна стратегия поиска такой структуры, стратегия структурной индукции (эта способность весьма развита в животном мире) плюс конкретный анализ и эвристические методы решения задач,** а не прямолинейный анализ очень общих классов сильно связанных сложных систем ...»

То же говорит физик Э. Теллер:

«Олицетворением простоты для меня служит один эпизод из истории музыки — искусства, традиционно любимого математиками и другими учеными. Четырнадцатилетний Моцарт услышал в Риме на тайной мессе «Мизерере» Аллегри. Сочинение это сохранялось в тайне; певцам запрещалось записывать его под страхом отлучения от церкви. Моцарт слышал его только раз и смог воспроизвести его целиком. Не следует думать, что он смог это сделать только благодаря своей изумительной памяти. Данная месса была произведением искусства, а следовательно, тяготела к простоте. **Сутью искусства является структура.** Мальчик, ставший потом одним из величайших композиторов мира, мог и не запомнить в подробностях это сложное произведение, но он определил основные моменты и запомнил их, а затем смог по ним восстановить и детали. Нелегко обнаружить эти основные моменты в музыке или науке. Для того чтобы их разглядеть, нужны труд и опыт. Однако **глубинная простота существует, и если ее найти, то можно открыть новые и более мощные взаимосвязи.»**

Еще дальше идет Дж. Уейнберг, когда предлагает определить науку о системах как науку об упрощении. Описав замечательное упрощение, с успехом примененное Ньютоном в механике, Уейнберг подводит итоги своим рассуждениям о простоте:

«Гений Ньютона не в исключительной вычислительной силе его мозга. Напротив, гениальной является его способность упрощать, идеализировать и выделять главное так, чтобы мир до некоторой степени становился обозримым и для обыкновенного человека. Изучая методы упрощения, удачно или неудачно применявшиеся, специалист по теории систем надеется сделать развитие человеческих знаний несколько менее зависящим от гениальности.»

Сложность и ее противоположность, простота, — понятия, связанные с многими фундаментальными философскими, математическими,

вычислительными и психологическими проблемами. В следующих разделах этой главы мы затронем некоторые наиболее общие проблемы, имеющие отношение к решению интеллектуальных задач.

7.2. Степени сложности

Иногда за кажущейся сложностью открывается простота, иногда, наоборот, простота скрывает исключительно сложные явления... С развитием средств исследования мы, несомненно, за простым будем открывать сложное, затем за сложным простое, затем снова сложное и так далее, и никогда не сможем предсказать, что же будет в конце концов. Где-то нужно остановиться, и наука требует, чтобы мы, остановились на простом. Только на этом фундаменте можно воздвигнуть здание обобщений.
Анри Пуанкаре

Перед тем как обсуждать вопросы оценки сложности, различные ее формы и то, как она представляется в схеме ФРИЗ, необходимо несколько расширить общепринятое понимание этого понятия и дать обзор разных интерпретаций сложности. Наиболее подходящим представляется исторический подход к определению различных методов трактовки понятия сложности, а именно, трактовка сложности в науке, технике и других областях человеческой деятельности.
История современной науки (с XVII в.) свидетельствует о том, что до XX в. наука занималась в основном очень простыми системами, **обычно из двух переменных**. Перечень основных событий в науке с XVII по XIX в. по большой части состоит из вариаций одной и той же темы: выявление скрытой простоты в ситуации, представляющейся сложной. Подобные ситуации характерны тем, что выделяются несколько существенных факторов, а множество других считаются несущественными. Это позволяет исследователю ввести сильные, экспериментально оправданные упрощения и, следовательно, рассматривать исследуемые характеристики «изолированно» от всех остальных.
Множество ситуаций, в которых из большого числа факторов удается выделить несколько существенных, имеется в физике, чего нельзя сказать о других областях науки. Именно этим объясняются значительные успехи физики, далеко обогнавшей другие науки.
Именно Ньютон положил начало этому развитию, показав, что в физике возможны существенные упрощения. Его закон всемирного тяготения, считающийся одним из высочайших достижений

человеческого ума, является следствием очень сильных упрощений. Тем не менее при правильных вычислениях он позволяет очень точно вычислить орбиты планет.
Для проведения расчетов по закону Ньютона необходим, разумеется, соответствующий математический аппарат. Этот аппарат (также предложенный Ньютоном, возможно Лейбницем)—дифференциальное исчисление. Ньютон разработал его как инструмент для решения задач, связанных с простыми физическими системами, подобными тем, что подчиняются его закону. Он, можно сказать, приспособил этот аппарат к физической простоте, им самим открытой.
Наука почти до 1900 г. находилась под исключительным влиянием достижений Ньютона. Его мощные упрощения применялись в самых разных областях, что при исследовании некоторых физических явлений, таких, как электричество, магнетизм, гидромеханика, дало отличные результаты, однако в других науках, особенно в биологии и медицине, этот подход не сработал. Задачи, которыми занималась наука и которые она умела решать, были в основном **задачами на детерминированных системах с двумя или тремя переменными**. **Они представлялись аналитически, обычно в виде систем дифференциальных уравнений. Подобные задачи — с малым числом переменных, высокой степенью детерминизма, решение которых ищется в аналитической форме, — обычно называют задачами *организованной простоты.***
В конце XIX в. некоторые физики занялись исследованием систем, описывающих движение молекул газа в замкнутом объеме Такие системы обычно состояли примерно из 10^{23} молекул. Эти молекулы обладают огромными скоростями, а их траектории из-за постоянных столкновений имеют причудливый вид. Нельзя отрицать, что это очень сложная система. Очевидно, что из-за сильных упрощений закон Ньютона при изучении таких систем неприменим.
Совершенно безнадежно пытаться решить задачу анализа движения молекул газа в замкнутом объеме (т. е. задачу анализа исключительно сложной и неорганизованной системы) с помощью средств и методов решения задач в условиях организованной простоты. В этом случае нужен совершенно новый подход. И группа ученых, прежде всего Л. Больцман и Дж. Гиббс, создала мощные статистические методы решения задач для систем с большим числом переменных, проявляющихся весьма случайным образом. **Подобные задачи получили название задач *неорганизованной сложности.***
Статистические методы не описывают отдельные переменные (например, движение отдельной молекулы). Они предназначены для

определения небольшого числа общих характеристик. Поясним это, приведя отрывок из У. Уивера:
«Классическая динамика XIX в. позволяла хорошо описывать движение шара из слоновой кости по бильярду. Типичной упрощенной задачей этого периода является определение моментов времени, в которые шар будет находиться в определенных положениях. Можно, хотя это много сложнее, анализировать движение двух и даже трех шаров по бильярду. Однако если мы попытаемся проанализировать одновременное движение 10 или 15 шаров, то окажется, что задача стала нерешаемой, причем не потому, что возникают какие-то теоретические проблемы, а из-за объема вычислений.
Представим теперь огромный бильярд и миллионы сталкивающихся друг с другом и с бортиками шаров. Как ни странно, эта задача оказывается проще, поскольку к ней применимы методы статистической механики. При этом, разумеется, нельзя описать траекторию какого-то отдельного шара, но зато можно достаточно точно ответить на такие важные вопросы, как: **сколько шаров в среднем ударится за 1 с в данный бортик; каков средний пробег шара до его столкновения с другим шаром, сколько в среднем столкновений происходит с шаром за 1 с?**
Выше говорилось, что новые статистические методы применимы к задачам неорганизованной сложности. Применимо ли слово «неорганизованная» к биллиарду с множеством шаров? Применимо, поскольку **методы статистической механики** обоснованно использовать, когда как по положению, так и по скоростям и направлениям шары распределены беспорядочным образом, т. е. неорганизованно. **Эти методы будут неприменимы, если расположить шары в один ряд параллельно одному из бортиков и заставить их двигаться строго параллельно друг другу и перпендикулярно этому бортику.** Тогда **шары будут сталкиваться только с двумя бортиками и никогда друг с другом и ситуации неорганизованной сложности не возникнет.»**
Разработанные статистические методы успешно применялись для решения многих задач неорганизованной сложности, возникающих как в науке, так и в других областях. Хорошо известны такие примеры успешного применения этих методов в статистической механике, термодинамике и статистической (или количественная) генетике. В технике эти методы играют важную роль при создании больших телефонных сетей и компьютерных систем с разделением времени, при решении задач обеспечения технической надежности и т. д. В деловой сфере эти методы широко используются при решении задач маркетинга, страхования и т. п.

В отличие от используемых при организованной простоте аналитических методов, которые оказываются неприменимы уже при относительно небольшом числе переменных (например, при пяти), **точность и уместность использования статистических методов возрастает с ростом числа переменных**. Таким образом, эти два метода являются взаимодополняющими. Они соответствуют двум противоположным концам спектра сложностей и, несмотря на взаимодополняемость, покрывают только небольшую часть всего спектра сложностей. В свою очередь, это означает, что, за исключением двух концов спектра сложностей, он остается не обеспеченным методологически в том смысле, что для него не годятся ни аналитические, ни статистические методы. **Задачи, связанные со средней частью спектра сложностей, называют *задачами организованной сложности.*** Происхождение этого названия хорошо описано У. Уивером:
«Этот метод решения задач неорганизованной сложности, столь мощный по сравнению со случаем двух переменных, для очень большой части спектра сложностей оказывается неприменимым. **Он имеет тенденцию к всеупрощению и в смысле научной методологии представляет собой переход из одной крайности в другую — от двух переменных к астрономическому числу переменных, оставляя нетронутой все промежуточные значения переменных.** Более того, важность этого класса задач не зависит напрямую от того, что число рассматриваемых переменных является средним или большим по сравнению с двумя и очень малым по сравнению с числом атомов в щепотке соли. На самом деле задачи из этого класса области часто имеют довольно значительное число переменных. По-настоящему важной характеристикой задач из данной области, задач, которые так мало изучены, является то, что в отличие ог неорганизованных ситуаций, к которым применимы статистические методы, в этих задачах существенно такое свойство как *организованность...* Такую группу задач можно назвать задачами *организованной сложности* Это новые задачи, их будущее во многом зависит от их решения. **Наука должна совершить третий рывок, может быть, более великий, чем «поколение» в XIX в. задач типа *организованной простоты* и победа в XX в. над задачами *неорганизованной сложности.*** Наука должна научиться решать задачи первого типа.
Примеров задач со свойствами организованной сложности великое множество, особенно в науках, изучающих жизнь, поведение, общество и окружающую среду, а также в таких прикладных областях, как искусственный интеллект, современная технология и медицина.

Некоторые из этих задач имеют фундаментальное значение, например проблема рака, изучение старения или такая обширная область, как сложные и разнообразные задачи, возникающие в искусственном интеллекте. Область современной технологии хорошо описана в лекции Дж. Данцига, прочитанной им в Международном институте прикладного системного анализа в Лаксенбурге (Австрия), сыгравшем важную роль в прорыве науки в новую для нее область организованной сложности:

Совсем не просто описать всю сложность современной технологии. Можно, например, начать с перечисления того, чем занято население небольшого города. С помощью телефонного справочника я составил такой список для г. Ричмонд, шт. Калифорния. Приведу пример занятий, начинающихся с букв BR: строительство мостов, столы для бриджа, радиостанции, брошюры, маклеры, бронза, щетки, броши, тормоза, коньяки, пайка, кирпичи, окраска зданий, безделушки. Всего я насчитал 6000 занятий.

Другой способ ощутить разнообразие материальной стороны жизни — посмотреть каталог электронных изделий. В нем перечислены тысячи и тысячи разного рода резисторов, конденсаторов, вакуумных трубок, транзисторов, кабелей, панелей, кнопок, переключателей, шкал, печатных схем, шкафов. Посмотрите каталог химических продуктов фирмы Sears Roebuck — в нем также предлагаются тысячи изделий. В современном университете может быть 100 разных кафедр. Правительство Соединенных Штатов только в Сан-Франциско имеет около 2000 контор, которые, как предполагается, выполняют различные функции на благо общества. Это мы говорим только о разнообразии, но у категории «сложность» есть и другие измерения. В модели *затраты-выход* Леонтьева для национальной экономики Соединенных Штатов вся промышленность разбита на 400 основных отраслей, и для каждой отрасли необходимо указать, сколько данных она получает от всех других отраслей. Полученная таблица 400x400 содержит 160 000 чисел. Каждый регион страны имеет свою таблицу *затраты-выход,* а таких регионов много. Число в таблице *затраты-выход* выражает зависимость одной отрасли от другой; другими зависимостями представлены сделки между регионами и отраслями, причем таких комбинаций может быть тоже очень много. Точно так же связаны друг с другом и страны.

Есть еще и *временные* зависимости: оборудование создается и содержится для использования в будущем, материалы накапливаются для использования в будущем; люди обучаются для работы в будущем. Есть также зависимости, связанные с *местоположением:* люди,

материалы и оборудование перемещаются на новые места, причем не только по поверхности Земли, но и под землей, и по воздуху.

Хотя можно достаточно легко определить «входы» и «выходы» для любой локальной области этой огромной и сложной деятельности, но задача заключается в том, чтобы одновременно отслеживать все эти взаимодействия. Нам известно, что такие мощные силы, как рост населения, недостаток сырья, продовольствия, энергии, концентрации накоплений и т. д., стремительно видоизменяют сложность этой задачи. Существует опасение, что структура взаимосвязей этих деятельностей может не выдержать подобных нагрузок. Если мы оставим эту систему без управления, может так случиться, что в ней возникнут разного рода отказы и провалы.»

По определению системы с организованной сложностью обладают многими свойствами, которыми, в принципе, нельзя пренебречь. По той же причине эти системы недостаточно сложны и случайны, чтобы для них можно было получить содержательные статистические оценки. Следовательно, к ним неприменима ни одна из двух стратегий упрощения, которыми располагает наука. И тем не менее упрощений в большинстве случаев избежать нельзя. Даже если задача, связанная с какой-то очень сложной системой, может быть успешно решена с помощью вычислительной техники без всяких упрощений, полученное решение должно быть в конечном счете упрощено настолько, чтобы человек, принимающий решение, мог им воспользоваться. Так как ни ньютонова, ни статистическая стратегия упрощения неприменимы, необходимо найти новые пути. В общем случае при хорошем упрощении для достигнутого сокращения сложности должна минимизироваться потеря нужной информации. Некоторые способы упрощения уже рассматривались в гл. 4 и 5.

Одним из способов работы с очень сложными системами, возможно самым важным, является допущение неточности при описании данных. В этом случае неточность имеет не статистическую, а более общую природу, хотя бы потому, что она включает возможность статистических описаний. Математический аппарат для этого подхода, разрабатываемый с середины 60-х годов, известен как «теория нечетких множеств». Суть и важность этой новой теории прекрасно описаны ее создателем Л. А. Заде:

«Согласно глубоко укоренившейся традиции научного мышления, отождествляющего понимание явления со способностью анализировать его количественно, все чаще проводятся попытки анализировать поведение «гуманитарных» систем так, будто это механические системы, описываемые разностными, дифференциальными или интегральными уравнениями.

Мы, по существу, утверждаем, что обычные количественные методы системного анализа совершенно не подходят для гуманитарных систем и вообще для любых систем, сложность которых сравнима с гуманитарными. **Это утверждение основано на принципе, который можно назвать *принципом несовместимости.*** Говоря неформально, суть этого принципа состоит в том, что с ростом сложности систем наша способность делать точные и содержательные утверждения об их поведении падает до определенного предела, за которым такие характеристики, как точность и содержательность (или реальность), становятся взаимоисключающими. (Отсюда следует принцип **«чем детальнее рассматривается реальная задача, тем более нечетким оказывается ее решение»**) **В этом смысле точный количественный анализ поведения гуманитарных систем не слишком подходит для решения реальных социальных, политических, экономических и других задач**

Альтернативный подход ... заключается в том, что ключевыми элементами мышления являются не числа, а метки нечетких множеств, т. е классов объектов, для элементов которых переход от принадлежности к непринадлежности классу является не резким, а постепенным. В самом деле, **«вездесущая» нечеткость человеческого мышления наводит на мысль, что логика рассуждений человека не является обычной двухзначной или даже многозначной логикой, но это — логика с нечеткими истинами, нечеткими отношениями и нечеткими правилами вывода. На наш взгляд, такая нечеткая и не вполне понятная логика является важнейшим компонентом одной из главных особенностей человеческого мышления, а именно способности *обобщать* информацию — выделять из огромных масс данных, обрушивающихся на мозг, только те, что нужны для решения конкретной задачи.**

По природе своей обобщение является аппроксимацией обобщаемого. Во многих случаях для удовлетворительного описания данных оказывается достаточно весьма приблизительного описания, поскольку **большинство выполняемых человеком работ не требует большой точности**. Используя эту терпимость к нечеткости, **мозг кодирует соответствующую информацию в терминах нечетких множеств, приближенно связанных с исходными данными** Таким образом, **поток информации, поступающий в мозг через зрение, слух, осязание и другие органы чувств, сокращается до тоненькой струйки информации, необходимой для очень приближенного решения определенной задачи.** А значит, возможность манипулировать нечеткими множествами (и, как следствие,

способность обобщать) является важнейшим достижением человеческого интеллекта, отличающим его от машинного интеллекта, реализованного на современных цифровых компьютерах. С этой точки зрения традиционные методы анализа систем не подходят для работы с «гуманитарными» системами, поскольку им не удается выявить реальную нечеткость мышления и поведения человека. **Таким образом, для радикального изменения работы с системами необходимы подходы, не фетишизирующие такие понятия, как точность, строгость, математический формализм, а использующие методологические схемы, содержащие неточность и неполную истинность.»**

Методологические средства для решения интеллектуальных задач, принадлежащих к категориям организованной простоты и неорганизованной сложности, достигли достаточно высокого уровня развития, и в прикладное математическое обеспечение компьютеров входят многочисленные пакеты статистического анализа, вычислительных методов и символьных преобразований. Иначе обстоит дело со средствами решения интеллектуальных задач из категории организованной сложности. Они разработаны еще недостаточно.

7.3. Меры сложности систем ИИ

При решении интеллектуальных задач сложность понимается двояко. Во-первых, сложность является свойством систем ИИ; во-вторых, она является свойством интеллектуальных задач. **Будем называть эти два типа сложности соответственно *сложностью систем ИИ* и *сложностью интеллектуальных задач.***

В этом разделе мы будем говорить о сложности систем ИИ. Некоторые проблемы, связанные со сложностью интеллектуальных задач, в литературе их часто называют проблемами *вычислительной сложности,* обсуждаются в разд. 7. 4 и 7. 5.

В схеме ФРИЗ системы ИИ имеют много граней, причем каждая грань соответствует одному из эпистемологических типов систем ИИ, введенных в гл. 1—6. Следовательно, и сложность для этих типов систем ИИ столь же многогранна. Иначе говоря, для разных типов систем ИИ из эпистемологической иерархии сложность понимается по-разному и по-разному должна исследоваться.

Вне зависимости от типа системы ИИ можно выделить **два общих *принципа оценки сложности систем ИИ;*** они применимы к системам любого типа и могут служить основой для сравнительного изучения сложности систем ИИ.

Согласно **первому принципу сложность системы ИИ (любого типа) должна быть пропорциональна объему информации, необходимой для описания этой системы ИИ. В данном случае слово «информация» понимается чисто синтаксически, а не семантически и не прагматически.** Одним из способов описания такой дескриптивной сложности (и, возможно, самым простым) является **оценка числа элементов, входящих в систему ИИ (переменных, состояний, компонентов), и разнообразия взаимозависимостей между ними.** В самом деле, при прочих равных условиях с ростом числа элементов или разнообразия взаимосвязей возрастают и трудности работы с системой ИИ. Есть множество других способов для выражения дескриптивной сложности, однако все они должны удовлетворять следующим требованиям.

Пусть X — множество всех систем ИИ определенного эпистемологического уровня, $\mathscr{P}(X)$ —мощность множества X, а C_X— *мера дескриптивной сложности на множестве X.* Тогда C_X — это функция

$$C_X : \mathscr{P}(X) \to R,$$

обладающая следующими свойствами:

C1) $C_X(\varnothing)=0$;

C2) если $A \subset B$, то $C_X(A) \le C_X(B)$;

C3) если A — гомоморфный образ B, то $C_X(A) \subseteq C_X(B)$;

C4) если A изоморфно B, то $C_X(A) = C_X(B)$;

C5) если 1) $A \cap B = \varnothing$, 2) A и B не взаимодействуют друг с другом, 3) A и B не являются гомоморфными образами друг друга, то

$$C_X(A \cup B) = C_X(A) + C_X(B)$$

Из свойств C1 и C2 следует, что **сложность любой системы ИИ характеризуется неотрицательным числом.** Свойства C2 и C3 связаны с таким **фундаментальным свойством, как монотонность: сложность не должна возрастать, если множество систем ИИ сокращается или они рассматриваются менее детально.** Условие C4 очевидно: **если переобозначить некоторые (произвольные) элементы заданных систем ИИ, а все остальное оставить без изменений, то сложность измениться не должна.** Свойство C5 — **свойство аддитивности**: **если объединяются два множества систем ИИ, не имеющих никаких общих компонентов (общих систем, взаимосвязей, морфизмов), то суммарная сложность должна быть равна сумме сложностей**

В соответствии со вторым общим принципом **сложность систем ИИ должна быть пропорциональна объему информации, необходимому для разрешения любой нечеткости, связанной с рассматриваемой нечеткостью.** В данном случае также имеется в виду синтаксическая информация, однако эта информация основывается на соответствующей *мере нечеткости* (разд. 4. 5).

Сложность систем ИИ изучается прежде всего для создания методов, с помощью которых она может быть снижена до приемлемого уровня. Упрощая систему ИИ, мы хотим упростить ее в обоих смыслах — и в смысле сложности, основанной на дескриптивной информации, и в смысле сложности, основанной на нечеткой информации. К сожалению, **эти два типа сложности друг с другом не согласуются**. Уменьшая в общем случае одну сложность, мы, как правило, увеличиваем вторую. Исходя из этих соображений, можно следующим образом сформулировать общую задачу упрощения.

Дана система ИИ некоторого эпистемологического уровня; обозначим через $\mathscr{X}$ множество всех ее содержательных упрощений.

Обозначим $\overset{d}{\preceq}$ и $\overset{u}{\preceq}$ соответственно упорядочения множества $\mathscr{X}$, основанные на **дескриптовой информации и на нечеткой информации**. Эти упорядочения, вообще говоря, являются слабыми (т. е. рефлексивными и транзитивными отношениями на $\mathscr{X}$).

Обозначим $\overset{\alpha}{\preceq}$, $\overset{\beta}{\preceq}$, ... другие (дополнительные) упорядочения (**слабые, частичные или полные**), которые представляют собой некоторые предпочтения, определенные пользователем на данной системе ИИ. Определим *объединенное упорядочение по предпочтению* $\overset{*}{\preceq}$ для всех имеющихся упорядочений следующей формулой:

$$(\forall x, y \in \mathscr{X})\left(x \overset{*}{\preceq} y \Leftrightarrow x \overset{d}{\preceq} y \text{ и } x \overset{u}{\preceq} y \text{ и } x \overset{\alpha}{\preceq} y \text{ и } x \overset{\beta}{\preceq} y \text{ и } \ldots\right)$$

Множество решений $\mathscr{X}_S$ задачи упрощения состоит из тех систем ИИ множества $\mathscr{X}$, которые либо являются эквивалентными, либо несравнимы в смысле объединенного упорядочения по предпочтению $\overset{*}{\preceq}$. Формально

$$\mathscr{X}_S = \{x \in \mathscr{X} \mid (\forall y \in X)(y \overset{*}{\preceq} x \Rightarrow x \overset{*}{\preceq} y)\}. \quad (1)$$

Эта общая формулировка задачи упрощения напоминает три частных случая этой задачи:

задачу определения подходящих систем ИИ с поведением (разд. 4.6);

задачу упрощения порождающих систем ИИ с помощью разрешающего укрупнения (разд. 4.9);

задачу реконструкции (разд. 5.7).

Все эти задачи соответствуют формулировке общей задачи упрощения, отличаясь друг от друга множеством $\mathscr{X}$ и математическими свойствами порядков предпочтения $\overset{d}{\preccurlyeq}$ и $\overset{u}{\preccurlyeq}$, определенных на $\mathscr{X}$.

Одним из основных принципов ФРИЗ является то, что пользователю позволяется определить собственное упорядочение по предпочтению для рассматриваемых систем. Если пользователь в качестве одного из критериев указывает сложность, но не определяет меру сложности, ФРИЗ должен предложить ему выбрать меру из списка возможных вариантов. Если пользователь не желает выбирать сам, ФРИЗ должен использовать некоторую меру сложности, выбираемую им по умолчанию.

Предлагаемые пользователю меры сложности должны быть соответствующим образом обоснованы и применимы к рассматриваемым задачам.

7.4. Предел Бреммерманна

Ханс Бреммерманн утверждает: «*Не существует системы обработки данных, искусственной или естественной, которая могла бы обрабатывать более чем $2 \cdot 10^{i7}$ бит в секунду на грамм своей массы.*»

Под «обработкой *N* бит» понимается пересылка *N* бит по одному или нескольким каналам вычислительной системы. Бреммерманн пришел к своему выводу, исходя из следующих соображений.

Для работы информация должна быть каким-то образом физически закодирована. Предположим, что она закодирована в виде энергетических уровней определенного типа энергии в интервале $[0, E]$, где E — количество энергии, которым мы располагаем для этой цели. Предположим далее, что энергетические уровни измеряются с точностью до ΔE. При этом весь интервал можно разделить максимум на $N=E/\Delta E$ равных подынтервалов, причем каждому будет соответствовать энергия, равная ΔE. Если всегда будет занято не более одного уровня (задаваемого маркером подынтервала), то максимальное число битов, представимых с помощью энергии E, будет равно

$$\log_2(N+1)$$

(в формуле $N+1$, поскольку следует учесть случай, когда не занят ни один уровень). Если вместо одного маркера с энергетическими уровнями из интервала $[0, E]$ использовать одновременно K маркеров $(2 \le K \le N)$, то можно представить

$$K\log_2(1+N/K)$$

бит. Оптимальное использование имеющейся энергии E получается при использовании N маркеров. В этом оптимальном случае можно представить N *бит* информации.

Для того чтобы представить больший объем информации при том же количестве энергии, необходимо сократить ΔE. Это возможно только до некоторого предела, так как нужно различать полученные уровни с помощью какой-то измерительной процедуры, которая независимо от ее сути всегда имеет ограниченную точность. Максимальная точность определяется принципом неопределенности Гейзенберга: энергия может быть измерена с точностью до ΔE, если выполняется неравенство

$$\Delta E \Delta t/h,$$

где Δt —длительность времени измерения, h=6.625×10^{-27} эрг/с— постоянная Планка, а ΔE определяется как среднее отклонение от ожидаемого значения энергии. Это значит, что

$$N \le E\Delta t/h. \qquad (2)$$

Представим теперь имеющуюся энергию E соответствующим количеством массы согласно формуле Эйнштейна

$$E=mc^2,$$

где c = 3·10^{10} см/с — скорость света в вакууме. Таким образом, верхняя, наиболее оптимистическая граница для N согласно неравенству (.2)

$$N = mc^2\Delta t/h. \qquad (3)$$

Подставив значения для *сяк,* имеем

$$N = 1.36m\Delta t \times 10^{47}. \qquad (4)$$

Для массы 1 г (m = 1) и времени 1 с (Δt =1) получаем указанное значение

$$N = 1.36 \times 10^{47}.$$

Используя полученный предел для обработки информации граммом массы за 1 с процессорного времени, Бреммерманн затем вычислил число бит, которые могла бы обработать гипотетическая компьютерная система, имеющая массу, равную массе Земли, за период, равный примерному возрасту Земли. Поскольку масса Земли оценивается примерно в 6·10^{27} г, а возраст в 10^{10} лет, причем год состоит приблизительно из 3,14·10^{7} с, этот воображаемый компьютер смог бы обработать порядка 2.56·10^{97} бит или, округляя до степени 10, порядка 10^{93} бит. **Это число (10^{93} бит) обычно называют <u>пределом Бреммерманна</u>, а задачи, требующие обработки более чем 10^{93} бит информации, называются *трансвычислительными задачами.***

Сразу видно, что предел Бреммерманна является весьма строгим ограничением, несмотря на то, что это число получено при весьма слабых условиях (более обоснованные предпосылки дают число значительно меньше 10^{93}). В самом деле, решение многих задач для систем даже относительно небольшого размера требует большего, чем

указанный предел, объема обработки информации. Рассмотрим, например, систему из n переменных с k разными состояниями каждая. Понятно, что эта система, в принципе, имеет k^n состояний. Множество обобщенных состояний конкретной системы является подмножеством этого множества. Всего таких подмножеств 2^{k^n}. Предположим, что нам нужно отобрать, выделить или классифицировать систему ИИ из множества всех систем ИИ этого типа. Тогда при условии, что используется самый эффективный метод поиска, при котором каждый бит информации (как ответ на дихотомический вопрос) позволяет разбить оставшееся множество вариантов пополам, необходимо обработать

$\log_2 2^{k^n} = k^n$ бит информации. Задача является

трансвычислительной, когда $k^n > 10^{93}$.

Это происходит, например, при следующих значениях k и n:

k	2	3	4	5	6	7	8	9	10
n	308	194	154	133	119	110	102	97	93

Проблема трансвычислительности возникает в самых разнообразных контекстах, например при распознавании образов. Рассмотрим, скажем, массив $q \times q$ типа шахматной доски, причем каждая клетка может быть одного из k цветов. Всего может быть k^n шаблонов раскраски, где $n=q^2$. Предположим, что нам нужно определить наилучшую (с точки зрения определенного критерия) классификацию этих шаблонов. Для этого необходимо провести поиск среди всех возможных классификаций этих шаблонов. Если классов всего два, то эта задача изоморфна предшествующей. Например, для массива 18×18 задача становится трансвычислительной при двух цветах, для массива 10×10— при девяти. **Проблема распознавания образов непосредственно связана с психологическими исследованиями сетчатки, но сложность этих исследований огромна. Сетчатка состоит приблизительно из миллиона светочувствительных колбочек.** Если даже (для простоты) считать, что колбочки имеют только два состояния (активное и пассивное), то **исследование сетчатки в целом потребует обработки**

$2^{1\,000\,000} = 10^{300\,000}$

бит информации. Это число находится далеко за пределом Бремерманна

Та же проблема возникает и в такой области, как тестирование больших интегральных микросхем (БИС). Они представляют собой

сложные электронные платы с большим числом входов и выходов. Для соответствующих электрических сигналов (каждый сигнал имеет два идеальных состояния) каждый выход представляет определенную логическую функцию от логических переменных, соответствующих входам. Тестирование определенной интегральной микросхемы представляет собой анализ «черного ящика»: **определение реализуемых ею логических функций осуществляется только с помощью задания значений входных переменных и наблюдения значений выходных переменных.** Для каждой выходной переменной тестирование, по существу, сводится к рассмотренной выше задаче при $k = 2$ (если только не используется многозначная логика). Отсюда следует, например, что тестирование схем с 308 входами и 1 выходом представляет собой трансвычислительную задачу. Однако хорошо известно, что для задачи тестирования на практике пределы сложности существенно ниже. На самом деле, многие выпускаемые БИС полностью проверить нельзя. Проблема состоит в разработке практически реализуемых методов тестирования, гарантирующих *почти полное* тестирование, например проверку более чем 90% возможных состояний.

Подробнее сложность этой задачи рассмотрена на рис. 1.

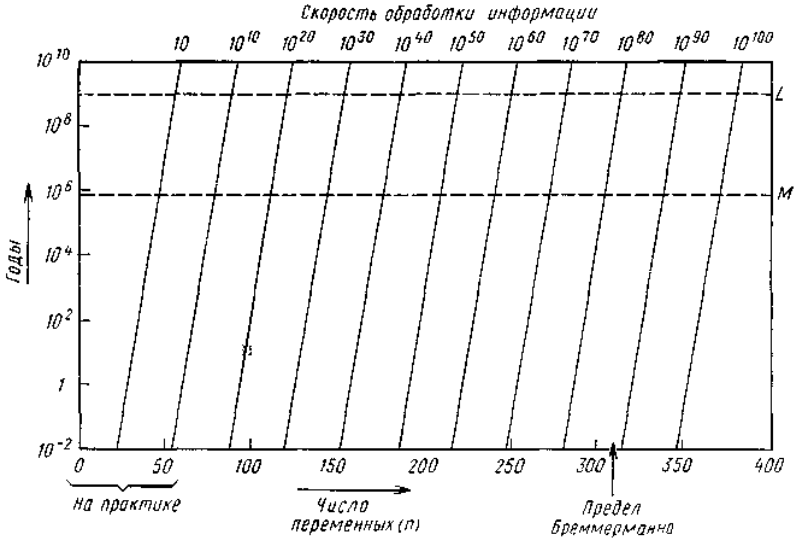


Рис. 1. Время, необходимое для выбора или идентификации логической функции от n переменных при скорости обработки информации, равной 10, 10^{10}, ..., 10^{100} бит/с

На рисунке показана зависимость времени (в годах), необходимого для выбора (идентификации, классификации и т. д.) логической функции, от числа *n* переменных при различных скоростях обработки информации, варьирующихся от 10 до 10^{100} бит/с. На рисунке также указаны два важных периода времени: *L* — это приблизительный возраст жизни на Земле, а *M* — примерное время появления человека.

Пример с тестированием схем ни в коем случае не является исключением. Настоящие интеллектуальные задачи требуют, к сожалению, огромных мощностей по обработке информации. В настоящей работе приводится много конкретных примеров таких задач. Об этом же пишет Бреммерманн:

«**Опыт ученых, решающих интеллектуальные задачи, занимающихся доказательством теорем и распознаванием образов, говорит об одном — это очень сложные задачи.** Вероятно, не существует столбовой дороги или какого-то простого метода, который позволил бы одним махом решить все наши задачи. Вывод, который можно сделать из данной статьи о пределах скорости и объема обработки данных, сводится к следующему: задачи, имеющие очень большое число вариантов, нельзя решить даже за счет абсолютного увеличения объема обрабатываемых данных. Необходимо разрабатывать всевозможные качественные приемы Разумеется, более быстрые компьютеры будут очень нужны. Однако, если говорить о принципиальных задачах, то уже нынешние компьютеры имеют почти такое же быстродействие, каким они будт обладать когда-нибудь.»

Если задача является трансвычислительной, то, чтобы ее можно было решить, она должна быть переформулирована. Наиболее естественный способ состоит в ослаблении условий. Например, **вместо лучшего решения можно искать просто хорошее, вместо точного — приближенное и т. д**. **Ослабление условий позволяет применять или эвристические методы, позволяющие отбрасывать множество неперспективных вариантов, или приближенные (нечеткие) методы, позволяющие работать с совокупностями вариантов.**

В оценке сложности задачи пользователю должен помочь ФРИЗ и, если задача не решается с помощью имеющихся вычислительных средств, предложить ему подходящие способы модификации его задачи. Предел Бреммерманна дает слишком простое разбиение интеллектуальных задач по сложности. Реальных, практических вычислительных ограничений он не отражает. Тем не менее, согласно Эшби, он является полезной вехой при предварительной оценке ситуации:

«Одно из наиболее очевидных следствий, которым сейчас практически пренебрегают, заключается в том, что, прежде чем начать исследование какой-то сложной системы, необходимо хотя бы приблизительно оценить информационные запросы. Если нужно 2000 бит, то все в порядке, но если оценка равна 10^{300} бит, то необходимо стратегически поменять метод.»

Конкретные вычислительные средства определяют, разумеется, более строгие ограничения на сложность задач, чем предел Бреммерманна— 10^{93}.

