3 р. 60 к.

Основная рассматриваемая задача — развитие концепций, связанных с системной ориентацией на привычные задания, т.е. имеющие вид выражений естественного языка (ЕЯ).

Решаемые вопросы:

- формализация семантического эквивалента, представление и использование знаний;
- автоматический ввод конструкций ЕЯ с выявлением семантического эквивалента, управление вводом за счет лингвистических знаний;
- организация системной активности, возрастающей в процессе накопления знаний;
- обучение по примерам, обобщение, формирование гипотез.

Области приложения:

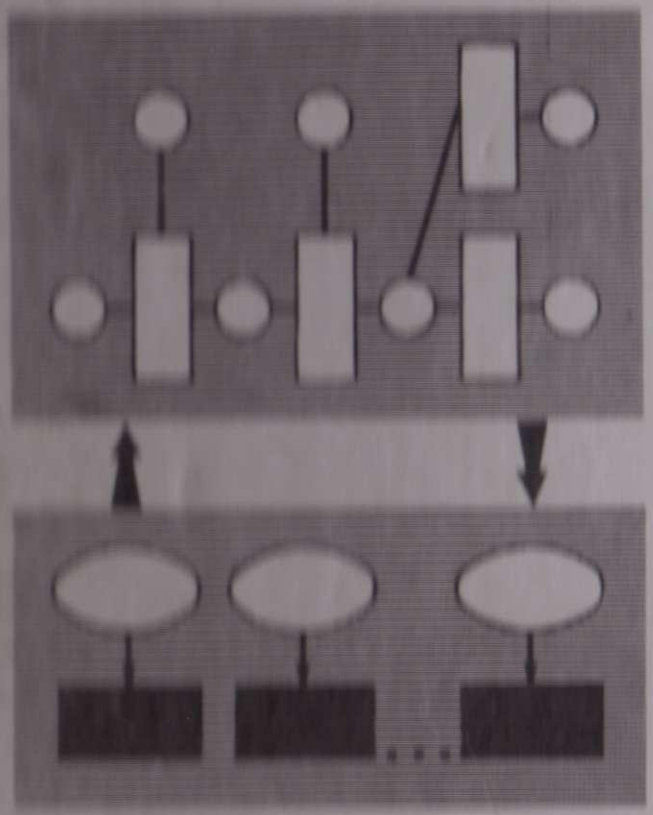
- базы знаний, обеспечивающие хранение и использование обобщенной информации, введение сложных концептуальных моделей;
- системы структурного распознавания, ориентированные на работу со сложными описаниями и учитывающие представления пользователей о классах объектов;
- экспертные системы, допускающие обучение по примерам, а также с помощью специалиста, пользующегося ЕЯ;
- "благожелательные" системы программирования, допускающие расширение входного языка до уровня форм ЕЯ, а также автоматическое конструирование алгоритма с устранением типовых ошибок;
- трансляторы, учитывающие семантическую компоненту и работающие согласованно с базами знаний.

Издательство «Наука»

И.П.КУЗНЕЦОВ СЕМАНТИЧЕСКИЕ ПРЕДСТАВЛЕНИЯ

И.П.КУЗНЕЦОВ

СЕМАНТИЧЕСКИЕ ПРЕДСТАВЛЕНИЯ



«НАУКА»

вать, что делать и как. Хотя такие указания могут даваться во все более приемлемых формах — процедуральной, декларативной, в виде специальных заданий и др., но все равно требуется постоянное участие разработчика. Отсюда существенные трудности в использовании систем, сопровождении. И проблема номер один — добиться необходимой автономии, самостоятельности систем. Ситуация в какой-то степени напоминает начальный этап в развитии воздухоплавания. Полетит или не полетит? Конечно, можно было создавать все более мощные метающие или бросающие устройства, совершенствовать крылья, прикрепляемые к рукам человека. Но было важно, чтобы аппарат полетел сам, за счет своих внутренних сил. Аналогом таких сил в области искусственного интеллекта является внутренняя организация создаваемой системы, ее знания и механизмы, за счет которых система должна постоянно обучаться, подстраиваться. В этом плане и написана данная книга. В основном она посвящена теоретическим аспектам. В то же время автором было рассмотрено большое количество примеров из различных областей приложений. Представляется, что книга вызовет интерес читателя в силу ее актуальности, пионерского характера.

Е. В. Золотов

ВВЕДЕНИЕ

Выражения внешнего языка (естественного или формального) можно рассматривать как определенного сорта задания, указывающие на необходимость *учесть, проверить, вычислить, ответить, понять, сделать* и т. д. Данная монография посвящена принципам организации систем, ориентированных на такие задания. Основное внимание уделяется средствам и методам их внутрисистемной обработки с учетом следующих факторов. Во-первых, в каждом задании следует четко различать форму выражения и сущность или семантический эквивалент. В привычных нам заданиях одно и то же может быть выражено по-разному. По возможности они должны быть отображены на один семантический эквивалент, что должна осуществлять сама система. Во-вторых, задания чрезвычайно разнообразны по своему характеру. Учесть все случаи их обработки не представляется возможным. Поэтому система должна быть расширяющейся или открытой. Любое очередное задание, с которым система успешно справилась, должно повышать ее возможности по выполнению других заданий. Никакое воздействие через внешний язык не должно проходить бесследно.

Данная проблематика относится к области информатики (искусственный интеллект), интерес к которой за последнее время неизмеримо вырос. Дело в том, что одно из магистральных направлений развития вычислительной техники — ее все бóльшая ориентация на категорию так называемых слабоподготовленных или необученных пользователей. Отсюда и необходимость приближения языка машины к привычным формам выражения заданий. Не удивительно, что многие технически высокоразвитые страны (США, Япония, Англия, Франция и др.) объявили о выполнении работ в указанном направлении на уровне национальных программ. В нашей стране исследования также приобретают широкий размах.

В настоящее время уже мало у кого вызывает сомнение, что системы, допускающие ввод заданий в привычных для человека формах, должны обладать знаниями. Иначе система не сможет самостоятельно учитывать все новые факторы. Существенны расхождения во мнениях в вопросах о том, как должны быть устроены такие знания, какой они должны иметь вид. Многие исследователи придерживаются определенной точки зрения, согласно которой в основе знаний должны быть привычные им средства. Отсюда различные подходы, направления.

Специалисты в области логики считают, что знания должны быть основаны на их формализмах, в частности, для этой цели может быть использован язык логики предикатов (или же одно из его расширений). Однако возможности систем, основанных на таких формализмах, оказыва-

ются невысокими. Многие привычные нам задания достаточно трудно выражать в языках логики и выполнять за счет правил вывода. Требуется что-то еще. И именно это «что-то» является источником развития самой логики, причиной появления прикладных, нестандартных логик.

Специалисты в гуманитарных областях недоумевают — почему в качестве знаний нельзя использовать привычный им естественный язык (ЕЯ). Хотя не представляют себе, как это можно сделать.

Специалисты в области программирования полагают, что роль знаний могут играть средства языков программирования. Все чаще говорят о необходимости их развития (с целью «приближения» к ЕЯ). Отсюда возникло новое научное направление, получившее название «логическое программирование». Речь идет о разработке специальных языков «программирования знаний», позволяющих человеку быстро создавать систему, настраивать ее на выполнение того или иного задания. Но выясняется, что для каждого задания (или типа заданий) требуются свои знания. Их нужно «разрабатывать». При попытках приближения к более естественным формам записи заданий объем подобных работ неизмеримо растет. Приходится учитывать все новые факторы. Выясняется, что задания различных типов сильно связаны между собой. Отсюда — трудности разработки и программирования.

Чтобы сказанное сделать более понятным, уместно провести следующую аналогию. Естественные задания чем-то напоминают достаточно свободное перемещение по пересеченной местности, что человек умеет делать. Подход, основанный на обычном программировании, аналогичен построению дорогостоящих дорог. По ним можно быстро ездить, возить большие грузы. Но удается связать лишь немногие пункты. Логическое программирование связано с построением более дешевых дорог. Удается связать уже больше пунктов. Но возникают ограничения по скорости перевозок, объему. И, наконец, чисто логический подход аналогичен созданию транспортных единиц, которые в принципе способны перемещаться по местности достаточно произвольно. Но их органы движения настолько слабы, что их приходится постоянно подталкивать, перетаскивать через препятствия.

Как же обеспечить достаточно произвольные перемещения, т. е. выполнение заданий с учетом их разнообразия? Как добиться высокой степени самостоятельности? Какими при этом должны быть системные средства, в частности знания? Здесь следует различать внешнее проявление какого-либо процесса и что за всем этим стоит. Например, видимые движения человека имеют совершенно другой характер, чем те «механизмы», которые их вызывают. Аналогично, внешний язык человека (формальный или ЕЯ) следует отличать от той «инструментальной поддержки», роль которой играет человеческий интеллект. Язык — его проявление. И знания человеку необходимы прежде всего для обеспечения указанной поддержки.

Следует ли в качестве знаний использовать какой-либо внешний язык человека? Конечно, такая альтернатива возможна. Знания в какой-то степени могут быть отображены на язык, а механизмы, обеспечивающие их использование, записаны в виде операторов над конструкциями языка. Но выясняется, что здесь важную роль играет форма, вид конструкций, от которых во многом зависит сложность операторов и решаемых

задач. Необходимо учитывать, что привычная нам последовательная форма выражения заданий во многом определяется спецификой органов речи и ее восприятия. Мозговые «структуры» имеют совершенно другой вид. И то, что просто на уровне этих структур, может оказаться неизмеримо сложным на уровне внешнего языка.

Приведенные соображения легли в основу настоящей работы. В ней устанавливаются требования, которым (по мнению автора) должны удовлетворять системные знания. Различаются структуры знаний, записываемые в языке представления, и принципы функционирования, выражаемые в виде механизмов манипулирования над структурами. Показывается, что в качестве структур знаний весьма перспективно использование однородных средств. Имеются в виду графы, сети, в том числе семантические. Но они (в своем традиционном виде) ориентированы на узкоспециализированные задачи. Требуется их развитие в направлении повышения изобразительных возможностей, создания необходимой надстройки. В связи с этим предлагается специальный класс семантических сетей, на которых задается операционная семантика. Показывается, каким образом на базе предлагаемых средств может обеспечиваться обработка различных видов заданий с учетом упоминавшихся ранее факторов. При этом речь в основном будет идти об обработке на уровне семантических эквивалентов (структур), что отражено в названии работы — семантические представления.

Данная монография явилась логическим продолжением работ автора, опубликованных ранее издательством «Наука» (см. [14, 24, 25]). Основная цель проводимых исследований — построение новых классов систем и машин, которые в удобных для человека формах допускают настройку на сравнительно широкий круг заданий. В связи с этим большое внимание будет уделяться вопросам унификации, т. е. поиску единых методик, «механизмов». Будет обсуждаться место индуктивных принципов, в том числе задач формирования гипотез, обучения по примерам. База внутрисистемной обработки должна быть семантико-индуктивной. Будет показана необходимость в специальном уровне структур, обеспечивающем сочетание вводимой информации и формируемых системой представлений.

Где же могут быть использованы предлагаемые в этой работе средства и методики? Прежде всего в задачах построения некоторых классов систем перспективного плана. Имеютя в виду следующие системы.

1. Базы знаний, ориентированные на автоматическое отображение и использование сложных описаний структурного и обобщенного характера (имеются в виду достаточно точные описания, т. е. допускающие однозначную интерпретацию терминов в рамках соответствующих языковых конструкций — ЕЯ или его формализованных частей).

2. Системы структурного распознавания, ориентированные на работу со сложными описаниями и допускающие в широких пределах использование вводимой извне информации — разного рода обобщенных сведений о классах объектов, в том числе сведений, содержащихся в толковых словарях.

3. Лингвистические процессоры, допускающие в достаточных пределах настройку на язык пользователя и ориентированные на более естественные формы выражения заданий.

4. Логические процессоры, допускающие за счет входной информации настройку на тот или иной способ выполнения задания.

5. Трансляторы, допускающие (за счет декларативной компоненты, т. е. знаний) настройку на используемый язык программирования; при этом обеспечиваются дополнительные возможности в плане расширения языка в направлении привычных человеку форм.

6. Многоязыковые системы программирования, обеспечивающие компоновку алгоритмов на уровне языка посредника (его роль могут играть семантические сети) и устранение ошибок с восстановлением недостающей информации.

7. Экспертные системы, допускающие в широких пределах учет поступающей от специалистов информации.

8. Системы для модельных экспериментов, ориентированные на задачи автоматического построения и ведения моделей за счет входных описаний (имеются в виду дискретные динамические модели и их точные описания — структуры, принципов функционирования).

9. Системы активного диалога, допускающего расширение форм внешней и внутренней активности за счет накопления декларативной компоненты — знаний.

10. Системы, ориентированные на работу с нечеткой информацией и обеспечивающие учет относительного характера расплывчатых понятий (например, те, которых юноши называют *пожилыми,* для стариков являются *молодыми*).

11. База математических конструкций, допускающая автоматическое отображение формальных текстов с их описаниями, а также построение и ведение соответствующих моделей (имеется в виду специальный класс математических машин, обеспечивающих в своих рамках хранение и ведение знаковых моделей).

Задачи построения перечисленных классов систем родственны по своему характеру. И если делать ориентацию на выявление семантической компоненты и обработку на уровне семантических эквивалентов, то системы вообще делаются трудноразличимыми. Проблемы их организации сильно переплетаются и в конечном счете сводятся к единым задачам — накопления и использования внутрисистемных знаний. Именно им посвящена данная работа. При этом основной акцент будет сделан на принципы построения или так называемую логическую организацию. Вопросы реализации пока останутся в стороне. Хотя и здесь открываются новые возможности — в плане построения специализированных машин, ориентированных на обработку логико-структурной информации, т. е. обеспечивающих накопление и использование знаний на уровне физических структур.

Большое внимание будет уделяться теоретическим аспектам, касающимся в основном некоторых математических формализмов. Будет учитываться тот факт, что любой формализм и инструментарий существуют в представлениях человека, имеющих выход на ЕЯ. Обучение формализму идет с помощью ЕЯ и предполагает определенного сорта знания и умения, в частности умения оперировать над символами, умозаключать и др. Будет рассматриваться отображение формализмов с их описаниями на предлагаемые средства. Речь идет о внутрисистемных представлениях о типовых формализмах. Такие представления образуют специального вида модели, позволяющие глубже понять, что же делает математик,

и соответственно, какой должна быть система-математик, способная воспринимать математические тексты. Будут анализироваться процессы, протекающие в рамках таких моделей, рассматриваться, в каких случаях такие процессы делаются неустойчивыми, недопустимыми. Отсюда станут более понятными причины типовых парадоксов, неразрешимых проблем.

Данная книга ориентирована на читателей, которые могут и не иметь специальной математической подготовки. Соответствующие разделы (в основном они даны петитом) могут быть опущены без ущерба понимания последующего материала. Более того, многие разделы могут быть легко усвоены за счет иллюстраций. Достаточно прочитать общие положения (в начале параграфов) и просмотреть рисунки. Если все в них понятно, то можно переходить к следующему параграфу. Так лучше делать при предварительном ознакомлении.

В процессе исследований и написания работы большая помощь была оказана член-корреспондентом АН СССР Е. В. Золотовым, которому автор выражает свою искреннюю признательность. Хочется также выразить благодарность А. С. Нариньяни, С. С. Гончарову, А. Н. Лященко, Д. И. Свириденко и другим, чьи советы и замечания во многом способствовали улучшению предлагаемого Вашему вниманию материала.

ГЛАВА 1

ПРЕДСТАВЛЕНИЕ ЗНАНИЙ

В этой главе будут рассматриваться общие принципы построения автономных систем, способных накапливать информацию и использовать ее для различных целей — ответа на запросы, распознавания, выявления смысла сообщений, проверки правильности и ряда других. Будет говориться, с каких методологических позиций следует проводить исследования подобного сорта «многоплановых» или комплексных задач, связанных по своей природе. Далее остановимся на необходимости в специальной парадигме, т. е. схемной организации, ориентированной на вопросы отображения естественно-языковой информации, ее представления и использования. Такая парадигма будет называться системами с семантической памятью. Имеется в виду «база структур» или «база знаний», основанная на унифицированных средствах и наделенная дополнительными возможностями в плане комплексного решения задач. В соответствии с указанной парадигмой будут установлены дополнительные требования к языку представления информации и методикам («механизмам») ее обработки. Будет показано, что существующие средства (по разным причинам) не удовлетворяют данным требованиям. В связи с этим ставится задача разработки нового языка. Ниже будут описаны его наиболее типовые конструкции и принципы их использования для представления информации достаточно простого вида.

§ 1.1. ОБ ОСОБЕННОСТЯХ ИССЛЕДОВАНИЯ

Речь будет идти о методологической основе создания систем, ориентированных на обработку сравнительно широкого круга заданий (см. Введение). Основное внимание будет уделяться средствам, необходимым для построения проекта или знаковой модели системы. Такие средства будем называть *логическим базисом* (аппаратом) и связывать с задачей *логической организации* систем. При этом будут учитываться следующие факторы. Во-первых, многие объекты и их описания имеют структурный характер. Их обработка на уровне семантических эквивалентов предопределяет необходимость соответствующих средств и методов структурной обработки. И, во-вторых, следует учитывать ориентацию систем на конечных пользователей — на их задания. Отсюда необходимость обеспечения все большего комплекса удобств, постоянное расширение системных возможностей для подстройки к пользователю, в том числе под его язык.