Выше неоднократно подчеркивалось, что **одной из главных задач ФРИЗ является создание человеко-машинного симбиоза (симбиоза пользователь-компьютер), в котором способности обоих симбионтов взаимно дополняют друг друга, позволяя эффективно решать задачи**. И хотя такой симбиоз представляется наилучшим методом решения сложных задач, он также не позволяет обойти предел Бреммерманна. **По сути, этот предел говорит о фундаментальных пределах наших знаний**. Эшби пишет:

«Очевидно, что наш мозг и мы сами материальны, а следовательно, ограничены. Ограничена и вся мировая наука, поскольку она также материальна. Вся информация, которой располагаю лично я, и вся информация, используемая мировой наукой, не превосходи: 10^{80} бит (Эшби получает значение 10^{80} из предела Бреммерманна, определенного для 1 с и 1 г, рассматривая «столетия и тонны компьютеров» (примерно 10 тыс. столетий и 10^{15} т). В данном случае разница между 10^{80} и 10^{93} несущественна.) Какой бы ни стала наука в будущем, этот потолок достигнут не будет

Мы не можем претендовать на какие-то особые преимущества в силу выдающегося положения в живой природе. Такие, как мы есть, получились в процессе естественного отбора. Являясь отбором, этот **процесс может быть оценен через некую информационную меру, и, следовательно, он имеет свои пределы.** При любом отборе на любой планеге поверхность планеты материальна, и приспособление не может идти быстрее этого предела. Как бы хорошо мы ни мыслили, предела в 10^{80} бит мы не достигнем. **Разум науки будущего не сможет использовать более чем 10^{80} бит информации, и сама наука будет только стремиться к этому пределу. Это наша информационная вселенная, и все, что находится за ее пределами, непознаваемо.**

7.5. Вычислительная сложность

Рассмотренный в предыдущем разделе предел Бреммерманна подходит для оценки задач с точки зрения объема информации, которую нужно обработать, однако есть и **другие критерии оценки задач**. Задача может укладываться в предел Бреммерманна и тем не менее быть практически нерешаемой. Следовательно, необходимо уточнить понятие сложности задачи.

Вычислительные характеристики задач исследуются общей теорией алгоритмов. В эту общую теорию входят три раздела: теория вычислимости, проектирование алгоритмов и теория вычислительной сложности. Подробный разбор этих разделов выходит за рамки данной книги. Однако было бы полезно дать краткий обзор этой теории, прежде всего результатов и подходов, используемых при решении интеллектуальных задач. Доказательств мы приводить не будем.

Интуитивно ***алгоритм* представляет собой набор команд на неком языке, описывающих выполнение действий по решению задачи определенного типа. Требуется, чтобы алгоритм был финитным (конечным), т. е. чтобы он завершался после конечного числа шагов (действий).**

Есть несколько формализации интуитивного понятия алгоритма, включая машины Тьюринга, алгоритмы Маркова и рекурсивные функции. Доказано, что эти определения эквивалентны. *Машина Тьюринга,* например, представляет собой простое устройство, состоящее из *автомата* с конечным числом состояний и *ленты*. Автомат обладает памятью, что позволяет ему находиться в одном из состояний, принадлежащих конечному множеству состояний, скажем множеству $Z=\{z_1, z_2,..., z_n\}$. **Потенциально бесконечная в обоих направлениях лента разбита на отрезки одинаковой длины — ячейки**. В каждой ячейке записана буква из конечного набора букв $X=\{x_0, x_1,...,x_m\}$ *алфавита.* Одна из букв, например x_0, всегда интерпретируется как *пробел* (пустая ячейка). **Связь между автоматом и лентой осуществляется с помощью *читающей-пишущей головки,* которая может считать букву с ленты или записать букву на ленту.** Одновременно головке доступна только одна ячейка

Автомат машины Тьюринга на каждом шаге изменяет свое состояние и выполняет действие одного из следующих трех типов:

I. Записывает на ленту вместо текущей буквы новую.
II. Сдвигается по ленте на одну ячейку влево или вправо.

III. Прекращает вычисление (операция остановки).

Новое состояние и выполняемое действие однозначно определяются текущим состоянием и считываемой с ленты буквой.

Пусть z_c и z_n соответственно текущее и следующее состояния машины Тьюринга, x_m — буква, читаемая с ленты, а y_p — выполняемая операция. Тогда при заданной на ленте начальной строке букв (эта строка не должна содержать пробелов) и определенном начальном состоянии работа машины Тьюринга определяется упорядоченным множеством четверок

$$(z_c , x_m, z_n, y_P).$$

Машина Тьюринга называется *детерминированной,* если запрещается, чтобы любые две четверки из этого множества начинались с одной и той же пары $z_c, x_r;$ в противном случае машина Тьюринга называется *недетерминированной.*

Общепринятая гипотеза, известная как тезис Черча (или тезис Черча — Тьюринга), утверждает, что если функцию можно вычислить на детерминированной машине Тьюринга, то эта функция считается вычислимой. Согласно этой гипотезе **машина Тьюринга является точным формальным эквивалентом интуитивного понятия алгоритма**. Математически эта гипотеза недосказуема, однако опыт и неформальные соображения ее подтверждают. Опровергнуть эту гипотезу можно, предложив альтернативное определение вычисления, полностью соответствующее интуитивным представлениям и допускающее описание таких вычислений, которые недоступны машине Тьюринга. Маловероятно, что такой формализм существует.

В общем случае задача считается неразрешимой, если не существует алгоритма, с помощью которого можно получить решение. Детерминированные машины Тьюринга совместно с тезисом Черча дают аппарат для формального определения существования алгоритмов решения различных задач. Для доказательства неразрешимости задачи достаточно доказать, что ее нельзя решить на машине Тьюринга. Подобное доказательство неразрешимости уже получено для ряда задач.

Сведения о неразрешимости задач должны включаться в ФРИЗ только в том случае, если схема ФРИЗ позволяет идентифицировать такие задачи. Если пользователь попросит решить задачу, про которую известно, что она неразрешима, ФРИЗ должен не только отказаться от ее решения, но и дать пользователю соответствующее объяснение причин отказа, включая ссылки на нужную литературу.

Неразрешимые задачи образуют один из трех классов задач. Во второй класс входят задачи, про которые не доказано, что они являются неразрешимыми, но для которых не найдены решающие

алгоритмы. Таким образом, это задачи, про которые неизвестно, разрешимы они или нет. Если пользователь потребует, чтобы ФРИЗ решил такую задачу, ему нужно выдать сообщение, что статус этой задачи неизвестен и что, к сожалению, ничего больше сделать нельзя. Остальные задачи являются разрешимыми, т. е. они в принципе разрешимы. Однако на практике многие из них решить нельзя, поскольку их решение требует слишком больших вычислительных ресурсов, таких, как время вычисления или память. Поскольку необходимое время вычислений обычно является единственным фактором, определяющим практическую разрешимость задачи, то и вычислительная сложность в основном изучается с точки зрения именно этого ресурса.

На практике разрешимость задачи зависит
от применяемого алгоритма решения задачи;
конкретной системы, рассматриваемой в задаче;
имеющихся вычислительных мощностей.

При заданном конкретном алгоритме решения задачи, время ее решения удобно представлять переменной, зависящей от размера рассматриваемых систем. **Эта переменная, которую часто называют *размерностью варианта задачи,* задает объем входной информации, необходимой для описания этих систем.**

Пусть n — размерность конкретных систем ИИ для некоторого варианта задачи. Тогда время выполнения алгоритма решения этой задачи задается функцией

$$f: R \rightarrow R \qquad (5)$$

такой, что $f(n)$—наибольшее количество времени, необходимое для выполнения алгоритма, решающего вариант задачи, размерность которого равна n. Функцию f обычно называют *временной функцией сложности.*

Считается, что можно выделить **два класса алгоритмов**, **отличающихся скоростью роста их временных функций сложности.** К первому классу принадлежат алгоритмы с **полиномиальными временными функциями сложности**. Они называются *полиномиально-временными алгоритмами.* Поскольку степень полинома имеет существенно большее значение, особенно при больших n, чем его коэффициенты и члены меньших порядков, то полиномиальные временные функции сложности можно характеризовать их порядком. Функция f имеет сложность $O(n^k)$, где k— положительное целое число, тогда и только тогда, когда существует константа $c>0$ такая, что

$$f(n) \le cn^k$$

для всех $n \ge n_0$, где n_0— наименьшая размерность рассматриваемой задачи. Например, функция

$$f(n)=25n^2+18n+31$$

имеет сложность $O(n^2)$, поскольку

$$f(n) \le 74n^2$$

при $n_0 = 1$ или

$$f(n) \le 42n^2$$

при $n = 2$ и т. д.

Ко второму классу алгоритмов относятся алгоритмы, временные функции сложности которых превосходят сложность $O(n^k)$ при любом k . Они обычно называются *экспоненциально-временными алгоритмами.*

Различие между полиномиальными и экспоненциальными временными алгоритмами, особенно для больших задач, очень велики. Это показано в табл. 1, в которой приведены скорости роста нескольких временных функций сложности.

Таблица 1.

Скорости роста некоторых полиномиально- и эксплуатационно-временных функций сложности

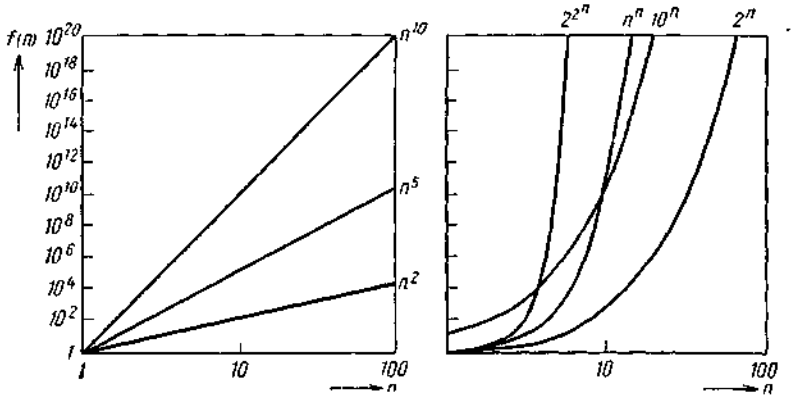
Временная функция сложности	Размерность варианта задачи n						
	1	10	20	30	40	50	100
n	0.000001 с	0.000001 с	0.00002 с	0.00003 с	0.00004 с	0.00005 с	0.0001 с
n^2	0.000001 с	0.0001 с	0.0004 с	0.0009 с	0.0016 с	0.0025 с	0.01 с
n^5	0.000001 с	0.1 с	3.2 с	24.3 с	1.7 мин	5.2 мин	2.8 ч
n^{10}	0.000001 с	2.8 ч	118.5 дней	18.7 лет	3 3 в	31.0 в	$3.2\ 10^4$ в
2^n	0.000002 с	0.001 с	1.0 с	17,9 мин	12.7 дней	35.7 лет	$4\ 10^{14}$ в
3^n	0.000003 с	0.059 с	58 мин	6.5 лет	3855 в	$2\cdot10^6$ в	$1.6\cdot10^{32}$ в
10^n	0.000001 с	2 8 ч	$3.2\cdot10^4$ в	$3.2\cdot10^{14}$ в	$3.2\cdot10^{24}$ в	$3.2\cdot10^{34}$ в	$3.2\cdot10^{24}$ в
2^{2^n}	0.000004 с	$5.7\cdot10^{292}$ в	$10^{3\ 10^5}$ в	$10^{3\ 10^8}$ в	$10^{3\ 10^{11}}$ в	$10^{3\ 10^{14}}$ в	$\approx10^{3\ 10^{24}}$ в
n^n	0.000001 с	2.8 ч	$3.3\cdot10^{10}$ в	$6.5\cdot10^{28}$ в	$3\ 8\cdot10^{48}$ в	$\approx2.8\cdot10^{69}$в	$\approx3.2\cdot10^{184}$в
$n!$	0.000001 с	3.6 с	771,5 в	$8.4\cdot10^{16}$ в	$2.6\cdot10^{32}$ в	$\approx9.6\cdot10^{48}$в	$\approx2\ 9\cdot10^{142}$в

Времена вычислений приведены для скорости вычислений, равной 1 млн. операций в секунду. Если сравнить, например, n^2 и n^{10}, то сразу видно, что практическая применимость алгоритмов сильно зависит от степени полиномиально-временной функции сложности. Однако (за исключением небольших значений n) полиномиально-временные алгоритмы существенно лучше «реагируют» на увеличение мощности вычислительных средств. Это отличие хорошо видно при сравнении графиков некоторых полиномиальных и экспоненциальных функций, изображенных на рис. 2, а также по тому, как в действительности растет диапазон решаемых задач с ростом скорости вычислений ЭВМ (см. формулы, приведенные в табл. 2).

Таблица 2.

Влияние роста скорости вычислений ЭВМ на диапазон решаемых задач для некоторых временных функций сложности

Временная функция сложности	Размерность наибольшего варианта задачи, решаемого за единицу времени с помощью: имеющейся вычислительной техники	в 10^2 раз более производительной вычислительной техники	в 10^3 раз более производительной вычислительной техники	в 10^6 раз более производительной вычислительной техники	в X раз более производительной вычислительной техники
n	n_1	$100n_1$	$1000\,n_1$	$1\,000\,000n_1$	Xn_1
n^2	n_2	$10n_2$	$31.6n_2$	$1000n_2$	$\sqrt{X}\,n_2$
n^5	n_3	$2.5n_3$	$3.98n_3$	$15.8n_3$	$\sqrt[5]{X}n_3$
n^{10}	n_4	$1.58n_4$	$2n_4$	$3.98n_4$	$\sqrt[10]{X}\,n_4$
2^n	n_5	$n_5+6.64$	$n_5+9,97$	$n_5+19.93$	$n_5+\log X/\log 2$
3^n	n_6	$n_6+4.19$	$n_6+6,29$	$n_6+12.58$	$n_6+\log X/\log 3$
10^n	n_7	n_7+2	n_7+3	n_7+6	$n_7+\log X$



Рие. 2. Графики некоторых типичных временных функций сложности; а) полиномиальных; с) экспоненциальных

Поскольку полиномиально- и экспоненциально-временные функции сложности отличаются весьма существенно, полиномиально-временные алгоритмы считаются *эффективными,* а экспоненциально-временные *неэффективными.* Как следствие, задачи, которые нельзя доказать, что они решаются с помощью полиномиально-временных алгоритмов, рассматриваются как *не поддающиеся решению,* а задачи, для которых известны полиномиальные алгоритмы,— как *поддающиеся решению.* Эти задачи

обычно называют Р-задачами (т. е. задачами, решаемыми за полиномиальное время); множество таких задач называется *классом* Р-задач,

Известно, что различия в системах кодирования, используемых для записи алгоритмов (как и отличия типов ЭВМ) не влияют на то, считается задача поддающейся решению или нет. Стандартные системы программирования и типы компьютеров отличаются друг от друга в лучшем случае полиномиально. Поэтому практическая разрешимость задачи зависит от системы программирования и типа ЭВМ, а поддается она решению в принципе или нет, не зависит. Поэтому для множества интеллектуальных задач неизвестно, существует ли полиномиально-временной алгоритм их решения, и не доказано, что они не поддаются решению. Общим для этих задач является то, что они могут быть «решены» за полиномиальное время на недетерминированных компьютерах, например на машинах Тьюринга. Такие задачи называются *NP-задачами* (недетерминированными полиномиально-временными задачами); они образуют *класс NP-задач.* Под решением здесь понимается следующее: **машина может проверить правильность предложенного решения за полиномиальное время.** Следовательно, в данном случае понятие недетерминированный полиномиально-временной алгоритм служит лишь обозначением того, что предложенное решение реальной задачи может быть проверено за полиномиальное время. Известно, что любая *NP* -задача решается с помощью детерминированного алгоритма сложности $O(2^{p(n)})$, где p — полиномиальная функция.

Класс *NP* -задач содержит класс *P,* так как любая задача, решаемая за полиномиальное время на детерминированной машине Тьюринга, решается (т. е. проверяется) за полиномиальное время на недетерминированной машине Тьюринга. Для знательного числа *NP* -задач доказано, что любая другая *NP* –задача может быть сведена к такой задаче за полиномиальное время. Эти задачи называются *NP-полными задачами.*

Поскольку в класс *NP* -задач входит много практически важных задач, чрезвычайно существенно определить его статус. Вопрос о том, поддаются *NP* -задачи решению или нет, является одним из самых важных вопросов математики, информатики и науки о системах ИИ. Он имеет огромное значение для решения интеллектуальных задач. Этот вопрос обычно формулируется следующим образом: «верно ли, что *NP* =P?». Ответить на него можно, доказав, что любая *NP* -задача либо является Р-задачей (т. е. решается за полиномиальное время), либо принципиально не поддается решению (т. е. решается за экспоненциальное время). Если про какую-то *NP* -полную задачу будет

доказано, что она не поддается решению, то $NP \neq P$. С другой стороны, если будет доказано, что такая задача решаема, то $NP = P$. Поскольку имеются сильные доводы в пользу того, что $NP \neq P$ при обычных правилах вывода, вопрос состоит прежде всего в том, чтобы найти некие нетрадиционные правила вывода, при которых можно было бы доказать, что какая-то *NP* -полная задача поддается решению.
На рис. 3 приведена классификация задач с точки зрения их разрешимости и вычислительной сложности.

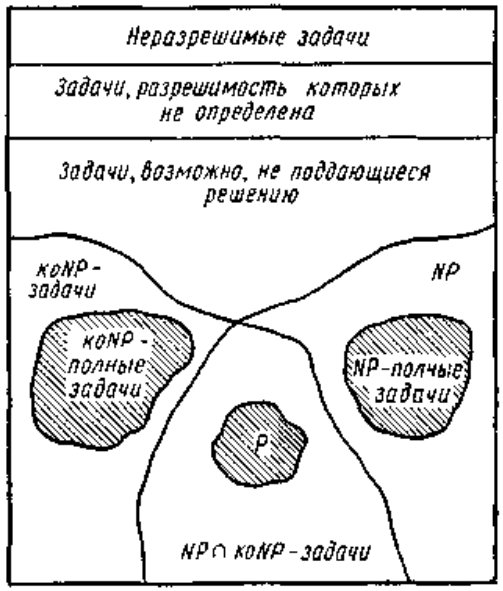


Рис. 3. Классификация задач с точки зрения их принципиальной разрешимости и того, поддаются ли они решению

Класс ко*NP* -задач состоит из задач, дополнительных к *NP* -задачам, т. е. задач, ответы на которые являются дополнением к ответам для соответствующих *NP* -задач. Неизвестно, верно ли, что *NP* =ко*NP* , однако известно, что пересечение *NP*∩ко*NP* непусто и содержит все P-задачи, а также некоторые другие.
Несмотря на то, что вычислительная сложность в основном выражается через время, необходимое для выполнения вычислений, бывает важно оценить и необходимый объем памяти компьютера. Это условие называется *пространственным.* Его можно оценить с помощью пространственной функции сложности, аналогичной *временной.* Однако известно, что любая задача, которую можно решить

за полиномиальное время, решается в полиномиальном пространстве. В самом деле, число ячеек, с которыми оперирует автомат машины Тьюринга при конкретном вычислении (это число определяет необходимое для решения пространство), не может быть больше числа шагов вычисления (что определяет время решения). Однако из этого не следует, что все задачи, разрешимые в полиномиальном пространстве, разрешимы за полиномиальное время. Именно поэтому для разбиения задач на поддающиеся и не поддающиеся решению используется временная сложность. Однако на практике важны оба эти условия.

7.6. Сложность в ФРИЗ

Ларе Лофгрен отмечает*: «С процессом обучения связана сложность описания (о-сложность). Ее можно оценить по тому, насколько трудно выделить описание системы... С процессом интерпретации связана сложность интерпретации (и-сложность). Ее можно оценить по тому, насколько трудно выделить из описания интерпретацию (смысл).»*
При разработке архитектуры ФРИЗ вопросы, связанные со сложностью систем ИИ и задач, возникают в нескольких случаях. Сложность систем ИИ играет в ФРИЗ двоякую роль: во-первых, определяет тип условия для некоторых интеллектуальных задач, которые идентифицируются в схеме ФРИЗ; во-вторых, используется для представления размерности систем ИИ, рассматриваемых в различных задачах и для оценки вычислительной сложности конкретных вариантов задач.
Интеллектуальную сложность как условие для интеллектуальных задач можно рассматривать, вообще говоря, как отношение предпочтения на множестве рассматриваемых систем ИИ. В таком смысле это понятие ориентировано прежде всего на пользователя. Поэтому желательно, чтобы и меру для оценки сложности систем ИИ определял сам пользователь. Однако в некоторых случаях пользователь может и не иметь своей меры сложности, и тогда ФРИЗ должен предложить ему на выбор собственные меры. Более того, если пользователь затрудняется в выборе или ему безразлично, какая из мер будет использована**, ФРИЗ должен для данной задачи сам выбрать *меру сложности систем по умолчанию.***
Вообще говоря, для систем ИИ разных эпистемологических типов нужны разные меры сложности. Могут понадобиться и разные меры сложности для систем ИИ одного эпистемологического типа, но имеющие другие методологические отличия. В гл. 4—6 при описании

неких «представительных» задач уже были введены некоторые очевидные меры сложности, которые можно использовать как меры по умолчанию (или как меры, предлагаемые в виде «меню»).

Мера сложности как способ представления размерности варианта задачи, по которой определяется временная функция сложности этой задачи, обычно определяется длиной описания рассматриваемой системы ИИ. Поскольку длина описания зависит от используемой системы кодирования, то в каждом случае она должна основываться на системе кодирования, примененной в конкретной реализации ФРИЗ для описания систем ИИ этого типа.

Сложность задач рассматривается как при создании, так и при использовании ФРИЗ. При создании ФРИЗ проблемы, связанные со сложностью задач, учитываются при исследовании методологических средств. К ним относится, например, **определение временных и пространственных функций сложности конкретных алгоритмов и программ, определение пределов практической разрешимости отдельных задач, сравнение конкурирующих алгоритмов с точки зрения их вычислительной сложности, поиск новых алгоритмов решения задач, для которых имеющиеся алгоритмы неудовлетворительны из-за их сложности, разработка эффективных эвристических алгоритмов для не поддающихся решению задач и поиск для таких задач упрощений, которые позволят их решить.**

Одной из функций ФРИЗ должен быть непременный анализ каждой предлагаемой задачи на сложность. Перед тем как продолжить работу, ФРИЗ должен представить пользователю краткий отчет о результатах этого анализа и, если нужно, список предлагаемых опций. Исходя из этого отчета, пользователь может подтвердить свой первоначальный запрос, выбрать одну из предложенных опций, как-то: изменить свою задачу или совсем отказаться от ее решения.

Несмотря на то, что анализ сложности может быть реализован самыми разными способами, он должен давать ответ на следующие фундаментальные вопросы. **Во-первых, о разрешимости**. Если задача неразрешима, приходится отказаться от ее решения, предложив, если возможно, пользователю модификации постановки задачи, делающие ее разрешимой. **Во-вторых, следует определить, к какому классу сложности принадлежит данная задача**. **Эти классы определяются тремя фиксированными значениями вычислительной сложности: пределом Бреммерманна (или его менее строгим аналогом), пределом для существующей вычислительной техники (это значение должно периодически корректироваться в соответствии с достижениями компьютерной**

техники) и пределом сложности для конкретно используемой реализации ФРИЗ.

Таким образом, интеллектуальные задачи разбиваются, как это , показано на рис. 4, на четыре больших класса.

Рис. 4. Приблизительная классификация интеллектуальных задач по сложности

Они введены для того, чтобы пользователь мог понять, поддается его задача решению или нет. Если задача принципиально неразрешима, ему следует отказаться от данной задачи и искать вычислительно менее сложную формулировку. Если задача потенциально решается, но нужные для этого ресурсы превосходят возможности вычислительной техники, то также нужно искать другую формулировку. Однако в этом случае остается возможность вернуться к первоначальной задаче в будущем.

Если поставленная задача не выходит за рамки возможностей техники, то необходимо провести более тщательный анализ ее сложности. Если задача может быть решена с помощью имеющейся реализации ФРИЗ, то пользователю следует дать приблизительную оценку стоимости вычислений. Если задачу решить нельзя, то нужно выдать пользователю сведения о необходимых для ее решения вычислительных ресурсах (скорости, памяти), и приблизительной стоимости вычислений.

В некоторых областях, таких, как предсказание погоды, контроль продукции, тестирование техники или принятие решений в управлении, в условия решения интеллектуальной задачи входит максимально допустимое время решения задачи. В других случаях указана максимально допустимая стоимость вычислений. Подобные ограничения должны учитываться при анализе сложности и отражаться в отчете, предоставляемом пользователю.

Необходимо отметить, что вычислительная сложность определяется не только размерностью задачи, т. е. задачи одного типа и размерности могут очень отличаться по сложности. В большинстве случаев в исследованиях по вычислительной сложности основное внимание

уделяется оценке наихудших вариантов задач. Несмотря на то, что теоретически это вполне обоснованно,такие оценки редко выполняются на практике и, следовательно, являются слишком пессимистическими. Чтобы как-то исправить это положение, в некоторых случаях помимо наихудших оценок приводятся также усредненные оценки. Однако такие оценки получаются в предположении о равновероятности всех вариантов задачи, что не всегда отражает действительное распределение вероятностей вариантов, встречающееся на практике. Проблема определения таких реальных распределений для разных типов задач является преимущественно эмпирической.

8. Целенаправленные системы ИИ

8.1. Базовые и дополнительные понятия

Единственным обоснованием наших понятий является то, что они используются для представления всей совокупности нашего опыта; вне его они не имеют права на существование
Альберт Эйнштейн

Для полного понимания концептуальной схемы ФРИЗ желательно взглянуть на нее со стороны, чтобы определить некоторые важные категории понятий. Такой подход позволяет выделить **три основные категории.** Назовем понятия в соответствии с этими категориями **примитивными, основными и дополнительными.**
Отличительным признаком *примитивных системных понятий* является их независимость от других понятий данной схемы. Однажды выбранные, они в основном определяют диапазон концептуальных схем ИИ, которые можно на них построить. Можно сказать, что богатство выбранных примитивных понятий определяет богатство получаемых из них концептуальных схем ИИ. **Исключение из рассмотрения каких-либо примитивных понятий может привести к значительному сужению диапазона получаемых концептуальных схем ИИ или даже сделать их практически бесполезными.**
В схеме УРСЗ к примитивным относятся понятия, связанные с исходными системами: атрибуты (входные и выходные) множества проявлений, базы и множества баз, конкретные и общие параметры и их параметрические множества, каналы наблюдения (четкие или нечеткие) и конкретизации (абстрагирования). Сюда также следует

отнести всевозможные методологические отличия, связанные с этими понятиями.
В схеме ФРИЗ все другие понятия определяются в терминах перечисленных понятий. При тщательном анализе можно естественным образом выделить **две категории этих производных понятий**. **К одной из них относятся понятия, связанные со всевозможными видами описаний связей между переменными.** Эти понятия используются при определении всех эпистемологических типов систем ИИ, кроме исходной системы ИИ, а именно: **при определении данных, правил сдвига на параметрических множествах, выборочных переменных (порождаемых, порождающих и входных), масок, поведения и ST-функций (основных и порожденных), среды (внутренней и внешней), подсистем и суперсистем, элементов структурированных систем, соединяющих переменных, связей (нейтральных или направленных), элементов метасистем и процедур замены.** Поскольку все эти понятия связаны с основной проблемой решения интеллектуальных задач — **описанием, определением и использованием связей между различными типами понятий** — **терминологически удобно называть их *базовыми системными понятиями.***
Остальные понятия ФРИЗ будем называть дополнительными системными понятиями. Эти понятия, не являющиеся ни примитивными, ни базовыми понятиями ФРИЗ, можно тем не менее определить через примитивные и базовые, а в некоторых случаях и через другие дополнительные понятия. Обычно они имеют различный смысл в зависимости от того, к каким типам систем ИИ относятся. *Системная сложность,* рассмотренная в гл. 7, одно из них. Это общее понятие, определенное своими общими аксиомами, которое относится к широкому спектру более конкретных понятий системной сложности. Их можно упорядочить по степени конкретности. Чтобы в решении интеллектуальных задач понятие системной сложности имело практический смысл, оно должно быть достаточно конкретным, по крайней мере должно определяться в терминах контактных типов систем ИИ.
К категории дополнительных системных понятий относятся **два важных класса понятий — *цель* и *характеристика* системы**. Это такие же общие понятия, как системная сложность, имеющие множество приложений. Цель настоящей главы — введение этих понятий на общем уровне, обсуждение их некоторых конкретных интерпретаций и определение роли при решении интеллектуальных задач.

8.2. Цель системы ИИ и ее характеристика

Цель системы ИИ можно определить различными способами. В соответствии с принятым в ФРИЗ общим подходом цель системы ИИ находится «в руках» пользователя. Это значит, что для заданной системы ИИ произвольного эпистемологического уровня, определенной ее первичными свойствами, ассоциируемая с системой ИИ **цель — это конкретное ограничение ее первичных или вторичных свойств, которое при данных обстоятельствах пользователь считает предпочтительным.**

Таким образом, **данная система ИИ может рассматриваться с точки зрения различных целей.** В некоторой степени система ИИ удовлетворяет любой цели. Эта степень, называющаяся *характеристикой системы ИИ относительно цели,* может быть измерена (в некотором смысле) близостью действительных и желаемых проявлений тех свойств системы ИИ, которые предусмотрены целью. Обычно она определяется в терминах соответствующей функции, называемой *характеристической функцией.*

Обозначим через $\mathscr{X}$ множество систем ИИ, отличающихся свойствами, которые в данном случае определяют понятие цели (остальные свойства совпадают). Тогда характеристическая функция, обозначим ее ω, имеет вид

$$\omega : \mathscr{X} \times \mathscr{X} \to [0, 1], \qquad (1)$$

где $\omega(\mathbf{x}, \mathbf{x}^*)$ представляет степень соответствия данной системы ИИ $\mathbf{x} \in \mathscr{X}$ целевой системе (хорошей, идеальной) $\mathbf{x}^* \in \mathscr{X}$. Характеристическую функцию удобно определять соответствующей функцией расстояния

$$\delta : \mathscr{X} \times \mathscr{X} \to \mathbf{R}^+ \qquad (2)$$

с помощью формулы

$$\omega(\mathbf{x}, \mathbf{x}^*) \frac{\delta_m(\mathbf{x}, \mathbf{y}) - \delta(\mathbf{x}, \mathbf{x}^*)}{\delta_m(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})} = 1 - \frac{\delta(\mathbf{x}, \mathbf{x}^*)}{\delta_m(\mathbf{x}, \mathbf{y})} \qquad (3)$$

где $\delta_m(\mathbf{x}, \mathbf{y}) = \max_{\mathbf{x}, \mathbf{y} \in \mathscr{X}} \delta(\mathbf{x}, \mathbf{y})$.

Пример 1. Предположим, что цель определена с помощью подходящей функции поведения f_B^* на множестве систем ИИ с поведением, характеризующихся одной и той же исходной системой ИИ и маской *M.* Тогда множество $\mathscr{X}$ в этом примере представляется множеством систем ИИ с поведением

$F_B=(S,M,f_B)$,

отличающихся только функциями поведения f_B. Возможным и подходящим в данном случае способом определения расстояния между системами является расстояние Хэмминга

$$\delta(\mathbf{x}, \mathbf{y}) = \sum_{\mathbf{c} \in \mathbf{C}} |{}^x f_B(\mathbf{c}) - {}^y f_B(\mathbf{c})| . \qquad (4)$$

Для вероятностных функций поведения $\delta_m(\mathbf{x}, \mathbf{y}) = 2$ и

$$\omega(\mathbf{x}, \mathbf{x}^*) = 1 - \delta(\mathbf{x}, \mathbf{x}^*)/2; \qquad (5)$$

для возможностных

$$\delta_m(\mathbf{x}, \mathbf{y}^*) = |\mathbf{C}|, \qquad (6)$$

$$\omega(\mathbf{x}, \mathbf{x}^*) = 1 - \delta(\mathbf{x}, \mathbf{x}^*)/|\mathbf{C}|.$$

Очевидно, что разные цели и характеристические функции применимы к разным типам систем ИИ. Тем не менее разные типы целей, требующих определенных характеристических функций, можно определить и для одних и тех же типов систем ИИ. Например, **для систем ИИ с поведением цели можно определить через подходящую функцию поведения, множество функций поведения, множество локальных функций поведения для определенных подмножеств параметрического множества, множество функций поведения, представляющих подходящие подсистемы, и т. д.** Очевидно, что **для каждого из этих типов целей необходима конкретная характеристическая функция**.

Понятия цели и характеристики являются базовыми для определения понятия целенаправленных систем.

8.3. Целенаправленные системы ИИ

Система, которая стремится улучшить характеристики своей работы при достижении заданной цели и осуществляет это без помощи извне, называется самоорганизующейся.

Ханс Бреммерманн

Предположим, что тип цели и соответствующая характеристическая функция определены для множества систем ИИ некоторого эпистемологического типа. Как отмечалось в разд. 8.2, с каждой системой ИИ из этого множества связано значение характеристической функции, определяющее степень соответствия системы ИИ данной цели. Этот факт подсказывает тривиальный способ определения целенаправленных систем ИИ: **система ИИ рассматривается как целенаправленная тогда и только тогда, когда ее характеристика**

относительно заданной цели больше некоторого заданного порога— обычно 0.5 или более.

Другой способ определения целенаправленных систем ИИ, представляющийся операционно более содержательным, заключается в рассмотрении понятия целенаправленных систем ИИ в относительных терминах. **Это значит, что одна система ИИ рассматривается как более целенаправленная относительно другой системы ИИ того же типа и определенной цели тогда и только тогда, когда ее характеристики *лучше* (в смысле некоторой характеристической функции) относительно данной цели.** Формально для двух заданных систем ИИ $\mathbf{x},\ \mathbf{y}\in\mathfrak{X}$ одного и того же типа, определенной цели $\mathbf{x}^*\in\mathfrak{X}$ и соответствующей характеристической функции ω система ИИ **x** является целенаправленной относительно системы ИИ **y** и цели **x*** при характеристической функции ω тогда и только тогда, когда

$$\omega(\mathbf{x},\ \mathbf{x}^*) > \omega(\mathbf{y},\ \mathbf{x}^*).$$

Назовем разность

$$\Delta\omega(\mathbf{x},\ \mathbf{y}\,|\,\mathbf{x}^*) = \omega(\mathbf{x},\ \mathbf{x}^*) - \omega(\mathbf{y},\ \mathbf{x}^*) \qquad (7)$$

степенью целенаправленности **x** относительно **y** при заданной цели **x***. Система ИИ с положительной степенью целенаправленности относительно другой системы ИИ должна обладать некоторыми свойствами, отличными от свойств последней, свойствами, связанными с целью и определяющими улучшение характеристики этой системы ИИ. **Будем называть их *свойствами выбора цели.*** Такими свойствами, например, являются некоторые дополнительные переменные или состояния в порождающих системах ИИ, дополнительные элементы или соединения в структурированных системах ИИ, дополнительные элементы или процедуры замены в метасистемах ИИ и др.

Для иллюстрации общих положений, связанных с целенаправленностью, применим их к системам ИИ определенного эпистемологического уровня — нейтральным системам ИИ с поведением. В этом случае свойствами выбора цели являются также переменные, включение которых в систему ИИ улучшает ее характеристику. **Назовем такие переменные *переменными выбора цели.***

Рассмотрим множество нейтральных систем ИИ с поведением обычного вида

$$\mathbf{F}_B=(\mathbf{S},\ M,\ f_B),$$

отличающихся только своими функциями поведения f_B. Поскольку далее в этом разделе рассматриваются системы ИИ только этого типа, то, не вызывая путаницы, упростим запись, опустив индекс *B*. Кроме того, нам удобно отождествлять любую систему ИИ из этого

множества с ее функцией поведения *f* или соответствующим распределением **f**.

Предположим, что рассматриваемые системы ИИ вероятностные. В соответствии с обозначениями, введенными в гл. 4, обозначим через **c** обобщенные состояния выборочных переменных, соответствующих маске *M,* и пусть $\mathbf{c}\in\mathbf{C}$. Предположим, что система ИИ, идентифицируемая распределением вероятностей

$$\mathbf{f}^*=(f^*(\mathbf{c})\,|\,\mathbf{c}\in\mathbf{C}),$$

рассматривается как цель. Тогда для каждой системы ИИ, идентифицируемой распределением вероятностей

$$\mathbf{f}=(f(\mathbf{c})\,|\,\mathbf{c}\in\mathbf{C}),$$

расстояние до цели можно, например, определить по формуле (4). Тогда характеристику каждой системы ИИ рассматриваемого множества можно определить, подставив соответствующее расстояние в формулу (5).

Рассмотрим теперь систему ИИ с поведением

$$\mathbf{F}'=(\mathbf{S}',\ M',\ f'),$$

исходная система ИИ которой содержит все элементы, входящие в систему ИИ **S**', и, кроме того, некоторые переменные, благодаря которым маска *M* расширяется до маски *M'*. Обозначим через **z** состояния выборочных переменных, соответствующих разности множеств *M'—M;* пусть $\mathbf{z}\in\mathbf{Z}$, а

$$\mathbf{f}'=(f'(\mathbf{c},\ \mathbf{z})\,|\,\mathbf{c}\in\mathbf{C},\ \mathbf{z}\in\mathbf{Z})$$

обозначает распределение вероятностей системы **F'**.

Для определения расстояния между системами ИИ, идентифицируемыми распределениями **f'** и **f***, необходимо преобразовать расширенное распределение **f'** к виду, сравнимому с распределением **f'**. Это можно сделать с помощью формулы

$$f''(\mathbf{c}) = \sum_{\mathbf{z}\in\mathbf{Z}} f'(\mathbf{c},\ \mathbf{z}). \qquad (8)$$

Пусть распределение

$$\mathbf{f}''(f''(\mathbf{c})\,|\,\mathbf{c}\in\mathbf{C})$$

используется для идентификации системы ИИ **F'**. Тогда формулы (4) и (5) применимы для вычисления соответственно расстояния δ(**f''**, **f***) и характеристики ω (**f''**, **f***). **Если вычисленная по формуле (7) функция Δω(f'', f|f*) больше 0, то система ИИ F' является *целенаправленной* относительно системы ИИ F при заданной цели f*;** тогда переменные, соответствующие разности *M'—M,* называются ***переменными выбора цели***.

Важно понять, что необходимое условие целенаправленности системы ИИ **F'** относительно системы **F** заключается в том, что **f** не является проекцией **f′**. Действительно, если **f** является проекцией **f′**, тогда **f''=f** и, следовательно,

$$\Delta\omega(\mathbf{f}'', \mathbf{f}|\mathbf{f}^*) = 0.$$

Это условие означает, что переменные **S'**, не содержащиеся в **S**, должны определять признаки, не свойственные объекту, для которого определена система ИИ **F**.

Пример 2. Рассмотрим задачу о максимальном использовании трех дорогостоящих устройств, входящих в вычислительную систему. Три переменные v_1, v_2, v_3 описывают состояние соответствующего устройства в зависимости от времени. Каждая из переменных имеет два значения: 0 — в момент наблюдения устройство неактивно и 1 — устройство активно.

Предположим, что с помощью аппаратуры было выполнено наблюдение (см. пример 4.8) и получено распределение вероятностей **f**, приведенное в табл. 1а, для маски без памяти.

Таблица 1.

Целенаправленная система и переменная выбора цели из примера 2

а)					б)					в)			
v_1	v_2	v_3	f	f*	v_1	v_2	v_3	v_4	f′	v_1	v_2	v_3	f″
0	0	1	0.15	0	0	1	1	1	0.10	0	1	1	0.10
0	1	0	0.20	0	1	0	0	0	0.02	1	0	0	0.02
1	0	0	0.10	0	1	0	1	0	0.03	1	0	1	0.03
1	1	0	0.25	0	1	1	0	0	0.04	1	1	0	0.05
1	1	1	0.30	1	1	1	0	1	0.01	1	1	1	0.80
					1	1	1	0	0.25				
					1	1	1	1	0.55				

Здесь также приведено распределение вероятностей **f***, представляющее цель. Используя формулы (4) и (5), получаем $\delta(\mathbf{f}, \mathbf{f}^*) = 1.4$ и $\omega(\mathbf{f}, \mathbf{f}^*) = 0.3$.

Предположим теперь, что вычислительная система дополнена новым устройством (например, каналом связи) таким образом, что оно влияет на работу трех рассматриваемых устройств. Предположим далее, что по сравнению с этими устройствами новое устройство относительно дешевле и степень его использования не так важна. Это устройство добавлено только с целью более эффективного использования трех других устройств. Цель, таким образом, остается прежней.

Пусть переменная v_4 определена для нового устройства так же, как и другие переменные для соответствующих устройств. Предположим,

что для новой системы, включающей переменную v_4, вновь выполнено аппаратное наблюдение, и результаты распределения вероятностей приведены в табл. 1,б. Воспользуемся формулой (8) для вычисления распределения **f''**, чтобы сравнить **f''**с **f*** (табл. 1в). Тогда $\delta(\mathbf{f}'', \mathbf{f}^*) = 0.4$ и $\omega(\mathbf{f}'', \mathbf{f}^*) = 0.8$. Следовательно,

$$\Delta\omega(\mathbf{f}'', \mathbf{f}|\mathbf{f}^*) = 0.8 - 0.3 = 0.5,$$

т. е. в данном случае расширенная система является целенаправленной, а переменная v_4 — это переменная выбора цели; степень целенаправленности (улучшение характеристики благодаря дополнительному устройству) равна 0.5.

Рассмотрим другую цель, у которой $f^*(c)=0.5$ для двух последних состояний, приведенных в табл. 1а. Тогда

$$\delta(\mathbf{f}, \mathbf{f}^*) = \delta(\mathbf{f}'', \mathbf{f}^*) = 0.9,$$

$$\omega(\mathbf{f}, \mathbf{f}^*) = \omega(\mathbf{f}'', \mathbf{f}^*) = 0.55.$$

Таким образом, в этом случае новая система не будет целенаправленной. Хотя новое устройство, представленное переменной v_4, существенно влияет на работу остальных устройств, оно не приближает систему к цели. Это значит, что переменная v_4 не является переменной выбора цели относительно альтернативной цели.

Рассмотрим еще одну цель, у которой $f^*(c) = 0.2$ для каждого состояния, приведенного в табл. 1в. Тогда

$$\omega(\mathbf{f}, \mathbf{f}^*) = 0.85, \ \omega(\mathbf{f}'', \mathbf{f}^*) = 0.27$$

и

$$\Delta\omega(\mathbf{f}'', \mathbf{f}|\mathbf{f}^*) = -0.58.$$

Переменная v_4 в этом случае является нежелательной и может быть названа переменной уклонения от цели — она отклоняет систему от цели, и таким образом, «понижает» ее характеристику.

8.4. Структурированные системы ИИ как парадигмы целенаправленных систем ИИ с поведением

Целенаправленные системы ИИ, определенные в предыдущем разделе, характеризуются отделением переменных выбора цели от остальных переменных и требованием того, чтобы переменные выбора цели способствовали достижению поставленной цели. Очевидно, что исследование различных способов порождения состояний переменных выбора цели чрезвычайно важно для понимания целенаправленных

систем ИИ, и в частности для развития методов проектирования соответствующих целенаправленных систем ИИ. При таком исследовании с необходимостью получаются , некоторые структурированные системы ИИ определенных типов. **Каждую из них можно рассматривать как парадигму, описывающую принцип (схему, форму) в терминах порождаемых состояний переменных поиска цели. Назовем эти парадигмы *структурными парадигмами целенаправленных систем ИИ.***

Во-первых, рассмотрим целенаправленные системы ИИ с поведением нейтрального типа. Как показано в предыдущем разделе, их переменные можно разделить на переменные выбора цели и остальные переменные. Поскольку цель определена в терминах последних переменных, то резонно назвать их *переменными, реализующими цель.* Переменные выбора цели воздействуют на переменные, реализующие цель, и в то же время подвергаются обратному воздействию. Поэтому естественно рассматривать целенаправленные системы ИИ как структурированные системы ИИ с двумя элементами. Один из них порождает состояния переменных, реализующих цель, а другой — состояния переменных выбора цели. Назовем их *элементом, реализующим цель,* и *элементом выбора цели* и будем формально описывать их как системы ИИ с поведением Итак,

$$^1\mathbf{F} = (^1\mathbf{S},\ ^1M,\ ^1f),$$
$$^2\mathbf{F} = (^2\mathbf{S},\ ^2M,\ ^2f)$$

соответственно. Поскольку недоразумения исключены, то и здесь индекс *B* опущен.

Обозначим ^1V, ^2V множества переменных в исходных системах $^1\mathbf{S}$, $^2\mathbf{S}$ соответственно. В общем случае $^2V \subseteq {}^1V$, поскольку переменные выбора цели воздействуют на переменные, реализующие цель (это требование следует из определения), тогда как обратное воздействие не требуется. Для описания способа порождения отдельных переменных необходимо рассматривать два элемента как направленные (это показано в разд. 5.3 и 5.4), даже если не рассматривается внешняя среда. Обозначим направленные соединения между элементами

$$\hat{C}_{2,1} = {}^2V \text{ и } \hat{C}_{1,2} = {}^1V',$$

где $^1V' \subseteq {}^1V$. Естественно различать три структурные парадигмы, отличающиеся друг от друга соединением $\hat{C}_{1,2}$:

$$\hat{C}_{1,2} = \varnothing;$$
$$\hat{C}_{1,2} \subset {}^1V;$$
$$C_{1,2} = {}^1V.$$

Эти парадигмы, отличающиеся объемом информации о переменных, реализующих цель, используются для порождения переменных выбора цели. Назовем их соответственно *парадигмой без информации, парадигмой частичной информации* и *парадигмой полной информации.* На рис. 1 показана схема этих трех парадигм.

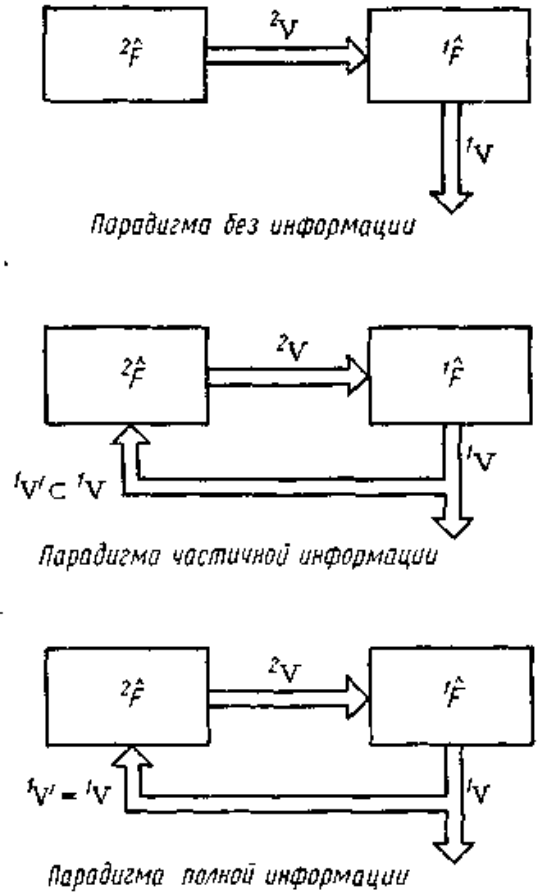


Рис. 1. Структурные парадигмы целенаправленной системы ИИ с поведе- нием нейтрального (без выходных переменных).

Рассмотрим теперь структурные парадигмы целенаправленных систем ИИ с поведением направленного типа. Помимо множеств переменных, содержащихся в их нейтральных аналогах, они содержат множество

входных переменных, которое обозначим X. Возможные направленные соединения между элементами, реализующими цель, элементом выбора цели и средой (обозначенной как элемент 0) сведены в таблицу:

	0	1	2
0	$\varnothing$	X	X'
1	${}^1V$	$\varnothing$	${}^1V^1$
2	$\varnothing$	${}^2V$	$\varnothing$

где $X' \subseteq X$ и ${}^1V' \subseteq {}^1V$. Для X' можно выделить три характерных состояния:

$X' = \varnothing$;

$X' \subset X$;

$X' = X$.

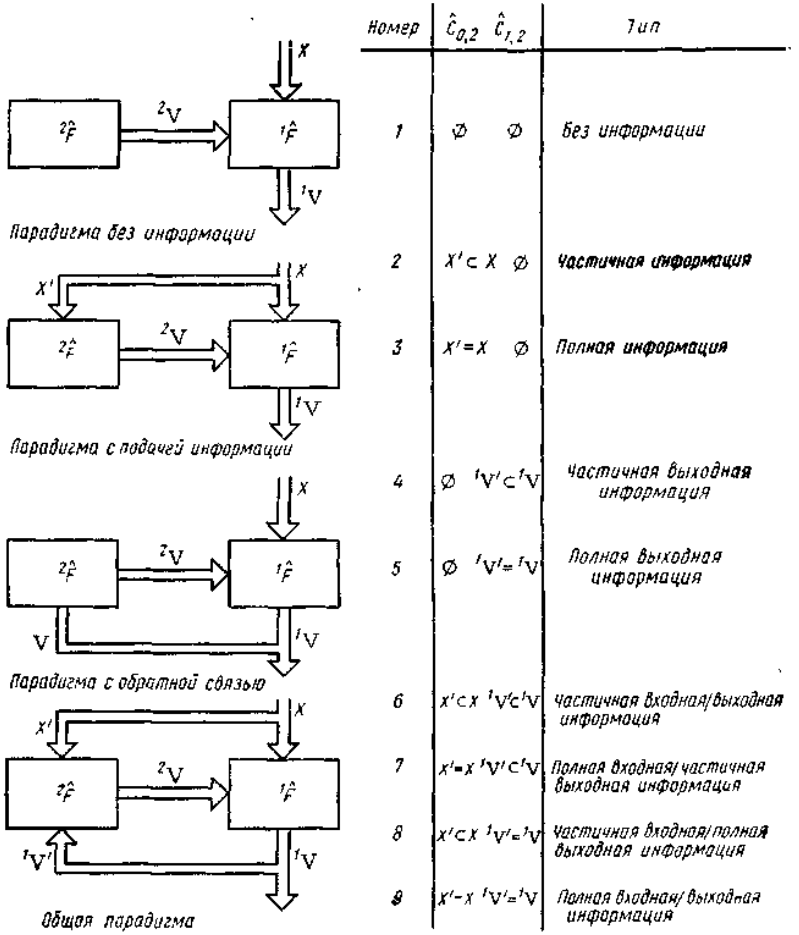
Аналогично может быть выделено три состояния ${}^1V'$:

${}^1V' = \varnothing$;

${}^1V' \subset {}^1V$;

${}^1V' = {}^1V$.

Каждое из состояний X' можно скомбинировать с любым из выделенных состояний ${}^1V'$. Это приводит к девяти парадигмам. Они показаны на рис. 2 и разделены на четыре класса, каждый из которых представлен одной из схем и соответствующим названием.



Номер	$\hat{C}_{0,2}$ $\hat{C}_{1,2}$	Тип
1	$\varnothing$ $\varnothing$	Без информации
2	$X' \subset X$ $\varnothing$	Частичная информация
3	$X' = X$ $\varnothing$	Полная информация
4	$\varnothing$ ${}^1V' \subset {}^1V$	Частичная выходная информация
5	$\varnothing$ ${}^1V' = {}^1V$	Полная выходная информация
6	$X' \subset X$ ${}^1V' \subset {}^1V$	Частичная входная/выходная информация
7	$X' = X$ ${}^1V' \subset {}^1V$	Полная входная/частичная выходная информация
8	$X' \subset X$ ${}^1V' = {}^1V$	Частичная входная/полная выходная информация
9	$X' = X$ ${}^1V' = {}^1V$	Полная входная/выходная информация

Рис. 2. Структурные парадигмы целенаправленной системы ИИ с поведением направленного типа

Понятие целенаправленной системы ИИ с поведением в целом и ее различные структурные парадигмы, перечисленные, в частности, на рис. 1 и 2, имеют широкий диапазон применений и множество различных интерпретаций. Одной из наиболее широко изучаемых интерпретаций является *регулирование.* Любая целенаправленная система ИИ, осуществляющая какое-либо регулирование, называется *регулятором.* Ее цель заключается в сохранении особого состояния

или особого подмножества состояний переменных '*V* несмотря на возмущения, представленные входными переменными. Переменные выбора цели выполняют функции регулирующих переменных; элементы **[1]F, [2]F** являются регулируемыми и регулирующими элементами соответственно. Регулятор можно также определить в терминах ST-системы. В этом случае цель заключается в сохранении системы ИИ в подмножестве состояний, включающем регулируемые переменные.

Другая интерпретация целенаправленной системы ИИ с поведением заключается в рассмотрении ее как обучающейся. Элементы: [1]**F** и [2]**F** являются соответственно обучающимся и обучающим. Цель заключается в получении **реакций (состояний переменных: в множестве *[1]V)*** на отдельные раздражители (состояния переменных в множестве *X),* которые рассматриваются (определены) каю. «правильные». Состояния переменных выбора цели в этом случае представляются как своего рода усиление.

Еще одна интерпретация заключается в рассмотрении целенаправленной системы ИИ с поведением как *системы ИИ, принимающей решения.* Элементы [1]**F** и [2]**F** становятся элементами реализаций решения и элементами принятия решения соответственно. Состояния входных переменных представляют так называемые «естественные состояния» (т. е. соответствующие внешние обстоятельства, возможные перемещения противника, определенные характеристики некоторого вида и т. п.). Состояния переменных в множестве *[1]V* представляют выходы, на которых определена *функция выгоды.* **Цель системы ИИ заключается в максимизации функции выгоды**. С помощью переменных выбора цели выбираются варианты из множества решений, положительно воздействующие на выходы. В соответствии с их ролью эти переменные можно, например, назвать **переменными принятия решения, переменными выбора или утилитарными переменными.**

Можно описать и некоторые другие интерпретации целенаправленной системы ИИ с поведением, например системы ИИ, корректирующие ошибки или самоорганизующиеся системы ИИ. Однако цель этой книги заключается не во всестороннем и детальном описании целенаправленных систем ИИ, а только в определении их роли в решении интеллектуальных задач.

8.5. Проектирование целенаправленных систем ИИ

В исследованиях искусственного интеллекта понятие структурных парадигм целенаправленных систем ИИ с поведением дает исследователю полезное описание систем ИИ, в соответствии с которым некоторые из исследуемых переменных рассматриваются как способствующие достижению выбранной цели. Основная проблема заключается в определении того, какие из переменных в данном исследовании в наибольшей степени влияют на достижение цели.

С другой стороны, при проектировании целенаправленных систем ИИ каждая структурная парадигма определяет структуру, в пределах которой должен действовать проектировщик. Другими словами, **каждая структурная парадигма представляет множество предположений (ограничений) относительно проектируемой системы ИИ, которые проектировщик не должен нарушать.**

Поскольку проектируемые системы ИИ всегда направленные, то при обсуждении различных вопросов, возникающих при проектировании целенаправченных систем ИИ, допустимы только показанные на рис. 2 структурные парадигмы.

Эти парадигмы могут быть частично упорядочены по строгости их ограничений. Чем строже ограничение, тем меньше свободы остается у проектировщика при решении его задачи и, следовательно, менее общей является парадигма. Будем называть структурную парадигму *менее общей* по сравнению с другой структурной парадигмой тогда и только, тогда, когда она включает больше ограничений (предположений), чем последняя. Применительно к множеству структурных парадигм, определенных на рис. 2, это упорядочение по уровню , общности образует решетку, которая описывается диаграммой Хассе на рис.3.

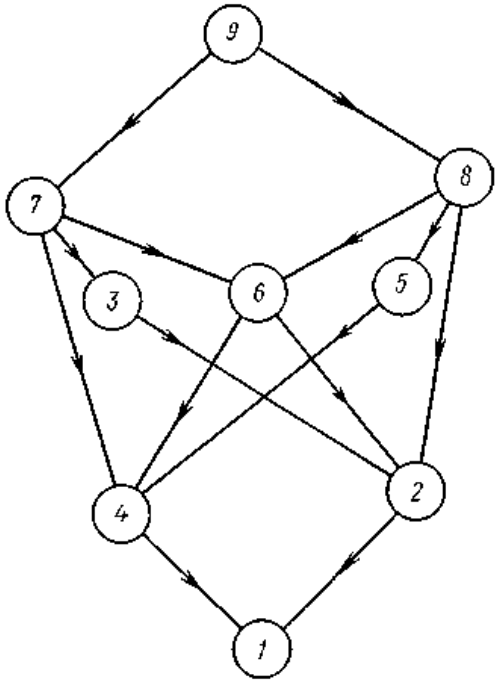


Рис. 3. Решетка структурных парадигм, показанных на рис 2, упорядоченных по общности

Для конкретных множеств X' и ${}^{1}V'$ более уточненную решетку можно определить, допустив, что X' и ${}^{1}V'$ представляют произвольные подмножества X и ${}^{1}V$ соответственно.
В типичной задаче проектирования целенаправленных систем ИИ имеются следующие составляющие:
1) направленная система ИИ с поведением ${}^{0}\mathbf{F}=({}^{0}\mathbf{S}, {}^{0}M, {}^{0}f)$, которая представляет элемент, реализующий цель, без переменных выбора цели;
2) цель или характеристическая функция, совместимая с системой ${}^{0}\mathbf{F}$;
3) список имеющихся элементов;
4) целевой критерий (функция);
5) структурная парадигма целенаправленных систем ИИ и, возможно, некоторые ограничения относительно проектируемой системы ИИ.

Проектирование целенаправленных систем ИИ включает следующие основные действия.
1. Необходимо явно **задать цель и характеристическую функцию**. **Если при постановке задачи задана только цель, то проектировщику предстоит выбрать соответствующую характеристическую функцию. Если задана только характеристическая функция, то естественно предположить, что цель представлена функцией поведения, для которой характеристическая функция достигает своего максимума.** Этот подход, конечно, может привести к нескольким функциям поведения. Проектировщик может либо выбрать одну из них в качестве цели, либо в качестве альтернативного варианта выбрать больше, чем одну цель, и рассматривать проектируемую систему ИИ как целенаправленную метасистему ИИ.
2. Необходимо **выбрать соответствующее множество переменных выбора цели.** Эти переменные не могут быть произвольными, они должны оказывать некоторое воздействие на выходные переменные данной системы ИИ ${}^{0}\mathbf{F}$ и, таким образом, приводить к нетривиальному расширению системы ИИ ${}^{0}\mathbf{F}$ в новую систему ИИ с поведением

$${}^{1}\mathbf{F}=({}^{1}\mathbf{S}, {}^{1}M, {}^{1}f).$$

Расширенная система ИИ представляет собой элемент, реализующий цель, для любой из структурированных парадигм целенаправленных систем ИИ с поведением. Выбор соответствующих переменных выбора цели является решающим. Однажды определенные, они задают наилучшую характеристику, которую с их по мощью можно получить. Если она не соответствует требованиям, то необходимо исследовать другие возможные переменные выбора цели.
3. Основная трудность в выборе соответствующих переменных выбора цели заключается в том, что воздействия различных множеств при рассмотрении их как целевых переменных (функции поведения ${}^{1}f$) обычно неизвестны. Следовательно, они должны быть определены как часть задачи проектирования. Описанные в гл. 4 процедуры можно использовать при решении этой задачи.
4. Как только множество переменных выбора цели оговорено и соответствующая система ИИ ${}^{1}\mathbf{F}$, реализующая цель, определена, следующий шаг состоит в определении какого-либо конкретного способа порождения переменных выбора цели, т. е. в определении системы выбора цели ${}^{2}\mathbf{F}$. Цель заключается в определении функции поведения ${}^{2}f$ системы ${}^{2}\mathbf{F}$, для которой характеристическая функция достигает своего максимума при ограничениях требуемой структурной парадигмы. При решении этой проблемы можно использовать

различные методы оптимизации. Их выбор зависит главным образом от методологического типа используемых систем ИИ, а также от типа характеристической функции.

5. Необходимо спроектировать из элементов имеющихся типов структурированную систему ИИ, реализующую систему ИИ выбора цели, с поведением $^2\mathbf{F}$. Эта стандартная задача проектирования рассмотрена в разд. 5.5.

8.6. Адаптивные системы ИИ

Спроектированная по процедуре, описанной в предыдущем разделе, целенаправленная система ИИ оптимальна (или близка к оптимальной) только при условии, что не меняются ни функция поведения ^1f элемента, реализующего цель, ни цель f^*. Если это условие не выполняется, то действительный рост характеристики системы ИИ относительно цели при воздействии элемента выбора цели может быть существенно меньше расчетного значения. Может даже возникнуть обратное влияние элемента выбора цели на элемент, реализующий цель, приводящее к уменьшению этой характеристики. Для сохранения высокого уровня характеристики вне зависимости от изменений функции ^1f или f^* необходимо, чтобы целенаправленная система ИИ была способна к адаптации.

Существует много причин, по которым функции ^1f и f^* могут изменяться. В общем случае изменение функции ^1f означает, что на этапе проектирования система ИИ ^1f, реализующая цель, не была соответствующим образом определена с учетом этого изменения. В конечном счете отсюда следует, что на переменные, реализующие цель, влияют некоторые входные переменные, которые не были выделены в функции ^1f. Как только целенаправленная система ИИ спроектирована и реализована для конкретной функции ^1f, пользователь уже не может управлять изменениями ^1f. С другой стороны, изменения цели полностью определены пользователем. В общем случае он может считать желательным изменение цели в связи с изменением обстоятельств, для которых была спроектирована система ИИ.

Чтобы обеспечить возможность адаптации целенаправленной системы ИИ при изменении цели, необходимо спроектировать ее элемент выбора цели для множества альтернативных целей и дополнить его специальной входной переменной, значение которой определяет цели. Назовем этот вид целенаправленной системы ИИ, адаптация которой ограничена конкретным множеством целей, *многоцелевой*

направленной системой ИИ. Схема соответствующей структурной парадигмы (общего типа) показана на рис. 4, где v^* обозначает переменную, определяющую текущую цель ($v^* \in V^*$), а система выбора цели, дополненная входной переменной v^*, обозначена $^2\mathbf{F}^+$.

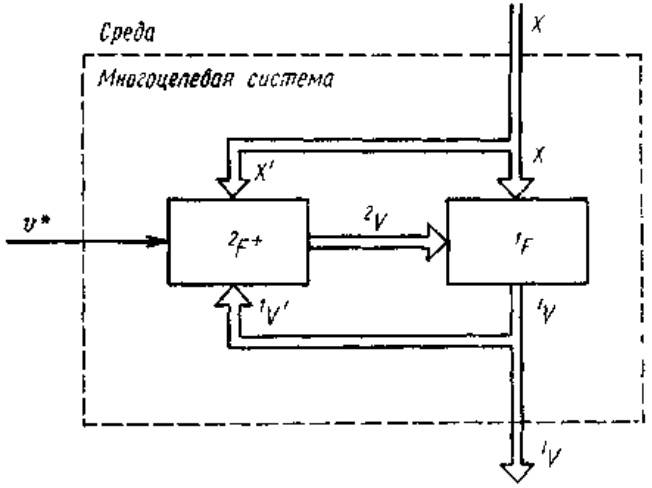


Рис. 4. Схема общей структурной парадигмы многоцелевой системы ИИ

В рассматриваемом случае цель определена либо пользователем, либо другой системой ИИ, связанной с входной переменной v^*.

Изменения цели могут определяться *целепорождающей системой ИИ,* которая сама является элементом многоцелевой направленной системы ИИ. Эта возможность показана на рис. 5,а.

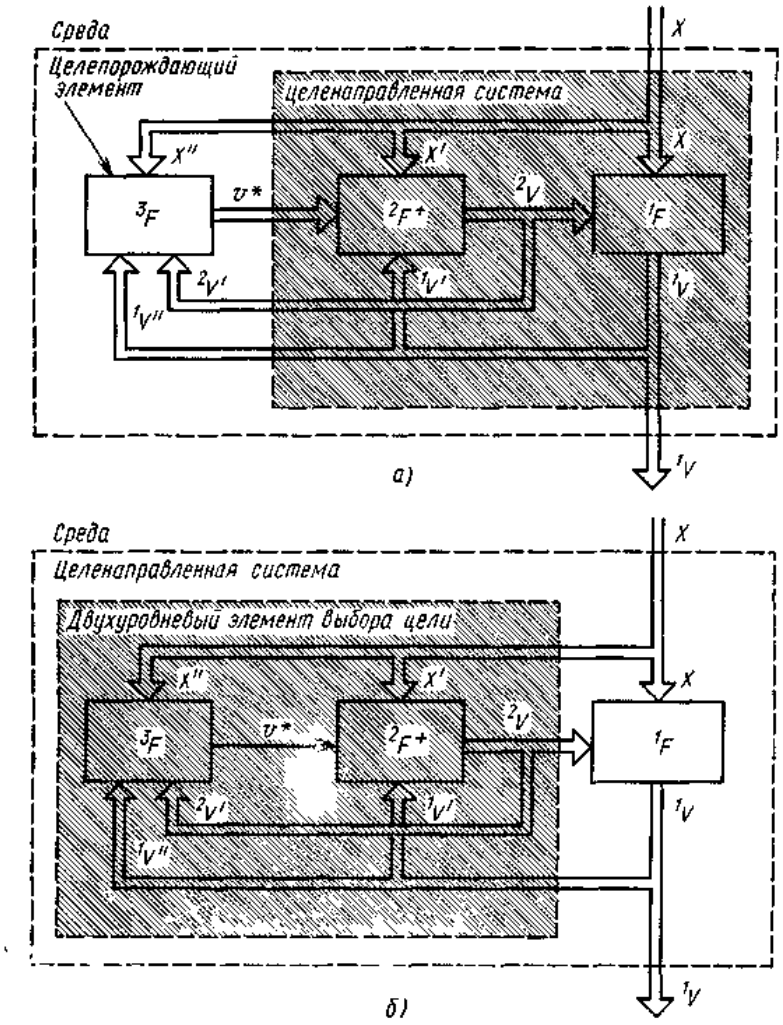


Рис. 5. Общая структурная парадигма целенаправленной системы ИИ, рассматриваемой как автономная многоцелевая система ИИ *(а)* или многоцелевая система ИИ с двухуровневым элементом выбора цели *(б)*

Элементы $^1\mathbf{F}$ и $^2\mathbf{F}^+$ образуют основную многоцелевую направленную систему ИИ (как на рис. 4), а $^3\mathbf{F}$ —целепорождающий элемент. 0бщая цель в данном случае заменяется последовательностями подцелей, представленными значениями переменной v^*. Эти последовательности

определяются функцией поведения ^3f в терминах переменных из множеств X, ^1V, ^2V *или* в зависимости от используемой структурной парадигмы некоторыми их подмножествами. Будем называть целенаправленные системы ИИ этого типа *автономными многоцелевыми направленными системами ИИ.*

Одно очевидное соображение в пользу рассмотрения автономных многоцелевых направленных систем ИИ состоит в том, что упрощается общая задача оптимизации, решение которой является частью процесса проектирования элемента выбора цели. Задача оптимизации распадается на несколько более простых оптимизационных задач, каждая из которых ограничена конкретными условиями, выраженными в терминах используемых переменных. С этой точки зрения более удобно рассматривать систему ИИ как обычную целенаправленную систему ИИ с двухуровневым элементом выбора цели (рис. 5,б).

Обе схемы на рис. 7.5 будут выглядеть одинаково, если речь идет о связях между тремя элементами. Они отличаются только способом, которым элементы концептуально скомбинированы в более крупные. Это тонкое отличие иллюстрирует основной феномен, связанный с системами ИИ всех видов: одну и ту же систему ИИ можно рассматривать с различных точек зрения в зависимости от того, применяется к ней операция укрупнения или уточнения.

Целенаправленные системы ИИ с k-уровневым элементом выбора ($k>2$) цели можно рекурсивно определить исходя из схемы, показанной на рис. 5,б. Любая такая система ИИ представляет собой иерархическую декомпозицию общей цели на подцели и содержит k уровней декомпозиции.

Рассмотрим теперь целенаправленные системы ИИ, которые адаптируются к изменениям функции поведения ^1f. Элемент выбора цели любой такой системы ИИ должен реализовывать следующие две функции:

1) обрабатывать данные, связанные с переменными множеств X, ^{1}V, ^2V, и образовывать модель системы ИИ $^1\mathbf{F}$;

2) применять модель $^1\mathbf{F}$ для порождения переменных выбора цели таким образом, чтобы их положительный эффект в смысле достижения цели был максимален или по крайней мере близок к максимуму.

Задача п. 1 решается многими способами. Можно непосредственно использовать методы, рассмотренные в гл. 4. Но удобнее рассматривать элемент, реализующий цель, и его модель как метасистему ИИ. Тогда элемент выбора цели должен отождествлять изменение так, как в разд. 6.6.

Пример 3. Для пояснения понятия целенаправленных систем, адаптирующихся к изменениям в системе, реализующей цель, рассмотрим в этом примере сложную адаптивную систему. Объектом, для которого определена эта система, является компьютер, оборудованный устройством, позволяющим ему передвигаться внутри квадратной области, разделенной в шахматном порядке на более мелкие квадраты. Эта область называется *рабочей (операционной) областью* компьютера. Предположим, что в рабочей области выделено 10 рядов и 10 столбцов, помеченных соответственно идентификаторами x и y ($x, y \in N_{0,9}$). Предположим далее, что каждый квадрат, определяющий местонахождение компьютера, помечен отдельным идентификатором

$$l=10x+y (l \in N_{0,99})$$

Компьютер также обладает широкими *потенциальными возможностями для исправления ошибок.* Их реализация при работе компьютера позволяет отождествлять неправильные действия в управлении вне зависимости от вида вызвавших их возмущений в окружающей среде. В определенных пределах это позволяет исключить эффект от сбоев в работе компьютера. Когда число сбоев в оборудовании становится слишком большим, превышающим возможности компьютера по исправлению ошибок, нормальная работа компьютера оказывается под угрозой. Поэтому желательно противостоять неизвестным возмущениям, **превратив компьютер в адаптивную целенаправленную систему, цель которой заключается в минимизации числа сбоев оборудования при длительной работе.** Поскольку компьютер может перемещаться, то естественный путь борьбы с внешними возмущениями состоит в избирательном перемещении внутри рабочей области в соответствии со стратегией, ориентированной на достижение этой цели.
Общая схема предполагаемой целенаправленной системы показана на рис. 6.