Неполнота. Это первая особенность исследований. Она является следствием высокого разнообразия «проявлений» человеческого интеллекта и соответственно поступающих от пользователя заданий. Такое разнообразие характерно для любой сферы деятельности. Возьмем для примера диалог на естественном языке. В процессе диалога человек широко пользуется разного рода анафорическими ссылками, эллиптическими конструкциями. Диалог протекает в рамках определенного сценария или модели. При этом предполагается умение объяснять, рассуждать, перехватывать инициативу, активно спрашивать. Как выясняется, способов объяснения, форм активности достаточно много. Все это каким-то образом должно учитываться системой по мере необходимости. Следовательно, она должна быть *расширяющейся* или открытой.

Разнообразие проявлений характерно и для области принятия решений, распознавания, информационного обслуживания, где большую роль играют всевозможные ограничения, уровни решений. Задания могут носить достаточно сложный характер, в том числе иметь вид обобщенных описаний. Система должна быть организована таким образом, чтобы для любого входного воздействия имелась заранее заготовленная «полочка», чтобы все укладывалось на свои места.

Системная ориентированность. Это вторая особенность исследований. Основной их целью является создание средств, необходимых для обеспечения автономной работы системы. Конечно, разработчик все должен знать об этих средствах. Однако пользователю о них может быть ничего неизвестно. Общение пользователя с системой должно идти на внешнем, удобном ему языке. Система должна поддерживать и расширять свои языковые возможности, предоставляя все больший комплекс удобств пользователю. Думать, что разрабатываемые средства, призванные решать широкий класс задач, могут быть использованы в качестве инструментария различными исследователями для моделирования их объектов, не совсем правильно. Внутренний язык должен быть таким, чтобы все представлять и хранить в явном виде, т. е. расшифровывать. Как станет ясно в дальнейшем, семантические сети, представляющие даже сравнительно простые объекты, содержат достаточно много фрагментов, которые человеку (в силу характера его мышления) трудно охватить. Наоборот, человеку свойственно многое умалчивать, по возможности сокращать количество передаваемых слов и предложений за счет использования различных синтаксических конструкций и т. д. Отсюда ясно, что удобные ему конструкции (языки) должны быть совсем другими.

Универсальность, единство. Третья особенность состоит в универсальности создаваемого аппарата при сохранении условия его единства. Опыт построения многих интеллектуальных систем показывает, что стоит взяться за какую-либо задачу, достаточно узкую, и выясняется, что при ее качественном решении она тянет за собой множество других задач — производных. Последние зачастую оказываются сложнее основной задачи.

Например, возьмем системы распознавания образов. Ясно, что проектируются они для кого-то и что степень доверия к ним людей, которые их используют, должна быть высокой. Отсюда следует, что в перспективе помимо задач обучения (формирования решающих правил по обучающей выборке) и распознавания система должна решать и другие задачи — доступа человека к системной информации. Человек должен иметь возможность узнавать, какие правила использует система для распознавания тех или иных классов объектов. Более того, пока еще

человеческие представления о многих классах объектов оказываются значительно более точными и полезными, чем системные. Человек (исследователь) должен иметь возможность вводить такие представления или эвристики, которым система должна следовать. Возникает задача согласования вводимых человеческих эвристик и системных решающих правил. Такое согласование значительно легче достигается при использовании единых представлений или единого внутреннего языка их записи. Условие единства непосредственно связано со свойством универсальности, степень которой должна повышаться по мере возрастания требований к функциональным возможностям системы. Если, например, система допускает ввод сообщений на (ограниченном) ЕЯ, то возникают дополнительные требования к внутреннему языку — он должен содержать необходимое число базовых элементов и конструкций, чтобы допускать отображение естественно-языковой информации, ее сути или смысла. И чем шире язык, тем таких элементов и конструкций должно быть больше.

Все сказанное справедливо не только для систем распознавания образов, но точно в такой же степени и для систем принятия решений, информационного обслуживания и др. Например, в системах, отвечающих на сложные вопросы, требуется преобразование представлений, порождение «новой» информации, т. е. логический вывод. Для обеспечения процесса понимания сообщений (перевода на внутренний системный язык) требуется организация соответствующей активности системы, необходимой для накопления знаний о языке, о предметной области. Аналогичная активность требуется в процессе поиска ответа. Качественная реализация упомянутых видов деятельности невозможна без других действий: обобщения, проверки правильности информации, принятия решений. В ряде случаев может потребоваться формирование стратегий, планирование и др. Подобных примеров очень много. Реализация подобных «взаимообусловленных» видов деятельности порождает все новые требования к внутреннему языку, его изобразительным возможностям. Прежде всего имеются в виду возможности представления структурной и обобщенной информации.

Единство и универсальность представлений непосредственно связаны с унификацией системных процедур или разработки единых механизмов обработки. Чем больше различных языков и представлений, тем больше видимых задач и процедур их решения. И наоборот, при разработке единых механизмов, единой базы решения число этих задач и процедур будет постоянно уменьшаться. Возможность такой разработки в какой-то степени следует из факта сводимости задач. Сравнительно давно подмечено, что распознавание сводится к принятию решений, к обнаружению закономерностей [12]. Перевод можно рассматривать как задачу принятия решения — какую структуру внутреннего языка следует выбрать на основе исходной ситуации (поступившего на вход предложения). Подобных примеров множество. Наиболее перспективный путь унификации задач и методов — переход на структурный уровень. В этом случае возможны единая постановка перечисленных выше задач и соответственно создание достаточно универсальных средств решения [24].

Цельность. Четвертая особенность исследований определяется характером результирующего продукта — цельностью создаваемого логического базиса. Исторически так сложилось, что многие направления области искусственного интеллекта начинали развиваться более или менее самостоятельно, формируя собственный базис. Так возникла логика предикатов, формальные грамматики, семантические сети различных видов, фреймы, расплывчатые логики, модели концептуальной зависимости [13, 53]. За последнее время появились идеи фокусных пространств [60] и многое другое. Конечно, каждый из перечисленных аппаратов отражает определенные стороны интеллектуальной деятельности человека. Их разработка основывалась на реальных примерах. Однако методика создания единого системного базиса путем какого-то комбинирования этих аппаратов, их «сшивания» представляется не совсем оправданной. Например, появились семантические сети — взять их в качестве основы, появились фреймы — добавить их к сетям, появилась идея фокусных пространств — поделить и то и другое на такие пространства и т. д. Конечно, пример довольно схематичен. Но ясно, что, пользуясь подобным приемом, нельзя достичь необходимого единства и цельности. Если создается цельный базис и на основе его организуется гипотетическая система, то в ней заведомо должны быть предусмотрены соответствующие возможности. Влияние должны оказывать лишь примеры, для которых должно находиться соответствующее место в рамках системы и ее средств.

Сводимость, умозрительное оперирование. Пятая особенность следует из только что рассмотренной и заключается в условии сводимости разрабатываемого логического базиса к существующим. Ясно, что если два различных базиса лежат в основе систем, которые должны решать одни и те же задачи (или срабатывать на одних и тех же примерах), то должно выполняться условие сводимости базисов. Если разрабатывается аппарат, предназначенный для реализации широкого круга интеллектуальных задач, то в него должна каким-то образом вкладываться логика предикатов, формальные грамматики, язык реляционной алгебры и др. Должны в каком-то виде присутствовать семантические падежи Филмора, нечеткости Заде и т. д. Словом, все формализмы, возникшие на основе исследования естественных рассуждений, должны естественным образом (как частные случаи) укладываться в рамки разрабатываемого аппарата. Итак, изучение указанных вопросов — важное звено исследований, необходимое для обоснования возможностей разрабатываемых средств и для переноса результатов, полученных в одной области, на другую.

Парадигма обучения. Шестая особенность исследований связана с необходимостью предварительной разработки специальной схемной организации, допускающей в достаточных пределах постоянное расширение системных возможностей. Словом, требуется умелый выбор парадигмы. Дело в том, что большинство реальных объектов (естественных и конструктивных), с которыми человек имеет дело, обладает огромной вариабельностью, т. е. допускает множество разнообразных проявлений. Например, даже такие, казалось бы, простые объекты, как буквы и цифры, имеют такое большое количество способов написания, что учесть их все и каким-то образом формализовать, связав с соответствующим классом, просто не представляется возможным. Отсюда вытекает необходимость организации обучения системы, которую давно уже поняли в области распознавания образов. Система по обучающей выборке сама должна строить решающие правила, алгоритмы.

Как уже говорилось ранее, весьма желательно, чтобы подобные правила могли также вводиться и корректироваться извне. Отсюда мы приходим

к схеме, содержащей наборы структур внутреннего языка, которые в режиме обучения формируются самой системой и однозначно определяют ее реакцию, т. е. ее решения. Помимо этого, допускается доступ человека к таким структурам на удобном ему внешнем языке. Подобную схему в дальнейшем будем называть системой семантической памятью (СП), где под СП понимается некое хранилище структур [1]. Подобные системы более подробно будут рассмотрены в следующем параграфе.

Высокой степенью вариабельности обладают и более сложные объекты — разного рода события, ситуации, а также языковые конструкции. Многочисленные попытки учесть все особенности ЕЯ и вложить в какой-либо алгоритм не увенчались успехом. Следовательно, необходимы средства настройки. Настройка только через словари словоформ и таблицы окончаний явно недостаточна. Система должна уметь настраиваться на различные виды синтаксического и семантического анализа. Учесть подобные виды анализа процедурным путем практически невозможно.

Однородность языка. Итак, СП — это хранилище разного рода структур (знаний), которые (без каких-либо ограничений) должны постоянно накапливаться, корректироваться, изменяться. Сравнительно просто это может быть обеспечено в том случае, если внутренний язык (язык записи системных знаний) допускает достаточно произвольное изменение своих конструкций. Известно, чем сложнее синтаксис языка, тем труднее корректировать его выражения. Поэтому весьма желательно, чтобы язык имел как можно более простой синтаксис, чтобы не требовалось специальных операций по анализу синтаксического оформления, чтобы можно было выражать суть сразу в явном виде. Отсюда приходим к идее удаления из языка таких синтаксических категорий, которые необходимы человеку для правильного «прочтения» выражений, т. е. к идее удаления разного рода скобок, знаков препинания и т. д. Должны оставаться только такие категории, которые естественным образом сочетаются с работой системных механизмов, а в перспективе — с элементной базой, используемой для реализации. Желательно, чтобы внутренний язык состоял из как можно более однотипных конструкций (кирпичиков), которые можно было бы изымать и добавлять достаточно произвольным образом, к примеру без дополнительного анализа количества открывающихся или закрывающихся скобок и их содержимого. Подобные соображения легли в основу понятия однородности. Однако требование однородности ни в коем случае не относится к внешним языкам, в том числе ЕЯ. Их высокая синтаксическая загруженность служит компактной упаковкой структур знаний в линейные конструкции, передаваемые и воспринимаемые соответствующими органами человека.

§ 1.2. ОСНОВЫ СЕМАНТИЧЕСКОГО ЯЗЫКА

При организации систем, ориентированных на обработку поступающей от человека информации, возникают естественные вопросы. Нужно ли различные сорта описаний представлять на различных внутренних языках? Должны ли системные знания быть различными по своей природе? Как выясняется, это чрезвычайно неудобно. Дело в том, что все знания

[1] Имеется в виду база знаний у определенного сорта интеллектуального автомата.

оказываются сильно связанными между собой, одни непосредственно вкрапливаются в недра других. Так, для правильного перевода предложений с внешнего (естественного) языка на внутренний необходимо учитывать тематику, тип пользователя, т. е. в знаниях о внешнем языке должны присутствовать знания о предметной области и соответствующей «модели» пользователей. Иными словами, лингвистические знания оказываются связанными с предметными. Более того, для обеспечения правильной реакции системы зачастую требуется учет сценария, в рамках которого формируется реакция. На способ реагирования могут влиять внешние факторы и многое другое. Следовательно, необходима запись различных сортов знаний в рамках единого языка. Системы с СП (базы знаний) должны иметь единый внутренний язык.

Требование к средствам. Вначале остановимся на вопросе, какие возможности в перспективе должны быть реализованы в рамках систем с СП. Ясно, что подобные возможности должны в какой-то степени сочетаться с особенностями, характерными для привычных форм общения. Одна из таких особенностей — постоянное расширение знаний людей о внешнем (естественном) языке за счет информации, выражаемой на этом же языке. При внимательном рассмотрении становится ясно, что такое расширение далеко выходит за рамки обычных определений и макроопределений, нашедших применение в искусственных языках, например в языках программирования — для ввода новых идентификаторов, операторов, описателей и т. д. Человек широко использует разного рода объяснения (в том числе с привлечением поясняющих примеров), аналогии и др. Именно в этом направлении должны развиваться лингвистические процессоры. Тогда можно исключить участие разработчика в формировании и накоплении лингвистических знаний, необходимых для общения системы с пользователем. Это сможет (и должен) делать сам пользователь. Чтобы добиться указанных возможностей, в базовое ядро системы должны быть включены средства, обеспечивающие выявление новых сведений лингвистического характера, их перенос, обобщение и многое другое.

Все сказанное в большой степени справедливо и тогда, когда речь идет о человеческих умениях. Такие умения накапливаются не только за счет освоения новых алгоритмов. Люди крайне редко изъясняются алгоритмическими выражениями, определяющими процедуру мышления. Как правило, умение складывается за счет опыта. При этом немаловажную роль играют сведения, извлекаемые из естественно-языковых выражений. Способы такого извлечения должны быть учтены при разработке системных механизмов или базового ядра. Тогда становится возможным объединение разработчика и пользователя, т. е. пользователь (без специального обучения) сможет выполнять функции разработчика. Более того, пользователю уже не потребуется специальных знаний о том, что делается внутри системы. Расширение системных знаний и умений будет осуществляться естественным для него образом.

Конечно, для реализации указанных возможностей еще требуются специальные исследования, прежде всего, естественного языка, его семантики и прагматики. Требуется выявление средств и конструкций, вызывающих совершенствование человеческих знаний и умений. Потребуется исследование операционного характера человеческих высказываний, их влияния на механизм мышления и др. В перспективе системы с СП (базы

знаний) должны строиться таким образом, чтобы для каждого человеческого высказывания можно было найти его место в СП. Более того, если человек использует данное высказывание для каких-либо целей, то аналогичные виды деятельности должны быть реализованы с помощью системных механизмов. Словом, должна иметь место так называемая относительная интерпретируемость [21] естественного языка (ЕЯ) в том формальном базисе, на котором основана система.

Важным обстоятельством является вопрос о выделении базового ядра (умений, механизмов), которое в рамках указанной организации обеспечивало бы поддержку и постоянное расширение системных функций и возможностей (А. С. Нариньяни связывает этот вопрос с разработкой так называемой базовой семантической машины). Причем расширение должно осуществляться как можно в более естественных формах, т. е. не за счет написания новых алгоритмов и программ, а за счет вводимой информации — предложений типа *Каждый и тот, и другой, и третий,* и т. д.

Согласованность. Обсуждаемая схема предъявляет сравнительно сильные требования к языку представления информации в СП. Уже говорилось о том, что, с одной стороны, внутренний язык должен обладать достаточно простым синтаксисом, быть однородным. С другой стороны, он должен обеспечивать представление различных видов информации, в том числе обобщенной (с оттенками сомнения или без них) и т. д. Наконец, язык должен удовлетворять еще одному важному требованию. Он должен, что называется, быть согласованным с внешним языком — ЕЯ. Это необходимо с точки зрения преобразования конструкций внешнего языка во внутренний (становится возможным использование продукционных систем, обеспечивающих последовательное «осмысление» компонент). Что же такое согласованность? Имеется в виду соотнесенность простых смысловых единиц (и более сложных компонент) ЕЯ с простыми элементами (более сложными конструкциями) внутреннего языка. Словам и словосочетаниям ЕЯ, имеющим достаточно самостоятельный смысл вне контекста, должны соответствовать базовые элементы или простые конструкции внутреннего языка. Если из таких слов слагаются более сложные осмысленные компоненты (словосочетания, отдельные предложения, имеющие самостоятельный смысл), то последним должны соответствовать более сложные конструкции, составленные из базовых. Например, в предложении *Король Франции лысый* всем трем словам должны соответствовать свои базовые элементы внутреннего языка. Словосочетанию *король Франции* также должна сопоставляться конструкция, составленная из двух предыдущих элементов. Из всех таких элементов и указанной конструкции должна составляться более сложная конструкция, соответствующая всему предложению.

Итак, согласованность предполагает, что если предложению сопоставлена некоторая правильная конструкция внутреннего языка, то любой осмысленной части предложения должна соответствовать также правильная конструкция, являющаяся частью первой. Самостоятельным элементам предложения (т. е. словам, словосочетаниям, имеющим достаточно самостоятельный смысл вне контекста) должны соответствовать базовые элементы внутреннего языка.