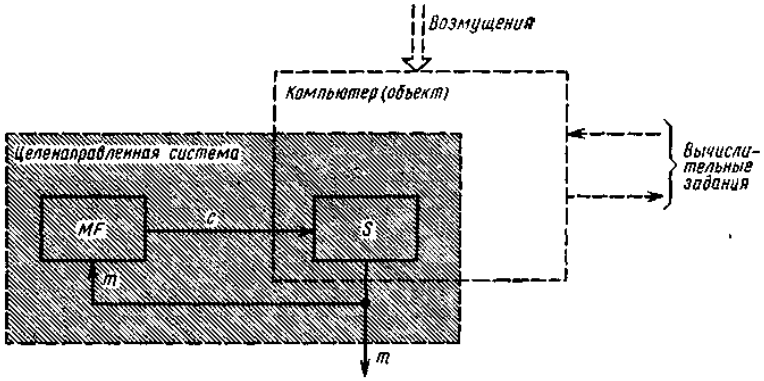


Рис. 6. Общая схема аддитивной целенаправленной системы, писанной в примере 3

На этом уровне во времени рассматриваются две переменные:
m — переменная, реализующая цель, она определяет количество сбоев компьютера в течение определенного промежутка времени ($m \in \mathbf{N}_0$);
c — переменная выбора цели, она определяет положение компьютера ($c \in N_{0,999}$).
Поскольку природа возмущений и их изменчивость неизвестны, то элемент, реализующий цель, должен рассматриваться только как исходная система **S**. В каждый момент времени принимается состояние переменной c и порождается состояние переменной m. Способ порождения переменной m неизвестен и может меняться.
Элемент выбора цели «следит» за переменной m и порождает переменную выбора цели c, управляющую передвижением компьютера. Он рассматриваетсякак метасистема

$$\mathbf{MF} = (T, \mathscr{F}, r),$$

где

$$\mathscr{F} = \{{}^{l}\mathbf{F} = ({}^{l}\mathbf{S}, {}^{l}M, {}^{l}f) \mid l \in N_{0,99}\},$$

T определено в регулярных интервалах времени, в течение каждого из которых компьютер не перемещается, а r указывает, что каждое состояние l переменной c однозначно определяет конкретный элемент поведения ${}^{l}\mathbf{F}$ метасистемы. Таким образом, элемент ${}^{l}\mathbf{F}$ метасистемы **MF** замещается в соответствии с заменой состояний переменной выбора цели с.
Отдельный элемент ${}^{l}\mathbf{F}$ метасистемы **MF** основан на маске, показанной на рис. 7.

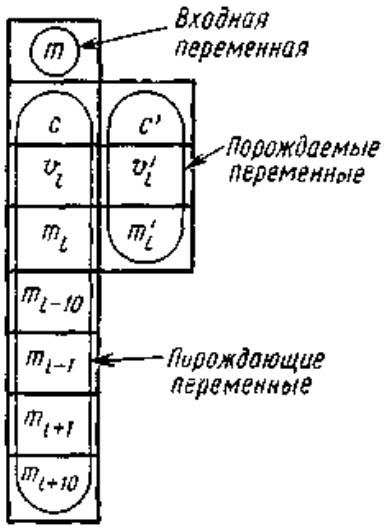


Рис. 7. Маски ${}_iM$ систем с поведением, в примере 3

Она определяет входную переменную m, выходную переменную c и шесть внутренних переменных. Переменная v_i соответствует суммарному числу переходов через позицию (квадрат) l, переменная m_l определяет среднее число сбоев компьютера по всем переходам через позицию l, остальные переменные определяют те же средние числа для четырех квадратов, соседних с квадратом l; переменные c', v'_l, m'_l представляют соответственно следующие состояния переменных c, v_l, m_l и являются порождаемыми переменными.

Переменные m'_l и v'_l порождаются детерминированно по формулам

$$m_l' = \frac{v_l m_l + m}{v_l + 1},$$
$$v_l' = v_l + 1; \tag{9}$$

будем считать, что m'_l вычисляетсяс точностью 0.01. Переменная c' в системе ${}^l\mathbf{F}$ может принимать значения только из множества

$$L = \{l-10,\ l-1,\ l,\ l+1,\ l+10\}.$$

Она порождается случайным образом, так что вероятности отдельных состояний обратно пропорциональны среднему числу сбоев в соответствующем квадрате. Тем не менее, чтобы сделать систему

адаптивной к неожиданным изменениям пространственно-распределенных неисправностей, необходимо присвоить всем состояниям ненулевую вероятность. Следовательно, если m_c ($c \in L$) настолько велико, что $1/m_c < 0.01$, то $1/m_c$ полагается равным α, где $\alpha > 0$ — некоторая постоянная величина; положим $\alpha = 0.01$.

Функция поведения ${}^lf$ сисемы ${}^l\mathbf{F}$ определяется следующим образом. Сначала определим

$$q_c = \begin{cases} 1/m_c, & \text{если } m_c \geqslant 0.01, \\ 0.01, & \text{если } m_c < 0.01, \end{cases}$$

при $c \in L - \{l\}$ и

$$q_l = \begin{cases} 1/m_l', & \text{если } m_l' \geqslant 0.01, \\ 0.01, & \text{если } m_l' < 0.01. \end{cases}$$

Тогда

$${}^lf(c' \mid m'_l,\ m_{l-10},\ m_{l-1},\ m_{l+1},\ m_{l+10}) = q_c \cdot k,$$

где $c' \in L$, а k — нормализующая константа, вычисленная по формуле

$$k = 1/(q_{l-10} + q_{l-1} + q_l + q_{l+1} + q_{l+10}).$$

Для клеток на границе рабочей области при вычислении вероятностей необходимо внести очевидные изменения.

На рис. 8 приведена схема, на которой показаны основные компоненты, входящие в целенаправленную метасистему **MF**.

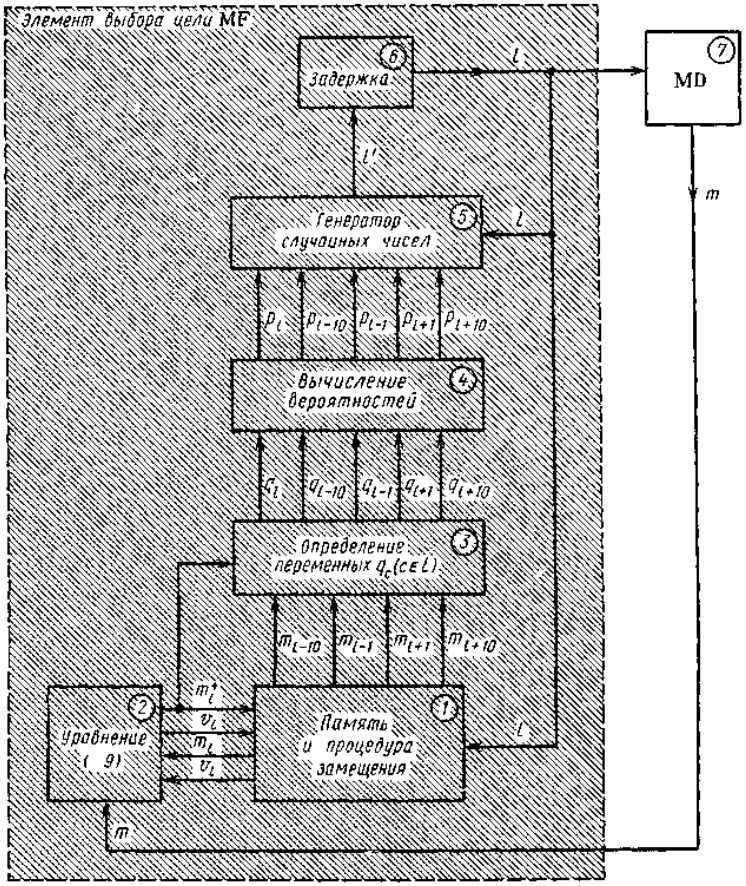


Рис. 8 Схема элемента выбора цели адаптивной системы, рассмотренной в примере 3

Блок 1—это память, в которой хранятся состояния выборочных переменных всех систем с поведением в **F**. Вход l представляет процедуру замещения: из памяти считываются состояния выборочных переменных, связанных с системой $^{l}\mathbf{F}$ и обеспечивающих сохранение состояния переменных m'_l и v'_l в соответствующей ячейке памяти. Блок 2 представляет уравнение (9), определяющее вычисление переменных m'_l и v'_l. Блок 3 определяет вспомогательные переменные q_c $(c \in L)$, исходя из которых вероятности

$$P_{c'} = {}^{l}f(c' \mid m_l', \quad m_{l-10}, \quad m_{l-1}, \quad m_{l+1}, \quad m_{l+10}), \quad c' \in L,$$

вычисляются в блоке 4. Блок 5 — это генератор случайных чисел, с помощью которого в соответствии с распределением вероятностей определяемся состояние переменной c' ($c' \in L$). Блок 6 — интервал времени, в течение которого компьютер не перемещается. До начала работы описанной целенаправленной системы в памяти (блок 1) может содержаться любая имеющаяся в этот момент времени информация о конкретном распределении отказов. Если никакой информации нет, то в памяти значения всех переменных полагаются равными нулю. Однажды запущенная целенаправленная система управляет перемещениями компьютера в соответствии с моделью окружающей компьютер среды (рабочей области). Модель определяется содержанием памяти целенаправленной метасистемы **MF** (блок 1 на рис. 8). Благодаря изменению переменных l и m модель постоянно обновляется. Система работает в соответствии с ее ожиданием реакции среды на соответствующее действие компьютера. Системы этого типа, способные приспосабливать модель к окружающей среде и использующие эту способность заранее, являются наиболее сложными адаптивными целенаправленными системами ИИ. Они обычно называются *упреждающими системами ИИ.*

Рассмотрим теперь некоторые свойства описанной адаптивной системы. Во-первых, можно выделить три типа сбоев оборудования:
сбои, связанные непосредственно с оборудованием;
сбои, обусловленные возмущениями, равномерно распределенными в рабочей области;
сбои, обусловленные локальными возмущениями в рабочей области.

Очевидно, что переменная выбора цели (положение в рабочей области) может влиять только на ошибки третьего типа.

Во-вторых, необходимо отметить, что описанная система реагирует на произвольное событие, угрожающее ее нормальной работе (коррекция выполнения требуемых действий компьютера). Нет необходимости в том, чтобы природа таких событий была заранее оговорена; все, что угрожает способности к нормальной работе, вызывает реакцию, направленную на сохранение этой способности. **Поэтому естественно назвать эту систему *самосохраняющейся системой* (т е. системой, стремящейся сохранить свою способность к нормальной работе).**

В-третьих, α — минимальное значение q_c $(c \in L)$, постоянное для конкретной системы, существенно влияет на способ адаптации системы к изменениям в окружающей среде. Если α слишком мало по сравнению с обычными значениями q_c, как в сбоях второго и третьего типа, то система медленно распознает изменения в окружающей среде.

Если α слишком велико, то система быстро распознает изменения, но ее модель окружающей среды становится непригодной, когда изменения происходят с меньшей скоростью, чем скорость, с которой компьютер может перемещаться.

8.7. Самовоспроизводящиеся системы ИИ

Самовоспроизводящаяся система (определенная как единое целое) представляет собой сеть процессов создания (преобразования и разрушения) компонент, производящих компоненты так, что: (1) при их взаимодействиях и преобразованиях непрерывно восстанавливается и осуществляется сеть процессов (отношений), которые их производят и (2) она составляет единое целое (машину) в пространстве, в котором она существует благодаря определенной топологической области ее реализации, представляющей собой подобную сеть.
Франсиско Дж. Варела

Цель настоящего раздела состоит в формулировании на языке ФРИЗ наиболее неортодоксального класса целенаправленных систем ИИ, которые мы будем называть *самовоспроизводящимися системами ИИ.* Слово autopoiesis — греческое и буквально переводится как «самовоспроизведение». Однако самовоспроизведение в самовоспроизводящихся системах ИИ не произвольно, а должно удовлетворять определенным требованиям.
В общем случае самовоспроизводящиеся системы ИИ функционируют на конечном и дискретном пространственно-временном параметрическом множестве. Пространство обычно имеет размерность 2 или 3, но бывает и *k* -мерное *(k≥1).* Полное описание самовоспроизводящихся систем ИИ состоит в том, что эти системы посредством набора правил перемещения и производства элементов нескольких различных типов, распределенных в пространстве, образуют и сохраняют во времени пространственно различимое целое.
Следовательно, самовоспроизводящиеся системы ИИ являются целенаправленными. Целью здесь является какой-либо тип границы, называемой *топологической границей,* позволяющей наблюдателю выделить часть пространства как единое целое. Некоторые из правил перемещения и производства, связанные с достижением цели в

конкретной самовоспроизводящейся системе ИИ можно, таким образом, рассматривать как свойства выбора цели данной системы ИИ.
Понятие самовоспроизводящихся систем впервые возникло в биологии, где можно найти множество примеров таких систем. К наиболее простым биологическим объектам, образование и сохранение которых как пространственных агрегатов можно описать в терминах самовоспроизводящихся систем, относятся, как отметил М. Зелены, биологические клетки:
«Мы интуитивно наблюдаем феномен самовоспроизведения в живых системах. Например, клетка — это сложная система, синтезирующая и производящая среди прочих макромолекул макромолекулы белков, липидов, ферментов, она состоит в среднем из 10^5 макромолекул. **За полное время жизни данной клетки все макромолекулы возобновляются приблизительно 10^4 раз.** Но **в течение всего процесса клетка сохраняет свои отличительные свойства, связность и относительную независимость. Она производит мириады компонент, но все же не производит ничего, *кроме самой себя.*** Хотя за время жизни клетка объединяет по меньшей мере 10^9 различных составных молекул, **она сохраняет свою индивидуальность и отличительные свойства. Сохранение единства и целостности в то время, как сами компоненты непрерывно или периодически распадаются и возникают, создаются и уничтожаются, производятся и потребляются, и называется самовоспроизведением.»**
Самовоспроизводящаяся система ИИ обычно описывается в терминах определенных типов, которым часто присваиваются такие названия, как **субстраты, катализаторы, дырки, звенья и т**. п. В каждый момент времени из определенного временного множества в любой выделенной области заданного пространства может находиться только один из компонентов определенного типа. Компоненты претерпевают пространственные преобразования во времени в соответствии с определенными правилами (правила перемещения и производства). При тщательном выборе этих правил система в состоянии образовывать и сохранять топологическую границу определенного типа (цель системы ИИ), и, следовательно, ее можно рассматривать как самовоспроизводящуюся систему ИИ.
Один из способов описания самовоспроизводящихся систем ИИ на языке ФРИЗ состоит в представлении их как метасистем ИИ вида

$$\mathbf{MD} = (T, \mathcal{D}, r).$$

Множество $\mathcal{D}$ в конкретной самовоспроизводящейся системе ИИ состоит из всех систем данных, которые можно определить в терминах

определенного пространства (как параметра) и единственной переменной, состояния которой представляют компоненты самовоспроизводящейся системы ИИ. Так, например, состояния 0, 1, 2, 3 и т. д. могут соответственно представлять дырки, катализаторы, субстраты и т. д. Параметрическое множество *T* метасистемы ИИ— это временное множество самовоспроизводящейся системы ИИ; оно полностью упорядочено, и ее элементы представляют соответствующие интервалы времени . Процедура замещения состоит из всех правил перемещения и производства самовоспроизводящейся системы ИИ. Замещение одной системы данных другой происходит каждый раз, когда одно время в множестве *T* заменяется следующим.

Пример 4. Опишем простую самовоспроизводящуюся систему, как метасистему. Она состоит из

временного множества *T,* которое удобно представить множеством неотрицательных чисел;

множества систем данных *D,* основанного на той же представляющей системе, параметром которой является двумерное пространство, определяемое двумя декартовыми координатами $x, y \in N_{0,9}$ и содержащее единственную переменную, имеющую пять состояний, которым присвоены следующие содержательные имена:

0 — дырка, 3 — звено,

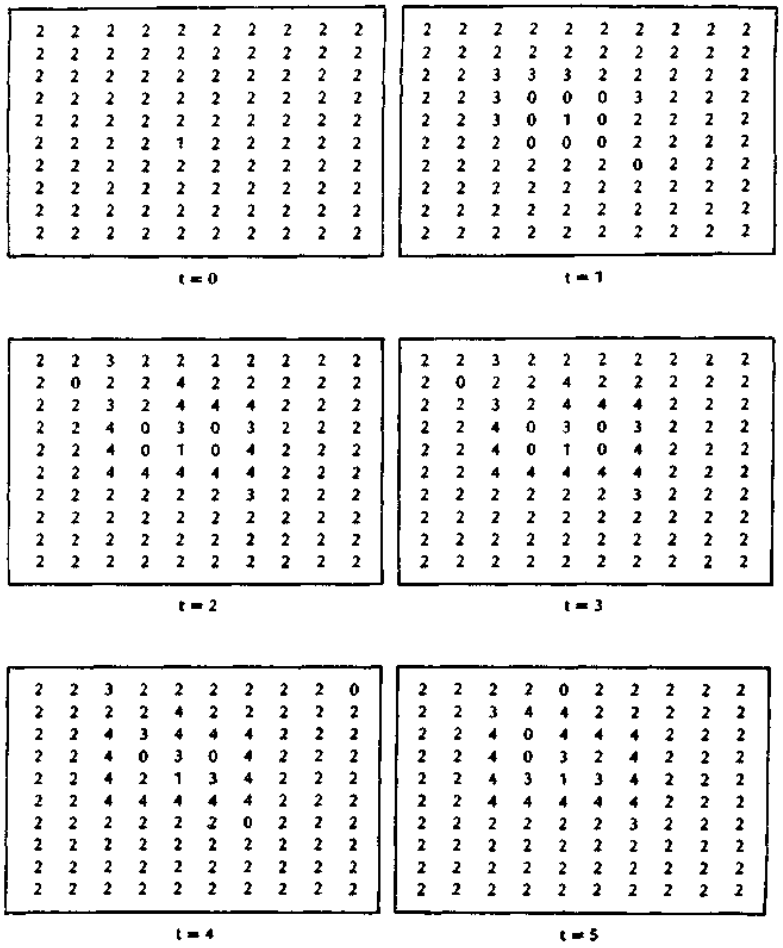
1 — катализатор, 4 — связующее звено;

2— субстрат,

процедуры замещения, определенной следующими правилами:

1. Два соседних субстрата, один из которых граничит с катализатором, соединяются с образованием звена.
2. Соседние звенья соединяются с образованием связных звеньев (замкнутое связное звено образует границу, сохранение которой является целью самосохраняющейся метасистемы) .
3. Произвольно выбранные свободные или связные звенья распадаются с образованием двух субстратов или двух звеньев соответственно. (В дальнейшем полученные компоненты могут вновь соединяться)
4. Субстраты могут передвигаться в любую соседнюю пустую подобласть, пересекая при необходимости отдельное звено или связные звенья. Звенья могут перемещаться в пустые подобласти, а также замещать субстраты, либо вытесняя их в соседние дырки, либо меняясь с ними местами. У катализаторов такая же, как и у звеньев, свобода в перемещении, и они могут замещать звенья. Тем не менее в отличие от субстратов ни звенья, ни катализаторы не могут пересекать сегменты связных звеньев. Связные звенья не перемещаются. Таким образом, в каждый момент времени однозначно

определяется система данных $D \in \mathcal{D}$. Системы данных замещают друг друга во времени в соответствии с правилами процедуры замещения. Процедура начинается с начального состояния (система данных), в котором в каждой позиции находится либо субстрат, либо катализатор. На рис. 9 показаны семь поспедовательных моментов времени из одной серии применений процедуры *r*.



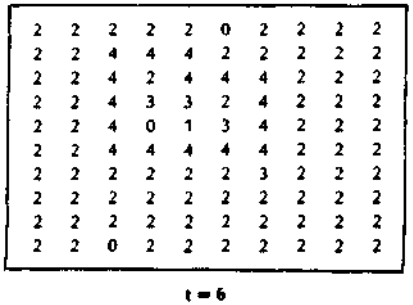


Рис. 9. Самосохраняющаяся систе ма (пример 4)

Целью этой системы является образование замкнутого пространства, ограниченного связными звеньями (топологическая граница). На рис. 9 видно, что система действительно является целенаправленной. Топологическая граница начинает образовываться в момент времени t=2 и полностью завершается при t=6.

Наиболее подробно изученными целенаправленными системами являются *регуляторы.* Основные принципы регулирования сформулировали У. Росс Эшби и Р. Конант. Очень важен закон Эшби о *необходимом многообразии.* Закон утверждает, что производительность любого физического устройства как регулятора не превышает его производительности как канала связи. Из этого закона следует, что многообразие регулирующих переменных можно понизить до желаемого уровня (что и является целью регулятора), только увеличив ряд регулирующих переменных по меньшей мере до соответствующего минимума. Закон этот часто формулируют следующим образом: **только многообразие может уменьшить многообразие.**

Иерархически организованные многоуровневые регуляторы изучены во многих работах. В наиболее общем виде результаты этих работ формулируются так: **чем слабее в среднем возможности регулирования и чем больше неопределенность имеющихся регуляторов, тем более высокая иерархия необходима при регулировании и управлении, чтобы достичь, если это возможно, тех же результатов регулирования Данное утверждение часто называют *законом необходимой иерархии*** Из этого закона следует, что недостаточные возможности регулирования можно до некоторой степени компенсировать с помощью представления регулятора как иерархической многоцелевой структурированной системы.

Кроме этих общих принципов регулирования, много результатов, связанных с регулированием, было опубликовано под рубрикой «Теория управления». Они, в первую очередь, относятся к регуляторам с обратной связью и посвящены конкретным классам систем (непрерывным, линейным).

9. Познавательные структуры и подобие систем ИИ

9.1. Познавательные структуры ИИ

По существу своему понятия познавательной структуры относятся к представлению внешней среды. Любой организм, приспосабливающийся к своей среде, должен иметь такое представление. Оно может быть выражено в неявном виде, через организацию реакции на стимулы, или в явной форме — в виде **системы познавательных объектов**, соотнесенных друг с другом способами, аналогичными отношениям между представляемыми ими объектами окружающей среды. Прежде всего рассмотрим, как следует представлять познавательные структуры ИИ в форме, совместимой с другими познавательными структурами.

Может оказаться полезным рассмотреть представление познавательных структур ИИ прежде всего на примере реализации некоего **понятия в программированной модели познавательных функций человека**. Структура, о которой идет речь, т. е. **система родственных связей**, является частью систематических усилий по созданию моделей интеллектуальных процессов, участвующих в процессе понимания. Будем считать **понимание, как процесс адаптивного усвоения новой информации об объектах и их взаимосвязях**. Здесь мы рассмотрим его структурную схему, чтобы выяснить, как можно представить *наличие* у индивидуума знаний о системе семейных отношений.

Рассмотрим следующий пример:

Фил — : муж (чей), Алиса;
родитель (чей), подсписок 1; ребенок (чей), ...
Подсписок 1: Джон, ...
Алиса — : жена (чья), Фил;
родитель (чей), подсписок 1;
ребенок (чей), ...

Подсписок 1: Джон, ...
Джон — : муж (чей), Мэри;
родитель (чей), . ..;
ребенок (чей), подсписок 1.
Подсписок 1: Фил, Алиса.
Мэри—: жена (чья), Джон;
родитель (чей), ...;
ребенок (чей), ... (1)

Мы можем понимать пример (1) как выражение некоторых представлений индивидуума о семейных отношениях между Филом, Алисой, Джоном и Мэри. Отметим, что взаимные отношения этих четырех представлений связывают их в более крупную структуру, которую можно изобразить так, как показано на рис. 1.

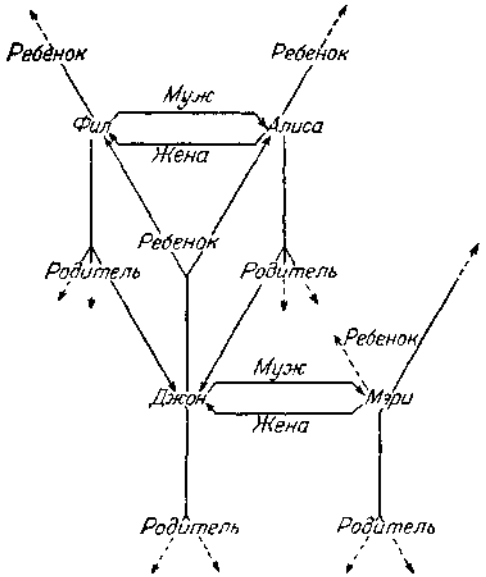


Рис. 1. Представление части познанагельной структуры (по 1).

Если окажется, что схеме рис. 1 недостает всего того множества связей, которые мы связываем с представлением о познавательной, структуре, вспомним, что мы в состоянии изобразить на бумаге лишь небольшую часть того, что является само по себе очень ограниченной системой по сравнению с реальной познавательной структурой. Если мы даже перейдем от листа бумаги к памяти вычислительной машины, мы все еще будем ограничены в том, что мы можем в нее вложить.

Но эти ограничения значительно менее жестки, и мы можем гораздо лучше аппроксимировать сложность познавательных структур в программированных моделях, систематически добавляя к ним новые элементы и новую информацию относительно взаимодействий между ними. Несмотря на фрагментарный и ограниченный характер этого примера, он, однако, ясно показывает, как представление концепций индивидуума приводит к представлению познавательной структуры в результате перекрестных связей между этими концепциями в их взаимных отношениях.

Область семейных отношений также хорошо иллюстрирует один из основных способов вывода новых отношений из более простых. Рассмотрим, например, систему семейных отношений, охватывающую только *n* различных лиц *a, b, c,..., n,* связанных между собой двумя бинарными отношениями: *C* (ребенок кого-то) и *P* (родитель кого-то). Мы можем написать *aPb,* чтобы указать, что *a* — родитель *b,* и *bCa,* чтобы указать, что *b* — ребенок *a.* Тогда *a — дедушка* (или *бабушка) c* в том и только в том случае, если *a* — родитель некоторого *x,* а *x* — родитель *c*. В символической форме, если символом *G* обозначить отношение «быть дедушкой (или бабушкой) кого-то», то *aGc* тогда и только тогда, когда *aPx* и *xPc* или, иначе, тогда и только тогда, когда *aPPc* Отношение *G* — простой пример отношения-произведения, которое Трэлл определяет так:

«Если *R*— отношение между элементами *a* множества *A* и *B* множества *B,* а S — отношение между элементами *b* множества *B* и *c* множества *C,* то *отношение-произведение* (или композиция) *RS* определяется как отношение между элементами *a* множества *A* и *c* множества *C* следующим образом: *aRSc* всегда, если для некоторого *b* в множестве *B* имеют место *aRb* и *bSs» (*Для нашего примера системы семейных отношений *A, B* и *C* в определении Трэлла являются эквивалентными названиями одного и гого же множества *n* различных персон.*)*.

Формирование отношений-произведений является источником **сложных понятий**, поскольку мы можем образовывать произведения любого числа отношений. В качестве другой иллюстрации, опять-таки на основе системы семейных связей, содержащей множество *n* различных персон с определенными выше отношениями С и *P,* рассмотрим отношение *K* (кузен кого-то), которое может быть определено через отношение-произведение *CCPP.* Это означает, что *a* — кузен *e(aKe)* тогда и только тогда, когда *a*—ребенок некоторого *b(aCb), b* в свою очередь — ребенок некоторого *c(bCc), c* — родитель некоторого *d(cPd)* и, наконец, *d* — родитель *e(dPe).*

Смысл в познавательной структуре.

Одно из классических прозвищ, которым награждали английских философов-ассоциационистов, — «умственные алхимики восемнадцатого и девятнадцатого столетия»— было связано с их трактовкой сложных представлений. Считалось, что значения сложных идей строятся из значений более простых идей. Но если, возражали скептики, представление о кирпиче действительно включает понятие глины, а более сложное представление о доме включает представления о кирпичах, окнах, дверях и т. п. до бесконечности, то сколь невероятно сложным должно быть представление о любом объекте!
Когда мы рассматриваем это утверждение в свете трактовки познавательной структуры, оно кажется уже менее безнадежным. Представление о чем бы то ни было не должно *содержать* что бы то ни было в большей степени, чем представление , например, о приемлемой служебной одежде должно содержать представления обо всех брюках, куртках и обуви, которые могли бы когда-то образовать часть такого наряда. Понятие чего-либо может *быть* чрезвычайно сложным в чьем-либо представлении. Но, как показывает этот пример, нет никакой *необходимости,* чтобы оно было более сложным, чем большинство других понятий.
Рассматриваемый вопрос не является острой проблемой в теориях познания и искусственного интеллекта . Тем не менее он хорошо иллюстрирует аналитическое значение модели, которая обеспечивает хотя бы возможный способ понимания таких познавательных проблем и явлений. Поскольку имеется весьма обширная литература по проблеме «смысла» (meaning), мы не будем поднимать здесь этот вопрос. Однако трудно представить себе, как мы могли бы обсуждать познавательные структуры *без* понятия «смысла», а схема, которую мы рассмотрели, дает по крайней мере полезный подход к этим проблемам.
Мы должны различать самодовлеющие системы ИИ и системы ИИ, служащие для представления других систем ИИ или применяемые к другим системам ИИ. Наша **обобщенная схема имеет своей задачей представление познавательных структур ИИ.** Но многие из элементов познавательных структур сами связаны, например, через системы восприятий и реакций с элементами, находящимися вне индивидуума. «Смысл» познавательного элемента, имеющего такие внешние отношения, будет при развиваемой нами интерпретации в значительной степени включать системы элементов, внешние по отношению к самой познавательной структуре ИИ. В частности, если индивидуум имеет доступ к внешнему элементу, он может получить новую информацию, которая может быть ассоциирована с познавательным представлением внешнего элемента. Для упрощения

мы, однако, сначала ограничим обсуждение, чтобы исключить те аспекты «смысла», которые вытекают из соответствий между познавательными элементами и элементами за пределами познавательных структур. Позднее мы смятим это ограничение; сейчас же мы рассматриваем познавательную систему ИИ совершенно изолированно от окружающей среды.
Для такой **изолированной познавательной структуры «смысл» некоторого элемента понимается как функция его взаимосвязей с другими элементами в системе ИИ**. Возможно, наиболее просто иллюстрировать это, сославшись на представления гипотетического испытуемого о ряде карточек в опыте Брунера и др. Предположим, что все карточки — положительные примеры понятия двух синих фигур. Если испытуемый, думая о них, игнорирует все признаки, кроме числа фигур и их цвета, то все карточки становятся для испытуемого эквивалентными и взаимозаменяемыми. Но с точки зрения полного описания ситуация здесь совершенно иная. Теперь уже взаимозаменяемыми будут лишь те элементы, которые тождественны по всем шести парам признак — значение. В остальных отношениях каждая карточка имеет единственный «смысл» с той точки зрения, что она связывается через шесть признаков с множеством значений, отличным от множества для какой-либо другой карточки Поэтому ее можно различить среди других познавательных элементов всецело на основании этой единственной конфигурации взаимосвязей Полезно будет рассмотреть тот же вопрос в более абстрактном виде.
Рассмотрим граф на рис. 2, на котором даны элементы *А1 — А6* некоторого множества А и их взаимосвязи.

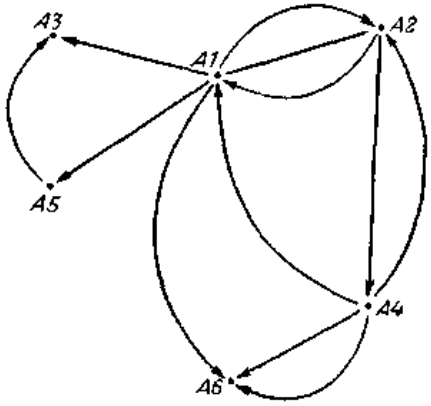


Рис.2. Граф С-отношений (кривые линии) и S-отношений (прямые линии) между парами точек множества *А*

Эти взаимные связи и представляют два отношения *C* и S и изображены соответственно дугами и прямыми линиями. При желании можно представить эквивалентную структуру в списковом виде. Она начиналась бы так:

A1—: S, подсписок 1; C, подсписок 2.
Подсписок 1: *A2, A3, A5*
Подсписок 2: *A2, A6.* (2)

Читатель может при желании убедиться в полной эквивалентности графа и спискового представления, выписав выражения для всех элементов из А. Граф может быть использован для представления некоторой части познавательной структуры какого-либо человека. Для конкретности мы можем интерпретировать шесть точек на графе как изображение людей, кривыми линиями обозначить, кто на кого похож, а прямыми — родительские отношения. Но оставим рисунок в абстрактной форме, потому что вопрос, который мы разбираем, является совершенно общим и не зависит от конкретного содержания отношений.

Здесь мы снова хотим **исследовать связь между «смыслом» элемента и его отношениями с другими элементами.** Если индивидуум игнорирует наименования шести точек в графе и не отличает линии, связывающие их, он лишается всякой возможности их различения. Все они будут для него одинаково бессмысленными. Но если не сообщить ему ничего, кроме информации о S-отношениях среди элементов, то ситуация изменяется. Теперь неразличимыми элементами будут только *A3* и *A5.* Каждый другой элемент имеет единственную структуру взаимосвязей по отношению к другим элементам сети.

Рассмотрим в связи с рис. 2 еще несколько примеров. Предположим, что обсуждаемому индивидууму дали описание (3) и попросили его обнаружить среди познавательных элементов, соответствующих точкам на рис. 2, элемент, удовлетворяющий описанию.

Описание —: *S,* подсписок 1; в, *A.*
Подсписок 1: подсписок 2, подсписок 3, подсписок 4.
Подсписок 2 — : в, *A.*
Подсписок 3 — : в, *A.*
Подсписок 4—: в, *A.* (3)

Что же определяет описание? Оно определяет множество, состоящее из всех тех элементов, которые есть в множестве *A* и находятся в S-отношении к трем другим элементам из множества *A* Как видно из графа, только один элемент *A1* удовлетворяет этим требованиям. Подобно этому, если нам дано описание некоторого *x,* такого, что *x* находится в S-отношении с *u(xSu), v*—в S-огношении с *x(vSu),* а *w* — в

S-отношении с *v(wSv),* где *u, v, w* — снова различные элементы из *A,* то, как следует из графа, только *A4* удовлетворяет описанию. Другими словами, если мы ограничимся рассмотрением познавательной структуры в некоторый момент времени, разумно рассматривать «смысл» по отношению к *контексту,* где **контекст любого заданного элемента определяется его прямыми и косвенными связями с другими элементами структуры.**

После того как мы дали такое определение «смыслу», мы можем теперь смягчить первоначальное ограничение, изолирующее познавательную структуру от окружающей индивидуум среды. Тогда познавательная структура становится подсистемой в гораздо более обширной структуре, охватывающей как индивидуума, так и его среду. Связь между познавательной структурой и этой большой системой достигается благодаря отношениям, определяемым перцептуальным и двигательным аппаратом, связанным с этой познавательной структурой.

Весьма интересна программа, моделирующая способы хранения и использования «смысла» человеком, разработана Квиллианом. Структура памяти в его модели в концептуальном отношении весьма близка к подходу, излагаемому нами, но разработана во всех необходимых подробностях для того, чтобы ее можно было использовать в реальном процессе обработки существенной информации.

«...**Полная структура ассоциаций.. образует просто большую, очень сложную сеть узлов и однонаправленных соединений памяти между ними...** Отсутствует предопределенная иерархия над- и подклассов; *каждое* слово— «патриарх» своей собственной иерархии, *если с него начинается некоторый процесс поиска.* Аналогично каждое слово находится на различных местах в иерархиях большого разнообразия словесных понятий, если процесс поиска начинается с них».

Программа Квиллиана начинает работать с каких-либо двух слов, «смысл» которых закодирован в памяти, и открывает семантические отношения между ними. **Это осуществляется поиском в сети надлежащих «траекторий», т. е. цепочек узлов и связей, соединяющих эти два слова.** Так как каждое слово может иметь несколько возможных значений («смыслов»), то в каждом случае должны устанавливаться и подходящие его значения. Когда траектория или отношение найдено, программа выражает отношение между словами в виде некоторого предложения.

9.2. Подобие

В толковых словарях понятие подобия обычно определяется как «наличие общих свойств» или «сходство по сути или по неотъемлемым признакам». В соответствии с этим определением два объекта рассматриваются как подобные, если они одинаковы или, по крайней мере, **сопоставимы по некоторым, но не обязательно по всем свойствам.** Кроме того, предполагается, **что свойства, по которым они сравниваются в данном контексте, являются существенными.** Таким образом, для заданного набора объектов можно определить множество различных видов подобия в зависимости от свойств, которые считаются существенными в каждом конкретном случае. По-видимому, *геометрическое подобие*— это первый вид строго сформулированного и разработанного подобия. Оно было определено Евклидом («Начала», т. 6) следующим образом: «Подобными называются такие прямолинейные геометрические фигуры, у которых углы одинаковы, а стороны, противолежащие одинаковым углам, пропорциональны». В соответствии с этим определением фигуры, показанные на рис. 3, подобны потому, что

1) их углы равны, т. е.

$$\alpha' = \alpha,\ \beta' = \beta,\ \gamma' = \gamma,\ \delta' = \delta;$$

2) их X стороны, противолежащие равным углам, пропорциональны, т. е.

$$a' = ka,\ b' = kb, \ldots, e' = ke,$$

где k — коэффициент пропорциональности.

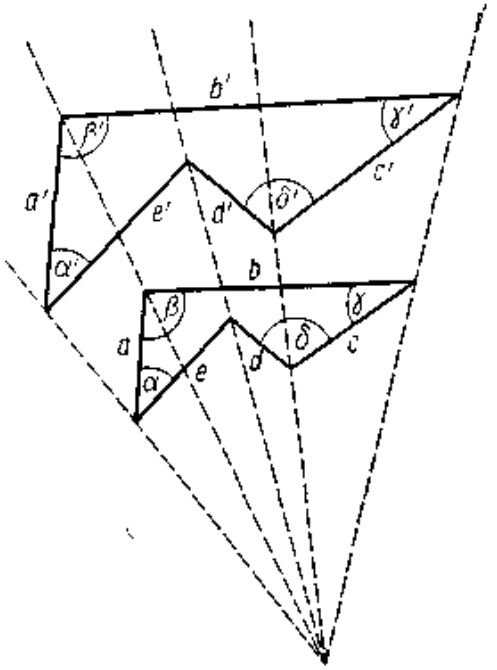


Рис. 3. Простое геометрическое подобие, основанное на группе линейных преобразований

Одна фигура может быть получена из другой простым линейным преобразованием, которое увеличивает или уменьшает фигуру, но не искажает ее формы. Такое преобразование обычно называется *конформным линейным преобразованием.*

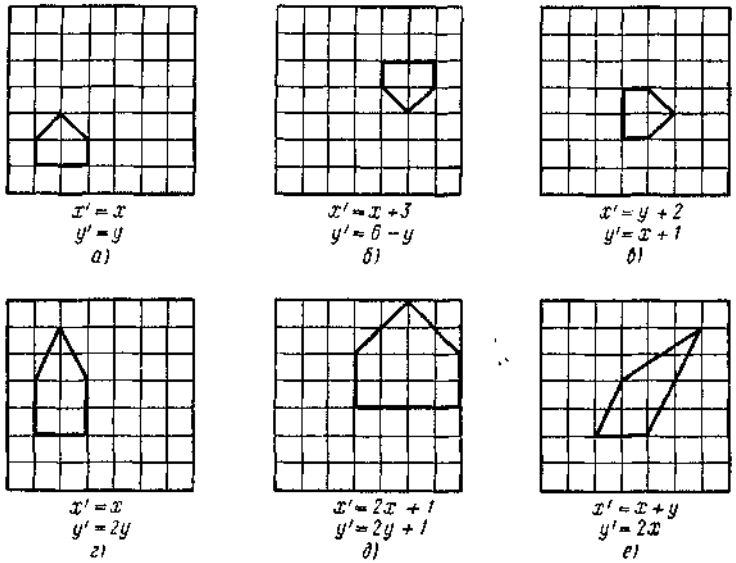
Понятие геометрического подобия плоских фигур можно значительно расширить, если рассматривать углы фигур как объекты линейного преобразования. При этом удобно представлять плоские фигуры как совокупности точек в двумерном декартовом пространстве с координатами x и y. Обобщенное геометрическое подобие плоских фигур, основанное на общем линейном преобразовании координат, может быть представлено уравнениями

$$x' = k_{1,x}x + k_{2,x}y + k_{3,x},$$

$$y' = k_{1,y}x + k_{2,y}y + k_{3,y}, \qquad (4)$$

где индексированные величины k являются постоянными коэффициентами; эти коэффициенты должны быть такими, чтобы уравнения (4) единственным образом определяли решения x и y при заданных x' и y'. Преобразование (4), естественным образом обобщающееся на случай трехмерного декартова пространства, обычно называют *общим аффинным преобразованием.* Оно включает различные виды преобразований (и подобия), такие, как преобразование, сохраняющее форму, симметричное отображение, одномерное растяжение или сжатие, вращение и т. д.

Пример 1. Рассмотрим дискретное декартово пространство $N^2_{0,7}$. На рис. 4 показано несколько фигур, которые являются подобными, т. е. переводятся друг в друга с помощью общих аффинных преобразований.



Рнс.4. Примеры плоских фигур, подобных относительно группы общих аффинных преобразований

Они представляют разные характерные частные случаи:
а) тождественное преобразование (исходная фигура); б) симметричное отображение относительно оси y и сдвиг; в) поворот и сдвиг;
г) растяжение вдоль оси y; д) растяжение и сдвиг (геометрическое подобие); е) общий случай.

Примером уравнений вида (4), не задающих аффинного преобразования, являются уравнения

$$x' = x + y,$$
$$y' = x + y.$$

Действительно, эти уравнения не обеспечивают единственного решения x и y: существует множество решений для x и y при условии $x'=y'$. Геометрическое преобразование (равнопропорциональное по всем направлениям) полностью сохраняет геометрический объект, если не считать его увеличения или сжатия. Поэтому можно говорить о том, что оно определяет строгое подобие для таких геометрических объектов, как плоские фигуры и твердые тела. Общее аффинное преобразование сохраняет геометрические объекты только приблизительно, так как допускает изменение формы в широком диапазоне; можно сказать, что этим преобразованием определяется слабое подобие геометрических объектов. Нарушения формы, допустимые при общем аффинном преобразовании, могут быть ограничены различными способами. Например, мы можем потребовать, чтобы все коэффициенты в уравнении (4), кроме $k_{1,x}$ и $k_{2,y}$, были равны нулю. Такое преобразование обобщает понятие геометрического подобия, так как допускает различные коэффициенты пропорциональности в разных направлениях, но определяет менее общий тип подобия, чем при общем аффинном преобразовании.

При замене одного элемента множества другим элементом подобие между этими элементами должно рассматриваться как *отношение эквивалентности,* определенное на этом множестве. Поэтому оно может быть представлено как *разбиение* этого множества определенной на нем *функцией* или *группой отображений* (взаимно однозначных и сюръективных) этого множества на себя (автоморфизмов). Например, в случае геометрического подобия (сохраняющего форму) множество всех равносторонних треугольников, всех квадратов, всех окружностей или всех прямоугольников, соответствующие стороны которых находятся в определенном отношении, являются примерами классов эквивалентности плоских фигур и в то же время блоками разбиения для данного вида подобия. С другой стороны, каждая определенная константа пропорциональности представляет один элемент преобразований, характеризующий подобие.

Хотя в этом и нет особой необходимости, интуитивное представление о подобии на заданном множестве элементов сначала формализуется с помощью группы преобразований. После того как эта группа определена, обычно делается попытка задать удобное для операций множество, элементы которого представляют собой классы эквивалентности, т. е. отдельные блоки отношения эквивалентности,

определенного на множестве группой преобразований. Такое множество обычно называется *множеством инвариантов* группы. В этом случае отношение эквивалентности может быть представлено соответствующей функцией из множества элементов, на котором подобие определяется на множестве инвариантов.
Рассмотрим, например, множество всех структурированных систем, которое может быть определено для заданного множества переменных в том смысле, как это рассматривалось в разд. 5.7. Группа перестановок переменных определяет на множестве отношение эквивалентности. Примером множества инвариантов этой группы является множество всех *непомеченных* схем, число входов которых равно числу рассматриваемых переменных. Различные виды подобия, определенные на одном и том же множестве, можно упорядочить по их общности. Один вид подобия является обобщением другого тогда и только тогда, когда множество разбиений, соответствующих первому виду, является укрупнением множества разбиений второго. Например, геометрическое подобие с различными коэффициентами пропорциональности является менее общим, чем подобие, основанное на группе общих аффинных преобразований, и в то же время более общим по сравнению с геометрическим подобием, основанным на группе преобразований, сохраняющих фигуру. Если некоторый объект считается подобным другому, то каждый из них сохраняет определенные свойства при определенных преобразованиях. В действительности свойства, которые при заданной цели желательно сохранить, являются обычно основой при определении подходящей группы преобразований. Например, если мы хотим сохранить вес геометрически подобных тел, сделанных из различных материалов, то константа пропорциональности c должна удовлетворять уравнению $c=(w/w')^{1/3}$,
где w и w'—удельные веса материалов, из которых сделаны тела, рассматриваемые как подобные.

9.3. Подобие и модели систем ИИ

При изложенном подходе ничего не стоит получить также некоторые важные результаты в отношении проблемы *подобия,* — проблемы, которая играет фундаментальную роль в теории познания и теории измерений. Обратимся снова к рис. 2. Являются ли элементы *A3* и *A5* подобными? Они не просто подобны, но и взаимозаменяемы, если рассматривается только отношение S в множестве *A* Но они совершенно различны, если мы рассматриваем С-связи. А при учете того и другого они подобны в некоторой степени, зависящей от

относительной важности каждого из этих отношений. То же касается и набора карточек в опыте Брунера, которые служат положительными примерами некоторого понятия, скажем понятия «две синие фигуры». Не имеет смысла говорить об абсолютном подобии каких-либо двух карточек. Мы должны спросить: подобие по отношению к чему? Если может быть применен ряд критериев, придающих больший вес тому или иному признаку, то понятие подобия приобретает смысл лишь тогда, когда оно оценивается путем задания некоторого критерия (правила, операции, признака и т. д.), который затем операционально фиксирует его смысл в данном контексте.
Джадсона и Кофера указывают на значение такого подхода к понятию подобия при объяснении эмпирических результатов. Испытуемым предъявляют карточки, содержащие по четыре слова каждая, и просят их отобрать «слово, которое является лишним». Пример (5) показывает одну из использованных при этом групп слов (слова означают прибавить, вычесть, умножить и увеличить *(англ)*) :

ADD SUBTRACT MULTIPLY INCREASE (5)

Как отмечает Кофер, «есть два способа классификации этих слов — как арифметических операций или как указаний на возрастание величины. В зависимости от принципа классификации, который использует испытуемый, он исключит, как лишнее, либо «increase», либо «subtract»... Subtract и increase можно назвать «недвусмысленными словами», а add и multiply — «двусмысленными словами». Джадсон и Кофер использовали две формы тестов, состоящих из сорока пяти таких слов. Формы были тождественны, за исключением двенадцати карточек, которые в этих двух формах отличались порядком слов. Так, карточка (4) во второй форме имела вид

ADD INCREASE MULTIPLY SUBTRACT. (5)

Одна группа испытуемых (около 65 человек) рассматривала первую форму, другая, сравнимая по величине группа — вторую форму. Важные различия в словах, отобранных как «лишние», содержались в девяти из двенадцати тестовых карточек.
Рассмотрим, например, результаты для двух форм карточки (4). Когда SUBTRACT является первым из двух недвусмысленных слов, оно выбирается как лишнее в 26,5% случаев. Когда это слово — второе, соответствующая цифра достигает внушительной величины 62%. Соответствующие результаты для слова INCREASE-30,7 и 70%.
По мнению Кофера, девять из двенадцати существенных изменений указывают, что «при некоторых условиях на решение задач может существенно повлиять относительный приоритет стимулов. Таким образом, весьма похоже на то, как будто первый недвусмысленный стимул, с которым испытуемый сталкивается в задаче из четырех слов,

оказывает огромное влияние на решение задачи; по-видимому, это влияние передается путем активации некоторого ответного семейства, категории или «понятия», которое становится доминантным вследствие того, что оно уже появлялось ранее в этом качестве, или из-за возрастания интенсивности, которое оно приобретает за счет двусмысленных слов, либо совместно от обеич причин. Эти результаты явно имеют отношение к проблеме воздействия вербального контекста...».

Итак, мы имеем экспериментальное исследование, показывающее большие **реальные эффекты**, которые, как отмечает Кофер, **имеют какое-то отношение к порядку слов, контексту, понятиям, признакам и подобию.** Но к чему именно? Можем ли мы с помощью рассмотрения интеллектуальных процессов выработать правдоподобную гипотезу, которая бы объясняла, *как* порядок слов, контекст и т. д влияют на суждения, которые выносят испытуемые Джадсона и Кофера? Другими словами, как передаются эти воздействия? Или, переходя к нашему вопросу-заменителю, **как мы могли бы запрограммировать систему интеллектуальных процессов, чтобы она вела себя таким же образом?**

Оказывается, мы уже *описали* систему, которая при минимальных видоизменениях будет обнаруживать именно такого рода поведение по отношению ко многим тестам Джадсона и Кофера.

Опишем семь основных шагов процесса.

1. Запомнить в качестве гипотезы значение каждого признака.
2. Если есть следующий стимул, взять его, если нет—остановиться.
3. Сравнить этот пример с гипотезой, чтобы определить, тождественны ли они, отметить и запомнить любое различие между ними.
4. Если пример тождествен гипотезе, снова перейти к шагу 2, если нет, то перейти к следующему шагу.
5. Установить обратную связь, указывающую, действительно ли стимул является примером понятия.
6. Если ответ „нет", вернуться к шагу 2; если „да"—перейти к следующему шагу.
7. Забыть любые значения предыдущей і и-потезы, которые отличаются от значений этого положительного ослабляющего примера. Перейти снова к шагу 1 (7)

Необходимые нам видоизменения целиком «связаны с различиями между ситуациями Брунера и Джадсона— Кофера, как их предъявляют испытуемым. В первой мы предполагаем, что испытуемому дается начальный положительный пример вырабатываемого понятия и что затем он получит информацию обратной связи о каждом примере,

после того как он встретится с ним. Задача испытуемого состоит в отыскании правила, определяющего положительные примеры. В случае конъюнктивных понятий это означает установление общих черт всех положительных примеров. Наша первоначальная модель не уточняла, как испытуемый может продемонстрировать свои знания, но мы можем потребовать, чтобы он отбирал только положительные примеры в карточках, которые он предварительно не видел, или чтобы он описывал характеристики, которыми должны обладать положительные примеры (например, две фигуры синего цвета).

В исследовании Джадсона и Кофера испытуемому сразу предъявляется четыре слова. Ему никогда не говорится, какие слова являются положительными примерами, и от него требуют ответить, какое из слов «является лишним». Чтобы приспособить программу Брунера к этим измененным экспериментальным условиям, предположим, что, хотя все слова в одной карточке предъявляются сразу, испытуемый имеет тенденцию читать и обрабатывать их в обычном порядке, т. е. слева направо. Кроме того, поскольку положительные примеры не указаны, программа должна *предполагать,* что каждый пример, с которым сталкивается испытуемый, является положительным, за исключением случаев, когда в нем нет *ничего* общего с текущей гипотезой о понятии. Когда встречается такой пример, система отбирает это слово как возможный ответ. Затем программа продолжает работать до тех пор, пока она не обработает последнее слово. Ввиду способа конструирования групп из четырех слов она должна в этот момент иметь только один возможный ответ, в этом случае программа выбирает это слово как «лишнее».

Может случиться, что какой-то испытуемый, обработав карточку, не найдет ни одного ответа или укажет сразу несколько возможных ответов. Такой результат может возникнуть в результате ошибок при переработке информации, а также из-за индивидуальных различий в ассоциациях со словами-стимулами. Если мы, однако, не желаем глубже разобраться в таких индивидуальных различиях, нам нужно лишь встроить генератор случайной функции, чтобы промоделировать поведение и в таких случаях. Тогда мы можем предположить, что при наличии более одного возможного ответа одно из слов отбирается среди них случайным образом. Если же не найдено ни одного возможного ответа, предполагается, что испытуемый образует ответ путем случайного выбора среди всех четырех слов.

При этих модификациях мы получаем модель процессов, участвующих в эксперименте Джадсона и Кофера. В соответствии с нашим обсуждением мы определяем «смысл», вызываемый в испытуемом словесным стимулом, как функцию взаимных связей

между использованным познавательным элементом и другими элементами в познавательной структуре. Используя отношения, предлагаемые Кофером, мы можем представить познавательный элемент некоторого испытуемого, соответствующий, например, слову MULTIPLY, следующим выражением:

multiply —: есть, арифметическая операция; указывает, возрастание величины; .. . (8)

Ниже дается схема программы.

1. Запомнить в качестве гипотезы значение каждого признака.
2. Если есть следующий стимул, взять его, если нет, *перейти к шагу 8.*
3. Сравнить этот пример с гипотезой, чтобы определить, тождественны ли они, отметить и запомнить любое различие между ними.
4. Если пример тождествен гипотезе, снова перейти к шагу 2; *если он не имеет ничего общего с гипотезой, поместить пример в список возможных ответов и снова перейти к шагу* 2; *в противном случае* перейти к следующему шагу.
5. Установить обратную связь, указывающую, действительно ли стимул является примером понятия. (Первый стимул Джадсона и Кофера и все последующие стимулы, имеющие что-либо общее с гипотезой, предполагаются положительными примерами.)
6. Если ответ „нет", вернуться к шагу 2, если „да", перейти к следующему шагу.
7. Забыть любые значения предыдущей гипотезы, которые отличаются от значений этого положительного ослабляющего примера. Перейти снова к шагу 1.
8. *Если это эксперимент Брунера и др., остановиться; если эксперимент Джадсона и Кофера, перейти к шагу 9.*
9. *Если имеется один возможный ответ, выдать этот ответ, если таких ответов больше одного, выбрать один из них случайным образом и выдать этот ответ; если возможных ответов нет, выбрать одно из четырех слов случайным образом и выдать этот ответ.* (9)

Следует отметить, что изменения, произведенные в шагах 2, 8 и 9, объясняются различиями входных и выходных условий в этих двух ситуациях. Они не означают изменения модели стратегии или участвующих информационных процессов. Изменение на шаге 4 отражает основную разницу целей в этих двух экспериментах. Если мы в опыте Брунера и др. начинаем с положительного примера, то стимул, не имеющий ничего общего с гипотезой о понятии, не может быть положительным примером этого понятия. Поэтому он будет

задержан на шаге 5 и просто удален на шаге 6. Изменение в шаге 4 позволяет выделить этот частный вид отрицательного примера и сохранить его для будущего использования. Такое изменение не обязательно в ситуации Брунера и др., но оно не помешает данной схеме образовывать понятия и в этой ситуации.

Если мы предложим модифицированной схеме такое сообщение, как (ADD, INCREASE, MULTIPLY, SUBTRACT), то произойдет следующее. Сделав шаг 1, схема сохраняет в качестве гипотезы значение ADD. Оно среди прочих содержит пары: (есть, арифметическая операция) и (указывает, возрастание величины). На шаге 2 выбирается INCREASE. На шаге 3 сравниваются значения ADD и INCREASE и отмечаются их различия. Шаг 4 приводит систему к шагу 5. Обратная связь предполагается положительной, и это заставляет систему через шаг 6 перейти к шагу 7, который стирает в гипотезе все те пары, которые не участвуют в ADD и INCREASE. Оставшаяся часть — это пара (указывает, возрастание величины) плюс, возможно, некоторые другие, которых мы не будем касаться. Теперь система возвращается к шагу 2, выбирает MULTIPLY, проходит ту же последовательность шагов и оставляет гипотезу неизменной. Возвращаясь опять к шагу 2, система выбирает слово SUBTRACT, для которого пара (указывает, возрастание величины) не входит в состав «смысла» слова. Этот факт отмечается на шаге 3, а на шаге 4 SUBTRACT помещается в список возможных ответов, так как у этого слова нет ничего общего с гипотетическим понятием (указывает, возрастание величины). Система снова возвращается к шагу 2. Стимулов больше нет, поэтому система делает шаг 8, который отсылает ее к шагу 9. Список возможных ответов проверен; выяснено, что он содержит одно слово — SUBTRACT, которое и становится ответом.

Что же мы теперь, после того как была представлена модель и продемонстрирована ее работа, можем сказать о ней? Ясно, что она очень проста и будет давать весьма маловероятные ответы на определенные сообщения. Но мы можем, конечно, при желании расширить ее несколькими путями, например включив в модель субъективные вероятности как функции подтверждения, так как слово MULTIPLY для некоторых испытуемых могло бы подтвердить гипотезу (указывает, возрастание величины). При желании мы могли бы также исследовать вариант, который смягчал бы предположение о последовательной пошаговой процедуре в пользу активной параллельной системы памяти. Но даже в данной простой форме кажется разумным предположить, что мы имеем модель интеллектуальных процессов, реализующую вероятную гипотезу

относительно способов, которыми приоритет и контекст оказывают влияние на суждения о сходстве и различии в стимулах. Иначе говоря, это модель общего посредничающего процесса в познании и мышлении. Кроме того, она обеспечивает конкретный механизм, посредством которого люди могут в процессе поиска решения какой-то задачи в неявном виде определять для самих себя подобие как некое отношение между объектами. Эта схема как раз и устанавливает операцию, дающую критерий подобия в рассмотренной ситуации. Наконец, если мы захотим запрограммировать схему и сможем реализовать программу в задуманном виде, эта модель явится также гипотезой относительно всех достаточных условий для рассмотрения приоритета, контекста, подобия и посредничающих процессов под таким углом зрения. Если программа работает, она тем самым доказывает свою способность служить для объяснения явления и способна предложить также другие эмпирические предсказания, которые могут быть подвергнуты исследованию.

В предшествующем обсуждении *«смысл»* (meaning) познавательного элемента определялся с учетом его местоположения в познавательной структуре (не физического, а относительного местоположения, определяемого общим множеством его взаимосвязей с другими элементами структуры). Если мы примем эту точку зрения на контекстуальную природу «смысла» (где контекст опять-таки относится к сети взаимосвязанных элементов, в которую вплетен каждый элемент), то что произойдет, когда мы «изымем элемент из контекста»?

Для иллюстрации рассмотрим снова рис. 2. Предположим, что мы хотим описать один элемент схемы, скажем *A6.* Для этого мы можем установить, что *A6* является элементом, «сходным» с двумя другими элементами в графе. Иначе говоря, *A6*— элемент, с которым два других, отличных от него элемента находятся в С-связи. Кроме того, один из этих других элементов также является родителем (т. е. находится в S-отношении к) *A6.*

Во многих случаях мы можем почувствовать, что такое описание говорит нам о некотором элементе меньше, чем могло бы дать более полное представление контекста, в котором он встречается. Исследователь познавательного несоответствия мог бы, например, пожелать иметь некоторую информацию об отношениях между этими двумя в остальном не описанными элементами. Действительно, обычно переработчик информации, не имеющий доступа к информации о других элементах, связанных с тем элементом, который мы непосредственно рассматриваем, окажется неспособным оценить полный «смысл» этого элемента. Ибо

существуют косвенные связи между элементами, и эти связи могут быть упущены, когда объект вырывается из контекста, как в приведенном выше примере; но именно эти косвенные связи создают возможность для установления новых , прямых связей между данным элементом и другими элементами сети.

Чтобы сделать эту мысль более ясной, предположим, что нам дано описание некоторого элемента *x,* где *x* — член познавательной подструктуры, представляющей семейные взаимосвязи между *n* различными индивидуумами. Предположим также, чго это описание *x* включает только связи между *x* и теми другими элементами, с которыми *x* непосредственно связан. Другими словами, пусть дано только это описание *x* и мы не желаем, например, иметь какую-либо другую информацию о родителях *x,* кроме той, что они родители *x.* В нашей познавательной структуре *t,* отец *x,* в свою очередь может быть ребенком *u (tCu),* а *u* — родителем некоторого *v (uPv)* и *v* — родителем некоторого *w (vPw).* Схема переработки информации, имеющая доступ к этой познавательной структуре и обладающая знанием эквивалентности между отношением *K* (кузен кого-то) и отношением-произведением *CCPP*, будет способна вывести *новую* прямую связь *xKw* между *x* и *w.* Но имея только описание *x* без описания всей познавательной структуры, та же интеллектуальная система, конечно, не будет в состоянии сделать такой вывод. Учитывая указанное выше влияние положения в контексте на «смысл», этот пример ясно показывает, в каком смысле можно считать эквивалентными понятия «изъятия элемента из контекста», «потери некоторой информации об элементе» и «изменения общего смысла элемента».

Не все изъятия из контекста оказываются одинаковыми по форме. Есть ряд процедур для отбора некоторых взаимосвязей и опускания других при описании элемента. Мы вовсе не предполагаем, что опускание информации всегда нежелательно и его следует избегать. Целесообразность его зависит от ограничений и целей, имеющих место в конкретном случае. Наша цель состоит лишь в прояснении отношений между проблемами контекста, «смысла» и описания, чтобы обеспечить более прямой и точный способ понимания этих проблем. Сама по себе модель познавательной структуры не решает задач и не имеет «мыслей». Чтобы показать, как много она оставляет без ответа, вообразим себе систему интеллектуальных процессов, которая могла бы проследить за ассоциативной системой, которую мы определили. На что была бы способна такая система? Нетрудно показать, как такая система могла бы отвечать на вопросы типа: «Каково *x* у *y*?», «Какое *y* имеет *z*?» и т. д., где *x, y* и *z*— познавательные элементы, а необ-

ходимые отношения между ними заданы непосредственно и в явной форме. Аналогично этому не представило бы трудностей заставить такую систему видоизменять эту структуру, добавляя к ней новые отношения, например *«x* есть *y* для *z* », сообщаемые ей извне. В этом весьма ограниченном смысле такая система могла бы быть способна обучаться и увеличивать свою познавательную структуру. Если такой системе задать адекватный словарь и простой синтаксис в нашей системе обозначений, она могла бы создавать познавательные элементы, соответствующие таким элементарным утверждениям, как «кошка — это млекопитающее», «кошка имеет четыре лапы и голову» или «Тимофей — это кот». Более того, не потребуется особых усилий, чтобы сделать систему способной описать Тимофея как нечто являющееся одновременно млекопитающим, котом, имеющим четыре лапы и голову, в соответствии с тем, что система знает. Разумно предположить, что описанная выше модель познавательной структуры дает адекватную основу для ограниченных моделей запоминания, простого вывода и элементарного взаимодействия со средой.

Но такие системы являются лишь зачаточными формами, если наша цель состоит в построении модели ИИ, которая была бы способна совершать это способами, сравнимыми с умственной деятельностью человека. Чтобы наглядно представить себе эту проблему, вообразите, что вы стали жертвой болезни, которая вызвала у вас временную слепоту и странную потерю памяти. Вы сохранили ваши знания грамматики, синтаксиса и фонетики родного языка. В остальном же, за исключением нескольких слов, вроде «что», «есть» и «определять», вы совершенно утратили понимание смысла слов. В вашей комнате есть два телефона. Один служит для связи с окружающей средой, другой соединен с хранилищем информации. Звонит первый телефон, и чей-то голос спрашивает: «Каково *x* у *y*?» Для вас слова *x* и *y* не имеют смысла. Это идеальные «бессмысленные слоги». Вы вызываете ваше хранилище информации и повторяете вопрос. Приходит ответ: *«z».* Вы сообщаете спрашивающему: *«x* у *y* есть *z* », либо: *«z* есть *x* у *y»* Он вам, далее, говорит, что Тимофей — это кот, что кот — это млекопитающее и т. д. Вы сообщаете эти данные вашему хранилищу и через некоторое время вы сможете восстановить их и сделать элементарный вывод, что Тимофей— это млекопитающее и имеет четыре лапы и голову. Но обычно вы проделываете все это без малейшего представления о том, о чем, собственно, идет речь. Вы можете с таким же успехом сказать, что *a* — это *b* и имеет *c* и *d.* Вы способны преобразовать форму высказывания, но только поточу, что вы сохранили ваши знания грамматики и

синтаксиса; вы знаете, что если «кудра ширяет бокренка», то «бокренок ширяется кудрой». Когда вы начинаете систематический поиск в вашем хранилище, каждое слово определяется цепочкой одинаково неизвестных слов, — нечто вроде ночного кошмара, который иной ученик хотел бы обрушить на своего учителя математики. Короче говоря, то, что вы можете делать в такой ситуации, весьма далеко от тех способностей, которые мы связываем с человеческим познанием и мышлением.

Определенное на множестве систем ИИ отношение подобия обычно называют *отношением моделирования.* Две системы ИИ подобны, если они сохраняют некоторые общие характеристики и могут быть преобразованы друг в друга соответствующими преобразованиями, примененными к другим характеристикам.

При решении интеллектуальных задач часто удобно (а иногда и необходимо) решать задачу для системы-заменителя, а не реальной системы, для которой была сформулирована эта задача. Использование подходящих систем-заменителей дешевле, быстрее, безопаснее, удобней, проще для понимания и контроля, точнее, более непротиворечиво и лучше приспособлено к человеческому восприятию. Две системы ИИ (реальная и ее заменитель) должны быть подобны в достаточно строгом смысле применительно к рассматриваемой задаче.

Рассмотрим две системы ИИ **x** и **y**, которые подобны относительно множества преобразований, применимых к некоторым их характеристикам. Будем считать, чго **x** — изучаемая система, а **y** — ее желаемое представление Тогда **x** называется *подлинной системой ИИ* (или просто подлинником), **y** - *модулирующей системой ИИ,* а **y** вместе с соответствующими преобразованиями называются *моделью* **x**.

Поскольку отношение подобия симметрично (как и любое отношение эквивалентности), то можно и систему **y** рассматривать как подлинную. Какая из ДВУХ систем рассматривается как подлинная, зависит от обстоятельств. Вопрос о том, является ли какая-то система подходящей моделью для подлинной системы ИИ, решается исключительно из прагматических соображений. Это решение принимается пользователем. Пользоватеть обычно выбирает нужную модель как заменитель подлинной, если, по его мнению, она имеет явные преимущества по сравнению с подлинником и в то же время не хуже любой другой из имеющихся в его распоряжении моделей.

Таким образом, термин «модель» используется нами в связи с определенным отношением между двумя системами. Он означает, что две системы в некотором смысле подобны и одна из них с определенной целью может быть заменена другой с помощью

соответствующих преобразований. Моделирующая система становится моделью, если ее дополнить преобразованиями, которые соответствующим образом связывают ее с подлинником. Другими словами, **каждой модели необходим подлинник**. **Для одного и того же подлинника можно построить различные модели**.

Помимо обычного, всем привычного значения термин «модель» в литературе используется для определения некоторых других понятий. К сожалению, этим термином злоупотребляют и небрежно с ним обращаются. Помимо обычного значения этот термин при решении интеллектуальных задач имеет по меньшой мере три различных значения.

Во-первых, термин «модель» часю используется для всех объектов, которые в настоящей книге называются системами ИИ, а термин «система ИИ» используется для обозначения того, что мы понимаем как «объект». Следовательно, в соответствии с данной терминологией **система ИИ - это часть мира, являющаяся предметом исследований, и всякое ее представление (образ) рассматривается как модель.** Терминологически оправдано назвать процесс исследования систем ИИ моделированием. В нашей терминологии все системы ИИ как абстракция ; некоторые из них являются описанием (образами) реальных явлении, и мы сохраним термин «моделирование» для отношения подобия между системами

Во втором значении термин «модель» употребляется как название множества предположений, при которых решается задача. Такими предположениями являются, например, аксиомы в математических теориях, используемые при решении некоторых задач. В этой книге для обозначения этого множества используется термин «парадигма». В литературе термин «модель» используется также для обозначения системы, являющейся упрощенной модификацией другой системы. Поскольку отношение упрощения антисимметрично, а отношение подобия симметрично, то терминологически следует различать эти два понятия.

Если отличать чисто абстрактные системы, не имеющие физических аналогов (без каналов наблюдения), от интерпретированных систем (будем называть их физическими системами), то можно выделить четыре типа моделирующих отношений в зависимости от природы подлинной и моделирующей системы:

Подлинная система	Моделирующая система	Тип
Физическая	Абстрактная	I
Абстрактная	Физическая	II
Физическая	Физическая	III
Абстрактная	Абстрактная	IV

К первому типу относятся все виды математических моделей. Эти модели основываются на известных физических и других законах природы. Они позволяют решать задачи, связанные с физическими системами, с помощью математических рассуждений (алгебраических преобразований или вычислительных методов) лучше, чем с помощью экспериментов с подлинными физическими объектами. Можно, например, ответить на многие вопросы, связанные с искусственными физическими системами до их создания. Вообще, **модели этого типа позволяют воспроизвести мысленный эксперимент (gedanken-эксперимент) на математических моделях гипотетического физического объекта.** Например, определение электрического тока и напряжения в гипотетической электрической цепи с помощью решения соответствующей системы алгебраических или дифференциальных уравнений предпочтительнее создания реальной цепи и выполнения соответствующих измерений.

Примером моделей второго типа являются всевозможные компьютеры: цифровые, аналоговые или смешанного типа. Сюда относятся также различные специальные физические системы, сконструированные как универсальные модели для математической системы определенного вида. В качестве примеров можно привести линейные анализаторы (для решения линейных алгебраических уравнений), полиномиальные анализаторы (для работы со степенными функциями), электролические ванны (для решения уравнений в частных производных).

Использование моделей этого класса состоит в изменении некоторых параметров физической системы и измерении значений других параметров, что позволяет описать решения соответствующих математических уравнений или ответить на некоторые математические вопросы.

Третий тип моделей играет важную роль в технике. Наиболее простым примером этого типа являются масштабные модели, представляющие собой системы, просто увеличенные или уменьшенные в определенном масштабе по отношению к подлиннику. Они используются, например, для изучения динамических свойств новых моделей самолетов, вертолетов или ракет в аэродинамических трубах, для испытаний новых типов кораблей в специальных водных бассейнах, для проверки плотин, мостов и других строительных проектов. Хотя все эти случаи сводятся к простому геометрическому подобию, результаты, полученные с помощью таких масштабных моделей, необходимо преобразовать соответствующим образом, чтобы они были применимы к подлинной системе. Действительно, при уменьшении (или увеличении) по отношению к подлиннику линейных размеров в *c* раз площади (т. е. площадь поперечного или продольного сечения крыла

самолета) уменьшаются (или увеличиваются) в c^2 раз, а объемы в c^3 раз. Задачи, возникающие в связи с изучением свойств преобразований для моделей этого типа, и их различные обобщения изучаются в *теории подобия* или *размерности.* Другими примерами этого типа являются компьютеры, моделируемые на других компьютерах, авиационные тренажеры для обучения пилотов, медицинские приборы, такие, как искусственное сердце или почка.

К четвертому типу относятся очень важные для прикладной математики модели. Они связаны с различными видами математических преобразований (преобразования Лапласа, Фурье и т. п.), используемых для моделирования математических систем одного типа (например, дифференциальных уравнений) системами другого типа (например, алгебраическими уравнениями). Вместо того чтобы работать с подлинной системой, мы можем использовать моделирующую систему, которая значительно проще, а затем применить полученные результаты к подлиннику. Для некоторых случаев (например, для преобразования Лапласа) имеются справочники, устанавливающие соответствие между подлинными и моделирующими системами.

Таково в общих чертах наше понимание отношения моделирования для систем; оставшаяся часть главы будет посвящена построению основных типов моделей в рамках концептуальной схемы ФРИЗ.