Отметим два важных момента. Не обязательно, чтобы в конструкции внутреннего языка было ровно столько базовых элементов, сколько слов в предложении. Дело в том, что в естественном языке многое подразумевается, дается по умолчанию. Подразумеваемые компоненты должны восстанавливаться в процессе ввода предложений, их преобразования во внутреннее представление, где все должно фигурировать в явном виде. Например, в предложении *шапка на меху* подразумевается, что шапка состоит из меха. В соответствующую конструкцию внутреннего языка должен быть введен элемент, соответствующий отношению *состоять из,* т. е. обеспечивается выявление подразумеваемого, представление в явном виде.

И второй момент — элементы и конструкции внутреннего языка должны сопоставляться не словам или словосочетаниям, а их смыслам или денотатам. Так слову *шапка* должен сопоставляться базовый элемент, соответствующий предмету — шапке. Система может знать или не знать, какая конкретная шапка имеется в виду, что должно быть соответствующим образом отражено во внутреннем представлении. Здесь важно, что речь идет о каком-то предмете, относящемуся к классу шапок. Когда мы говорили об осмысленных единицах и частях предложения, то имели в виду подобную предметную соотнесенность.

При выполнении условия согласованности облегчается процедура ввода и корректировки. Делается возможным последовательный анализ предложения, начиная с выявления наиболее простых осмысленных единиц и перехода ко все более сложным. Такой анализ определяет процесс последовательного составления конструкций внутреннего языка. Более того, допускаются корректирующие предложения типа *Не король, а принц, не лысый, а курчавый* и т. д. В них имеются осмысленные компоненты, по которым всегда можно выделить и изменить соответствующие части в уже построенной конструкции.

Итак, условие согласованности необходимо прежде всего с точки зрения организации работы механизмов перевода и корректировки входной информации. При этом предполагается возможность ввода в язык необходимого числа базовых элементов и конструкций. Имеется в виду свойство расширяемости. Перейдем к рассмотрению языка, который удовлетворяет всем перечисленным требованиям.

Итак, свойство однородности и согласованности внутреннего языка должно сочетаться с его высокими изобразительными возможностями. Как нетрудно показать, существующие языки в должной степени не обладают этими свойствами. Языки программирования имеют достаточно сложный синтаксис и (за редким исключением) не предусматривают данных структурного характера. Язык логики предикатов также нельзя считать однородным. О его недостатках, ограниченных возможностях, затрудняющих использование в качестве декларативных знаний, писалось достаточно много [2, 9]. Еще более ограниченными возможностями обладают языки формальных грамматик, допускающие лишь фиксированные виды анализа. Подобный перечень можно было бы продолжить, включить сюда фреймовые языки (с операционными вставками), *ATN*-грамматики, различные математические языки. Конечно, в корне неверно думать, что такие языки вообще не следует использовать в интеллектуальных системах. Но такое использование предполагает уже другую парадигму — когда знания конструируются человеком, т. е. человек постоянно сопровождает систему, зная обо всем, что в ней происходит. Тогда важно иметь хороший инструментарий для человека (разработчика).

И такой инструментарий может иметь формы одного их перечисленных (внешних) языков. Внутренний же язык должен быть устроен по-своему.

Предыстория. Общая структура внутреннего системного языка стала ясна в 1974 г. [17, 23]. В основу языка был положен специальный класс семантических сетей с вершинами связи (в ранних работах они назывались *Т*-графами, так как понятие «семантическая сеть» было еще не устоявшимся). Семантические сети были выбраны вследствие их прекрасного свойства однородности. Однако обычные сети, состоящие из вершин-понятий и связывающих их дуг, как оказалось, ограниченны с точки зрения возможностей представления, в частности обобщенной информации, отношений между ситуациями, самими отношениями и т. д. В связи с этим в них были введены необходимые наборы вершин-понятий, а также «развязывающие элементы» — вершины связи, стоящие на местах дуг и как бы разрывающие эти дуги.

В зарубежной литературе сети с вершинами связи впервые описаны в работе Церкона и Шуберта [58] и в дальнейшем развиты Шубертом [63]. Они были названы «сетями с пропозициональными вершинами». Слово «пропозициональный» представляется не совсем удачным в силу несоответствия его смысла аналогичному слову в логике высказываний. Поэтому в дальнейшем будем следовать своей терминологии. Однако в сетях Шуберта еще не было четкого деления вершин-понятий на классы и не обеспечивалась достаточная однородность представления логических и обобщенных структур. То же самое можно сказать и о сетях Хендрикса [50], Ригера [62]. Совершенно в стороне стоят сети Шенка [53]. С нашей точки зрения, при их разработке не совсем удалось справиться с задачей однородного представления логической и сложноструктурированной информации (возможно, такая задача и не ставилась). Не будем останавливаться на подробном описании упомянутых сетей. Интересующимся дополнительно можно предложить работы [39, 40, 57]. Перейдем к краткому изложению основных черт используемых нами сетей. Более подробно такие сети будут описаны в § 1.3, 1.4, 2.1 и 2.2.

Сети с вершинами связи. Напомним, что обычные семантические сети состоят из вершин, соответствующих объектам или понятиям, а также дуг, соответствующих отношениям и связывающих эти вершины. В рассматриваемых нами сетях вершины могут соответствовать не только объектам или понятиям, но и отношениям, логическим составляющим информации (фактам истинности, лжи), комплексным объектам и др. Всему, что может рассматриваться как самостоятельная единица, должна быть сопоставлена собственная вершина. Например, вершины могут сопоставляться законченным событиям или ситуациям. Вершины делятся на два класса: определенные (*о*-вершины) и неопределенные (*н*-вершины). Первые соответствуют узнанным объектам, выявленным отношениям, распознанным событиям, ситуациям, а вторые — неузнанным, невыявленным.

Помимо указанных, вводятся вершины совсем другого типа — вершины связи. Они соединяются помеченными ребрами (ребрами различных типов) с вершинами, взятыми из множества упомянутых выше вершин. Фактически ребра метятся цифрами, определяющими семантический падеж отношения. В результате образуется фрагмент, соответствующий элементарной ситуации, т. е. объектам, связанным отношением. Такой фрагмент называется элементарным.

Элементарный фрагмент можно представить в виде паука с помеченными лапками. При этом тело такого паука есть вершина связи, а лапки — ребра. Последними он как бы цепляется за другие вершины. Номер или тип лапки определяет роль, которую играют схваченные им вершины в представленной ситуации, т. е. есть ли это вершина-объект, вершина-отношение, вершина, соответствующая факту *истинности—лжи*, или вершина, соответствующая всей элементарной ситуации. Специальное деление перечисленных выше вершин на непересекающиеся множества не производится. Каждая из них может играть любую роль. Словом, ситуация или логические составляющие могут быть связаны своими отношениями, отношения также могут быть объектами другого отношения и т. д. В результате обеспечиваются широкие возможности представления.

Таким образом, в описываемых сетях вместо дуг обычных семантических сетей используются паукообразные фрагменты, в середине которых находится вершина связи. Последние играют роль развязывающих элементов. Они обеспечивают равнозначность вершин, соответствующих отдельным компонентам или единицам информации. Все они могут быть связаны отношениями, т. е. паукообразными фрагментами. Из последних составляются сети, которые имеют вид множества пауков, сцепившихся помеченными лапками (ребрами).

Помимо указанных фрагментов, вводятся средства еще одного вида, служащие для представления наборов перечисленных объектов (множеств), для записи найденных неопределенных компонент и др. Такие средства имеют вид пары, где первой компонентой является *н*-вершина, соответствующая неопределенной компоненте, а второй — множество конкретизирующих *о*-вершин, соответствующих найденным компонентам. Подобное множество называется значениями *н*-вершин, а сама пара — *з*-сетью. В § 1.3 и 2.2 будет дано более детальное описание сетей и *з*-сетей, которые вводятся в рамках единого языка, названного семантическим языком (СЯ). Будут описаны более сложные *з*-сети, задающие значения *п*-ок *н*-вершин. Например, с помощью таких *з*-сетей могут быть представлены и перечислены всевозможные варианты пар *поставщик—поставляемая деталь*, *поставщик—место поставки—поставляемая деталь* и т. д. Конечно, подобную информацию можно записать и с помощью общепринятых таблиц, используемых в реляционных базах данных (см. § 2.3). Однако с точки зрения обеспечения однородных представлений удобнее использование *з*-сетей.

Графы. Возьмем предложения *Каждый человек смертен*, *Только те, кто смертны, могут быть людьми*, *Если имеется человек, то он должен быть смертным*, *Никто не является бессмертным*. С точки зрения логики предикатов они выражают одно и то же, т. е. представляются с помощью одного и того же предикатного выражения. Однако некоторое смысловое отличие в них ощущается. И это прежде всего акцентация ударений, оттенки долженствований, изменение так называемых тем и рем. Для представления подобных отличий в дальнейшем будем использовать специальные конструкции, называемые семантическими графами [36]. С их помощью представляются такие же отношения, что и с помощью сетей, но дополнительно учитывается операционная семантика форм ЕЯ — какую компоненту нужно искать вначале, какую затем и т. д. Каждый граф задает собственные операции, которые выполняются над исходной сетью и приводят

к последовательному нахождению значений *н*-вершины графа, т. е. к уточнению представленных в графе неопределенных компонент. Например, вопрос *Сколько площадей в Москве*? представляется с помощью графа, задающего последовательное выполнение двух операций: с помощью первой ищутся все площади Москвы, а второй — подсчитывается их количество. Поиск осуществляется с использованием системных знаний, т. е. сети, с помощью которой представлена информация о г. Москве. Если сказать более точно, то ищутся, конечно, не площади, а вершины сети, соответствующие этим площадям. Словом, операции, которые задаются графом, выполняются над сетью и приводят к нахождению или выделению множеств ее вершин.

Приведенный пример намеренно упрощен. В общем случае графы могут задавать различные операции перебора (с объединением результатов, формированием альтернативного множества и пр.), а также операции теоретико-множественного пересечения, объединения, проверки включения множеств, их непустого пересечения и т. д. (см. гл. 4). Например, предложение *Каждый человек смертен* может быть представлено с помощью графа, задающего операции поиска двух множеств — первое составлено из тех, кто смертен, а второе — из тех, кто является людьми. Далее проверяется теоретико-множественное включение второго множества в первое.

Подобные операции вводятся в рамках специального языка, названного операторным языком (ОЯ). При этом каждый граф задает собственные операции, которые в общем-то могут выполняться над любой сетью. Например, графы, соответствующие утверждениям типа *Каждый поставщик должен поставлять хотя бы одну деталь*, *В каждом городе должны быть свои поставщики*, ...задают операции, которые выполняются над сетью, представляющей входную информацию — описание новых поставщиков или поставок. В результате обеспечивается проверка полноты, правильности или допустимости таких описаний. Графы объединяются в наборы, где дополнительно указывается порядок их применения или стратегия применения. В результате образуются так называемые обязательные знания, играющие важную роль — роль семантических фильтров. С помощью обязательных знаний обеспечивается решение множества задач, связанных с распознаванием объектов (по их структурным описаниям), с прогнозированием, с дополнением подразумеваемой информации и т. д. (см. § 5.3).

Остановимся на нескольких важных моментах. Подход, выбранный нами, очень близок к подходу, используемому в реляционных базах данных, где вначале запрос преобразуется в операции реляционной алгебры, которые выполняются, в результате чего находится ответ [10, 47]. Отличие заключается в том, что операции, задаваемые графами, это операции не над таблицами («отношениями»), а над сетью и промежуточными результатами — множествами вершин. Используемый нами набор операций (входящих в ОЯ) значительно шире, чем в реляционной алгебре, т. е. выходит за ее рамки. Более подробно эти вопросы будут рассмотрены в § 4.1.

Граф есть более сложный формальный объект чем сеть. Сеть можно считать частным случаем графа, т. е. сеть есть граф, в котором не задано направления обработки (например, такое направление не имеет значения и можно использовать любое из возможных). Более того, и граф может быть представлен в виде сети, в которой с помощью специальных фрагментов указан порядок использования других фрагментов — представляющих

содержательную информацию. В дальнейшем будет удобно пользоваться как понятием сети, так и графа.

Отметим, что одним из наиболее привлекательных свойств сетей и графов является их однородность. Как станет ясно ниже, они составляются из однотипных фрагментов весьма простым способом. Каждый из фрагментов может быть изъят из сети. Любой другой может быть добавлен к ней без нарушения синтаксической правильности и осмысленности конструкций. Этим обеспечивается необходимая гибкость. Чрезвычайно простой синтаксис языка сетей и графов предопределяет сравнительно несложные механизмы выполнения указанных действий. И в тоже время с помощью сетей и графов может быть представлена информация чрезвычайно высокой степени структурированности и сложности. Однотипность представления предопределяет возможность использования одних и тех же способов для обработки информации, представляющей объекты, ситуации, логические связи и др. Словом, открывается путь к вопросам унификации механизмов.

Механизмы. Уже говорилось, что обработка основывается на принципе наложения сетей, последовательного сопоставления их фрагментов. Процедура такого сопоставления определяется графами, т. е. задаваемыми ими операциями. В результате находятся неопределенные компоненты информации. В отличие от обычного сопоставления таблиц или образцов (фреймов) при наложении сетей используется более сложный подход — окрестностный. Для поиска неопределенных компонент используются их окрестности. При этом приходится постоянно искать сопоставляемые компоненты, выбирать направление поиска. Все это делает процедуру наложения достаточно сложной, хотя и в большей степени универсальной.

Указанные принципы легли в основу механизмов, обеспечивающих обработку информации, реализацию различных видов деятельности. Обработка в многих случаях сводится к формированию графов и выполнению операций, задаваемых графами. Таким способом обеспечивается решение двух основных задач — конкретизации и преобразования. Первая задача заключается в нахождении неопределенных составляющих входной информации, в выполнении разнообразных проверок, а вторая задача — в преобразовании информации. Преобразование управляется с помощью специальных средств, называемых сетевыми продукциями. Такие средства можно считать некой разновидностью графов, задающих специальные операции поиска сетей определенных конструкций, их изъятия и замены на другие сети. С помощью таких продукций представляются различные определения, а также некоторые виды условных предложений, см. § 2.1 и 5.3.

Предложение *Если имеется человек, то он смертен* представляется с помощью продукции, где в левой части представлена принадлежность к классу людей, а в правой — свойство *быть смертным*. Подобная продукция может быть применена к любой входной информации. И если в ней речь идет о каких-либо людях, то будет добавлено их свойство — смертность.

Итак, сетевые продукции это некоторые «расширенные» правила подстановки, добавления, имеющие вид сетей и выполняемые над сетями. При этом в продукциях (с помощью *н*-вершин) могут быть указаны места произвольного заполнения.

§ 1.3. СЕМАНТИЧЕСКИЕ СЕТИ

В данном параграфе будет начато рассмотрение языка представления знаний — семантического (СЯ). Как уже говорилось в § 1.1, под СЯ понимаются конструкции определенного вида, состоящие из однотипных элементов — вершин. Из последних составляются фрагменты и сети, которые в свою очередь используются для построения более сложных конструкций — графов. В дальнейшем речь будет идти об одном классе сетей и графов, так называемых семантических. При этом будут использоваться семантические сети определенного вида — с вершинами связи [14, 23]. Напомним, что под семантическим графом понимаются сети, на которых задан порядок обработки их фрагментов, так называемый порядок конкретизации.

Вершины. СЯ служит для представления предметных областей, в которых выделяются объекты $\{A_i\}$ и отношения $\{R_j\}$. Будем считать, что $\{A_i\}$ и $\{R_j\}$ включены в множество компонент $\{D_k\}$, различаемых в данной предметной области. Под компонентами будем понимать не только объекты и отношения, но и составленные из них комплексные объекты и различные ситуации (рассматриваемые как единое целое). Компонентами могут быть и логические составляющие: истинности, лжи.

Компонентам сопоставляются элементы одного и того же набора — вершины. Они еще называются вершинами-понятиями. Будем обозначать вершины, соответствующие объектам A_i, отношениям R_j или произвольным компонентам D_k, через a_i, r_j, d_k, т. е. аналогичными малыми латинскими буквами с индексами.

Обозначим все множество вершин-понятий (т. е. соответствующих различным компонентам) через D. Оно включает в себя два подмножества: A и R, где элементы из A соответствуют объектам, а из R — отношениям. Здесь для обозначения множеств использованы заглавные латинские буквы без индексов, которые не следует путать с обозначениями объектов и отношений (где использованы заглавные латинские буквы с индексами).

Как показывает опыт, стандартизировать наборы объектов и отношений в реальных предметных областях оказывается чрезвычайно трудно. Поэтому в дальнейшем будем считать множества A и R расширяющимися — допускающими пополнение все новыми, удобными для представления, вершинами.

Элементарные фрагменты. Деление на объекты и отношения является в некотором смысле условным. Отношения могут рассматриваться как объекты, связанные своими отношениями. Объекты могут указывать на тип отношения. В дальнейшем не будем стараться проводить четкое разграничение. Будем считать существенной роль вершин, которую будем задавать с помощью кортежей $\langle d_1, \ldots, d_k \rangle$, где $d_1, \ldots, d_k \in D$. Будем называть кортежи элементарными фрагментами (ЭФ). В каждом ЭФ имеется множество вершин, соответствующих объектам, и одна вершина — отношению. Будем отделять эту последнюю вершину, назначив ей определенное место в кортеже — третье. Последующие места — с четвертого по k-е — занимаются вершинами-объектами.

Таким образом, кортеж $\langle d_1, \ldots, d_k \rangle$ представляет отношение типа D_3 между объектами $D_4, \ldots, D_k$. Первые два места имеют специальное назначение. Если указанные объекты с их отношением рассматриваются как единый комплексный объект или ситуация (D_1), то последнему сопоставляется собственная вершина d_1, которая занимает первое место. Второе место занимает вершина d_2, соответствующая логической составляющей D_2, например указывающей на истинность или ложность представленного отношения. Для подобного указания будем использовать собственные вершины: t соответствует факту *истинности*, а f — *лжи*.

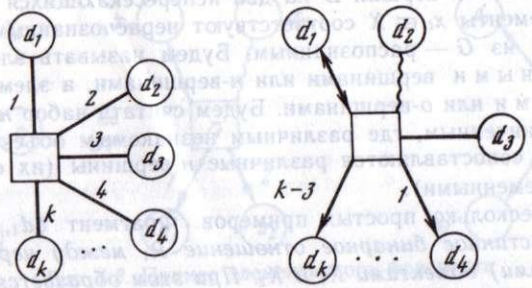


Рис. 1.1. Элементарный фрагмент (ЭФ)

Рис. 1.2. Другой способ изображения ЭФ

В дальнейшем будем допускать возможность нахождения на первом месте достаточно произвольных элементов из множества A. Каждый такой элемент может соответствовать комплексному объекту. На других местах вершин-объектов могут находиться элементы из A и R, в том числе вершины-отношения. В то же время второе и третье места всегда будем оставлять только за вершинами, соответствующими логическим составляющим, и за вершинами-отношениями.

Будем изображать фрагмент $\langle d_1, \ldots, d_k \rangle$ в виде сети рис. 1.1, где квадратиком изображена вершина связи. Она не соответствует ни объектам, ни отношениям и носит чисто служебный характер, т. е. используется для указания связи. Роли вершин задаются нумерацией ребер. В дальнейшем будем вместо нумерации использовать различные типы ребер (рис. 1.2), где стрелки служат только для задания таких типов. Ребро $\leftrightarrow$ присоединяется к вершине, которая в ЭФ стоит на первом месте, ребро $\sim\sim$ — на втором, ребро без стрелок — на третьем, а ребра $\overset{i}{\rightarrow}$ — на последующих. При этом цифры $i = 1, \ldots, k$ указывают на относительные места последующих вершин в ЭФ, что будет служить заданием роли соответствующих объектов в отношении.

Известно, что в различных отношениях могут участвовать различные компоненты — субъекты, объекты и др. Отношения могут иметь различную местность, что будем представлять с помощью кортежей ЭФ различной длины. N-местным отношениям соответствуют кортежи $N + 3$-местности. Местность кортежей определяется количеством семантических падежей типа *быть субъектом*, *быть действием* и т. д., характеризующих компоненты отношения.

***H*-вершины и *o*-вершины.** Будем различать объекты (отношения, логические составляющие и прочие компоненты) двух типов: известные (выявленные) или определенные и неизвестные (невыявленные) или неопределенные. Последние образуют множество $\{X_i\}$. В соответствии со сказанным будем

делить все множество вершин D на два непересекающихся подмножества X и G, где элементы $x_i \in X$ соответствуют нераспознанным компонентам X_i, а элементы из G — распознанным. Будем называть элементы X н е о п р е д е л е н н ы м и вершинами или *н*-вершинами, а элементы G — о п р е д е л е н н ы м и или *о*-вершинами. Будем считать набор *н*-вершин практически неограниченным, где различным незнакомым объектам или отношениям всегда сопоставляются различные *н*-вершины (их еще называют вершинами-переменными).

Приведем несколько простых примеров. Фрагмент $\langle d_1, t, r_1, x_1, x_2 \rangle$ *представляет истинное бинарное отношение* R_1 *между нераспознанными (неопределенными) объектами* X_1 *и* X_2. *При этом образуется комплексный объект* D_1. Фрагмент $\langle x_1, x_2, r_1, a_1, a_2 \rangle$ представляет *отношение* R_1 *между объектами* A_1 *и* A_2, *для которого не ясно, истинно оно или ложно, а также не известно, какой же образуется комплексный объект*. Подобные фрагменты изображены на рис. 1.3 и 1.4, где обозначения *н*-вершин вынесены за границы кружочков, а обозначения *о*-вершин находятся внутри их.

В дальнейшем будем пользоваться **сокращенным изображением** фрагментов. Во-первых, не будем рисовать *о*-вершину t с ее ребрами. Будем считать, что если к вершине связи не подсоединены ребра вида ~~~, то с помощью соответствующего фрагмента представляются истинные (t) отношения. Это позволит значительно упростить рисунок. В самом деле, очень редко говорится о том, что является ложным или чего нет. Во-вторых, фрагмент, представляющий заданное отношение (т. е. на третьем месте ЭФ стоит *о*-вершина), будем изображать более простым способом: обозначение такой *о*-вершины будем помещать внутрь вершины связи. Это позволит приблизить обсуждаемые сети к общепринятому виду.

С учетом сказанного фрагмент $\langle x_1, x_2, r_1, a_1, a_2 \rangle$ может быть изображен проще, чем фрагмент, изображенный в правой части на рис. 1.4. При этом заметим, что убрать вершину x_2 с ее ребром можно было бы только в том случае, если на ее месте стояла бы *о*-вершина t.

Сети. Для представления ситуаций, состоящих из множеств объектов и отношений, будем пользоваться множествами ЭФ, которые будем называть сетями, а части сетей ф р а г м е н т а м и. Будем считать, что порядок записи ЭФ в сети или место их расположения в соответствующем изображении сети не играет никакой роли. Другими словами, изменение порядка или места никак не влияет на интерпретацию сетей. Будем записывать сети в виде

$$\Phi_1 \circ \Phi_2 \circ \ldots \circ \Phi_n, \qquad (1.1)$$

где Φ_i для $i = 1, \ldots, n$ — есть ЭФ, а «$\circ$» — специальный разделительный знак.

Будем изображать сети в виде набора ЭФ, которые могут содержать одни и те же (общие) вершины. Связь между ЭФ задается только с помощью этих вершин. Например, на рис. 1.5 изображена сеть, представляющая *отношение* R_1 *между объектами* A_1 *и* X_1, *а также между* X_1 *и* X_2. Такая сеть записывается в виде

$$\langle x_3, t, r_1, a_1, x_1 \rangle \circ \langle x_4, t, r_1, x_1, x_2 \rangle,$$

где x_3 и x_4 соответствуют парам объектов, связанных отношением, т. е. совокупностям. В этой сети общими для двух ЭФ являются три вершины — t, r_1, x_1. При этом информационная связь между представленными отношениями осуществляется через *н*-вершину x_1. Имеется в виду зависимость неизвестного объекта X_1 от обоих отношений, а также объекта X_2 от отношения R_1 между A_1 и X_1. Если заменить x_1 на *о*-вершину, то такой зависимости уже не будет.

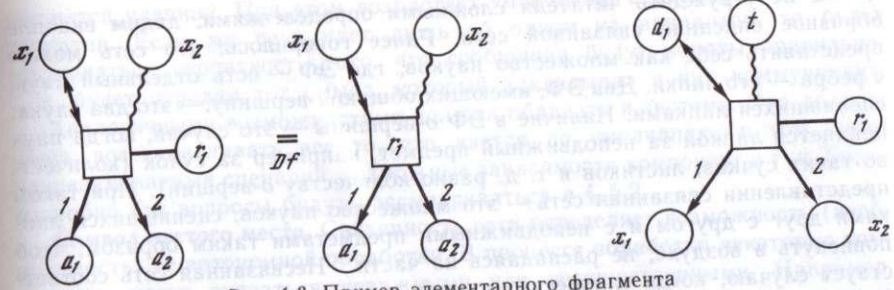


Рис. 1.3. Пример элементарного фрагмента

Рис. 1.4. Сокращенное обозначение ($r_1 \in R$)

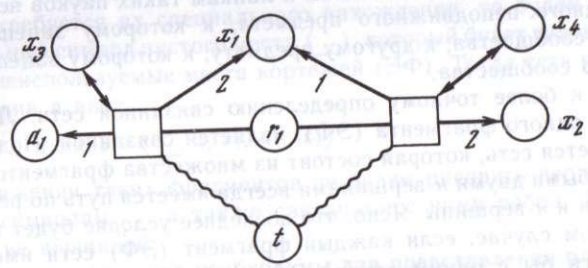


Рис. 1.5. Пример простой сети

Следует оговориться, что далеко не всякая сеть (1.1), составленная из фрагментов, будет представлять правильную или допустимую информацию. Фрагменты могут представлять отношения, которые никак не «стыкуются» один с другим, например отношения *быть братом* (R_1) и *иметь квадратную форму* (R_2). Тогда сеть

$$\langle x_3, t, r_1, a_1, x_1 \rangle \circ \langle x_4, t, r_2, x_1 \rangle$$

будет представлять *Некто* X_1, *являющийся братом* A_1, *имеет квадратную форму*. В дальнейшем будем придерживаться схемы обработки, в которой такие каламбуры выявляются в процессе ввода информации и проверки ее на допустимость, для чего будут использованы так называемые обязательные знания. Их роль будут играть семантические графы (см. § 5.2).

Будем обозначать сети заглавными буквами T_j и указывать, что сеть содержит вершины $d_1, \ldots, d_k$ (не обязательно только их) через $T_j(d_1, \ldots, d_k)$. Будем обозначать соответствующие объекты с их отношениями через $\mathcal{T}_j(D_1, \ldots, D_k)$.

Понятие связанной сети. В основе понятия связанной сети лежит зависимость объектов от их окружения. Отражаются общепринятые представления о связанном вопросе, высказывании, связанном условии задачи и др.

Чтобы не утруждать читателя сложными определениями, дадим вначале образное описание связанной сети. Ранее говорилось, что сеть можно представить себе как множество пауков, где ЭФ — есть отдельный паук, а ребра — его лапки. Два ЭФ, имеющих общую *н*-вершину,— это два паука, сцепившихся лапками. Наличие в ЭФ *о*-вершины — это случай, когда паук цепляется лапкой за неподвижный предмет, например за сучок (количество таких сучков, листиков и т. д. равно количеству *о*-вершин). При таком представлении связанная сеть — это множество пауков, сцепившихся лапками друг с другом и с неподвижными предметами таким образом, чтоб повиснуть в воздухе, не распадаясь на части. Несвязанная сеть соответствует случаю, когда множество пауков делится на сообщества, которые висят независимым образом. Такие сообщества называются несвязанными или независимыми частями. По телам и лапкам таких пауков нельзя перебраться от одного неподвижного предмета, к которому зацеплена лапка паука одного сообщества, к другому предмету, к которому зацеплена лапка паука другого сообщества.

Перейдем к более точному определению связанной сети. Любая сеть, состоящая из одного фрагмента (ЭФ), является связанной. Под связанной также понимается сеть, которая состоит из множества фрагментов и в которой между любыми двумя *н*-вершинами всегда имеется путь по ребрам через вершины связи и *н*-вершины. Ясно, что последнее условие будет выполняться только в том случае, если каждый фрагмент (ЭФ) сети имеет общую *н*-вершину хотя бы с одним другим ЭФ этой же сети. В общем случае такая связь может иметь вид цепочек.

Отметим, что несвязанные сети могут содержать одни и те же *о*-вершины, но не могут иметь одних и тех же *н*-вершин, а также *н*-вершин, связанных цепочками фрагментов через *н*-вершины.

С помощью связанных сетей представляются неделимые (в некотором смысле) куски информации, которые целиком участвуют в обработке. Если сеть не является связанной, то она может быть разбита на самостоятельные (связанные) части, которые можно использовать автономно.

Например, пусть известно, что *имеется отношение R_1 между объектами A_1 и A_2 и требуется найти объект, связанный отношением R_2 с A_2*. Ясно, что в данном случае условие никак не связано с требованием, т. е. условие не может быть использовано для нахождения объекта. Хотя в условии и требовании упоминается один и тот же объект A_2. Аналогично в сообщении *В огороде бузина, а в Киеве дядька* также имеются две независимые части, которые представляются с помощью двух не связанных друг с другом сетей. Анализ правильности такого сообщения сводится к независимой проверке правильности ее частей.

Следует оговориться, что в обсуждаемом сообщении имелись в виду *некие дядька и огород*, никак не связанные друг с другом. Хотя это не всегда так. К примеру, если системе известно, что речь идет о *дядьке, который имеет данный огород, не выносит запаха бузины и поэтому живет в Киеве*, то сеть будет связанной. Пример взят из [12]. Он является прекрасной иллюстрацией одного из важнейших свойств человека — умения связывать, казалось бы, независимые куски информационного материала, используя для этой цели свои знания. Если в речевом акте дается два последовательных сообщения, то всегда делается попытка связать их. Чем больше человек знает, чем больший имеет опыт, тем в больших случаях такая попытка

окажется удачной. При этом возрастает количество вариантов связи, альтернатив. Если же речь идет лишь об одном из вариантов, то будет уменьшаться вероятность того, что сообщения будут поняты правильно, т. е. будет передан тот смысл, который закладывал в них коммуникант.

Аналогичными возможностями должна обладать и система. Она должна уметь восстанавливать все то, что дается по умолчанию, в том числе подразумеваемый сценарий, возможные зависимости компонент и т. п. Более подробно эти вопросы будут рассматриваться в § 5.2.

Символ пустого места. Связанность сети определяет возможность (необходимость) ее автономной обработки. В процессе обработки некоторые компоненты могут оказаться неважными или несущественными. Например, для нахождения объекта X_2, по отношениям, представленным с помощью сети рис. 1.5, несущественными являются комплексные объекты X_3 и X_4. И если не требуется их специального нахождения, то будем сопоставлять им специальный символ пустого места (__), который будет указывать на незанятые или неиспользуемые места кортежей (ЭФ). Тогда сеть рис. 1.5 может быть записана в виде

$$\langle \_\_,\ t,\ r_1,\ a_1,\ x_1\rangle \circ \langle \_\_,\ t,\ r_1,\ x_1,\ x_2\rangle. \qquad (1.2)$$

При изображении таких фрагментов не будем рисовать вершины, соответствующие символам __, а также связанные с ними ребра. Будем считать символ __ не вершиной.

Символы пустого места необходимы для представления различных отношений, действий, для которых задаются не все их компоненты, а только нужные. Человек, как правило, сообщает только то, что является существенным в данный момент. Например, в предложении *Иван взял книгу* не указывается, *откуда взял, когда*. Важно, что в настоящий момент *Иван имеет книгу* (что следует из первого предложения). Пусть действие *взять* рассматривается как отношение типа R_1 с четырьмя семантическими падежами; *кто взял, что взял, откуда взял и когда*. Тогда указанное предложение будет представлено с помощью кортежа $\langle \_\_,\ t,\ r_1,\ a_1,\ a_2,\ \_\_,\_\_\rangle$, где a_1 соответствует конкретному *Ивану* (который имеется в виду), а a_2 — *книге*.

Следует помнить, что символы __, находящиеся в разных местах или в разных кортежах сети, могут соответствовать различным неопределенным объектам (не принимающим участия в обработке). При этом особо будем различать случай, когда символ __ стоит на месте отношения. Будем считать, что кортеж $\langle d_1,\ t,\ \_\_,\ a_1,\ \ldots,\ a_n\rangle$ представляет факт, что *объекты A_1, ..., A_n являются частью комплексного объекта D_1*, т. е. представлено только отношение *часть—целое* (*ч-ц*).

Особенности. Не останавливаясь на описании существующих семантических сетей, что достаточно подробно изложено в работах [2, 9, 40, 63], а также на описании различных логических и реляционных средств, с чем можно познакомиться в [18, 21, 32, 51], отметим некоторые моменты, характеризующие введенный язык. Первая особенность состоит в наличии специальных вершин-ситуаций (стоящих на первом месте в кортежах или в графическом изображении связанных ребрами ↔ с вершиной связи) и вершин-логических составляющих (стоящих на втором месте и связанных ребрами в виде волнистых линий).

Наличие вершин-ситуаций является одним из существенных факторов, определяющих высокие изобразительные возможности языка сетей. Можно показать, что такие сети не укладываются в рамки языка логики предикатов [25]. В самом деле, каждому фрагменту вида $\langle \_\_, t, r_1, d_1, ..., d_n \rangle$ можно поставить в соответствие истинный предикат $R_1(D_1, ..., D_n)$. Множеству или композиции таких фрагментов будет соответствовать конъюнкция предикатов. В то же время фрагменту $\langle d_0, t, r_1, d_1, ..., d_n \rangle$ трудно найти эквивалент в языке предикатов. Требуется его расширение с введением имен предикатов D_0, т. е. $R_1(D_1, ..., D_n) \overline{\overline{D_0}}$, а также возможности нахождения имен D_0 на аргументных местах предикатов. Становятся возможными конструкции типа $R_1(..., D_0, ...) \overline{\overline{p_1}} D_0$, где имена предикатов являются аргументами их самих же. В более сложных случаях, имена предикатов могут быть аргументами других предикатов, а имена последних — аргументами первых. Возникают контуры обратных связей. Как с ними бороться? Что означают такие конструкции и к каким неприятностям они могут привести? Ответ на эти вопросы требует специальных исследований.