9.4. Модели исходных систем ИИ

Исходные системы ИИ просты в том смысле, что в них нет информации об отношениях межлу их переменными. Для двух заданных нейтральных исходных систем ИИ скажем систем **S** и **S'**, единственный разумный способ определения их подобия заключается в том, чтобы потребовать сохранение существенных свойств каждого определенного множества состояний и параметрического множества при преобразованиях, являющихся взаимно однозначным соответствием между

1) множествами переменных этих двух систем ИИ ;
2) множествами параметров этих двух систем ИИ;
3) множествами сосюянии переменных из п. 1;
4) параметрическими множеивами из п. 2

Это, по существу, означает, что системы ИИ **S** и **S'** изоморфны в смысле их обобщенных систем ИИ, но при этом могут быть семантически совершенно различны (в смысле каналов конкретизации и наблюдения).

Формально две нейтральные системы ИИ

$$= (\mathbf{O},\ \mathbf{I},\ \mathbf{i},\ O,\ \mathscr{O}),$$
$$= (\mathbf{O}',\ \mathbf{I}',\ \mathbf{I}',\ O',\ \mathscr{O}'),$$

где

$$\mathbf{I} = (\{v_i,\ V_i) \mid i \in N_n\},\ \{(w_j,\ W_j) \mid j \in N_m\});$$
$$\mathbf{I}' = (\{v_i',\ V_i') \mid i \in N_n\},\ \{(w_j',\ W_j') \mid j \in N_m\})$$

подобны тогда и только тогда, когда

v_i соответствует $v_{p(i)}$ ($i \in N_n$), где p обозначает перестановку N_n, т. е. p: $N_n \leftrightarrow N_n$;

w_j, соответствует $w_{q(j)}$ ($j \in N_m$), где q обозначает перестановку N_m, т е. q: $N_m \leftrightarrow N_m$.

для любого $i \in N_n$ $V_i \leftrightarrow V_{p(i)}$ существует взаимно однозначное соответствие, при коюром все существенные свойства множества V_i сохраняются в множестве $V_{p(i)}$;,

для любого $j \in N_j$, $W_j \leftrightarrow W_{q(j)}$ существует взаимно однозначное соответствие, при котором все существенные свойства W_j сохраняются в $W_{q(j)}$.

Если две нейтральные системы ИИ **S** и **S'** подобны в том смысле, что их обобщенные системы ИИ изоморфны, то любую из них можно рассматривать как подлинную систему ИИ, а другую как моделирующую. Моделирующую систему ИИ вместе с преобразованием, определенным взаимно однозначными соответствиями 1—4, можно рассматривать как модель другой системы ИИ (подлинника). Тем не менее необходимо подчеркнуть, что отношение моделирования не имеет практического смысла для исходных систем ИИ, так как не индуцирует никакого подобия между подлинником и моделирующей системой ИИ, поскольку возможности их варьирования весьма ограничены. Оно имеет практическое значение только в случае, когда рассматриваемые исходные системы ИИ являются компонентами систем ИИ эпистемологически более высокого типа. Понятие подобия в этом случае уточняется с помощью дополнительных требований. Подобие исходных систем ИИ в ряде случаев может возникать как необходимое условие подобия эпистемологически более сложных систем ИИ, в которые они входят, но никогда не является достаточным для такого подобия.

Пример 2. Примером подобия исходных систем в музыке является транспонирование из одной тональности в другую. Рассмотрим, например, исходную систему **S**, определенную в примере 6 раз.2.8 (рис. 8,б раз.2.8) и другую исходную систему **S'**, в которой каждая нота из множества V_1 понижена на один тон, а все остальное в точности так же, как в системе **S**. Будем далее считать, что пе-

ременные, описывающие высоту тона, ритм и тональность, в одной системе соответствуют переменным, описывающим те же свойства в другой системе, и что все взаимно однозначные соответствия между множествами состояний и параметрическими множествами одинаковы Тогда системы **S** и **S'** можно рассматривать как подобные относительно преобразования их исходных систем. Произвольная партитура, описываемая в терминах системы **S**, однозначным преобразованием транспонируется в аналогичную запись, описываемую в терминах системы **S'**. Эти соответствия имеют смысл для исходных систем, поскольку музыкальные записи являются их вторичными характеристиками. Но, если система **S** является частью некоторой системы данных **D**, а система **S'**— частью другой системы данных **D'**, то подобия систем **S** и **S'** недостаточно для определения имеющего смысл подобия систем данных **D** и **D'**. Действительно, данные **d**, **d′** систем **D**, D' могут быть произвольными записями, описываемыми в соответствующей исходной системе, но никоим образом не связанными отношением подобия.

Рассмотрим теперь две направленные исходные системы, скажем, как подлинную и моделирующую. Поскольку каждая переменная $\hat{\mathbf{S}}$ и $\hat{\mathbf{S}}'$ этих двух систем объявлена как входная или выходная, исходное управление (как самой системы, так и внешней среды) однозначно определено. Как следствие, взаимно однозначные соответствия между множествами переменных и множествами состояний $\hat{\mathbf{S}}$ и $\hat{\mathbf{S}}'$, которые требуются для подобия нейтральных исходных систем, могут быть заменены входными и выходными соответствиями (не обязательно взаимно однозначными) так, чтобы обеспечить управление каждой переменной только из одного источника. При условии однозначности контроля (аналогичное рассмотренному в гл. 5 требованию для структурированных систем) предполагается, что входные соответствия между множествами переменных направлены из $\hat{\mathbf{S}}'$ к $\hat{\mathbf{S}}$, а выходные соответствия между множествами переменных — из $\hat{\mathbf{S}}$ к $\hat{\mathbf{S}}'$. Соответствия между множествами состояний направлены в точности наоборот.

Поскольку не требуется взаимно однозначного соответствия между элементами систем $\hat{\mathbf{S}}$ и $\hat{\mathbf{S}}'$, то отношение моделирования между направленными исходными системами не обязательно симметрично; оно является квазиупорядочением (рефлексивное и транзитивное отношение), определенным на произвольном множестве направленных исходных систем. Это означает, что из возможности использования системы $\hat{\mathbf{S}}'$ для моделирования системы **S** не следует,

что последняя может использоваться для моделирования первой. Это, в свою очередь, означает, что, не являясь изоморфными, подобные системы $\hat{\mathbf{S}}$ и $\hat{\mathbf{S}}'$ связаны друг с другом двумя гомоморфными отношениями, одно из которых связано с входными переменными, а другое с выходными, и изоморфным отношением между их параметрами. Система $\hat{\mathbf{S}}$ —гомоморфный образ $\hat{\mathbf{S}}'$ для входных переменных, когда $\hat{\mathbf{S}}'$ —гомоморфный образ S для выходных переменных; они изоморфны относительно их параметров.

Формально направленная исходная система

$$\hat{\mathbf{S}}' = (\hat{\mathbf{O}},\ \hat{\mathbf{I}},\ \hat{\mathbf{I}}',\ \hat{O}',\ \hat{E}')$$

является моделирующей для другой направленной ИСХОДНОЙ системы

$$\hat{\mathbf{S}} = (\hat{\mathbf{O}},\ \hat{\mathbf{I}},\ \hat{\mathbf{I}},\ \hat{O},\ \hat{E}),$$

где

$$\hat{\mathbf{I}} = (\{(v_i,\ V_i)\,|\,i \in N_n\},\ u,\ \{(w_j,\ W_j)\,|\,j \in N_m\}),$$

$$\hat{\mathbf{I}}' = (\{(v_k',\ V_k)\,|\,k \in N_m\},\ u',\ \{(w_j',\ W_j')\,|\,j \in N_m\}),$$

тогда и только тогда, когда

1) для всех $k \in N_n$, таких, что $u'(k)=0$, переменная v'_k представляет переменную $v_{p0(i)}$, где

$$p_0 : Y_0 \to X_0 \quad 1$$

— функция входных переменных $\hat{\mathbf{S}}$ и $\hat{\mathbf{S}}'$, определенная на множествах

$$X_0 = \{i \mid i \in N_n,\ u(i) = 0\},$$

$$Y_0 = \{k \mid k \in N_{n'},\ u'(k) = 0\}$$

соответственно;

2) для всех $k \in N_n$ отображение $v_{p_0(k)} \to V_k$ гомоморфно относительно существенные свойств $v_{p_0(k)}$;

3) для всех $i \in N_n$, таких что $u(i) = 1$, переменная и, представляет переменную $v_{p_0(i)}$, где

$$p_1 : X_1 \to Y_1$$

— функция выходных переменных $\hat{\mathbf{S}}$ и $\hat{\mathbf{S}}'$ соответственно, определенная на множествах

$$X_1 = \{i \mid i \in N_n,\ u(i) = 1\},$$

$$Y_1 = \{k \mid k \in N_{n'}\ u'(k) = 1\}$$

4) для всех $i \in N_n$ отображение $V_{p_1(i)} \to V_i$ гомоморфно относительно существенных свойств множества V_i,
5) w_j соответствует $w_{q(j)}$ $(j \in N_m)$, где q обозначает перестановку N_m,
6) для каждого $j \in N_m$ взаимно однозначное соответствие $W_j \to W_{q(j)}$ изоморфно относительно существенных свойств W_j.

Система **S'** совместно с отображениями 1—6 является моделью системы **S**. Введем следующие названия для пар отображений:
входные отображения: 1, 2;
выходные отображения: 3, 4;
параметрические отображения: 5, 6.
Некоторые примеры этих отображений будут рассмотрены для моделей направленных систем более высокого эпистемологического уровня.

9.5. Модели систем данных

Суть моделирования заключается в установлении соответствия между парами систем.
Легко обобщить понятие моделей исходных систем ИИ на модели систем данных. Рассмотрим две нейтральные системы данных

$$\mathbf{D} = (\mathbf{S}, \mathbf{d}),\ \mathbf{D}' = (\mathbf{S}', \mathbf{d}'),$$

и пусть **D** рассматривается как подлинная система. Тогда **D'** можно считать системой, моделирующей систему **D**, если

существует преобразование (множество взаимно однозначных соответствий) между **S** и **S'**, при котором **S'** является моделирующей системой **S** (разд. 9.3);

отношение моделирования между **S** и **S'** сохраняет данные **d** в системе **D'**.

Таким образом, **модель системы данных — это такая модель исходной системы, которая сохраняет и ее данные.** Другими словами, **если D — подлинная система данных и D' — моделирующая система данных, то они изоморфны (при соответствующих взаимно однозначных соответствиях) в смысле подобных систем и относительно данных.**

Пример 3. Рассмотрим мелодию, партитура которой приведена на рис. 5.

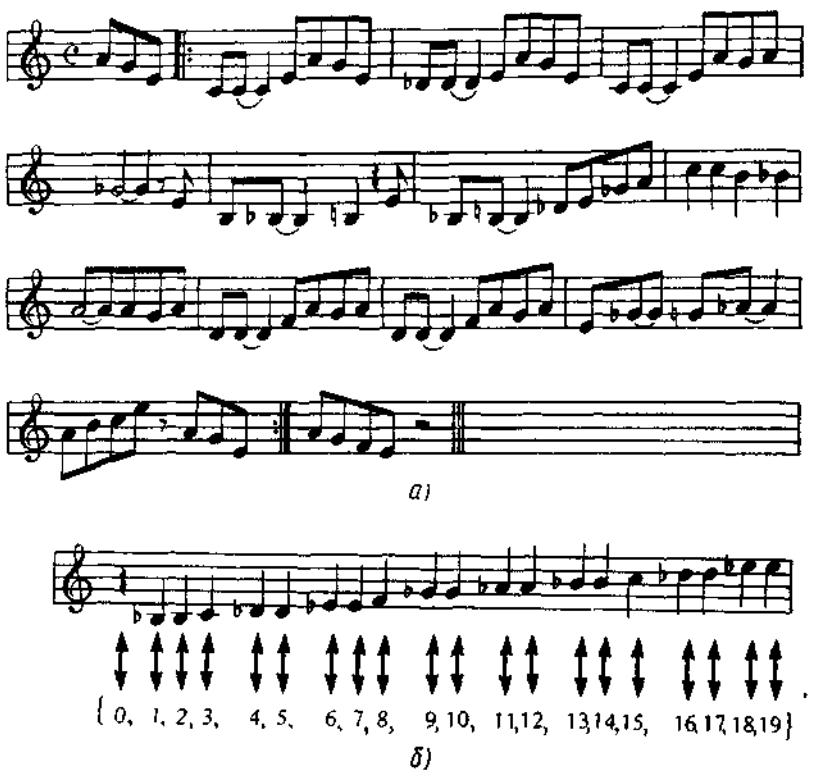


Рис. 5. Моделирование системы данных (пример 3)

Пусть мелодия описывается системой данных **D'**. Будем считать, что временное множество и переменная, соответствующая ритму, например переменная v_1', определены так же, как и в примере 6 р.2.7. Положим далее, что переменная v_2', соответствующая высоте тона, определена так, как показано на рис. 5,б. Обозначим через **D** систему данных, определенную в примере 6 р.2.7, но без переменной v_3 (тональности). Тогда легко убедиться, что **D'** — моделирующая система для **D** (и наоборот) при следующих взаимно однозначных соответствиях:
1) v_1' соответствует v_2, v_2' соответствует v_1;
2) время t' соответствует времени t;
3) $V'_1 \leftrightarrow V_2$ —тождественное отображение, а $V'_2 \leftrightarrow V_1$ определяется уравнениями

$$v_2' = \begin{cases} 19 - v_1 + 1 & \text{при } v_1 \neq 0, \\ 0 & \text{при } v_1 \neq 0; \end{cases}$$

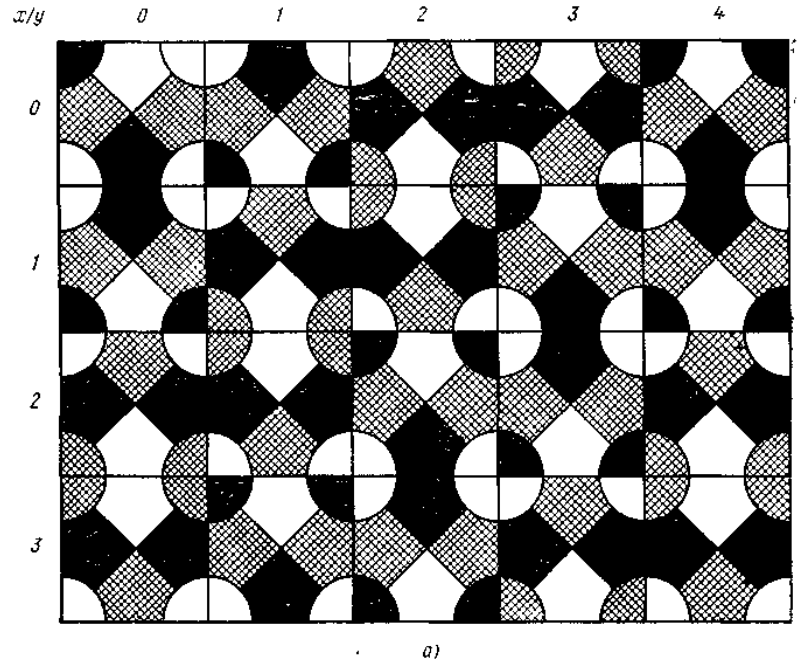
4) $T' \leftrightarrow T$— взаимно однозначное отображение двух временных множеств.

Принятая в музыке терминология описывает связь этих двух мелодий как комбинацию инверсии и транспозиции. Мелодия, описываемая системой данных **D'**, получена из мелодии, описываемой системой данных **D**, в результате инверсии и понижения на один тон; аналогично мелодия, описываемая системой данных **D**, получена из мелодии, описываемой системой данных **D'**, в результате инверсии и повышения на один тон.

Иногда достаточно определить модель системы данных менее строго, отказавшись от требования выполнения отношения моделирования между **S** и **S'**. В этих случаях отношение моделирования между системами данных **D** и **D'** представляется в терминах взаимно однозначных соответствий **V↔V'** и **W↔W′** между множествами обобщенных состояний и обобщенных состояний параметров двух систем данных, при которых данные сохраняются.

Необходимо подчеркнуть, что применимость моделей такого типа несколько ограничена до тех пор, пока в них не учитываются конкретные отношения между отдельными переменными и параметрами двух систем. Используемые для описания глобальных связей системы назовем *глобальными моделями данных.*

Пример 4. Рассмотрим мозаику, показанную на рис. 6,а.



v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v'_1	v'_2
Ч	Б	Ч	С	С	Б	Ч	Б	0	0
Б	Ч	Б	С	С	Ч	Б	Ч	0	1
Б	С	Б	Ч	Ч	С	Б	С	1	1
С	Б	С	Ч	Ч	Б	С	Б	1	0

г)

Рис. 6. Глобальное моделирование данных (пример 4)

Ее можно описать как систему данных со следующими компонентами: параметр, являющийся двумерным пространством x, y ($x \in N_{0,4}$,

$y \in N_{0,3}$), которое представляет собой прямоугольную область, разбитую в шахматном порядке на квадраты;

восемь переменных v_i ($i \in N_8$), каждая из которых описывает цвет определенной подобласти каждого квадрата, как это показано на рис. 4,*б*;

множества состояний всех переменных одинаковы и состоят из трех цветов: черного — Ч, белого — Б, серого — С;

данные **d**, формально описывающие мозаику: матрица 4×5, элементами которой являются восьмерки букв Ч, Б, С, представляющие комбинации цветов в определенных областях параметрического множества.

Рассмотрим другую систему данных (назовем ее **D'**), состоящую из меняющихся во времени t переменных v_1', v_2', каждая из которых может принимать два значения. Значения **d′** меняются периодически; один период показан на рис. 4,в. Читатель может легко убедиться, что данные **d** сохраняются в данных **d'** при преобразовании

$$t = x + 5y \quad (t \leqslant 19)$$

между множествами параметров, а также при преобразовании, показанном на рис. 4,г, для множества состояний двух систем.

9.6. Модели порождающих систем ИИ

Наибольшая опасность аналогии заключается в том, что подобие принимается за доказательство тождественности.

Если в моделях систем ИИ требуется, чтобы данные сохранялись, то в моделях порождающих систем ИИ требуется сохранение свойств ST-функций. Тем не менее нет оснований требовать сохранения масок систем ИИ с поведением в их моделях. В самом деле, те же результаты можно получить или с помощью параметрических поледовательностей состояний одной переменной или совместных состояний нескольких переменных. Это утверждение лучше всего пояснить на примере разных устройств последовательных и параллельных компьютеров (например, сумматоров).

Если допускаются преобразования масок подлинной и моделирующей порождающих систем ИИ, то это заменяет преобразования их исходных систем ИИ, причем предполагается, что исходные системы ИИ подлинной и моделирующей порождающих систем ИИ связаны только через параметры и параметрические множества. Существуют особые случаи, когда преобразования между масками являются на

самом деле преобразованиями между переменными исходных систем (например, систем с поведением без памяти).

Как было показано в гл. 4, каждая ST-система может быть единственным образом преобразована в изоморфную систему ИИ с поведением Таким образом, при обсуждении моделей порождающих систем ИИ можно без потери общности ограничиться системами ИИ с поведением. Это также позволит нам упростить запись, исключив индексы *B* и S, используемые обычно для различения систем с поведением и ST-систем.

Рассмотрим две нейтральные системы ИИ с поведением

$$\mathbf{F} = (S,\ M,\ f),$$
$$\mathbf{F}' = (S',\ M',\ f'),$$

и пусть **F** рассматривается как подлинная система ИИ. Тогда **F'** является моделирующей **F** системой ИИ, если

1) существуют взаимно однозначные соответствия между параметрами и параметрическими множествами в **S** и **S'**, при которых свойственные параметрам особенности сохраняются, как это требуется при подобии исходных систем ИИ (разд. 9.3);

2) существует взаимно однозначное соответствие $M \leftrightarrow M'$ между масками двух систем ИИ;

3) для любых выборочных переменных s_i из системы **F**, определяющих выборочные переменные s_k из системы **F'** ($i, k \in N_{|M|}$) в соответствии с п. 2, существует взаимно однозначное соответствие $S_i \leftrightarrow S_k$ между множествами их состояний, при котором существенные особенности S_i сохраняются в S_k;

4) при преобразованиях (взаимно однозначных соответствиях) 1—3 функция сохраняется в системе **F'**.

Назовем систему **F'** вместе с взаимно однозначными соответствиями 1—3, сохраняющими функцию **f**, моделью системы **F** со *строго определенным поведением.* Как показано ниже, в ряде случаев это понятие можно различными способами обобщить.

Пример 5. На рис. 7 приведен пример отношения моделирования между двумя системами с поведением (направленными, вероятностными).

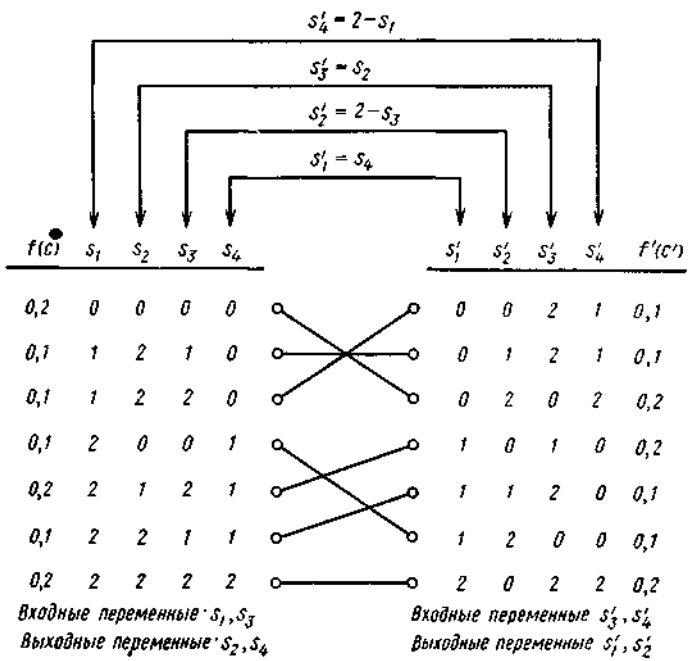


Рис.7. Отношение моделирования двух систем ИИ с поведением (пример 5)

Предполагается, что параметрические множества обеих систем полностью упорядочены, а множества состояний никаких особых свойств не имеют. Исходные системы этих систем не описаны, поскольку для описания отношения моделирования они не нужны. Взаимно однозначные соответствия между выборочными переменными двух систем с поведением показаны стрелками сверху таблиц функций поведения. Преобразования между множествами состояний соответствующих переменных описаны уравнениями, помечающими СЕЯЗИ. На рисунке показано также предполагаемое преобразование между обобщенными состояниями двух систем. Очевидно, что необходимое (но не достаточное) условие для отношения моделирования между двумя системами с поведением состоит в том, что распределение *f'(c')* (распределение вероятностей, возможностей или какое-либо иное) является перестановкой распределения *f(c).* Предполагается также, что обе системы являются системами без памяти и что их направленные функции поведения получены из их нейтральных аналогов (см. рис.5).

Пример 6. Рассмотрим цилиндрический и конический объемы, соединенные трубкой с пренебрежимо малым объемом (рис. 8).

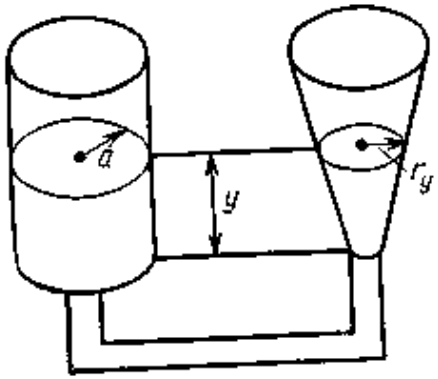


Рис. 8. Гидравлическая модель математической системы с поведением

На этой гидравлической системе можно определить детерминированную систему без памяти, состоящую из двух переменных и отношения между ними, определяемого объемами сосудов, где

x — объем воды, содержащейся в сосудах, измеряется в кубических сантиметрах с точностью до 1 см^3;

y — высота уровня воды, измеряется в сантиметрах с точностью до 0,1 см.

Очевидно, что значения переменных меняются во времени. Рассмотрим теперь математическую систему с поведением, состоящую из двух переменных *x′* и *y′,* функция поведения которой *y′=f′(x)* определяется действительными решениями уравнения

$$py'^3 + qy' = x',$$

где *p* и *q* — постоянные коэффициенты. При соответствующих объемах гидравлическая система может быть использована для решения кубического уравнения.

Получим выражения для отношения моделирования между этими двумя системами; обозначим радиус цилиндра (см) и отношение радиуса к высоте конуса соответственно *a* и *b.* Тогда объем воды в цилиндре равен

$$\pi a^2 y.$$

Объем воды в конусе равен

$$1/3\pi r^2_y y,$$

где r_y — радиус плоского сечения конуса, соответствующего высоте y (см). Поскольку $b = r_y/y$, то выражение для объема воды в конусе можно переписать в следующем виде:

$$1/3\pi b^2 y^3.$$

Полный объем воды в двух емкостях обозначим через x и тогда

$$\pi a^2 y + 1/3\pi b^2 y^3 = x.$$

Чтобы преобразовать это уравнение в уравнение математической системы, задаваемой переменными x' и y', значения величин a и b должны быть выбраны таким образом, чтобы удовлетворять соотношения

$$\pi a^2 = q;$$

$$1/3\pi b^2 = p.$$

Отсюда сразу следует, что гидравлическую систему с параметрами

$$a = (q/\pi)^{1/2},$$

$$b = (3p/\pi)^{1/2}$$

можно использовать для решения кубического уравнения, поскольку отношение между x и y описывается тем же уравнением, что и отношение между x' и y'.

Для направленных систем ИИ с поведением отношение моделирования не предполагает существования обязательных взаимно однозначных соответствий между множествами переменных и множествами состояний подлинной и моделирующей систем ИИ. Вместо этого при желании можно использовать входные и выходные отображения, как это обсуждалось в разд. 9.3 для направленных исходных систем ИИ.

Пример 7. В качестве простого примера отношения моделирования между двумя направленными системами с поведением можно привести принцип действия, использующийся в логарифмической линейке: произведения двух действительных чисел определяются сложением соответствующих отрезков на двух линейках. Соответствие входных и выходных данных этого хорошо известного примера приведено на рис.9.

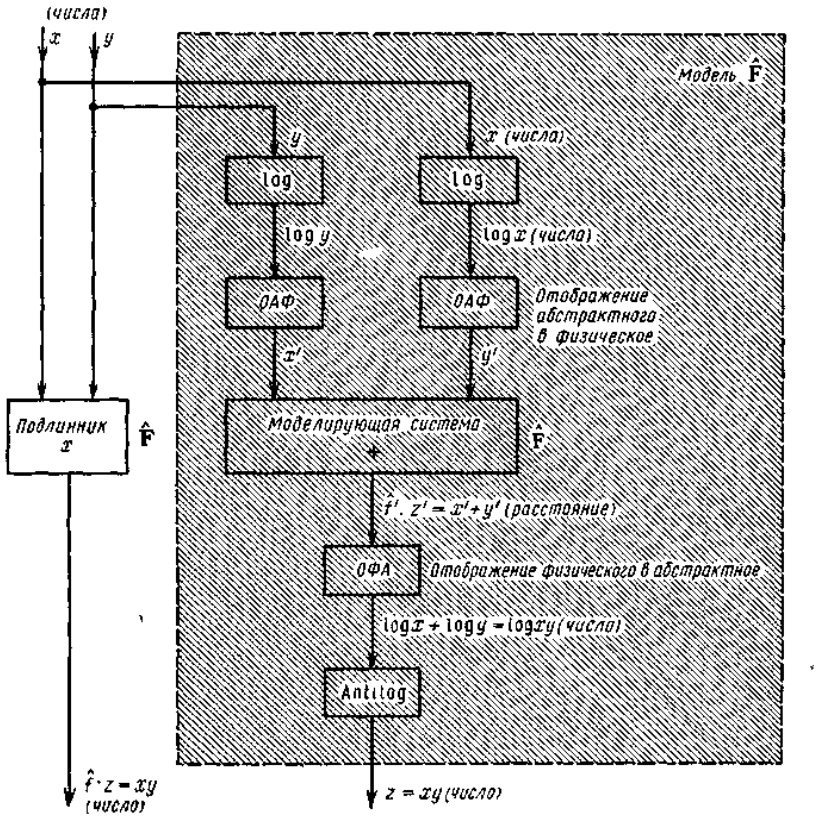


Рис.9. Правило сдвига на логарифмической линейке как модель поведения

Пример 8. Чтобы проиллюстрировать отношение моделирования между направленными системами с поведением, для которого входные соответствия не взаимно однозначны, рассмотрим подлинную систему, содержащую входные переменные x и y, выходную переменную z и функцию поведения (детерминированную без памяти)

$$z = 3\sin(x+y);$$

моделирующую систему, содержащую входные переменные x_1, x_2, y_1, y_2, выходную переменную z' и функцию поведения

$$z' = x_1x_2 + y_1y_2.$$

Положим, далее, что эти системы основываются на одном и том же параметре.
Хотя эти две функции поведения выглядят совершенно различно, вторая система может быть использована как моделирующая при соответствиях входных и выходных данных. Такие соответствия показаны на рис. 10.

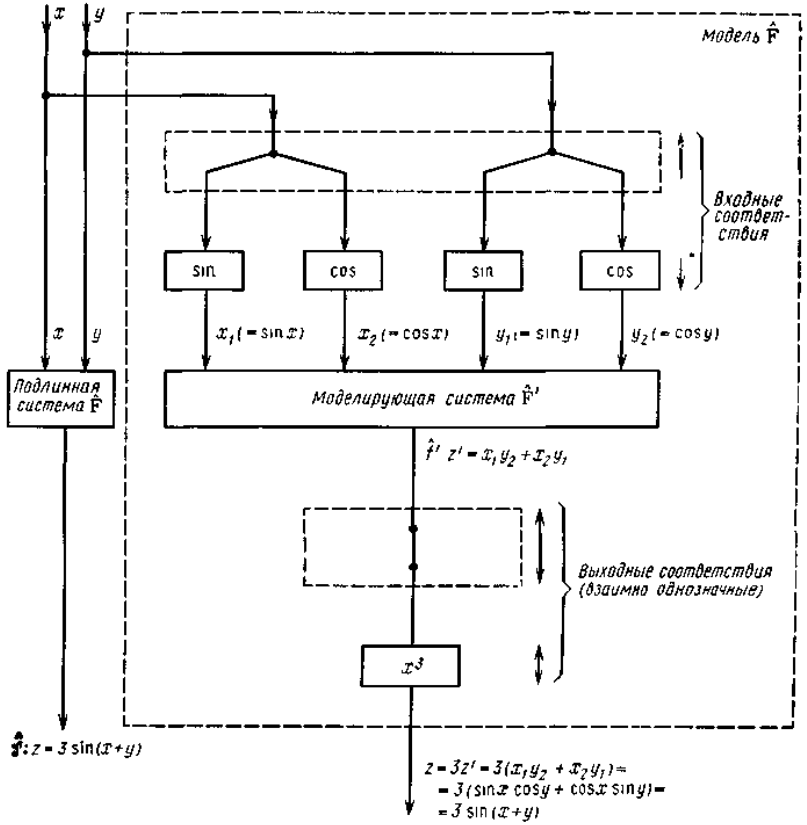


Рис.10. Отношение моделирования между двумя направленными системами с поведением, у которых входные данные не взаимно однозначны (пример 7)

Рассмотрите обратные отображения между множествами переменных и соответствующими множествами состояний.

Пример 9. Рассмотрим простую электрическую цепь с двумя полупроводниковыми диодами, схема которой показана на рис.11. Она содержит две входные переменные *x* и *y* и одну выходную *z,* которые определяют напряжения. Предположим, что на вход подаются два различных значения напряжений, одно из которых меньше, а другое больше постоянного значения *V*. Обозначим эти две величины (состояния переменных) *L* и *H* соответственно. Подавая на вход различные комбинации этих двух напряжений и измеряя выходную переменную, можно построить функцию поведения *f,* приведенную на рис.11, где буквы *L* и *H* для выходной переменной имеют тот же смысл.

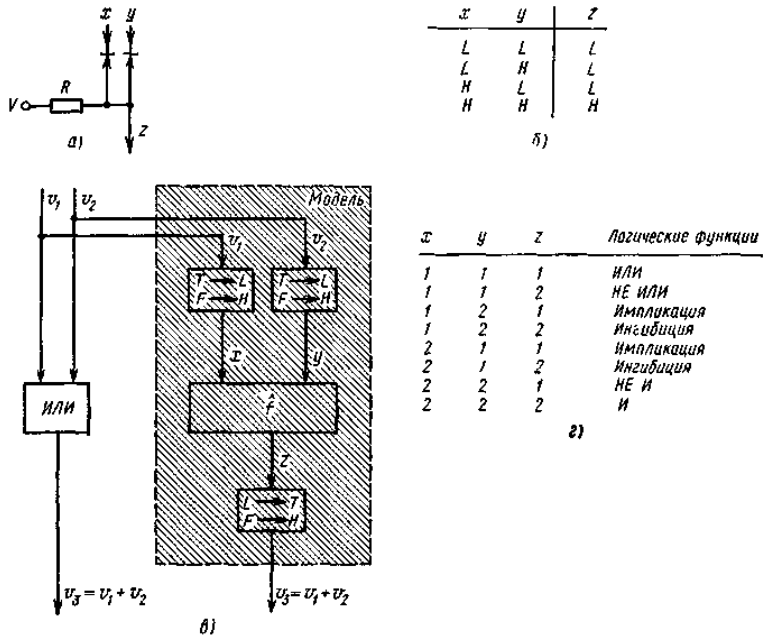


Рис. 11. Физическая система, которую можно использовать для моделирования некоторых абстрактных систем (пример 9)

Предположим теперь, что мы хотим использовать систему с поведением, определенную на данной электрической схеме, для моделирования логических функций. Это можно сделать с помощью соответствующих предположений относительно данных трех физических переменных и использования функции *f* для определения

истинности каждого предположения. Они здесь могут быть двух типов: либо «переменная *x* (или *y,* или *z)* находится в состоянии *L»,* либо «переменная *x* (или *y,* или *z)* находится в состоянии *H*». Эта физическая система может использоваться для моделирования различных логических функций в зависимости от комбинаций двух типов предположения для трех переменных. Например, если первый тип предположения относится ко всем трем переменным, то состояния *L* или *H* делают соответствующее предположение истинным или ложным. Обозначим истинность и ложность для данного предположения соответственно *T* и *F.* Подобное использование физической системы для моделирования логической функции ИЛИ (дизъюнкции) показано на рис.11,в. Все логические функции, которые могут быть смоделированы рассматриваемой физической системой сведены в таблицу на рис. 11,г, цифрами 1 и 2 обозначены предположения соответственно первого и второго типа.

Также можно ввести понятие *глобального моделирования поведения,* аналогичное понятию глобального моделирования данных. Оно определяется через взаимно однозначное соответствие между обобщенными состояниями выборочных переменных двух систем ИИ с поведением, при котором их функции поведения сохраняются. Таким образом, глобальное моделирование поведения есть изоморфное отношение между двумя множествами обобщенных состояний.

Пример 10. Отношение глобального моделирования данных, рассмотренное в примере 4 (мозаика), можно легко переформулировать как отношение глобального моделирования поведения. Рассмотрим маску из двух столбцов, и пусть v_1'' и v_2'' определяются уравнениями

$$v''_{1,t} = v'_{1,t+1} \quad \text{и} \quad v''_{2,t} = v'_{2,t+1}.$$

Тогда

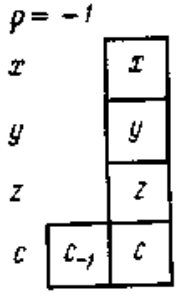
v_1'	v_2'	v_1''	v_2''
0	0	0	1
0	1	1	1
1	1	1	0
1	0	0	0

есть множество всех состояний выборочных переменных (в соответствии с данными, определенными на рис. 8 4,в). Эта система детерминирована: состояния порождаемых выборочных переменных v''_1, v''_2 однозначно определяются состоянием порождающих переменных v_1, v'_2. При взаимно однозначном соответствии, показанном на рис.6,г, она изоморфна системе, представляющей мозаику,

при условии, что последняя переформулирована аналогичным образом в виде системы с поведением.

Отношение моделирования между системами с поведением может быть обобщено с помощью замены преобразования между масками преобразованием между участками (ячейками) выделенных сегментов массивов данных, при котором порожденные данные при тех же начальных условиях сохраняются.

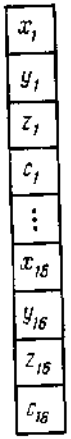
Пример 11. Рассмотрим отношение моделирования между последовательным (см. пример 9 раз.5) и параллельным двоичными сумматорами. Будем считать, что эти сумматоры предназначены для сложения пар двоичных чисел, содержащих по 16 цифр. Эти маски и функции поведения последова тельного и параллельного сумматоров показаны на рис.12,а, *б* соответственно.



$$z = x + y + c_{-1} \pmod 2$$

$$c = [x + y + c_{-1} - (x + y + c_{-1}) \pmod 2]/2$$

а)

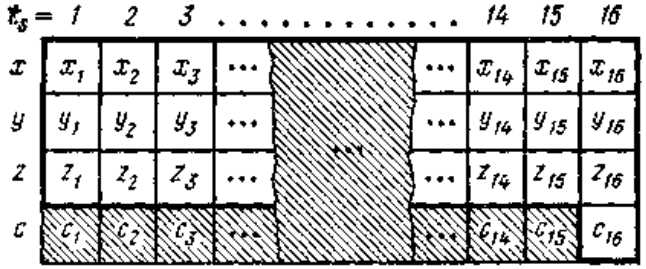


б)

$$z_i = x_i + y_i + c_{i-1} \pmod 2$$

$$c_i = [x_i + y_i + c_{i-1} - (x_i + y_i + c_{i-1}) \pmod 2]/2$$

$$(i \in N_{16},\ c_0 = 0)$$



в)

Рис. 12. Отношение моделирования между последовательным и параллельным сумматорами (пример 10)

Переменные *x , y* (или *x′, y′)* представляют цифры в складываемых числах; *z* (или *z ′)* —сумму цифр; *c* (или *c*') —то, что следует перенести в следующий разряд. В обоих случаях параметром является время.

У параллельного сумматора нет памяти. Он содержит 64 переменные и складывает два двоичных числа, в каждом из которых 16 цифр. Для

выполнения того же сложения последовательному сумматору потребуется 16 последовательных тактов. В последовательном сумматоре два числа представляются временной последовательностью из 16 состояний входных переменных *x, y,* а результат представляется временной последовательностью выходной переменной *z* и последним состоянием переменной *c,* как показано на рис. 12,в; последнее состояние *c* — это наибольшая значащая цифра суммы. Предположим, что сразу после завершения одной операции сложения начинается другая с соответствующими начальными условиями (переносимая часть вновь приравнивается к нулю).

Для описания отношения моделирования между двумя сумматорами, очевидно, недостаточно определить его только в терминах взаимно однозначных соответствий между их выборочными переменными, множествами состояний и параметрическими множествами.

Необходимо ввести комбинированное преобразование, включающее выборочные переменные и параметрические множества. Таким преобразованием в данном случае является взаимно однозначное соответствие между состояниями переменных x_i, y_i, z_i , c_i в моменты времени t_p и состояниями переменных *x, y, z, c* в моменты времени $t_s = i+16(t_p—1)$ соответственно ($i \in N_{16}$), где t_p и t_s — времена для параллельного и последовательного сумматоров. Видно, что если

$$x_{i,t_p} = x_{t_s} \text{ и } y_{i\,t_p} = x_{t_s}$$

для всех $i \in N_{16}$, то

$$z_{i\,t_p} = z_{t_s} \text{ и } c_{i\,t_p} = c_{t_s}$$

для всех $i \in N_{16}$ при комбинированномпреобразовании, т. е. сумматоры взаимозаменяемы. На самом деле достаточно, чтобы второе уравнение выполнялось только при i=16 (последний перенос).

В комбинированном преобразовании, проиллюстрированном на примере последовательного и параллельного сумматоров, предполагается сдвиг в параметрическом множестве одной из систем, отношение моделирования которого рассматривается. При этом вводятся новые переменные, в некотором смысле аналогичные выборочным переменным. Поскольку эти переменные вводятся для установления отношения моделирования с другой системой, их можно назвать *моделирующими переменными.*

Существует много различных способов, с помощью которых общая модель поведения может быть строго определена с помощью

комбинированных преобразований между моделирующими переменными, так же, как в примере 11.

9.7. Модели структурированных систем ИИ

Маркс Вартофски отметил:
«...несмотря на то, что отношение моделирования представляется бинарным, на самом деле это триада; нечто может считаться моделью чего-то другого тогда и только тогда, когда можно выделить некие аспекты, в смысле которых одна сущность похожа на другую, а соответствующие их свойства являются» общими.
Как отмечалось ранее, целью замены одной системы ИИ (подлинной) другой (моделирующей) является получение определенных преимуществ при решении какой-то задачи, связанной с подлинной системой ИИ. При использовании одной структурированной системы ИИ для моделирования другой структурированной системы ИИ достаточно, чтобы сохранялись связи между некоторыми множествами переменных (при определенных отображениях переменных, параметров, множеств состояний и параметрических множеств) и совершенно не важно, сохраняется ли в модели структура подлинной системы ИИ (ее элементы и связи между ними). Представляющие интерес ограничения получаются как для подлинной структурированной системы ИИ, так и для ее модели с помощью определенной композиционной процедуры, как объяснено в гл. 5. Модели этого вида в действительности являются моделями порождающих систем ИИ, хотя рассматриваемые порожденные системы ИИ получаются из соответствующих структурированных систем ИИ. Примерами отношения моделирования этого вида являются любые две структурированные системы, имеющие одну и ту же несмещенную реконструкцию (см. задачу реконструкции в разд. 5.7).
Настоящая модель структурированной системы ИИ должна сохранять не только связи между входящими переменными, но и структуру (т. е. элементы и связи) подлинника. При этом важно понимать, что элементы моделирующей системы ИИ не обязательно должны быть в точности такими, как в подлинной системе ИИ. Требуется только, чтобы они удовлетворяли отношению моделирования, соответствующему эпистемологическому уровню, на котором они определены Рассмотрим две структурированные системы ИИ

$$\mathbf{SX} = \{({}^{i}V',\ {}^{i}\mathbf{X}) \mid i \in N_q\},$$
$$\mathbf{SX}' = \{({}^{j}V'',\ {}^{j}\mathbf{X}') \mid j \in N_q\},$$

где ${}^{i}\mathbf{X}$, ${}^{j}\mathbf{X}'$ — это исходные системы ИИ, системы данных или порождающие системы ИИ, нейтральные или направленные, а **X, X'** — обозначения системных типов. Пусть **SX** рассматривается как подлинная система ИИ. Тогда **SX'** определяется как система ИИ, моделирующая систему ИИ **SX**, если существует взаимно однозначное соответствие между элементами **SX** и **SX'**, например,

$$z: N_q \leftrightarrow N_q,$$

при котором
1) если $f=z(i)$, то ${}^{j}\mathbf{X}'$ вместе с соответствующими отображениями является моделью элемента ${}^{j}\mathbf{X}'$;
2) все связи из системы ИИ **SX** сохраняются в системе ИИ **SX'** при включенных в отношение моделирования связях между элементами ${}^{i}\mathbf{X}$ и ${}^{j}\mathbf{X}'$ $(i, j \in N_q)$.

Пример 12. Рассмотрим структурированную систему $\widehat{\mathbf{SF}}$, описываемую блок-схемой рис.13,а.

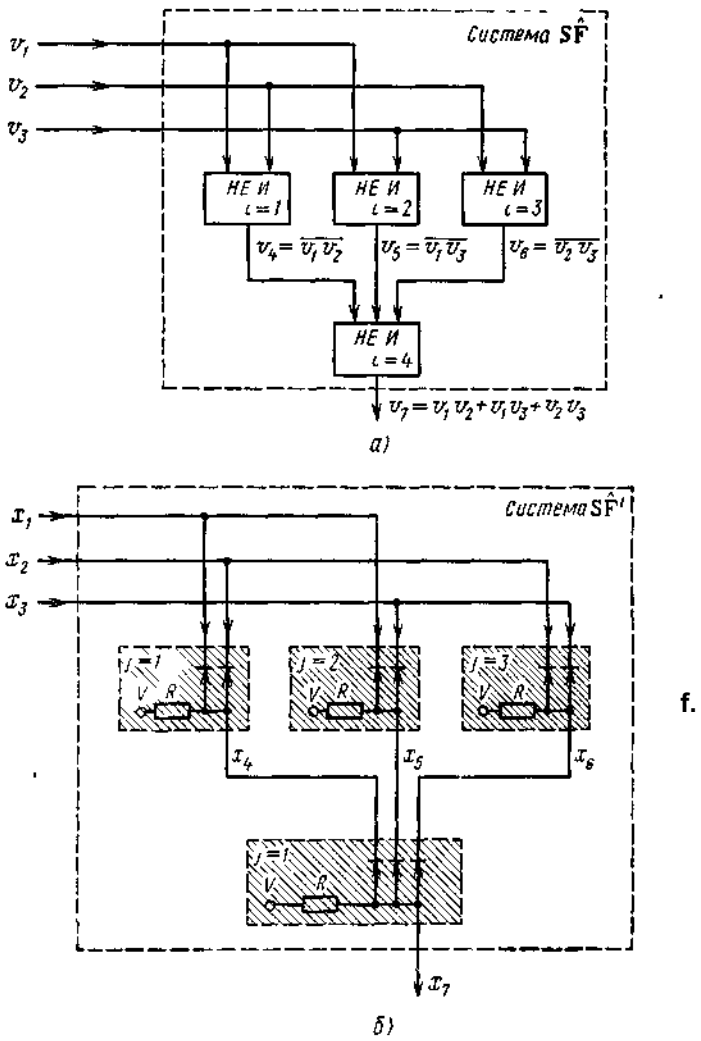


Рис. 13. Структурированная система из примера 12

Переменные $v_1, v_2, \ldots, v_7$ представляют утверждения. Каждая из них может иметь два значения: истинное *(T)* и ложное *(F)*. Система реализует функцию большинства (см. пример 5.8): ее выходная переменная принимает значение *T* тогда и только тогда, когда две из трех входных переменных v_1, v_2, v_3 принимают значение *T*.

Рассмотрим теперь структурированную систему $\hat{SF}'$, описываемую блок-схемой на рис. 13,б. Ее элементами являются электрические цепи с полупроводниковыми диодами, поведение которых описано в примере 9. Эту систему можно использовать для моделирования структурированной системы при однозначном отображении

$$z:j=i$$

в предположении, что введены соответствующие отображения между переменными и множествами состояний двух систем, при которых элементы системы $\hat{SF}'$ становятся моделями соответствующих элементов системы $\hat{SF}$, полная модель системы $\hat{SF}$, основанная на системе **SF'**,показана на рис.14; $\hat{f}'$обозначает функцию поведения, определенную на рис.11,б, а *f* — расширение той же функции для трех входных переменных (т. е. выходная переменная находится в состоянии *H* только при условии, что все входные переменные находятся в состоянии *H)*.

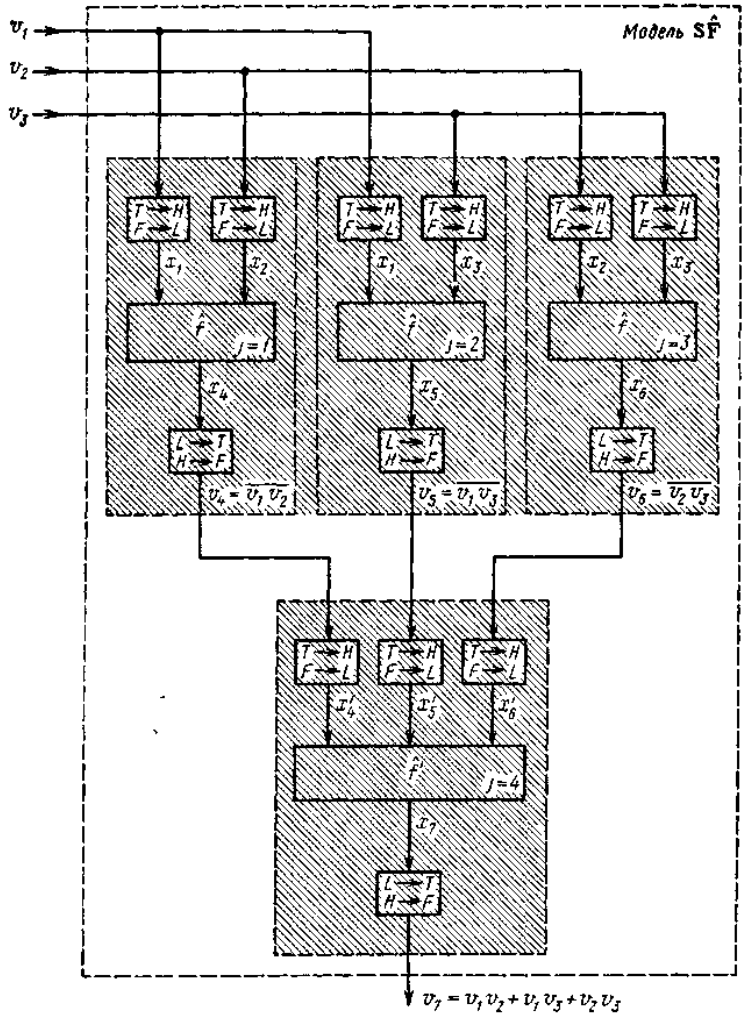


Рис. 14. Модель структурированной системы, показанной на рис.9,а (пример 12)

Легко видеть, что

1) каждый элемент системы $\widehat{\mathbf{SF}}$ представляется моделью в полной модели структурированной системы $\widehat{\mathbf{SF}}$;

2) все соединения между элементами $\widehat{\mathbf{SF}}$ сохраняются в полной модели $\widehat{\mathbf{SF}}$ как соединения между их моделями.

9.8. Модели метасистем ИИ

Хотя для целей моделирования в некоторых случаях удобно использовать метасистемы ИИ, обычно достаточно, чтобы моделирующая метасистема ИИ сохраняла только ограничения между некоторыми переменными (при соответствующих отображениях переменных, параметров, множества состояний и параметрических множеств), представленными подлинной метасистемой ИИ. Как правило, не важно, сохраняются или нет в модели элементы и процедура замены подлинной метасистемы ИИ. Если они не сохраняются, то в действительности модель не является моделью метасистемы ИИ, даже в том случае, когда используются метасистемы ИИ.

Настоящая модель метасистемы ИИ должна сохранять не только ограничения на рассматриваемые переменные, но также элементы и процедуру замены подлинника.

Рассмотрим две метасистемы ИИ

$$\mathbf{MX}=(\mathbf{W}, \mathscr{X}, r),$$

$$\mathbf{MX}'=(\mathbf{W}', \mathscr{X}', r'),$$

где $\mathscr{X}$, $\mathscr{X}'$ — множества их элементов, и пусть **MX** рассматривается как подлинная метасистема ИИ. Тогда **MX'** считается моделирующей **MX** метасистемой ИИ, если существует взаимно однозначное соответствие между множествами $\mathscr{X}$ и $\mathscr{X}'$, например

$$z: \mathscr{X}' \leftrightarrow \mathscr{X},$$

при котором

1) если **x**=z(**x**), то **x'** вместе с соответствующими отображениями является моделью **x**;

2) процедура замены *r* сохраняется в метасистеме **MX'** при взаимно однозначном соответствии между **W** и **W'**, и отображения, входящие в отношения моделировния между элементами множеств $\mathscr{X}$ и $\mathscr{X}'$, связаны друг с другом с помощью функции z.

Предоставим читателю возможность в качестве упражнения обобщить это определение на метасистемы более высокого порядка и построить несколько примеров моделей метасистем.

10. Архитектура ФРИЗ

10.1. Эпистемологическая иерархия систем ИИ

Эпистемологические типы систем ИИ, рассматриваемые в схеме ФРИЗ, включают исходные системы ИИ и их компоненты (объектные системы ИИ, конкретные и общие представляющие системы ИИ), системы данных, порождающие системы ИИ (системы с поведением или ST-системы), структурированные системы ИИ различных типов и уровней. Кроме того, каждый из этих системных типов может быть как направленным, так и нейтральным.

Эпистемологические типы систем ИИ в ФРИЗ можно частично упорядочить. Рассмотрим два типа систем ИИ, например **x** и **x'**, и будем считать, что системный тип **x** *эпистемологически ниже* системного типа **x'** тогда и только тогда, когда:

1) для любой заданной системы ИИ типа **x'** существует единственная процедура, основанная исключительно на этой системе и использующая всю имеющуюся в ней информацию, с помощью которой при подходящих начальных и других соответствующих условиях получается хотя бы одна система ИИ типа **x**;

2) не существует процедуры, с помощью которой только из заданной системы ИИ типа **x** можно получить единственную систему ИИ типа **x'**, т. е. всегда есть некоторая неопределенность и, следовательно, некоторая степень произвола при определении системы ИИ типа **x'** из системы ИИ типа x.

Пусть для двух данных эпистемологических системных ИИ типов **x** и **x'**

$$\mathbf{x} \preceq \mathbf{x}'$$

означает, что тип **x** эпистемологически ниже типа **x'**. Тогда пара

$$\mathcal{H}_l = (\mathcal{E}_l \prec_l,$$

где $\mathcal{E}_l$ обозначает множество всех эпистемологических типов систем ФРИЗ (как нейтральных, так и направленных), таких, что общее число уровней структурированных систем ИИ и метасистем ИИ в любом системном типе не превышает l, определяет *эпистемологи-ческую иерархию в системе ФРИЗ* ($l \in \mathbf{N}$).

Для каждого $l \in \mathbf{N}$, $\mathcal{H}_l$ —полурешетка. Только при рассмотрении гомогенных структурированных систем ИИ или метасистем ИИ $\mathcal{H}_l$ состоит из

$$|\mathcal{E}_l| = 3\ (2^{l+1} - 1)$$

элементов. В качестве примера на рис. 1 показан случай $l = 2$

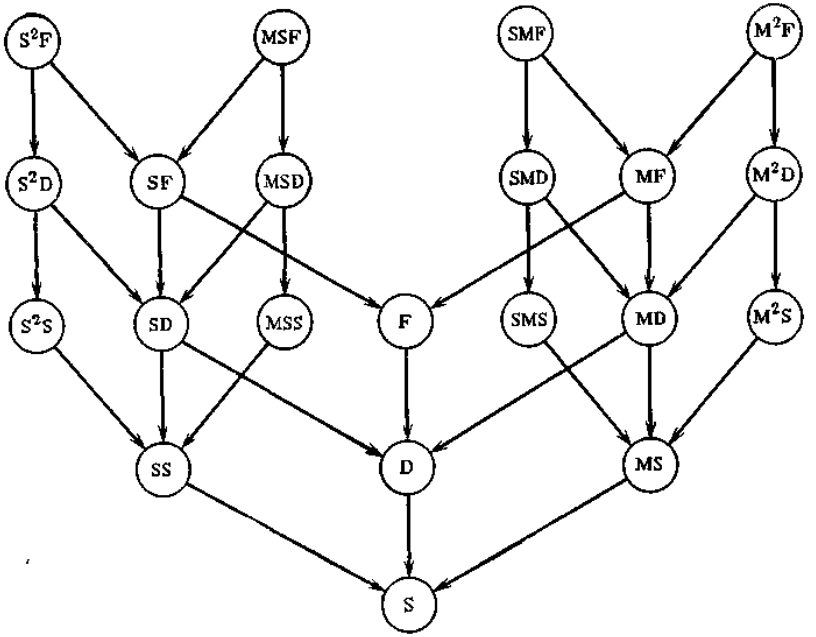


Рис. 1. Полурешетка эпистемологических типов систем в ФРИЗ, где число структурированных систем комбинируемых типов и метасистем для любой системы не превышает двух $\mathcal{H}_2 = (\mathcal{E}_2, \preceq)$

В табл. 1 определена полурешетка $\mathcal{H}_l$ при произвольных конечных l (с учетом ограничения условия гомогенности структурированных систем и метасистем).

Таблица 1.

Эпистемологическая иерархия, полурешетка типов систем в ФРИЗ: $\mathcal{H}_l = (\mathcal{E}_l, \preceq)$

Тип системы	Непосредственный преемник в полурешетке
S	Нет
D	**S**
F	**D**
$(\mathbf{S}^{s_i}\mathbf{M}^{m_i})^j$ **S**	$(\mathbf{S}^{s_i-1}\mathbf{M}^{m_i})^j$**S** для определенного i
	$(\mathbf{S}^{s_i}\mathbf{M}^{m_i-1})^j$ **S** для определенного i
$(\mathbf{S}^{s_i}\mathbf{M}^{m_i})^j$ **D**	$(\mathbf{S}^{s_i-1}\mathbf{M}^{m_i})^j$ **D** для определенного i
	$(\mathbf{S}^{s_i}\mathbf{M}^{m_i-1})^j$ **D** для определенного i
	$(\mathbf{S}^{s_i}\mathbf{M}^{m_i})^j$ **S**
$(\mathbf{S}^{s_i}\mathbf{M}^{m_i})^j$ **F**	$(\mathbf{S}^{s_i-1}\mathbf{M}^{m_i})^j$ **F** для определенного i
	$(\mathbf{S}^{s_i}\mathbf{M}^{m_i-1})^j$ **F** для определенного i
	$(\mathbf{S}^{s_i}\mathbf{M}^{m_i})^j$ **D**
$(\mathbf{M}^{m_i}\mathbf{S}^{s_i})^j$ **S**	$(\mathbf{M}^{m_i-1}(\mathbf{S}^{s_i})^j$ **S** для определенного i
	$(\mathbf{M}^{m_i}\mathbf{S}^{s_i-1})^j$ **S** для определенного i
$\mathbf{M}^{m_i}\mathbf{S}^{s_i})^j$ **D**	$(\mathbf{M}^{m_i-1}\mathbf{S}^{s_i})^j$ **D** для определенного i
	$(\mathbf{M}^{m_i}\mathbf{S}^{s_i-1})^j$ **D** для определенного i
	$(\mathbf{M}^{m_i}\mathbf{S}^{s_i})^j$ **S**
$(\mathbf{M}^{m_i}\mathbf{S}^{s_i})^j$ **F**	$(\mathbf{M}^{m_i-1}\mathbf{S}^{s_i})^j$ **F** для определенного i
	$(\mathbf{M}^{m_i}\mathbf{S}^{s_i-1})^j$ **F** для определенного i
	$(\mathbf{M}m_i\mathbf{S}^{s_i})^j$ **F**

Обозначения

$(\mathbf{S}^{s_i}\mathbf{M}^{m_i})^j\mathbf{X}$,

$(\mathbf{M}^{m_i}\mathbf{S}^{s_i})^j\mathbf{X}$

используются соответственно для последовательностей

$\mathbf{S}^{s_1}\mathbf{M}^{m_1}\mathbf{S}^{s_2}\mathbf{M}^{m_2}\ \ldots\ \mathbf{S}^{s_j}\mathbf{M}^{m_j}\mathbf{X}$,

$\mathbf{M}^{m_1}\mathbf{S}^{s_1}\mathbf{M}^{m_2}\mathbf{S}^{s_2}\ \ldots\ \mathbf{M}^{m_j}\mathbf{S}^{s_j}\mathbf{X}$,

где **X** обозначает системный тип, а j, s_i, m_i $(i \in N_q)$—некоторые натуральные числа; допускается случай, когда $m_j = 0$ в первой и $s_j = 0$ во второй последовательностях, и $\mathbf{S}^0\mathbf{X}$, $\mathbf{M}^0\mathbf{X}$ интерпретируется как **X**. Необходимо, чтобы

$$\sum_{i \in N_l} (s_i + m_i) \leqslant l$$

Каждый класс систем ИИ, характеризующийся неким эпистемологическим типом, будет в дальнейшем классифицироваться с помощью соответствующих ему методологических отличий. Как было показано выше (разд. 1.7), суть методологических отличий состоит в том, чтобы при решении интеллектуальных задач отличать системы ИИ, требующие применения различных методов.

Существенной проблемой архитектуры ФРИЗ является выбор соответствующих методологических отличий для различных эпистемологических типов систем ИИ. Выбор можно начинать с наиболее общих методологических отличий, соответствующих различным системным свойствам, и продолжать по мере конкретизации. Желательно упорядочить результирующие методологические отличия в зависимости от степени конкретизации. Поскольку всегда комбинируется несколько категорий, то результирующее упорядочение является только частичным.

На архитектурном уровне желательно выделить основные категории методологических различий и свести их к небольшому набору важных и наиболее общих отличий для каждой категории. Но в то же время в архитектуре должна учитываться возможность расширения этих множеств в различных реализациях ФРИЗ.

В разных местах этой работы в связи с отдельными эпистемологическими типами систем вводились методологические отличия, которые считались достаточно значимыми, чтобы выделить их в архитектуре ФРИЗ. Список категорий этих существенных отличий приведен в табл. 2.

Таблица 2.

Сводка важнейших методологических отличий, рассматриваемых на одном уровне архитектуры ФРИЗ:

Характеристика системы	Методологические отличия	Эпистемологические типы систем	Варианты возможного развития	Ссылки
Переменные	Нейтральные Направленные	Все	Смешанные	Разд. 2.5
Каналы наблюдения	Четкие Нечеткие	Все	Смешанные	Уравнения (2.2), (2.3)
Множества состояний и параметрические множества	Без свойств	Все	Интервальные шкалы	Уравнения (2.8) — (2.10) Разд. 2.3, 2.4 (рис. 2.2, табл. 2.1)
	Содержательные комбинации свойств упорядочения, расстояния и непрерывности		Рациональные шкалы Другие шкалы	
Данные	Полностью определенные	D или XD		Разд. 2.6
	Не полностью определенные			
	С несущественными элементами			
	Четкие	D или XD	Смешанные	Уравнение (2.21) Уравнение (2.27)
	Нечеткие			
	Периодические	D или XD	Смешанные	Разд. 2.6
	Апериодические			
Маска	Без памяти	F или XF	Компактная	Разд. 3.2
	С памятью			
Ограничивающая функция	Функция поведения ST-функция	F или XF	—	Разд. 3.8
Функция поведения или ST-функция	Детерминированная	F или XF	Основаны на других подмножествах нечетких мер	Уравнения (3.14), (3.16) Уравнения (3.18), (3.19)
	Вероятностная Возможностная			Уравнения (3.20), (3.21)
Элементы структурированных систем	Согласованные Несогласованные	SXF	—	Разд. 4.3, 4.4

X обозначает последовательность операторов **S** и **M**, **F** — порождающие системы любых типов.

Каждая категория характеризуется системным свойством, для которой она определена, списком индивидуальных методологических отличий, эпистемологическими типами систем, к которым они приложимы, примерами возможных расширений и соответствующими ссылками.

10.2. Условия задачи

Как только задача четко сформулирована, решить ее обычно достаточно просто.

Условия задачи определяют задачи на множестве рассматриваемых систем ИИ. В каждом условии рассматривается либо одна система ИИ, либо пара систем ИИ. Аналогично тому, как системы классифицируются в соответствии с их типами, так и условия классифицируются по типам. При правильном определении любой тип условий должен быть таким, чтобы изменения отдельных условий, принадлежащих этому типу, не приводили к изменению методологии. Хотя конкретные условия задачи так же, как и их типы, имеют смысл только для отдельных типов систем, можно в общем случае **выделить четыре широкие категории условий задачи.** Каждое условие может быть либо требованием ответа на ***вопрос,*** либо требованием удовлетворить ***запрос,*** либо требованием достижения ***цели,*** либо требованием удовлетворить какому-либо ***ограничению.*** В одной задаче может быть скомбинировано несколько условий, т. е. одна или больше целей могут быть скомбинированы с одним или несколькими ограничениями.

Как уже было отмечено, в ряде случаев ФРИЗ должен предоставить пользователю возможность сделать свой собственный выбор конкретных условй в пределах каждого выделенного типа условий. Как только тип условия определен, что может быть сделано только в контексте заранее определенных одного или двух типов систем ИИ, можно предложить пользователю *определить его собственный выбор* конкретного варианта этого типа. Если он не воспользуется этой возможностью, то ФРИЗ должен предложить ему *меню подходящих вариантов,* один из которых объявлен как *вариант по умолчанию.* Если пользователь все же не выберет ни один из них, то в ФРИЗ должен использоваться именно этот вариант.

На уровне архитектуры для каждого типа системы ИИ или пары типов желательно зафиксировать только ограниченное множество типов существенных условий. Предполагается, что это начальное множество будет постепенно в процессе эволюции ФРИЗ расширяться (разд. 10.6). Типы условий задач нельзя определить отдельно от типов систем ИИ, к которым они применяются. Каждый из них вместе с рассматриваемым типом системы ИИ образует тип задачи. Таким образом, типы условий могут быть определены только как части типов задач. Наиболее важны типы условий, введенные в различном контексте в этой работе и перечисленные в следующем разделе.

10.3. Интеллектуальные задачи

Системы ИИ и условия, относящиеся к этим системам, образуют интеллектуальные задачи. Поэтому их естественно классифицировать в соответствии с рассмотренной классификацией систем ИИ и типов условий.

В ряде задач может рассматриваться одна-единственная система ИИ. Назовем эти задачи, имеющие вид вопроса или условия относительно каких-либо свойств данной системы ИИ, *элементарными задачами.* Например, для данной конкретной ST системы можно сформулировать следующий вопрос: всегда ли можно получать из одного состояния другое состояние? — или в качестве альтернативы можно потребовать для каждой пары состояний список кратчайших последовательностей преобразований одного состояния в другое.

Для каждого из рассматриваемых типов систем ИИ необходимо обеспечить наиболее понятную работу ФРИЗ с элементарными за-, дачами. Одно из наиболее значимых подмножеств элементарных задач состоит из *тестов на противоречивость (корректность)* для определенных пользователем систем. Сюда, например, относятся проверка необходимых свойств рассматриваемых вероятностных или возможностных распределений, всевозможных условий совместимости и т. п. Эти тесты должны выполняться ФРИЗ автоматически в процессе формулирования пользователем задачи для каждой, определяемой пользователем системы. Если какие-то тесты не выполняются, система должна быть отвергнута как некорректная, а пользователю сообщены причины.

Все задачи, не входящие в категорию элементарных, содержат две или более систем ИИ. Задачи, содержащие две системы ИИ, в ФРИЗ явно определены и описаны методологически; назовем их *базовыми задачами.* Все остальные задачи, содержащие более двух систем ИИ, формулируются и рассматриваются в терминах соответствующих последовательностей базовых и элементарных задач.

Базовые интеллектуальные задачи классифицируются по типам задач. Каждый из них состоит из упорядоченной пары типов систем ИИ и множества типов условий, применимых к этим типам систем. Поскольку при решении задачи всегда приходят от некоторых заданных объектов к неким неизвестным объектам, то порядок типов систем ИИ неразрывно связан с этим направлением. Первый из двух типов систем ИИ рассматривается как общее описание *начальнойси стемы,* т. е. системы ИИ, заданной в конкретной задаче данного типа. Второй тип

описывает класс систем решения; назовем их *терминальными системами.* В соответствии с функцией терминальных систем можно выделить два вида интеллектуальных задач.

Для задач первого вида приведем следующую каноническую формулировку. Для данной конкретной начальной системы ИИ типа z определяются такие терминальные системы типа z', для которых выполняются данные требования (связанные с системными типами z и z'). Таким образом, решение задач первого вида — это множество конкретных систем ИИ типа z'.

Для задач второго вида используем другуюя каноническуюя формулировку. Для данной конкретной начальной системы типа ИИ z и конкретной терминальной системы типа z' определяется некоторое свойство терминальной системы, специфицированное заданными требованиями относительно начальной системы ИИ. Таким образом, решением задачи второго вида является то, что связывает две данные системы ИИ.

Примером задач первого вида является задача получения из заданной системы данных всех порождающих систем, удовлетворяющих требованиям максимальной согласованности, минимальной сложности и порождающей неопределенности, маски которых являются подмасками определенной наибольшей допустимой маски. Этот тип задач был сформулирован в разд. 4.4, обсуждался в разд. 4.6 и показан на рис. 2.

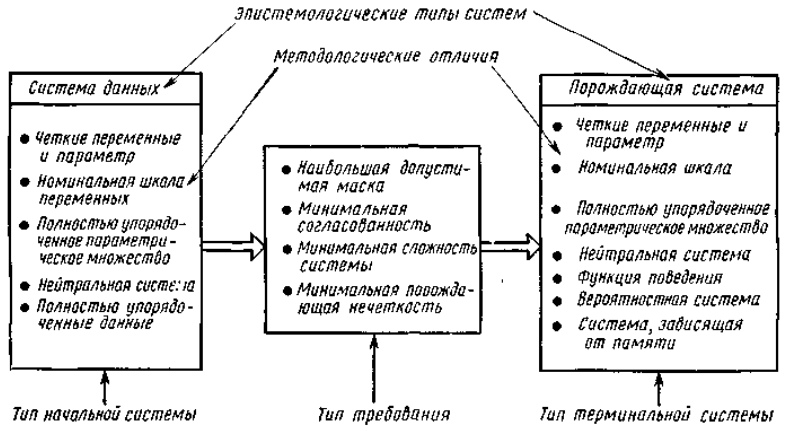


Рис. 2. Пример базового типа задач первого вида (см. разд. 4.4 и 4.6)

Примером задач второго вида является задача определения изменения характеристики систем ИИ с поведением, обусловленного некоторыми конкретными переменными. В этом типе задач (рис. 3), рассмотренном в разд. 8.3, некоторое свойство начальной системы ИИ представляет собой цель, а характеристики терминальной системы отдельно и вместе с этими переменными сравниваются с точки зрения заданной характеристической функции.

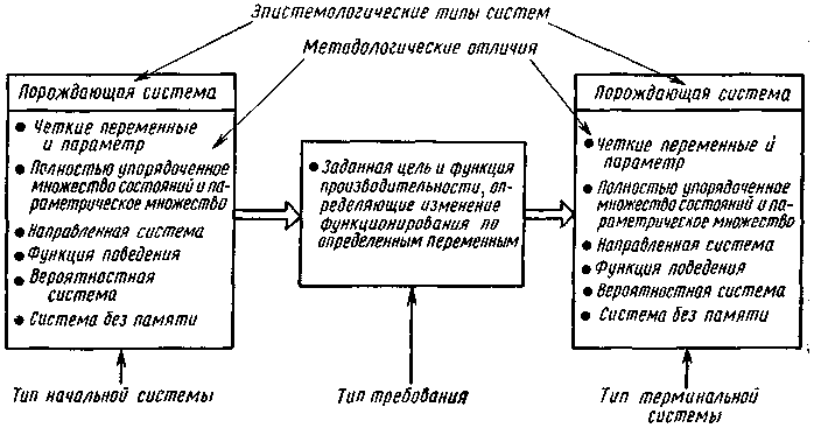


Рис. 3. Пример базового типа задач первого вида (см. разд. 4.4 и 4.6)

На рис. 4 показаны категории задач, составляющие ядро ФРИЗ.