Вершины-логические составляющие делают возможным использование однородных представлений логических конструкций (см. § 1.4). В результате изменением системных знаний (сетей, представляющих таблицы истинности логических операций) может быть обеспечена настройка систем на ту или иную логику. Подобные сети состоят из фрагментов типа: $\langle \_\_, t, \wedge, t, f, f \rangle$ — соответствует $1 \wedge 0 = 0$, $\langle \_\_, t, \vee, t, t, t \rangle$ — соответствует $1 \vee 1 = 1$ и т. д. Словом, таблицы истинности для логических связок представляются как тернарные отношения, где символу 1 сопоставляется *о*-вершина t, а 0 — *о*-вершина f. Следует отметить, что с помощью подобных знаний может быть представлена и информация типа $1 \wedge 0 = 1$ — *есть ложь*, для чего используется фрагмент $\langle \_\_, f, \wedge, t, f, t \rangle$. Таким образом, возможен более высокий уровень логических обобщений.

Другая особенность состоит в четком различении *о*-вершин, *н*-вершин, а также спецвершин. Ниже еще будут введены некоторые типы вершин: вершины-классы, факультативные вершины и др. Такое различие сочетается с их «равнозначностью».

Третья особенность заключается в отсутствии полного набора семантических падежей (типа *быть субъектом, быть действием, быть отношением* и т. д.), задаваемых в явном виде. Считается, что каждое отношение однозначно определяет свои падежи, а также их распределение по местам соответствующего кортежа. Словом, с местом (для каждого конкретного отношения) связан семантический падеж. При графическом изображении такой падеж определяется типом или номером ребра, исходящего из вершины связи. Подобный подход позволяет упростить процедуру сопоставления, избавиться от лишних указаний и никак не связан с потерей общности. В естественном языке слова, называющие отношения и действия, как правило, определяют формы, т. е. категории слов — имен объектов, субъектов и т. д. Такие формы (вне зависимости от порядка расположения слов) могут быть преобразованы в кортежи, где денотаты слов будут расставлены по своим местам. При этом облегчается процедура сопоставления кортежей, представляющих одно и то же отношение. Как правило, сопоставляются элементы, стоящие на одном и том же месте.

Обычно к недостаткам указанного подхода относят высокую местность кортежей. Для каждого семантического падежа (а их более десятка) якобы должно быть свое место. Но это не совсем так. Разумно семантические падежи, указывающие на особенности действия или ситуации в целом, представлять как отношения, связывающие, например, ситуацию или событие со временем его протекания, местом, причиной и т. д. Такие отношения представляются с помощью собственных фрагментов (кортежей) (см. §1.4).

§1.4. ПРИНЦИПЫ ПРЕДСТАВЛЕНИЯ ИНФОРМАЦИИ

В данном параграфе будут рассмотрены достаточно простые случаи использования сетей для представления некоторых теоретико-множественных соотношений, логических высказываний, структурных зависимостей. Материал ни в коем случае не претендует на какую-либо полноту и приводится только в качестве примера, характеризующего принципы использования сетей как средств представления информации.

Вершины-классы. Для называния и описания человек редко использует имена конкретных объектов (такие имена в основном служат для называния мест, городов — *Москва, Киев, ...*). Как правило, в описаниях используются слова-понятия или имена классов. Для их представления будем использовать специальное множество *о*-вершин $M = \{m_i\}$, соответствующих классам. Например, такие *о*-вершины могут соответствовать *классу людей, классу животных* и т. д. Будем обозначать их через '*кл. людей*', '*кл. животных*' и т. д., т. е. с использованием верхних запятых (апострофов).

Введенные *о*-вершины из M могут стоять в ЭФ на местах вершин-объектов и служат для представления обобщенной информации. Имеется в виду представление общих свойств и отношений, характерных для классов объектов. Например, с помощью ЭФ

$$\langle \_\_, t, r_1, \text{'кл. людей'} \rangle \qquad (1.3)$$

представляется *общее свойство* R_1, *характерное для всех людей*. Это может быть свойство *быть смертным, быть разумным* и др.

Обозначение *о*-вершины. В дальнейшем для облегчения понимания материала будем широко пользоваться разного рода мнемоническими обозначениями, наподобие тех, которые были введены для вершин-классов. Будем использовать три способа обозначения *о*-вершин, соответствующих отношениям: во-первых, с помощью верхних запятых, во-вторых, используя первые буквы этих отношений (если они имеют длинную запись), и, в-третьих, опираясь на общепринятую символику (это касается операций, логических связок и др.). Например, '*быть смертным*' — есть *о*-вершина, соответствующая указанному свойству. Эта же *о*-вершина может быть обозначена через *б. см.* Другая *о*-вершина $\in$ соответствует отношению *быть элементом множества*. Чтобы в последнем случае избежать путаницы, будем оговаривать, где знак $\in$ обозначает *о*-вершину, а где — операцию. Это же относится и к другим операциям.

Будем использовать верхние запятые для обозначения различных *о*-вершин, в том числе вершин, соответствующих индивидуальным объектам и действиям. Например, '*Хабаровск*', '*Москва*', ... — есть *о*-вершины, соответствующие конкретным городам, а '*взять*', '*предназначаться*', ... — есть *о*-вершины, соответствующие названным действиям.

Заметим, что в силу полисемичности глаголов они в различных контекстах могут выражать различные действия. В этом случае будем особо оговаривать, какое из действий имеется в виду. Хотя следует помнить, что каждому такому действию в СП сопоставляется собственная вершина (если СП реализована в виде однородной ассоциативной среды) и собственный код (если СП реализована на ЭВМ). Все сказанное справедливо и для имен существительных, используемых для названия объектов. Вообще, каждому знакомому системе объекту (отношению) в СП лучше сопостав-

лять не имя, а вершину или код, которые можно считать в некотором роде смысловыми единицами. Представляется разумным на уровне лингвистического процессора постараться избавиться от многозначности слов естественного языка, чтобы облегчить обработку.

Представление соотношений. *Принадлежность объекта* D_1 *к классу* M_1 будем представлять в следующем виде:

$$\langle \_\_,\ t,\ \in,\ d_1,\ m_1 \rangle. \tag{1.4}$$

Для представления *включения объектов одного класса* (M_1) *в другой* (M_2) будем использовать ЭФ вида

$$\langle \_\_,\ t,\ !\subseteq,\ m_1,\ m_2 \rangle, \tag{1.5}$$

где $!\subseteq$ — есть *о*-вершина, соответствующая операции *теоретико-множественного включения* (над классами). Операции пересечения, объединения и дополнения представляются аналогичным образом, только с помощью шестиместных кортежей. Например, соотношение $M_1 \cap M_2 = M_3$ представляется в виде $\langle \_\_,\ t,\ !\cap,\ m_1,\ m_2,\ m_3 \rangle$, где $!\cap$ соответствует операции над классами.

Классы, множества, совокупности. Будем отличать классы от множеств и совокупностей, хотя в этих понятиях много общего. Множество наблюдаемых объектов — это некоторая реальность. Для ее квантования используются понятия или классы. Классам в СП соответствуют собственные *о*-вершины, с помощью которых задаются общие свойства объектов. Таких вершин ограниченное количество. Множества же объектов могут постоянно меняться. Например, возьмем множества людей, проживающих в том или ином городе. Таких множеств может быть бесконечно много. Конечно, им нет смысла сопоставлять собственные *о*-вершины. Ниже будем различать два типа множеств: задаваемых перечислением и выделяемых по их свойствам и отношениям. Для представления первых множеств будут введены специальные средства — *з*-сети (см. §1.5). Вторым же будут сопоставляться *н*-вершины с их окрестностями — связанными фрагментами. Например, *множеству объектов класса* M_1, *обладающих свойством* R_1, сопоставляется *н*-вершина x_1 следующей сети:

$$\langle \_\_,\ t,\ \in,\ x_1,\ m_1 \rangle \circ \langle \_\_,\ t,\ r_1,\ x_1 \rangle.$$

Для представления операций над множествами будем использовать собственный набор *о*-вершин. Например, если *н*-вершины x_1 и x_2 соответствуют множествам X_1 и X_2, то для представления соотношения $X_1 \cup X_2 = X_3$ будем использовать фрагмент $\langle \_\_,\ t,\ \cup,\ x_1,\ x_2,\ x_3 \rangle$, где $\cup$ — есть *о*-вершина, соответствующая операции над множествами. Напомним, что аналогичная операция над классами обозначалась бы через $!\cup$. Знак «!» соответствует дополнительной операции выделения элементов классов, что осуществляется по отношению $\in$. После этого выполняется сама операция $\cup$.

Под совокупностью объектов будем понимать такое множество, которое в тот или иной момент времени (или в каком-то определенном месте) рассматривается как единое целое. В §1.3 были введены средства представления совокупностей. Если *множество объектов* $\{A_1, \ldots, A_n\}$ *рассматривается как совокупность* D_1, то это представляется в виде $\langle d_1,\ t,\ \_\_,\ a_1, \ldots, a_n \rangle$, т. е. совокупности D_1 сопоставляется своя собственная *о*-вершина d_1.

Представление количеств. Одним из существенных параметров, характеризующих множества, является количество их элементов. Для представления факта, что *некоторое множество* X_1 *содержит* N *объектов*, будем использовать фрагмент

$$\langle \_\_,\ t,\ б.\ кол,\ 'n',\ x_i \rangle, \tag{1.6}$$

где 'n' — есть *о*-вершина, соответствующая заданному количеству, а *б. кол* — отношению *быть количеством*. Вершину x_i, входящую в фрагмент (1.6), будем обозначать в виде x_i^n, подразумевая наличие фрагмента. Например, возьмем сеть

$$\langle \_\_,\ t,\ б.\ кол,\ '1',\ x_1 \rangle \circ \langle \_\_,\ t,\ \in,\ x_1\ 'кл.\ людей' \rangle \circ \langle \_\_,\ t,\ 'быть\ умным',\ x_1 \rangle, \tag{1.7}$$

где x_1 соответствует некоему *одному умному человеку*. С учетом введенных обозначений она будет записана в виде

$$\langle \_\_,\ t,\ \in,\ x_1^1,\ 'кл.\ людей' \rangle \circ \langle \_\_,\ t,\ 'быть\ умным',\ x_1^1 \rangle. \tag{1.8}$$

Будем изображать ее, как показано в средней части рис. 1. 6.

Таким образом, *н*-вершины вида x_i^1 будут соответствовать *единичным объектам*, x_i^2 — *парам*, ..., а *н*-вершины x_i (без индексов) — *множествам с неизвестным* (произвольным) *количеством элементов*. Аналогичная информация в естественном языке выражается с помощью категории числа, которая здесь представляется с помощью индексов. Например, если говорится об *одной вычислительной машине* (но не известно, какая имеется в виду), то ей сопоставляется *н*-вершина x_i^1. Если же речь идет о *вычислительных машинах* (*о некотором их множестве*), то им сопоставляется другая *н*-вершина x_i. При этом следует различать два случая: когда имеются в виду *все машины* или *некоторые из них*. В § 7.3 для их представления будут введены специальные средства.

Количества играют существенную роль в процессе представления и обработки информации. Они указывают на необходимость образования альтернативных множеств, определяют разного рода смысловые нюансы. Сравнительно часто категория количества характеризует объекты одного и того же класса. Будем обозначать *н*-вершины x_i, x_i^1, ..., x_i^n, сопоставленные *объектам класса* M_j, через $x_i|m_j|$, $x_i^1|m_j|$, ..., $x_i^n|m_j|$. Если класс имеет имя в естественном языке, то вместо m_j будем подставлять это имя в соответствующем числе. Например, *н*-вершина x_i^1 $|человек|$ соответствует *некоему человеку*, а *н*-вершина x_2 $|собаки|$ — *множеству собак* и т. д. При изображении таких вершин будем подразумевать наличие соответствующих фрагментов. С учетом сказанного сеть (1.8) записывается в виде

$$\langle \_\_,\ t,\ 'быть\ умным',\ x_1^1\ |человек| \rangle \tag{1.9}$$

и изображается, как показано в правой части рис. 1.6.

Рассмотрим еще два типичных случая использования сетей.

Представление логических высказываний. Начнем с простого примера. В левой части рис. 1.7 изображена сеть, представляющая *факт, что объект* A_1 *обладает свойством* R_1 *или* R_2. При этом свойства рассматриваются как унарные отношения, а логическая операция $\vee$ как тернарное отношение, связывающее логические составляющие (X_1 и X_2) двух других отношений (R_1 и R_2) с фактом истинности t, т. е. представляется логическое выражение $X_1 \vee X_2 = 1$, где X_1 и X_2 — пропозициональные переменные. В со-

ответствии с ранее введенными обозначениями упомянутая сеть может быть изображена в сокращенном виде, как показано в правой части рис. 1. 7. Она записывается следующим образом:

$$\langle \_\_,\ x_1^1,\ r_1,\ a_1 \rangle \circ \langle \_\_,\ x_2^1,\ r_2,\ a_1 \rangle \circ \langle \_\_,\ t,\ \vee,\ x_1^1,\ x_2^1,\ t \rangle, \qquad (1.10)$$

где последняя вершина t соответствует логической составляющей всей представленной информации, т. е. логическому значению выражения $X_1 \vee X_2$.

С помощью сетей могут быть представлены достаточно сложные логические выражения, составленные из различных операций. Например, выражение $(X_1 \vee X_2) \wedge X_2 = 1$ представляется в виде сети (1.10), в которой

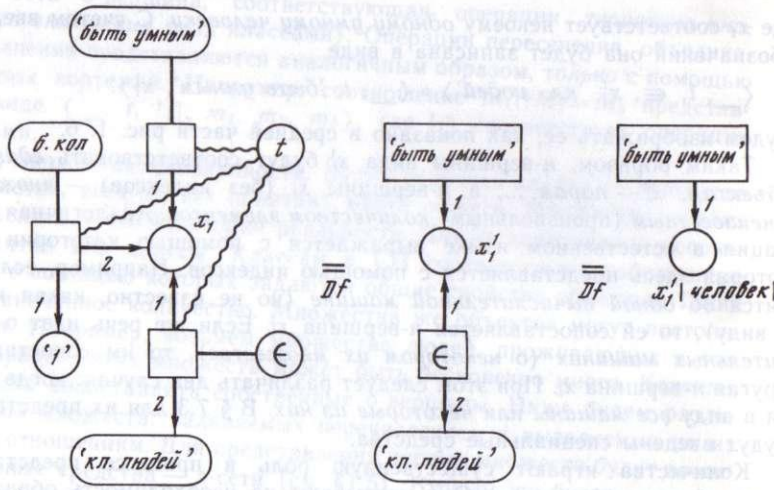


Рис 1.6. Способы изображения факта, что *имеется некий человек* X_1, *и он умный*

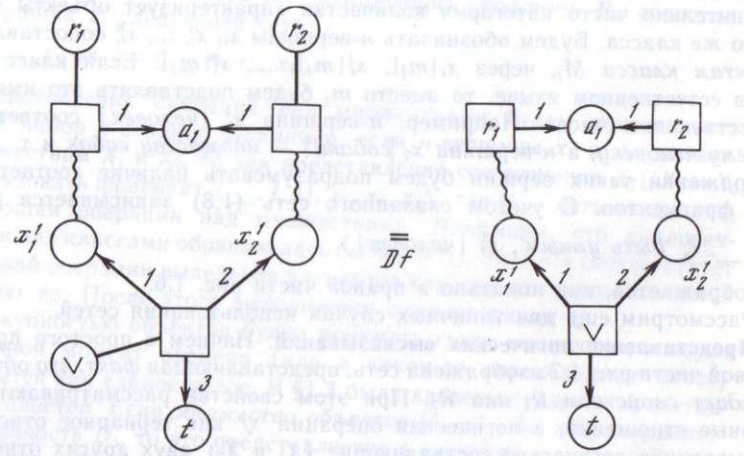


Рис. 1.7. Пример сети, представляющей логическое выражение $R_1(A_1) \vee R_2(A_2)$

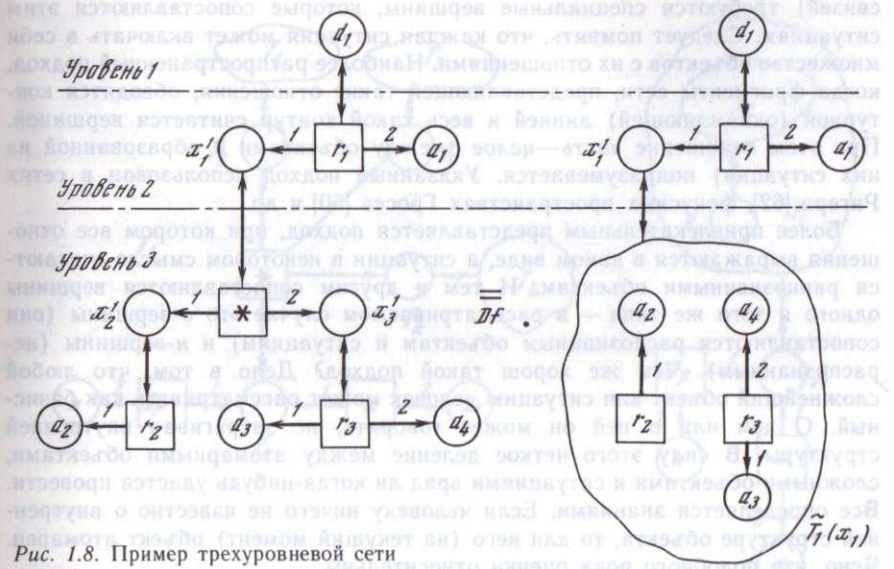


Рис. 1.8. Пример трехуровневой сети

о-вершина t заменяется на x_4^1 и к которой добавляется фрагмент вида $\langle \_\_,\ t,\ \wedge,\ x_3^1,\ x_4^1,\ t \rangle$. Следует отметить, что однородность представления логических выражений предопределяет возможность их обработки унифицированными процедурами. Выражения могут обрабатываться как обычные отношения, т. е. не требуются специальные алгоритмы или процедуры (см. § 4.2).

Представление уровней детализации. В левой части рис. 1.8 изображен пример сети, представляющей три уровня детализации: на одном уровне представляется основной объект, на другом — его части, на третьем — части этих частей. Таких уровней может быть больше трех. Человек довольно широко пользуется подобными многоуровневыми представлениями.

На верхнем уровне (см. рис. 1.8) представлен *объект* D_1, *который состоит из двух других объектов* X_1 *и* A_1, *связанных отношением* R_1. *Объект* X_1 *в свою очередь состоит из двух несвязанных частей: первой является объект* A_2 *с его свойством* R_2, *а второй — объекты* A_3 *и* A_4, *связанные отношением* R_3. Этим двум частям сопоставляются *н*-вершины x_2^1 и x_3^1, а с помощью $\langle x_1^1,\ t,\ \_\_,\ x_2^1,\ x_3^1 \rangle$ представляются составные части объекта X_1.

Будем изображать независимые составные части объектов более простым способом, как показано в правой части рис. 1.8. Здесь отношение *часть—целое* подразумевается. Считается, что вершины и фрагменты, находящиеся внутри контурной линии, представляют части объекта X_1. Ясно, что связанные вершины и фрагменты будут представлять связанные части, например связанные отношением R_3 объекты A_3 и A_4. В то же время несвязанные фрагменты типа $\langle \_\_,\ t,\ r_2,\ a_2 \rangle$ и $\langle \_\_,\ t,\ r_3,\ a_3,\ a_4 \rangle$ представляют относительно независимые части.

Замыкания и *с*-вершины. Для представления сложных зависимостей между ситуациями (т. е. причинно-следственных, временных и других

связей) требуются специальные вершины, которые сопоставляются этим ситуациям. Следует помнить, что каждая ситуация может включать в себя множество объектов с их отношениями. Наиболее распространенный подход, когда фрагменты сети, представляющей такие отношения, обводятся контурной (окаймляющей) линией и весь такой контур считается вершиной. При этом отношение *часть—целое* (между объектами и образованной из них ситуации) подразумевается. Указанный подход использован в сетях Ригера [62], фокусных пространствах Гросса [60] и др.

Более привлекательным представляется подход, при котором все отношения выражаются в явном виде, а ситуации в некотором смысле считаются равнозначными объектам. И тем и другим сопоставляются вершины одного и того же типа — в рассматриваемом случае это *о*-вершины (они сопоставляются распознанным объектам и ситуациям) и *н*-вершины (нераспознанным). Чем же хорош такой подход? Дело в том, что любой сложнейший объект или ситуацию человек может рассматривать как базисный. О нем или о ней он может говорить, не затрагивая внутренней структуры. В силу этого четкое деление между атомарными объектами, сложными объектами и ситуациями вряд ли когда-нибудь удастся провести. Все определяется знаниями. Если человеку ничего не известно о внутренней структуре объекта, то для него (на текущий момент) объект атомарен. Ясно, что подобного рода оценки относительны.

Чтобы избежать трудностей классификации (с поиском некоей «абсолютной» истины) и специальных средств обработки ситуаций (выделения их компонент и т. д.), будем использовать для их представления обычные вершины и фрагменты. Объектам и отношениям, составляющим сложный объект или ситуацию, сопоставляются обычные сети, а самой ситуации (если это необходимо) — отдельная вершина, которую будем называть *с*-вершиной сети. *С*-вершина связывается с самой сетью таким образом, чтобы обеспечить представление принадлежности объектов и их отношений к ситуации. Как правило, для этого будем использовать внутреннюю структуру фрагментов (кортежей), т. е. вершины, стоящие в кортежах на первом месте и соответствующие всей представленной информации. Например, *с*-вершиной сети

$$\langle x_2,\ t,\ r_2,\ a_2\rangle \circ \langle x_3,\ t,\ r_1,\ a_1\rangle \circ \langle x_1,\ t,\ \_,\ x_2,\ x_3\rangle$$

является x_1. При этом с помощью последнего фрагмента представляется, что *объект* X_1 *состоит из объектов* X_2 *и* X_3, а с помощью первых двух фрагментов — *свойства объектов* A_2 *и* A_1, *составляющих* X_2 *и* X_3.

Далеко не каждая сеть может иметь свою *с*-вершину. Если сеть представляет информацию, которая не рассматривается как самостоятельная единица, то *с*-вершины и не требуется. Другой случай, когда сеть начинает использоваться как отдельная, самостоятельная единица. Тогда требуется специальное формирование *с*-вершины, что зачастую связано с достройкой сети. Рассмотрим, как это делается.

Пусть имеется некоторая сеть T_1, представляющая отношения между объектами. Процедуру формирования ее *с*-вершины будем называть замыканием сети по отношению *часть—целое*. Такая процедура заключается в заполнении первых мест кортежей сети собственными *н*-вершинами (если их там не было) и введении фрагментов, представляющих отношение *часть—целое*. Например, пусть T_1 состоит из k фрагментов (ЭФ), у которых на первых местах стояли соответственно *н*-вершины $x_i^1, \ldots, x_{i+k}^1$ (ранее не входившие T_1). Тогда вводится фрагмент $\langle x_{i+k+1}^1, t, \_, x_i^1, \ldots, x_{i+k}^1\rangle$, где x_{i+k+1}^1 и будет *с*-вершиной сети. Более подробно процедура замыкания рассмотрена в работе [14], где указано, как избежать дублирования при представлении отношения *часть—целое*.

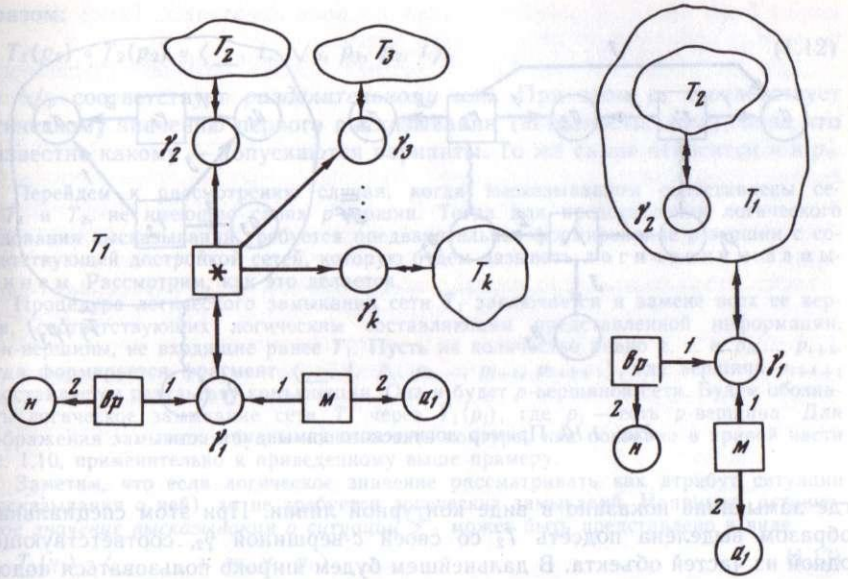


Рис. 1.9. Сети, представляющие *статический объект* $\mathcal{T}_1$ *с его частями* $\mathcal{T}_2, \ldots, \mathcal{T}_k$; *объект существует в настоящее (н) время и в месте* A_1. Справа представлена одна из частей — $\mathcal{T}_2$

Будем обозначать замыкание сети T_1 через $\tilde{T}_1(\gamma_1)$, где γ_1 — ее *с*-вершина. Для изображения замыканий будем пользоваться контурами, как показано в правой части рис. 1.8. При использовании замыканий можно сравнительно простым образом выделять все компоненты сети — по ее *с*-вершине. Для этого используется стандартный механизм, обеспечивающий ответ на запросы типа *Какие части имеет объект или ситуация*?

Представление ситуаций. Введенные ранее *о*-вершины позволяют удобным способом представлять факторы, характеризующие ситуацию в целом, например время протекания, место, кто информировал (сообщал) о ней, когда и др. Такой ситуации сопоставляется *с*-вершина γ_1, к которой подсоединяются фрагменты

$$\langle \_,\ t,\ м,\ \gamma_1,\ d_1\rangle \circ \langle \_,\ t,вр,\ \gamma_1,\ d_2\rangle \circ \ldots, \qquad (1.11)$$

где *м*, *вр* — есть *о*-вершины, соответствующие отношениям *место протекания*, *время протекания*, а d_1, d_2 — соответствуют указанному месту, времени.

Время существования объекта или протекания какого-либо события может быть описано по-разному: указанием конкретного момента (года, месяца, дня...), интервала или же с помощью традиционных категорий *настоящего*, *прошедшего* и *будущего* времен. Последним будем сопоставлять специальные *о*-вершины *н*, *п* и *б*. Например, то, что *объект или ситуация* $\mathcal{T}_1$ *существует в настоящем*, представляется в виде $\langle \_,\ t,\ вр,\ \gamma_1,\ н\rangle$, где γ_1 — *с*-вершина соответствующей сети. На рис. 1.9 изображена более сложная сеть. С помощью нее представлен *объект* $\mathcal{T}_1$, *состоящий из частей* $\mathcal{T}_2, \ldots, \mathcal{T}_k$. *Данный объект существует в настоящее время и в месте* A_1. В правой части рис. 1.9 проиллюстрирован другой способ изображения,

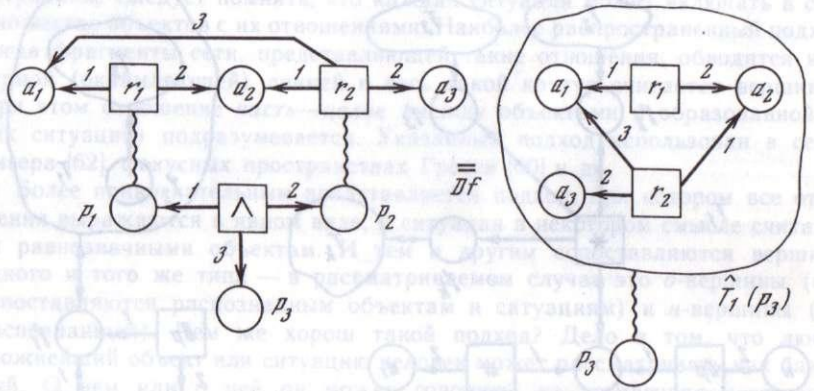


Рис. 1.10. Пример логического замыкания сети

где замыкание показано в виде контурной линии. При этом специальным образом выделена подсеть T_2 со своей c-вершиной γ_2, соответствующей одной из частей объекта. В дальнейшем будем широко пользоваться подобными изображениями.

Следует заметить, что время и место существования какого-либо цельного объекта относится ко всем его частям, что может быть представлено отдельно для каждой части. Однако это очень неудобно. Лучше категории времени и места представлять лишь для наиболее крупных объектов, как это было сделано на рис. 1.9. Предполагается возможность перенесения указанных категорий на любую часть, т. е. окрестность вершины γ_1 (связанные с ней фрагменты) в данном случае может быть перенесена на любую из вершин $\gamma_2, \ldots, \gamma_k$. Это осуществляется путем использования специальных знаний, связанных с преобразованиями (см. § 2.1 и 5.3).

Логические зависимости, p-вершины. Помимо указанных, могут быть и другие типы зависимостей, среди которых важное место занимают логические связи. На примере рис. 1.7 был рассмотрен простейший вид такой связи — между логическими составляющими или логическими значениями (а не между отношениями). В более сложном случае целый «кусок» информационного материала (наборы объектов и отношений) может рассматриваться как единый в логическом смысле, например, как единый факт, из которого может следовать другой факт. Таким фактам сопоставляются собственные вершины типа x^1, которые будем обозначать символами p. На рис. 1.10 такой вершиной является p_3. Она сопоставляется логическому значению всего высказывания, утверждающего *наличие* (*или отсутствие*) *двух отношений*: *отношения* R_1 *между объектами* A_1 *и* A_2, *а также тернарного отношения* R_2 *между объектами* A_2, A_3 *и* A_1.

Будем называть вершину, которая соответствует логическому значению всего высказывания, представленного с помощью сети T_1, p-вершиной этой сети. P-вершины будут служить для представления логических зависимостей. Пусть имеются две сети $T_1(p_1)$ и $T_2(p_2)$, представляющие два высказывания. Пусть p_1 и p_2 — их p-вершины. Тогда *факт, что имеет место или первое, или второе высказывание*, представляется следующим

образом:

$$T_1(p_1) \circ T_2(p_2) \circ \langle \_\_, t, \vee_p, p_1, p_2, t \rangle, \qquad (1.12)$$

где $\vee_p$ соответствует *разделительному или*. При этом p_1 соответствует логическому значению первого высказывания (*истинности*, *лжи*), пока что неизвестно какому — допускаются варианты. То же самое относится и к p_2.

Перейдем к рассмотрению случая, когда высказываниям сопоставлены сети T_1 и T_2, не имеющие своих p-вершин. Тогда для представления логического следования высказываний требуется предварительное формирование p-вершин с соответствующей достройкой сетей, которую будем называть л о г и ч е с к и м з а м ы к а н и е м. Рассмотрим, как это делается.

Процедура логического замыкания сети T_1 заключается в замене всех ее вершин, соответствующих логическим составляющим представленной информации, на n-вершины, не входящие ранее T_1. Пусть их количество равно k, т. е. $p_i, \ldots, p_{i+k}$. Тогда формируется фрагмент $\langle \_\_, t, \wedge, p_i, \ldots, p_{i+k}, p_{i+k+1} \rangle$, где вершина p_{i+k+1} сопоставляется результату конъюнкции. Она и будет p-вершиной сети. Будем обозначать логическое замыкание сети T_1 через $\hat{T}_1(p_j)$, где p_j — есть p-вершина. Для изображения замыкания будем использовать контуры, как показано в правой части рис. 1.10, применительно к приведенному выше примеру.

Заметим, что если логическое значение рассматривать как атрибут ситуации (высказывания о ней), то не требуется логических замыканий. Например, *истинностное значение высказывания о ситуации* $\mathcal{T}_1$ может быть представлено в виде

$$T_1(\gamma_1) \circ \langle \_\_, \_\_, л.\ зн, \gamma_1, p_1 \rangle, \qquad (1.13)$$

где *л. зн — иметь логическое значение*, а γ_1 — c-вершина сети T_1. Тогда p_1 и будет играть роль p-вершины. В частности, если на место p_1 поставить t, то будет представлен *факт истинности высказывания*, а если f — *то ложности*. Подобное представление также является приемлемым и может быть использовано в реальных случаях. При этом второе место в кортежах может иметь другую роль.

Отношения со свободными ролями. Ранее в основном рассматривался случай, когда роли всех объектов, входящие в отношение, известны. Объекты

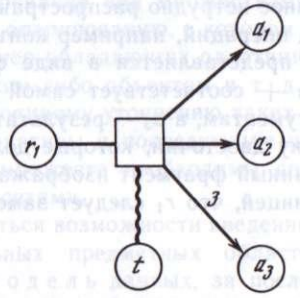


Рис. 1.11. Фрагмент, представляющий *тернарное отношение* R_1, *у которого роли объектов* A_1 *и* A_2 *не зафиксированы*

могут быть однозначно расставлены по своим аргументным местам, что представлялось с помощью кортежей. При такой расстановке облегчается поиск. Он сводится к последовательному сопоставлению кортежей, играющих роль образцов (см. § 4.2). Однако в ряде случаев роли отдельных объектов или семантические падежи отношений по каким-либо причинам могут быть неизвестны. Например, может быть не ясным, кто является субъектом, а кто объектом действия. Семантические падежи однозначно расставлены быть не могут. Более того, в симметричных отношениях типа *находиться слева-направо*, *быть братом* и т. д. при перестановке аргументов сущность не меняется. В отношении *часть—целое*, где

может быть много аргументов-частей, их роли идентичны. Такие аргументы можно произвольно переставлять местами.