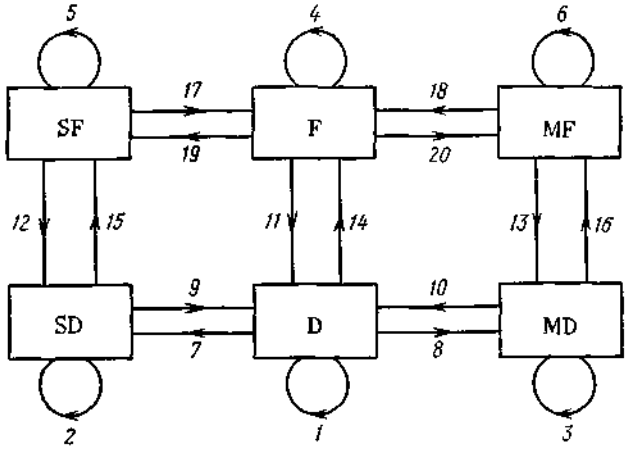


Рис. 4. Ядро ФРИЗ: важнейшие категории задач

Каждая **категория задач** характеризуется эпистемологическими типами входящих в нее систем ИИ; на рисунке это показано стрелками, помеченными числами. Категории задач образуют **кластеры типов задач**. Каждая категория содержит типы задач, имеющих особые методологические отличия, но содержащие одни и те же пары эпистемологических типов систем ИИ.

Некоторые основные типы задач, показанные на рис. 4, были в предыдущих главах определены либо явно, либо как части более крупных задач. Ниже приводится классификация задач по категориям. Очевидно, чго категории 1—6 содержат все простейшие типы задач, связанные с отдельными эпистемологическими типами систем ИИ, но также и многие базовые типы задач. В частности, они содержат следующие классы базовых типов:

различные типы задач, связанные с упрощением систем так же, как из разд. 4.9;

различные типы задач, связанные с отношением моделирования между системами, как это рассматривалось для некоторых эпистемологических уровней в гл. 9;

различные типы задач, в которых системы определенного эпистемологического уровня сравниваются по некоторому критерию, например по их сложности (гл. 7), характеристике относительно различных целей (гл. 8), потере информации (разд. 5.6—5.8) и т. п.

Категории 7 и 8 содержат типы задач, в которых в соответствии с определенными критериями данные разбиты относительно рассматриваемых переменных или параметрического множества.

Категории 9 и 10 состоят из типов задач, в которых полная система данных определяется соответственно из элементов структурированных систем ИИ или метасистем ИИ. Они, например, включают типы задач, в которых требуется разрешать несогласованности в данных, как это описано в разд. 5.3 для структурированных систем данных.

Категории 11—13 содержат различные типы задач, в которых данные порождаются эпистемологически более высокими типами систем ИИ при определенных начальных и других условиях. Во всех задачах этих категорий ФРИЗ используется в качестве средства при компьютерном моделировании.

Категория 14 состоит из типов задач, в которых порождающие системы ИИ получены из заданных систем данных при определенных типах требований. Как говорилось в разд. 4.4, 4.6 и 4.10, обычно требуется, чтобы результирующая порождающая система ИИ основывалась исключительно на масках, не превышающих определенную

наибольшую допустимую маску, а порожденные ими данные полностью соответствовали имеющимся данным и чтобы их порожда- . емые неопределенности и сложности были минимизированы. Эти задачи могут быть дополнены другими требованиями, такими, как дальнейшие ограничения на результирующие порождающие системы ИИ или критерии оптимизации, представленные как конкретные отношения предпочтения, определенные на соответствующем множестве порождающих систем ИИ. Сюда также относятся задачи, связанные с введением внутренних переменных (см. разд. 4.10).

Категории 15 и 16 состоят, в основном, из тех же типов задач, что и категория 14, но каждая отдельная задача должна быть повторена для всех элементов данной структурированной системы данных или метасистемы данных.

Категория 17 содержит типы задач, связанных с определением семейств реконструкций, однозначных реконструкций, основанных на различных типах требований, и других вопросах, обсуждавшихся в разд. 5.6—5.8.

Категорию 18 составляют типы задач, связанных с моделированием порождающих систем ИИ метасистемами ИИ различных методологических типов.

Категория 19 включает различные типы декомпозиционных задач, решаемых при проектировании систем (разд. 5.5), а также некоторые задачи реконструкции (разд. 5.7, 5.8).

Категория 20 содержит все типы задач, связанные с идентификацией параметрических требований, обусловленных изменениями связей между переменными данной порождающей системы ИИ. Примерами таких задач являются задачи, рассмотренные в разд. 6.6.

Для большинства задач, имеющих наибольшее практическое значение, требуется решение последовательности базовых и простейших типов задач. Например, задачи реконструкции, введенные в разд. 5.7, представляют собой последовательность задач из категорий 4, 5, 17, 19; для некоторых типов задач проектирования необходимо решение задач из категорий 11, 14, 17, 19; ряд задач компьютерного моделирования включает задачи из категорий 11 и 17 или 11 и 18 и т. д.

10.4. Формализация концептуальной схемы ФРИЗ

Следующие определения связывают прилагательные «идентифицируемый» и «допустимый» с понятиями «система ИИ», «условие», «задача». Назовем систему ИИ, условие или задачу *идентифицируемыми,* если их можно сформулировать на языке ФРИЗ; назовем

их *допустимыми,* если они идентифицируемы и их можно рассматривать в контексте конкретной реализации ФРИЗ. Кроме того, назовем задачу *решаемой,* если она допустима и может быть решена с помощью методологических средств, доступных в конкретной реализации ФРИЗ. Эти прилагательные относятся к конкретным системам ИИ, условиям и задачам, а также к их типам.

Обозначим $\mathscr{X}$ множество всех *допустимых эпистемологических типов систем ИИ.* Тогда

$$\mathscr{X} \subset \mathscr{E},$$

где $\mathscr{E}$ обозначает множество всех идентифицируемых эпистемологических системных типов. Обычно

$$\mathscr{X} \subseteq \mathscr{E}_l$$

при некотором фиксированном $l \geq 1$, где $\mathscr{E}_l$ $(l \in \mathbf{N})$—это множество всех идентифицируемых эпистемологических системных типов, в которых общее число структурированных систем меньше или равно l.

Обозначим $\mathscr{Y}$ множество *допустимых типов методологических отличий,* и пусть $\mathscr{Z}$—это множество всех *допустимых типов систем ИИ.* Тогда

$$\mathscr{Z} \subset \mathscr{X} \times \mathscr{Y};$$

$\mathscr{Z}$ — собственное подмножество $\mathscr{X} \times \mathscr{Y}$, поскольку некоторые методологические отличия неприменимы ко всем эпистемологическим типам систем ИИ. Обозначим $\mathscr{S}$ множество всех допустимых систем ИИ (т. е. конкретных систем, а не системных типов). Тогда $\mathscr{Z}$ индуцирует разбиение

$$\mathscr{S}/\mathscr{Z} = \{\mathscr{S}_z \mid z \in \mathscr{Z}\}$$

на $\mathscr{S}$, где $\mathscr{S}_z$ —множествовсех допустимых систем типа z . Пусть

$$s_{z,i} \in \mathscr{S}_z \quad (z \in \mathscr{Z}).$$

Тогда $s_{z,i}$ —допустимая система ИИ типа z , идентифицированная (отличающаяся от других систем ИИ типа **z**) идентификатором **i**.

Обозначим $\mathscr{R}_z$ множество всех допустимых типов требований, применимых к одной системе ИИ типа **z** , а $\mathscr{Q}_{z,z'}$ множество всех допустимых типов требований, применимых к паре систем типов **z** и **z'**. Тогда мы можем определить

$$^1\mathscr{Q} = \bigcup_{z \in \mathscr{Z}} Q_z$$

и

$$^2\mathscr{Q} = \bigcup_{z,\, z' \in L} \mathscr{Q}_{z,z'}.$$

Очевидно, что множество

$$\mathcal{Q}=\,^1\mathcal{Q}\cup\,^2\mathcal{Q}$$

состоит из всех *допустимых типов требований.*

Обозначим $\mathcal{R}_z$ множество всех допустимых требований (т. е. конкретных требований, а не типов требований), применимых к одной системе ИИ типа **z** . Тогда $\mathcal{Q}_z$ индуцирует разбиение

$$\mathcal{R}_z/\mathcal{Q}_z=\{\mathcal{R}_{z\,j}\mid \mathbf{q}_{z,\,j}\in\mathcal{Q}_z\}$$

на $\mathcal{R}_z$, где $\mathcal{R}_{z,j}$—множество допустимых требований типа **j**, применимых к одной системе типа z . Пусть

$$r_{z,j,u}\in\mathcal{R}_{z,j},$$

т. е. $\mathbf{r}_{z,\,j,u}$ — допустимое требование типа **j** , применимое к одной системе типа **z** и однозначно определенное идентификатором **u**.

Обозначим $\mathcal{R}_{z,z'}$ множество всех допустимых требований, применимых к паре систем типов **z** и **z'**. Тогда $\mathcal{Q}_{z,z'}$ индуцирует разбиение

$$\mathcal{R}_{z,z'}/\mathcal{Q}_{z,z'}=\{\mathcal{R}_{z,z'k}\mid \mathbf{q}_{z,z',k}\in\mathcal{Q}_{z,z'}\}$$

на $\mathcal{R}_{z,z'}$, где $\mathcal{R}_{z,z',k}$—множество всех допустимых, требований типа **k**, применимых к паре систем типов **z** и **z'**. Пусть

$$\mathbf{r}_{z,z'k,u}\in\mathcal{R}_{z,z',k},$$

т. е. $\mathbf{r}_{z,\,z',k,u}$ обозначает допустимое требование типа **k**, применимое к паре систем типов **z** и **z'** и однозначно определенное идентификатором **u**.

Введенные понятия допустимых систем ИИ, требований и их типов позволяют определить допустимые задачи и их типы. Обозначим соответственно $^1\mathcal{P}$ и $^2\mathcal{P}$ множество всех *допустимых типов элементарных задач* и множество всех *допустимых типов базовых задач.*

Тогда

$$^1\mathcal{P}=\{(\mathbf{z},\,\mathbf{q}_{z,j})\mid \mathbf{z}\in\mathcal{Z},\,\mathbf{q}_{z,j}\in\mathcal{Q}_z\},$$

$$\mathcal{P}=\{(\mathbf{z},\,\mathbf{z}',\,\mathbf{q}_{z,z',k})\mid \mathbf{z},\,\mathbf{z}'\in\mathcal{Z},\quad \mathbf{q}_{z,z',k}\in\mathcal{Q}_{z,z'}\},$$

а

$$\mathcal{P}=\,^1\mathcal{P}\cup\,^2\mathcal{P}$$

— множество всех допустимых типов задач. Хотя в выражении для $^2\mathcal{P}\mathbf{z}$ может быть тем же типом системы ИИ, как и $\mathbf{z}'$, они используются для двух различных систем в каждой конкретной базовой задаче. Это является отличием базовых задач от элементарных, в каждой из которых рассматривается только одна конкретная система.

В результате описания типов требований в множестве $\mathcal{Q}_{z,z'}$ $(\mathbf{z},\,\mathbf{z}'\in\mathcal{Z})$ множество $^2\mathcal{P}$ разбивается на два подмножества $^{21}\mathcal{P}$ и

$^{22}\mathcal{P}$ базовых типов задач соответственно первого и второго видов.

Следовательно,

$$\mathcal{P}=\,^1\mathcal{P}\cup\,^{21}\mathcal{P}\cup\,^{22}\mathcal{P}.$$

Множество $\mathcal{Z}$ индуцирует разбиение

$$^1\mathcal{P}/\mathcal{Z}=\{^1\mathcal{P}_z\mid \mathbf{z}\in\mathcal{Z}\}$$

на $^1\mathcal{P}$, где $^1\mathcal{P}_z$ — множество всех допустимых типов элементарных задач, применимых к системам типа **z**. Аналогично множество $\mathcal{Z}^2$ индуцирует разбиение

$$^2\mathcal{P}/\mathcal{Z}^2=\{^2\mathcal{P}_{z,z'}\mid \mathbf{z},\,\mathbf{z}'\in\mathcal{Z}\}$$

на множестве $^2\mathcal{P}$, где $^2\mathcal{P}_{z,z'}$ — множество всех допустимых типов базовых задач, применимых к паре систем типов **z** и **z'**. Аналогичные разбиения можно определить на множествах $^2\mathcal{P}$ и $^{22}\mathcal{P}$. Отметим, что множества $^1\mathcal{P}_z$ и $^2\mathcal{P}_{z,z'}$ являются категориями задач в том смысле, как это обсуждалось в разд. 10.3 и показано на рис. 4.

Обозначим $^1\mathcal{PP}$ множество всех допустимых простейших задач.

Тогда множество $^1\mathcal{P}$ индуцирует разбиение

$$^1\mathcal{PP}/^1\mathcal{P}=\{^1\mathcal{PP}_{z,j}\mid \mathbf{z}\in\mathcal{Z},\ \mathbf{q}_{z,j}\in\mathcal{Q}_z\}$$

на $^1\mathcal{PP}$, где $^1\mathcal{PP}_{z,j}$ —множество всех допустимых элементарных задач, определенных для одной системы типа **z**, и требования типа $\mathbf{q}_{z,j}$.

Обозначим $^2\mathcal{PP}$ множество всех допустимых базовых задач.

Тогда множество $^2\mathcal{P}$ индуцирует разбиение

$$^2\mathcal{PP}/^2\mathcal{P}=\{^2\mathcal{P}_{z,z',k}\mid \mathbf{z},\,\mathbf{z}'\in\mathcal{Z},\ \mathbf{q}_{z,z',k}\in\mathcal{Q}_{z,z'}\}$$

на $^2\mathcal{P}_{\mathcal{P}}$, где $^2\mathcal{P}_{z,z'k}$ —-множество всех допустимых базовых задач, определенных в терминах пары систем типов **z** и **z'** и требования типа $\mathbf{q}_{z,z'k}$. Каждое множество $^2\mathcal{P}_{z,z',k}$ в дальнейшем разбивается на подмножества базовых задач первого и второго вида, обозначаемые соответственно $^{21}\mathcal{P}_{z,z',k}$ и $^{22}\mathcal{P}_{z,z',k}$.

Тогда

$$^{21}\mathcal{PP}=\bigcup{}^{21}\mathcal{P}_{z,z',k},$$

$$^{22}\mathcal{PP}=\bigcup{}^{22}\mathcal{P}_{z,z',k},$$

где объединения множеств взяты по всем $\mathbf{z},\ \mathbf{z}'\in\mathcal{Z}$, а $\mathbf{q}_{z,z',k}\in\mathcal{Q}_{z,z}$ — множества всех базовых задач, соответственно, первого и второго видов.

Теперь можно формально описать три основных типа допустимых задач:

элементарные задачи, обозначаемые $^1\mathrm{p}_\alpha$, имеют вид

$^{1}p_{\alpha} = (\mathbf{s}_{z,i},\ \mathbf{r}_{z,j,i})$,

где α — однозначный идентификатор четверок $(\mathbf{z},\ \mathbf{i},\ \mathbf{j},\ \mathbf{u})$.

Базовые задачи первого вида, обозначаемые $^{21}p_{\beta}$, описываются так:

$^{21}p_{\beta} = (\mathbf{s}_{z,i},\ \mathbf{z}',\ \mathbf{r}_{z,z',k,u})$,

где β — уникальный идентификатор пятерок $(\mathbf{z},\ \mathbf{i},\ \mathbf{z}',\ \mathbf{k},\ \mathbf{u})$.

Базовые задачи второго вида, обозначаемые $^{22}p_{\gamma}$, описываются следующим образом:

$^{22}p_{\gamma} = (\mathbf{s}_{z,i},\ \mathbf{s}_{z',t}\ \mathbf{r}_{z,z',k,u})$,

где γ — уникальный идентификатор шестерок $(\mathbf{z},\ \mathbf{i},\ \mathbf{z}',\ \mathbf{t},\ \mathbf{k},\ \mathbf{u})$.

Обозначим $\mathcal{PP}$ множество всех допустимых задач, определяемых ФРИЗ. Тогда

$\mathcal{PP} \subset (^{1}\mathcal{PP} \cup ^{21}\mathcal{PP} \cup ^{22}\mathcal{PP})^{*}$,

где звездочка обозначает множество всех последовательностей, которые могут быть образованы элементами множества. Аналогично определяется множество $\mathcal{P}$ всех допустимых типов задач ФРИЗ:

$\mathcal{P} = (^{1}\mathcal{P} \cup ^{21}\mathcal{P} \cup ^{22}\mathcal{P})^{*}$,

где $^{1}\mathcal{P}$, $^{21}\mathcal{P}$, $^{22}\mathcal{P}$ — конечные множества элементарных и базовых типов задач.

Теперь ясно, что на глобальном уровне концептуальная схема ФРИЗ может рассматриваться как **язык для описания допустимых типов интеллектуальных задач**, **алфавит которого состоит из всех допустимых простейших и базовых типов задач, определяемых, в свою очередь, основными эпистемологическими и методологическими типами систем ИИ и типами требований.**

10.5. Описание архитектуры ФРИЗ

Назначение архитектуры заключается в определении, тщательном описании тех функций проектируемого искусственного объекта (будь то здание, механизм или система ИИ), которые необходимы для достижения заданной цели. Ранее с различной степенью детализации обсуждались выбор и описание основных функций ФРИЗ, цель которого состоит в том, чтобы пользователь, обладающий знаниями в области искусственного интеллекта , смог решать интеллектуальные задачи. Цель настоящего раздела — дать сжатый и исчерпывающий обзор этих функций. На рис. 5 показана блок-схема, которую рекомендуется использовать как руководство при чтении следующего описания.

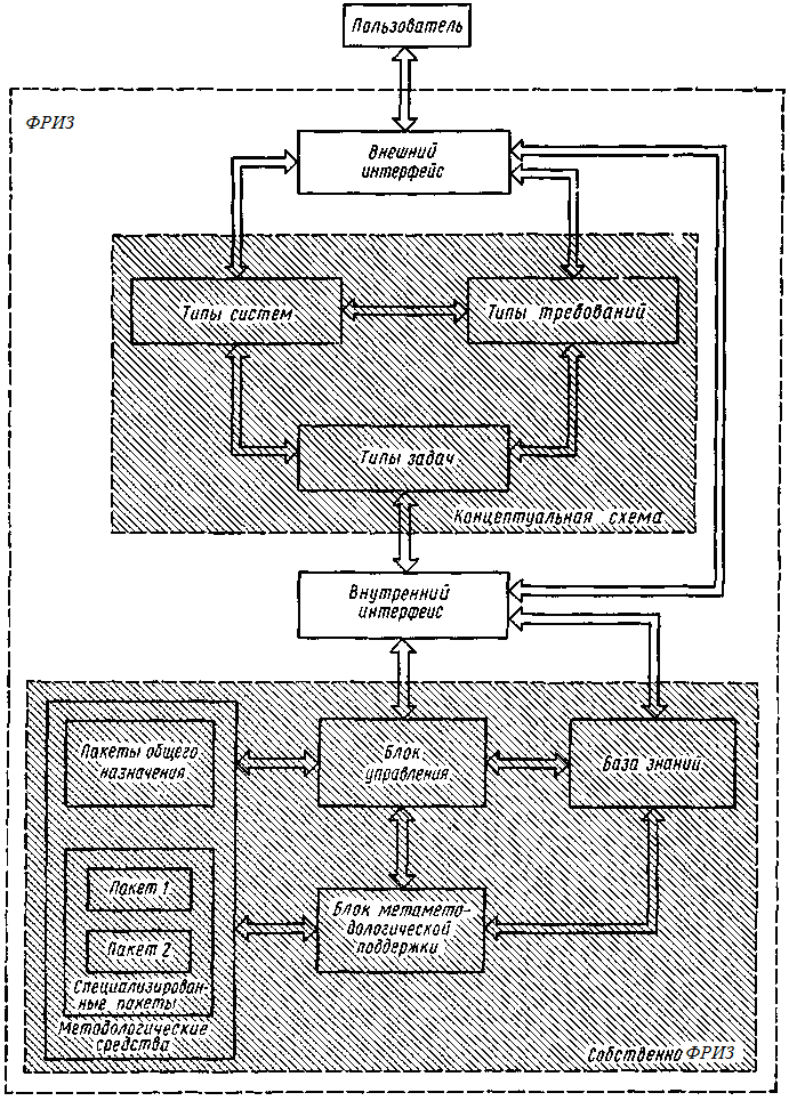


Рис .5. Схема архитектуры ФРИЗ

Концептуальная схема, разработанная в гл. 1—9 и кратко описанная в разд. 10.4, является ядром ФРИЗ. Она представляет собой язык, используемый в ФРИЗ для описания выделенных типов систем ИИ, требований и задач. Когда речь идет об архитектуре, то как указывается в разд. 10.1, фиксируется только эпистемологическая иерархия системных типов. Остальные понятия, такие, как типы требований или методологические отличия систем ИИ, описываются в архитектуре ФРИЗ только в самых общих терминах. Более точное их описание является предметом конкретной реализации ФРИЗ, базирующейся на конкретном множестве системных типов, каждый из которых определяется совокупностью эпистемологических и методологических особенностей, а также конкретными множествами типов требований и, как следствие, типами задач; они называются *допустимыми типами систем ИИ,* требований и задач (т. е. они допустимы для данной реализации ФРИЗ).

Как было показано в разд. 10.4, содержательные типы интеллектуальных задач образованы последовательностями простейших и базовых типов задач. Последние, в свою очередь, образованы интеллектуальными типами и типами требований, являющихся совместимыми в том смысле, что они могут быть применимы друг к другу. Таким образом, три основные категории, образующие концептуальную схему ФРИЗ: **системные типы, типы требований и типы систем ИИ — взаимосвязаны.** На рис. 5 это показано двусторонними связями между соответствующими блоками.

Концептуальная схема — это лингвистическая среда, в которой пользователь общается с ФРИЗ, состоящим из соответствующей базы знаний, множества метаметодологических средств, а также блока управления. Связь *пользователь* — ФРИЗ двусторонняя, и с каждой стороны снабжена соответствующим интерфейсом. Назовем интерфейс со стороны пользователя *внешним интерфейсом,* а со стороны ФРИЗ — *внутренним интерфейсом.*

Выделим два типа внешнего интерфейса, каждый из которых может быть включен в конкретную реализацию ФРИЗ. Первый спроектирован в расчете на *опытного пользователя,* который в достаточной степени знаком с концептуальной схемой ФРИЗ (по меньшей мере на уровне этой книги) и ограничениями той реализации ФРИЗ, которую он собирается использовать. Этот тип интерфейса основан на предположении, что пользователь не нуждается в помощи при формальном определении его систем и требований, и, следовательно, единственная функция интерфейса состоит в проверке на возможные несоответствия в формулировках пользователя.

Другой тип внешнего интерфейса спроектирован в расчете на *обычного пользователя,* чье знание концептуальной схемы ФРИЗ является недостаточным. Функция этого интерфейса состоит не только в проверке возможных несоответствий в формулировках пользователя, но и в том, чтобы предоставить пользователю широкий спектр услуг при формулировании его задачи. Это означает, что внешний интерфейс должен содержать соответствующие *процедуры опроса* для идентификации типов систем ИИ и требований, а также конкретных систем ИИ и требований заданных типов. Такие процедуры могут оказаться очень сложными (а может быть, и совершенно нереальными) при условии, что пользователь ничего не знает о концептуальной схеме ФРИЗ. Поэтому представим себе, что в любых реализациях ФРИЗ, доступных, по крайней мере, в ближайшем будущем, от обычного пользователя будет требоваться некоторое *минимальное знание концептуальной схемы ФРИЗ.* Эти минимальные сведения можно сделать стандартной частью любого руководства по ФРИЗ.

Упомянутые процедуры опроса, позволяющие провести идентификацию и являющиеся важным элементом внешнего интерфейса ФРИЗ, могут быть усовершенствованы на различных уровнях в зависимости от того минимального знания концептуальной схемы ФРИЗ, которое требуется от пользователя. **Чем меньше знаний, тем более сложными и менее совершенными становятся процедуры**. В предельном случае, когда не требуется никаких знаний, создание удачных процедур опроса является главной исследовательской проблемой. **Затронутые вопросы в полной изучаются мере преимущественно в области искусственного интеллекта.**

***Собственно ФРИЗ* состоит из четырех функциональных элементов: множества методологических средств, базы знаний, блок мета-методологической поддержки и блока управления.** Все понятия ФРИЗ представлены в ФРИЗ в некотором стандартизованном виде, что осуществляется посредством внутреннего интерфейса. Понятно, что такой блок как база знаний, а также методологические и метаметодологические блоки должны обладать соответствующими стратегиями рассуждения.

***Методологические средства* — это пакеты методов (и соответствующих компьютерных программ), с помощью которых могут быть решены некоторые из допустимых типов задач.** Они разделяются на **общие и специализированные пакеты**. *Общие пакеты* предназначены для типов задач, сформулированных в терминах наиболее общих методологических отличий, доступных в данной реализации ФРИЗ; *специализированные пакеты* предназначены для

всех типов задач, основанных на менее общих методологических отличиях.
Любое методологическое средство — это множество методов (и соответствующих компьютерных программ) для решения некоторых простейших или базовых типов задач и процедура (компьютерная программа), определяющая порядок, в котором отдельные методы должны применяться. Таким образом, **методологические средства образуются из общего набора методов (программ), имеющегося для решения допустимых простейших и базовых типов задач, с помощью процедуры (управляющей программы), применяющей необходимые методы в соответствующем порядке.**
В ФРИЗ учтены различные метаметодологические соображения. Они реализуются с помощью *метаметодологического блока.* Этот блок содержит информацию об упорядочении всех допустимых задач, следующей из их общности методологического статуса. Упорядочение по общности отражает, в основном, упорядочение методологических отличий систем ИИ и требований. Методологический статус задачи связан с ее разрешимостью, запросами на время решения и требуемую память, с характеристиками соответствующих методологических средств.
Теоретически неразрешимые задачи отличаются от задач, неразрешимость которых обусловлена исключительно ограничениями данной реализации ФРИЗ. Если задача неразрешима в последнем смысле, то возможны два ответа ФРИЗ. **Во-первых, пользователю могут быть предложены альтернативные формулировки задачи, основанные на более строгих предположениях и допускающие решение ее с помощью ФРИЗ**; при этом блок методологической поддержки осуществляет переход от данной методологической парадигмы к более конкретным парадигмам. **Во-вторых, может быть вызван блок базы знаний, чтобы предоставить пользователю полез- ную информацию относительно исходной задачи, например ссылки на соответствующие библиотеки программ, статьи или книги.**
Блок метаметодологической поддержки должен осуществлять необходимый анализ вычислительной сложности каждой конкретной задачи, для которой в данной реализации ФРИЗ имеются средства решения. Если задача оказывается практически неразрешимой, то этот блок должен, по возможности, выработать альтернативные формулировки задачи (основанные на более строгих предположениях), которые можно численно реализовать. Кроме того, для каждой задачи, которая может быть решена данной реализацией ФРИЗ, этот блок

должен определить приблизительные вычислительные затраты и другие соответствующие характеристики используемого метода.
Как было отмечено выше, ***блок знаний* содержит полезную информацию о задачах, которые не могут быть решены данной реализацией ФРИЗ**. И, кроме того, этот блок может содержать другую необходимую информацию о системах ИИ и интеллектуальных задачах. К ней, например, относятся теоретические или экспериментальные законы, принципы или эмпирические правила науки о системах, такие, как закон необходимого многообразия или закон необходимой иерархии.
Связь пользователя с блоками ФРИЗ осуществляется либо через концептуальную схему, либо непосредственно через внешний и внутренний интерфейсы. Первая связь содержит формулировки задач, а вторая связана с различными метаметодологическими подходами и использованием базы знаний.
Необходимая координация работы трех описанных блоков ФРИЗ осуществляется *блоком управления.* Им в соответствии с требованиями и другими условиями, в основном принимается решение о том, какой блок и как следует активизировать.

10.6. Развитие и совершенствование ФРИЗ и СИИ

В соответствии с принятой терминологией ФРИЗ — система искусственного интеллекта. Системы искусственного интеллекта на больших компьютерах позволяют неопытному пользователю использовать в качестве руководства записанное ранее «знание» интеллектуала или «коллективное знание» многих интеллектуала. Используя системы искусственного интеллекта, Вы обсуждаете с «коллегой» проблемы медицинской диагностики или поиска нефти, или сложные процессы математического открытия. Вы дискутируете с ними, спрашиваете их мнение и в результате сотрудничества с теми, кто, может быть, давно умер, приходите к решению
Вычислительная техника затрагивает самую суть нашего существования, поскольку **общение — это сердце человеческой цивилизации.** Мы представляем собой уникальный вид благодаря нашим способностям к адаптации и обучению, влияние которых чрезвычайно усиливается благодаря нашей способности общаться. Вполне достаточно, чтобы только один человек получил знания непосредственно из собственного опыта. Остальным можно рассказать, а среда позволяет перенести рассказанное через пространство и время.

Новая среда *действительно* изменяет наш мир, а компьютер обеспечивает самое необходимое для кодирования не самого разговора, а возможности разговаривать, не самой картины, а возможности ее воссоздать

В то время как большинство описанных в литературе систем искусственного интеллекта предоставляют пользователю знания о традиционных дисциплинах (например, о конкретных областях медицины или права), роль ФРИЗ заключается в том, чтобы помочь пользователю при рассмотрении интеллектуальных задач. Таким образом, ФРИЗ объединяет интеллектуальные знания и методологию, и, следовательно, его применение выходит за рамки традиционных дисциплин.

По сравнению с системами искусственного интеллекта другого вида, цель которых заключается в решении задач в целом (фундаментальные решатели задач) возможности ФРИЗ ограничены, он решает исключительно интеллектуальные задачи. Это ограничение умышленное и отражает нашу философскую точку зрения, что **симбиоз человек — компьютер представляет собой наилучшее средство для решения задач в целом и что потенциальные возможности компьютера максимальны в задачах, которые рассматриваются как интеллектуальные.** Цель ФРИЗ заключается в максимальной реализации этих потенциальных возможностей.

Одна из главных, но недостаточно четко акцентированных ранее особенностей архитектуры ФРИЗ, которой мы пока недостаточно уделяли внимание, является то, что по своей природе она эволюционна. Для того чтобы более тщательно пояснить этот аспект, необходимо сначала показать, что **концептуальная схема ФРИЗ, так же, как и используемые знания и методологические основы, не должна развиваться в изоляции от традиционных дисциплин.** В самом деле, традиционные дисциплины рассматривались как источники системных идей, из которых и должна была возникнуть архитектура ФРИЗ.

Фактически все традиционные дисциплины тем или иным образом были связаны с интеллектуальными задачами определенных типов и, следовательно, были разработаны методы для решения этих задач. Развитие любой из них внесло свой вклад в изучение систем ИИ и методологию их решения; результаты фактически были получены в рамках каждой из этих дисциплин.

Следует подчеркнуть, что описанная в этой работе концептуальная схема ФРИЗ в значительной степени развивалась в течении мгогих лет в процессе тщательного определения понятия системы ИИ и соответствующих интеллектуальных задач при изучении многих

отдельных дисциплин, абстрагирования их от контекста, введения соответствующих категорий и, в конце концов, объединения их в одно связное целое. Хотя настоящая схема весьма устойчива, эти процессы продолжаются, и в будущем может возникнуть необходимость в расширениях и других эволюционных изменениях этой схемы. Аналогичные процессы происходят и в процессе эволюции базы знаний ФРИЗ, методологических и метаметодологических основ. Очевидно, что отбор понятий, методов и знаний из различных традиционных дисциплин — только часть всего процесса развития архитектуры ФРИЗ. В действительности это только основа для настоящих исследований ФРИЗ, цель которых состоит в том, чтобы, определяя и заполняя пробелы в концептуальной схеме, сделать ее по возможности полной и всесторонне совершенствовать знания и методологические основы, особенно в тех важных областях, которые пока недостаточно развиты.

Еще один дополнительный и весьма важный аспект эволюционной природы архитектуры ФРИЗ заключается в необходимости того, чтобы ФРИЗ был адаптируем к нуждам его пользователей. На обычном языке это означает, что предполагается хранение записей о всех взаимодействиях *пользователь — ФРИЗ* и использование этой информации в дальнейших исследованиях ФРИЗ. Для этой цели определенный интерес представляют записи и неудачных взаимодействий пользователя и ФРИЗ, поскольку они могут обозначить недостатки интерфейса пользователь — ФРИЗ, методологически недостаточно исследованные типы задач или даже необходимость расширения концептуальной схемы.

На рис. 6 показан описанный выше процесс развития архитектуры ФРИЗ.

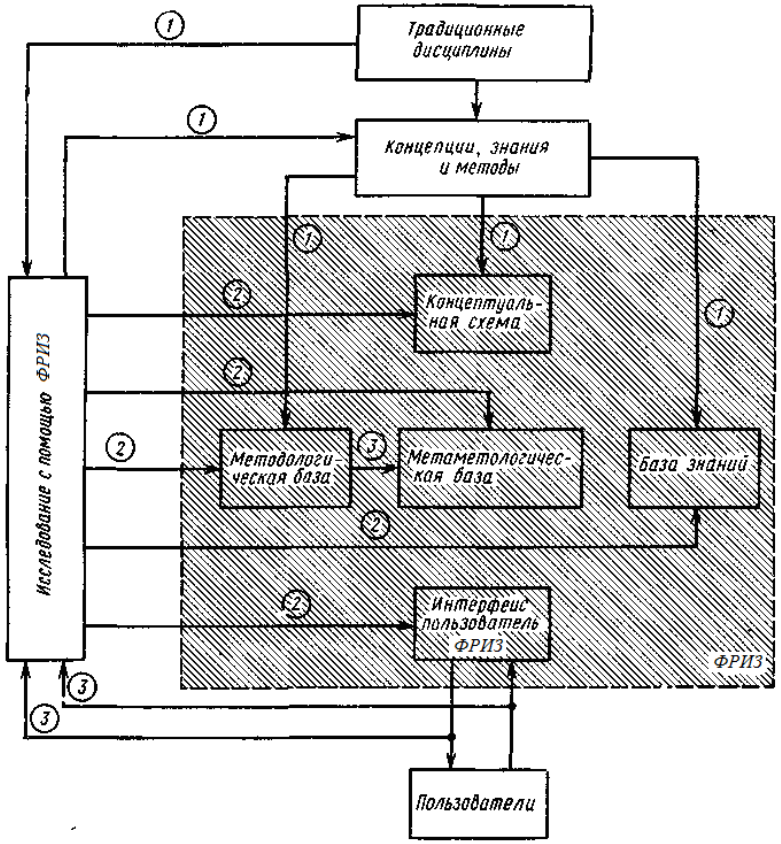


Рис.6. Схема процесса развития архитектуры ФРИЗ

Цифры при соответствующих связях на диаграмме имеют следующий смысл:
1 — процессы выделения, абстрагирования, введения категорий и объединения соответствующих понятий, знаний и методов из традиционных дисциплин;
2 —четыре области исследования ФРИЗ;
3 —используемая как руководство при исследовании ФРИЗ информация о взаимодействиях пользователей и ФРИЗ.
Поскольку среда, в которой развивается ФРИЗ (традиционные дисциплины и представляющие эти дисциплины пользователи ФРИЗ),

имеет определяющее значение для его эволюции, то представляется возможным охарактеризовать архитектуру ФРИЗ как естественную.
Закончим этот раздел и эту книгу цитатой:
«Полезна полнота без полноты. Желательна завершенность без завершения.»

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. А.Е. Кононюк. Дискретно-непрерывная математика. Математическая логика. Кн.9. Ч.1 Киев: «Освіта України», 2017. 464с.
86. А.Е. Кононюк. Дискретно-непрерывная математика. Математическая логика. Кн.9. Ч.2 Киев: «Освіта України», 2017. 564с.
87. А.Е. Кононюк. Дискретно-непрерывная математика. Математическая логика. Кн.9. Ч.3 Киев: «Освіта України», 2017. 424с.
88. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 1. Киев: «Освіта України», 2017. 730с