Для представления подобных отношений будем использовать фрагменты, у которых на нескольких (или на всех) ребрах отсутствует нумерация. Например, один такой фрагмент изображен на рис. 1.11, где представлено тернарное отношение R_1. При этом роли объектов A_1 и A_2 четко не зафиксированы. Допускается как вариант $\langle \_\_, t, r_1, a_1, a_2, a_3 \rangle$, так и $\langle \_\_, t, r_1, a_2, a_1, a_3 \rangle$. Роли же других составляющих, в том числе и объекта A_3, указаны. Соответствующие вершины связаны с вершиной связи отмеченными ребрами.

Будем записывать фрагменты типа рис. 1.11 в следующем виде:

$$\langle \_\_, t, r_1, :a_1, :a_2, a_3 \rangle, \qquad (1.14)$$

где знак «:» впереди вершины говорит от том, что она, возможно, стоит не на своем месте. Такую вершину можно поменять местами с другой вершиной, также помеченной знаком «:», или поставить на место какого-либо символа __

Отношения типа часть—целое. Некоторые отношения сами определяют аргументы со свободными ролями. К ним относятся симметричные отношения типа *часть—целое* и ряд других. При представлении таких отношений в виде кортежей знаки двоеточия впереди вершин специально ставить не будем, подразумевая их. Хотя при изображении соответствующих фрагментов будем использовать ребра в виде стрелок без номеров. Например, кортеж, представляющий отношение *часть—целое*, будем записывать обычным способом $\langle d_1, t, \_\_, a_1, ..., a_n \rangle$, но в изображении будут стрелки без номеров. Считается, что отношение *часть—целое* (ему сопоставляется __) определяет расстановку знаков двоеточия. Они должны быть у всех вершин $a_1, ..., a_n$, соответствующих частям.

Сказанное нетрудно распространить на симметричные отношения, а также на ряд операций, например конъюнкции, сложения и др. Так, операция сложения представляется в виде фрагмента $\langle \_\_, t, +, a_1, a_2, a_3 \rangle$, где *о*-вершина + соответствует самой операции (тернарному отношению), a_1 и a_2 — аргументам, а a_3 — результату (см. § 2.4). *О*-вершина + определяет расстановку двоеточий, которые должны быть перед *о*-вершинами a_1 и a_2. Кстати, данный фрагмент изображается так же, как показано на рис. 1.11, с той разницей, что r_1 следует заменить на *о*-вершину +.

СПЕЦИАЛЬНЫЕ СРЕДСТВА ПРЕДСТАВЛЕНИЯ

В предыдущей главе были рассмотрены общие принципы, связанные с представлением системных знаний. Были введены семантические сети (с вершинами связи) достаточно простого вида, возможности которых иллюстрировались на сравнительно простых примерах. Однако введенных средств оказывается недостаточно для представления реальных предметных областей (ПО), где могут иметься разного рода зависимости, закономерности. Они могут носить групповой характер, выражать достаточно сложные изменения, преобразования. Для представления подобной информации в самом начале данной главы будут введены специальные средства — так называемые сетевые продукции. Не следует путать это понятие с продукциями в их традиционном понимании (с правой частью, указывающей на способ воздействия). Сетевые продукции — это разновидность семантических сетей. Их левые и правые части — это сети. Наборы продукций образуют определенного сорта внутрисистемные знания (об изменениях, преобразованиях). Такие знания могут постоянно пополняться, корректироваться. Они могут быть использованы для прогнозирования, прослеживания происходящих изменений, преобразования представлений, что осуществляется путем применения продукций. Ниже будет в общих чертах описано действие применения, обеспечивающее необходимые преобразования на уровне внутрисистемных представлений, т. е. преобразование сетей, структур.

Далее, в рамках языка семантических сетей будут введены средства, необходимые для представления групповых отношений, а также для реализации некоторых способов оперирования над множествами. Внутрисистемная обработка сводится к такому оперированию, в котором участвуют различные множества объектов, например обладающих однотипными свойствами, связанных отношениями с каким-либо объектом и т. д. Ответ на запросы сводится к нахождению и постоянному уточнению таких множеств, которые во многом определяют альтернативы и последующие шаги обработки. Для представления подобных множеств необходимы специальные средства, которые будут называться *s*-сетями.

В данной главе будут иллюстрироваться возможности введенных средств для представления некоторых реальных предметных областей. Будет рассматриваться реляционная модель данных, за последнее время ставшая уже классической. Будет проводиться сравнительный анализ этой модели и семантической, основанной на семантических сетях введенного типа. Далее будут затрагиваться вопросы представления алгоритмических конструкций, построения так называемых вычислительных моделей [46]. Отметим, что эти вопросы относятся к области концептуального программирования, интерес к которой за последнее время значительно возрос. Речь идет о построении автономных систем, ориентированных на задачи непосредственных пользователей[1], в том числе задачи вычислительного характера. Такие системы должны иметь свои

[1] Э. Х. Тыугу называет такие системы благожелательными.

внутренние представления — об условии задачи (она может быть задана в виде соотношений), средствах решения. На основе имеющихся знаний система сама должна выбирать направление решения, формировать модель вычисления с построением готового алгоритма. На уровне внутрисистемных представлений должна обеспечиваться компоновка моделей, проверка их правильности и т. д. Для этого необходима разработка соответствующих средств представления и способов манипулирования. Здесь весьма перспективно использование семантических сетей, с помощью которых можно добиться необходимой гибкости, обеспечить постоянную подстройку системы под пользователя. В конце главы будут обсуждаться некоторые вопросы такого использования.

§ 2.1. СЕТЕВЫЕ ПРОДУКЦИИ

Наш внешний мир динамичен. В нем постоянно что-то изменяется. Некоторые изменения происходят сами по себе, другие — вызваны действием человека. Изменения могут касаться и самого человека, например его пространственного расположения, его знаний. Изменяться могут и отношения, связывающие человека с другими объектами внешнего мира. Все это выражается с помощью специальных языковых форм (типа *если..., то, вначале..., затем...*) и глаголов (типа *становиться, взять, передвинуться* и т. д.).

Часто изменения рассматриваются как определенного сорта д и н а м и ч е с к и е о т н о ш е н и я. Хотя отличить последние от статических можно только по их функциональной роли или так называемой операционной семантике, указывающей, какие изменения стоят за такими отношениями. Если подобные указания отсутствуют, то динамические отношения в некотором смысле делаются равнозначными статическим. И в том и в другом случае проследить какие-либо изменения невозможно. Например, *Иванов взял ручку*. Если не знать, какие изменения вызывает действие *взять*, то описанная ситуация будет рассматриваться следующим образом: *Субъект Иванов находился в отношении взять с объектом ручка.*

Динамические отношения дискретного характера. Рассмотрим вначале идеальный случай, когда объекты и отношения предметной области (ПО) однозначно распознаваемы. Им сопоставляются свои *о*-вершины. Пусть в такой ПО происходят следующие изменения: *вначале имела место ситуация* $\mathcal{T}_1$, *затем* $\mathcal{T}_2$, *затем* $\mathcal{T}_3$, ..., *затем* $\mathcal{T}_k$. Самый простой способ представления таких изменений основан на использовании сети, изображенной на рис. 2.1. С помощью фрагментов $\langle \_\_, t, в\text{-}з, \gamma_i, \gamma_{i+1} \rangle$ представляется отношение типа *вначале—затем*. Вершины γ_i и γ_{i+1} ($i=1, 2,...$) сопоставлены ситуациям, которые рассматриваются как самостоятельные. Представлено, что изменения затрагивают не всю ситуацию, а лишь часть. Контурные линии, служащие для изображения T_i, T_{i+1}, пересекаются. Это означает, что некоторые свойства и отношения остаются, но они рассматриваются в рамках уже другой ситуации — возникающей. В текущий момент имеет место лишь последняя ситуация $\mathcal{T}_k$, что представляется с помощью специального фрагмента $\langle \_\_, t, вр, \gamma_k, н \rangle$. В общем-то такой фрагмент всегда может быть сформирован за счет отношения *вначале—затем*. Изменениям в целом сопоставлена *с*-вершина γ_{k+1}. Такие изменения рассматриваются как отдельные события.

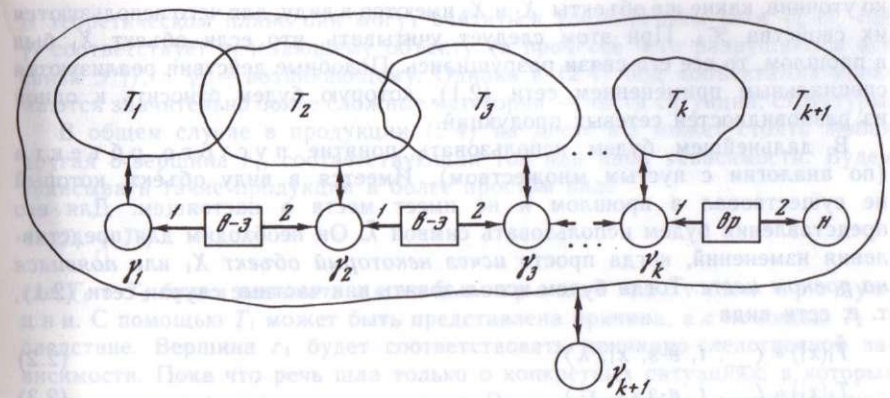


Рис. 2.1. Сеть, представляющая последовательность ситуаций $\mathcal{T}_1$, $\mathcal{T}_2$, ..., $\mathcal{T}_k$, образующих более сложную ситуацию $\mathcal{T}_{k+1}$

Описанный способ представления неудовлетворителен в тех случаях, когда в каком-либо объекте или ситуации изменяется лишь незначительная часть. Стоит ли рассматривать такие изменения как замену одной ситуации другой? Может оказаться, что изменения предусматриваются в рамках самой ситуации и не затрагивают ее сущности. Тогда требуются свои представления. На рис. 2.1 изображена сеть, представляющая *изменение объекта* A_1 *на* A_2, *что происходит в рамках ситуации* $\mathcal{T}_{k+1}$. Фактически один объект замещается на другой, а сама ситуация не изменяется.

В рассмотренной сети предполагалось, что объекты как бы имеют свои уникальные имена (обозначения) A_1 и A_2. На самом деле многие объекты могут быть выделены лишь по их свойствам и отношениям. Как же тогда должны представляться изменения? Здесь необходимо учитывать, что объекты с одними и теми же свойствами и отношениями просто неразличимы. Если в какой-либо ситуации заменить один объект на другой, сохранив все свойства и отношения, то, по сути говоря, ничего не изменится. Такая замена просто не будет замечена. Если речь идет о каких-либо изменениях, то предполагается исчезновение одних свойств или отношений с возникновением других, которые привязываются к оставшимся частям. Итак, **изменения происходят в контексте.**

В простейшем случае подобные изменения могут быть представлены в виде

$$T_1(x_1^1, x_2^1) \circ \langle \_\_, t, в\text{-}з, x_1^1, x_2^1 \rangle . \qquad (2.1)$$

Предполагается, что с исчезновением у объекта X_1 разрушились все его связи. Соответственно фрагменты сети T_1, содержащие *н*-вершину x_1^1, должны быть отмечены как несуществующие. Однако с появлением объекта X_2 возникли новые связи, также представленные в T_1. Соответственно фрагменты с вершиной x_2^1 должны быть отмечены как существующие, т. е. представляющие то, что в настоящий момент имеет место. Для этого могут быть использованы традиционные категории настоящего и прошедшего времени. Проследить изменения, представленные с помощью сети (2.1), можно, толь-

ко уточнив, какие же объекты X_1 и X_2 имеются в виду, для чего используются их свойства $\mathcal{T}_1$. При этом следует учитывать, что если объект X_1 был в прошлом, то все его связи разрушились. Подобные действия реализуются специальным применением сети (2.1), которую будем относить к одной из разновидностей сетевых продукций.

В дальнейшем будем использовать понятие пустого объекта (по аналогии с пустым множеством). Имеется в виду объект, который не существовал в прошлом и не имеет места в настоящем. Для его представления будем использовать символ $\tilde{\lambda}$. Он необходим для представления изменений, когда просто *исчез некоторый объект* X_1 или *появился на ровном месте*. Тогда будем использовать как частные случаи сети (2.1), т. е. сети вида

$$T_1(x_1^1) \circ \langle \_,\ t,\ в\text{-}з,\ x_1^1,\ \tilde{\lambda} \rangle; \tag{2.2}$$

$$T_1(x_1^1) \circ \langle \_,\ t,\ в\text{-}з,\ \tilde{\lambda},\ x_1^1 \rangle. \tag{2.3}$$

Понятие сетевой продукции. В описанном способе представления не учитывался тот факт, что изменения часто затрагивают и сам контекст.

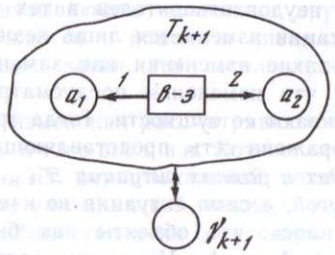


Рис. 2.2. Сеть, представляющая, что *в ситуации* $\mathcal{T}_{k+1}$ *объект* A_1 *исчез и на его месте появился* A_2

Если объект исчез, то, как правило, изменяются отношения между оставшимися объектами. Например, если из слова *AFBC* убрать букву *F*, то изменится отношение между буквами *A* и *B* — они окажутся рядом. Аналогично при добавлении букв изменяются отношения в том месте, куда они вставляются. Такие изменения не могут рассматриваться под углом зрения только разрушения (формирования) всех связей. В процессе разрушения возникают новые связи, а в процессе формирования — разрушаются имеющиеся ранее. Для представления таких изменений требуются конструкции следующего вида:

$$T_1(\gamma_1) \circ \langle \gamma_3,\ t,\ в\text{-}з,\ \gamma_1,\ \gamma_2 \rangle \circ T_2(\gamma_2), \tag{2.4}$$

где сеть T_1 представляет все исчезнувшие связи, а T_2 — все возникшие. Им сопоставлены *с*-вершины γ_1 и γ_2. Вершина γ_3 соответствует самим изменениям. Это и есть *с*-вершина продукции. В отличие от сети рис. 2.1 с помощью (2.4) представлено только то, что изменилось. В частности, при изменениях типа *AFBC*→*ABC* не будет представлена часть *BC*. Будем называть конструкции (2.4) сетевыми продукциями (подразумевая слово «семантический»).

Сетевые продукции будут играть важную роль в задачах представления разного рода зависимостей не только временно́го, но и причинно-следственного (условного) характера, а также определений, пояснений и т. д.

В теоретическом плане они могут считаться расширением сети (2.1), где γ_1 соответствует исчезающему объекту (в процессе чего разрушаются все связи $\mathcal{T}_1$), а γ_2 — возникающему. Однако в (2.4) под «объектами» понимаются значительно более сложные категории — части ситуаций, структуры.

В общем случае в продукции (2.4) на месте *в-з* может стоять любая другая *о*-вершина r_1, соответствующая той или иной зависимости. Будем записывать такие продукции в более простом виде

$$T_1 \underset{r_1}{\rightarrow} T_2,$$

где T_1 и T_2 будем называть левой и правой частями продукции. С помощью T_1 может быть представлена причина, а с помощью T_2 — следствие. Вершина r_1 будет соответствовать причинно-следственной зависимости. Пока что речь шла только о конкретных ситуациях, в которых описываются происходящие изменения. Рассмотрим более сложный случай.

Как уже говорилось, очень мало объектов имеет свои уникальные имена. Чтобы указать на какой-либо объект, выделить его, необходимо обратить внимание на его характерные свойства и отношения. Например, в описании *Буква, стоящая между A и B, переместилась и встала вслед за буквой C* прямо не указывается, какая буква имеется в виду. Предполагается, что она всегда может быть найдена в том слове, к которому относится описание. Для представления подобного сорта объектов ранее были использованы *н*-вершины. Они вводятся в левую и правую части продукции (2.4). В связи с чем (2.4) примет вид

$$T_1(z_1) \underset{r_1}{\rightarrow} T_2(z_2), \tag{2.5}$$

где z_1 и z_2 — наборы *н*-вершин. Здесь изменения представлены отдельно от самой ситуации. Чтобы проследить изменения, требуется специальное совмещение продукции (2.5) и сети, представляющей ситуацию. В результате уточняется, о каких же множествах Z_1 и Z_2 идет речь. Становится возможной привязка описываемых изменений к конкретной ситуации, добавление новой информации. Будем называть подобного сорта совмещение применением продукции.

Будем различать несколько видов описаний зависимостей и соответственно типов сетевых продукций (2.5). Рассмотрим вначале, когда в описании указана конкретная ситуация, в которой происходят изменения. При уточнении множеств Z_1 и Z_2 следует учитывать подобное указание. Для его представления будем использовать *с*-вершину продукции. На рис. 2.3 показана продукция (см. середину рисунка), которая записывается следующим образом:

$$\langle \gamma_1,\ t,\ r_1,\ x_1^1,\ x_2^1 \rangle \circ \langle \gamma_k,\ t,\ в\text{-}з,\ \gamma_1,\ \gamma_2 \rangle \circ \langle \gamma_2,\ t,\ r_2,\ x_1^1,\ a_1 \rangle.$$

Представляются изменения, происходящие в ситуации $\mathcal{T}_k$ (ей сопоставлена сеть T_k с *с*-вершиной γ_k). Изменения касаются *объектов* X_1 *и* X_2, *связанных отношением* R_1. *Такая связь разрушилась и возникла новая связь — объект* X_1 *вступил в отношение* R_2 *с объектом* A_1.

Чтобы понять, о каких изменениях идет речь, необходимо найти, что это за объекты (X_1 и X_2), имеются ли такого сорта объекты (связанные R_1) в ситуации $\mathcal{T}_k$. Это осуществляется совмещением или применением продук-

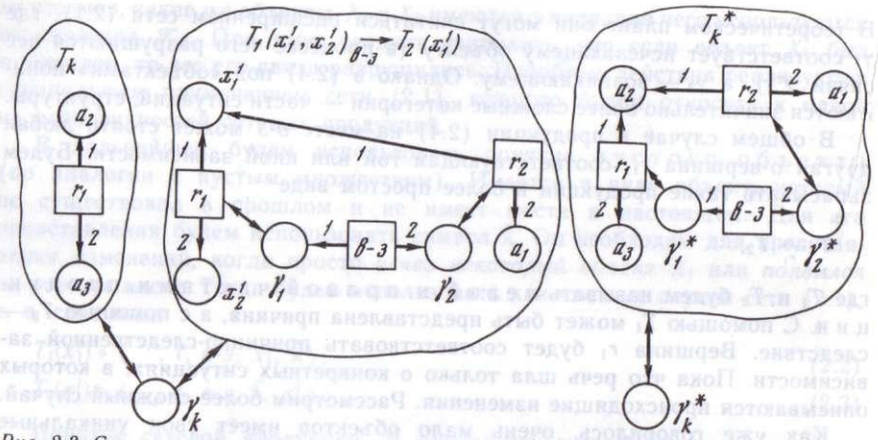


Рис. 2.3. Сетевая продукция, представляющая *исчезновение у объекта* (X_1) *его связи типа* R_1 *и возникновение связи типа* R_2 *с* A_1

ции к сети T_k. Рассмотрим пример такого применения при работе продукции в одном из режимов — дополнения (с сохранением предыстории). В левой части рис. 2.3 представлены *объекты* A_2, A_3, *связанные отношением* R_1. Совмещение возможно — продукция применима, что приводит к изменению сети T_k, пополнению соответствующей информации. В правой части изображена сеть T_k^*, получившаяся в результате пополнения. В ней представлено, что *отношения объекта* A_2 *изменились: вначале он был связан отношением* R_1 *с* A_3, *а затем отношением* R_2 *с* A_1. Видно, что новое отношение «привязано» к имевшемуся ранее объекту A_2. Но возник и новый объект A_1. Соответственно новый фрагмент $\langle \_\_, t, r_2, a_2, a_1 \rangle$ сформирован на базе имевшейся вершины a_2 и вершины a_1.

Процесс применения продукций будет описан более подробно в конце параграфа, а также в § 5.3. Сейчас только отметим, что здесь возможны различные варианты. В проиллюстрированном случае рис. 2.3 в совмещении участвовала лишь левая часть продукции. Хотя может участвовать и правая часть. Применение продукции привело к сохранению предыстории, т. е. было представлено отношение типа *вначале—затем*. Хотя такое применение может сводиться и к простому замещению, как в формальных грамматиках (при подстановке слов). Более того, возможны различные способы представления того, что изменяется. Например, изменять можно метки времени, а можно — знаки существования (λ) или несуществования ($\tilde{\lambda}$). Об этом уже говорилось ранее. Характер требуемых изменений во многом зависит и от задач. Для некоторых задач необходимо лишь знание текущей ситуации. Тогда имеет место случай замещения, т. е. полного «забывания» того, что было ранее. Хотя для многих задач оказывается полезным помнить предысторию (как возникла исходная ситуация). Тогда можно вернуться назад, учтя предыдущий опыт. Это наиболее общий случай, к которому сводится случай замещения. Всегда можно изъять то, что было отмечено метками прошедшего времени.

Зависимости обобщенного характера, закономерности. Перейдем к рассмотрению случая, когда в описаниях нет специальных указаний на конкретную ситуацию. Описываемые изменения могут касаться любой ситуации, которая может быть уточнена в соответствии с характером описания. Последнее приобретает оттенок зависимости, закономерности. Указывается исходная часть или то, что было (условие, причина), и то, что получилось (результирующая часть, следствие). Зависимость может быть справедлива для любой ситуации, на компонентах которой выполнимо условие, в частности, имеются указанные исходные объекты. Тогда зависимость может быть «прослежена», использована для целей прогнозирования. При этом прослеживание может осуществляться в обе стороны. Например, по текущей ситуации может быть сделана попытка угадать ее причину. Тогда уже предполагается совмещение следствия. Более того, в процессе такого совмещения могут уточняться и компоненты причины. «Настройка» закономерности предполагает весьма сложные действия, в которых в общем случае могут участвовать обе части — левая и правая.

Для представления таких зависимостей в дальнейшем будут использоваться сетевые продукции вида (2.5). Прослеживание зависимости сводится к специальному совмещению продукции с сетью $T_{\text{исх}}$, соответствующей исходной ситуации. Такое совмещение было названо применением. Будем различать несколько случаев или режимов применения. Ранее говорилось о прослеживании зависимости в ту или иную сторону. Специальные режимы обеспечивают такое прослеживание с представлением зависимости и без. В одном из режимов может осуществляться проверка наличия зависимости. Подобные режимы будут подробно рассматриваться в § 5.3. Ниже в основном будет описываться случай применения, основанный на совмещении левой части. Если в $T_{\text{исх}}$ имеется аналог этой части, то продукция считается применимой. Тогда и формируется сеть, представляющая произошедшие изменения.

Заметим, что зависимость может быть привязана к конкретной ситуации. Соответственно в продукции (2.4) может быть заведомо указано, к какой сети ее следует применять. Для этого используется c-вершина γ_3 продукции и фрагмент типа $\langle \gamma_k, t, \_\_, \gamma_3 \rangle$, где γ_k — есть c-вершина сети. С помощью фрагмента представляется, что *зависимость*, которой сопоставлена вершина γ_3, *относится к* (*является частью*) *ситуации*, которой сопоставлена γ_k. Фрагмент добавляется к (2.4). Фактически представляется справедливость той или иной зависимости применительно к отдельным ситуациям. Если подобных указаний нет, то имеет место случай так называемого свободного применения, т. е. ситуация, к которой относится зависимость, заранее не фиксируется.

Еще один случай — когда зависимость или закономерность относится лишь к ситуациям определенного класса. Тогда для представления будем использовать сетевые продукции вида

$$T_1(z_1, \gamma_1) \circ \langle \gamma_3, t, r_1, \gamma_1, \gamma_2 \rangle \circ \langle \gamma_k, t, \_\_, \gamma_3 \rangle \circ \langle \_\_, t, \in, \gamma_k, m_1 \rangle \circ T_2(z_1, \gamma_2). \quad (2.6)$$

По сравнению с (2.5) добавляются фрагменты $\langle \_\_, t, \in, \gamma_k, m_1 \rangle \circ \langle \gamma_k, t, \_\_, \gamma_3 \rangle$, представляющие *соотнесенность зависимости к ситуации класса* M_1. Продукция (2.6) может быть совмещена лишь с сетью, представляющей такую ситуацию. Конечно, для совмещения требуется выполнение соответствующих условий, связанных с применимостью продукции.

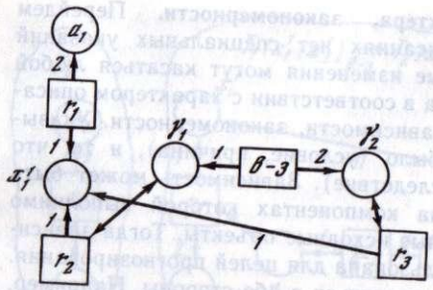


Рис. 2.4. Сетевая продукция, представляющая *изменение свойств объекта, находящегося в отношении* R_1 *с* A_1

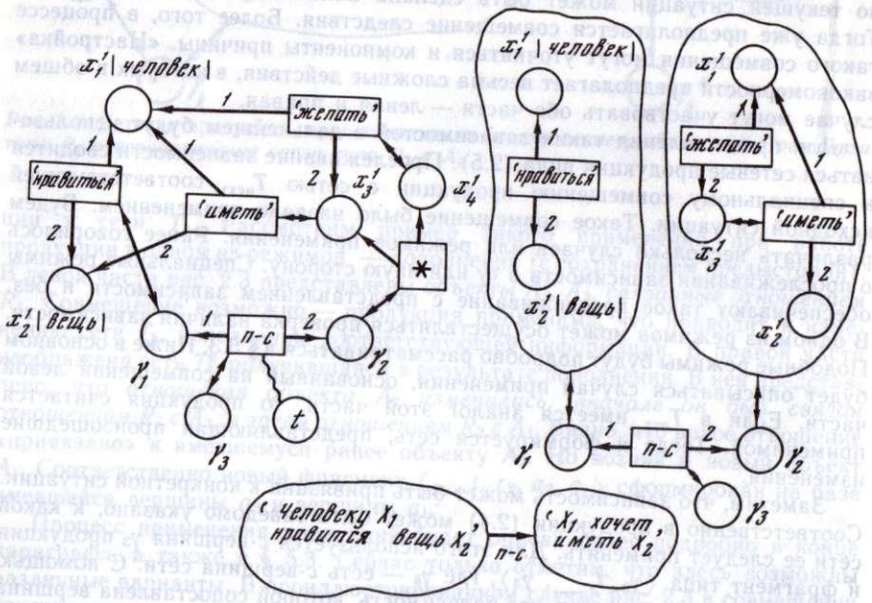


Рис. 2.5. Три способа изображения сетевой продукции, представляющей *Если* X_1 *нравится вещь* X_2, *то он желает иметь ее*

Отметим, что в сетевой продукции наборы *н*-вершин z_1 и z_2 могут совпадать, а могут и различаться. Пусть левая и правая части T_1 и T_2 содержат одни и те же *н*-вершины $z_1 = z_2$. Они соответствуют неопределенным объектам, которые как бы сохраняются в представленной закономерности. Например, если объект присутствует в исходной ситуации, то он есть и в измененной; если объект имеется в причине, то он есть и в следствии, и т. д. Конечно, имеются в виду лишь объекты, связанные отношениями так, как представлено в левой части продукции T_1, т. е. на возможность их варьирования наложены ограничения.

Будем различать сетевые продукции по способу уточнения множеств объектов Z_1 и Z_2. Ранее в основном рассматривался случай, когда для уточнения объектов набора Z_1 достаточно было описания исчезающей части (см. рис. 2.3). Объекты X_1 и X_2 уточнялись по отношению R_1, которое исчезает, разрушается. В общем случае объекты наборов Z_1 и Z_2 могут уточняться с использованием **окружения, контекста.**

В качестве примера рассмотрим рис. 2.4. На нем изображена продукция, представляющая следующие изменения, которые касаются свойства объекта X_1: *вначале он обладал свойством* R_2, *а затем* — R_3. При этом имеется в виду *объект, связанный отношением* R_1 с A_1.

Заметим, что контекст $\mathcal{T}_3$, по которому уточняются объекты, но который сам не изменяется, в общем случае может быть представлен следующим образом:

$$T_1 \circ T_3 \underset{r_1}{\rightarrow} T_2 \circ T_3,$$

где T_3 соответствует отношениям. Последние как бы исчезли и тут же возникли. В частности, на рис. 2.4 таким отношением будет R_1 с объектами A_1 и X_1. Они и образуют контекст.

Изображение продукций, возможности представления. В дальнейшем будем использовать несколько способов изображения сетевых продукций. На рис. 2.5 изображены три варианта продукции, представляющей пояснение типа *Если некоторому человеку* X_1 *нравится вещь* X_2, *то он хочет иметь ее.* Здесь имеет место причинно-следственная зависимость, которой сопоставляется *о*-вершина *п-с*. Всему событию, заключающемуся в том, что X_1 *нравится вещь* X_2, сопоставлена *с*-вершина γ_1, а событию типа *желает иметь...*— *с*-вершина γ_2 (верхние индексы 1 у таких вершин будут подразумеваться). При этом неважно, о каком конкретном человеке или вещи идет речь. Им сопоставляются *н*-вершины x_1^1 и x_2^1. Нетрудно убедиться, что чем больше таких *н*-вершин (при фиксированном числе фрагментов), тем в большем количестве случаев будет справедлива представленная закономерность, т. е. последняя будет носить более обобщенный характер.

Первый способ изображения (слева рис. 2.5) иллюстрирует, что продукция — это та же семантическая сеть, только специальной конструкции. В ней (путем замыкания) указываются две сравнительно самостоятельные части со своими *с*-вершинами γ_1 и γ_2. Фрагмент, отмеченный *, служит для такого указания. Напомним, что он представляет отношение *часть—целое*. Второй способ изображения (в середине) является в некотором смысле упрощенным. В нем нет фрагментов (со звездочками), представляющих отношение *часть—целое*. Для выделения частей используются контурные линии. При этом одни и те же вершины (и фрагменты) левой и правой частей продукции продублированы. Например, вершина x_1^1 изображена два раза. Хотя подразумевается, что это одна и та же вершина, т. е. имеется только один ее экземпляр.

Второй вариант по своей композиции смотрится несколько лучше. Он будет использоваться для представления сложных продукций, где первый вариант приводит к чрезвычайному загромождению рисунка. Хотя изображение в определенном смысле делается некорректным. И, наконец, третий вариант основан на мнемонических обозначениях, которые будут введены в конце параграфа.

Сетевые продукции, как уже говорилось, будут использоваться не только для представления изменений, зависимостей временно́го характера, но для представления разного рода **преобразований**, которые реализуются на уровне внутрисистемных представлений. В частности, имеются в виду преобразования, основанные на использовании разного рода определений, пояснений, выражаемых с помощью форм типа *это есть*, *это значит* и др. Например,

пояснение *Клюква — это красная кислая ягода* представляется с помощью продукция $T_1 \underset{Df}{\rightarrow} T_2$, где *Df* соответствует специальному отношению типа *определять, пояснять*. Сеть T_1 состоит из одного фрагмента $\langle \gamma_i, t, \in, x_1, $ *'клюква'* $\rangle$, а T_2 — из трех фрагментов, представляющих, что X_1 *является ягодой, имеет красный цвет и кислый вкус*. Высказывание X_1 *является тещей* X_2 *это значит, что* X_1 *является матерью жены* X_2 также представляется в виде $T_1 \underset{Df}{\rightarrow} T_2$, где T_1 первая часть высказывания, а T_2 — вторая.

Каждое такое высказывание может быть представлено несколькими способами. В частности, если имеется в виду не отношение *быть клюквой*, а реальный объект, то пояснение указанного типа будет представлено с помощью продукции вида (2.1), где вместо *в-з* будет стоять *Df*, а x_1^1 и x_2^1 будут соответствовать отождествляемым объектам. Подобные представления более подробно будут рассматриваться в § 5.3. Итак, будем допускать несколько способов представления одного и того же определения — с учетом имеющих место нюансов. Сказанное относится и к определениям разного рода действий, стратегий, основанных на использовании операторов (акций) базового набора (см. § 2.4).

В примерах, которые рассматривались выше, предполагалось, что обе части продукции основаны на одних и тех же представлениях. Так, и левая и правая части продукции представляли смысл высказываний, составляющих определяемую и определяющую части. Хотя в ряде случаев предполагаются чисто формальные преобразования — без вникания в смысл. Имеются в виду преобразования знакосочетаний, связанные с перестановкой знаков (слов), раскрытием скобок и т. д. Еще один случай сводится к преобразованиям, связанным с интерпретацией, выявлением семантической компоненты. Анализируется знакосочетание (словоформа) и формируется сеть, представляющая ее смысл. В последующих разделах будут рассматриваться примеры таких преобразований, для чего будем различать уровни поверхностных и семантических структур (см. § 2.3). Сейчас важно отметить, что во всех указанных случаях могут быть использованы сетевые продукции. Они позволяют представлять разного вида языковые формы, с помощью которых вводятся новые понятия, описываются значения слов и символов, указываются эквивалентные формы. Соответствующие преобразования реализуются применением продукций к исходным сетям. При этом чем больше в таких продукциях *н*-вершин, тем чаще она будет применимой. Будет представляться зависимость все более обобщенного характера.

Отметим, что в частном случае продукции сводятся к графовым грамматикам. Однако в продукциях общего вида допускается наличие *н*-вершин, вершин с заданными значениями. Продукции могут работать в режимах простого дополнения, в том числе с сохранением предыстории. Все это выходит за рамки грамматик.

Составные продукции. В общем случае изменения могут описываться указанием на те действия, которые к ним приводят. Предполагается, что с каждым действием могут быть связаны такие изменения. Подобного рода связь представляется с помощью специальных продукций, называемых составными. Их правыми частями также являются продукции. Словом, имеется как бы материнская продукция, в данном случае представляющая определение действия, связь с изменениями, а также дочерняя продукция, представляющая сами изменения.

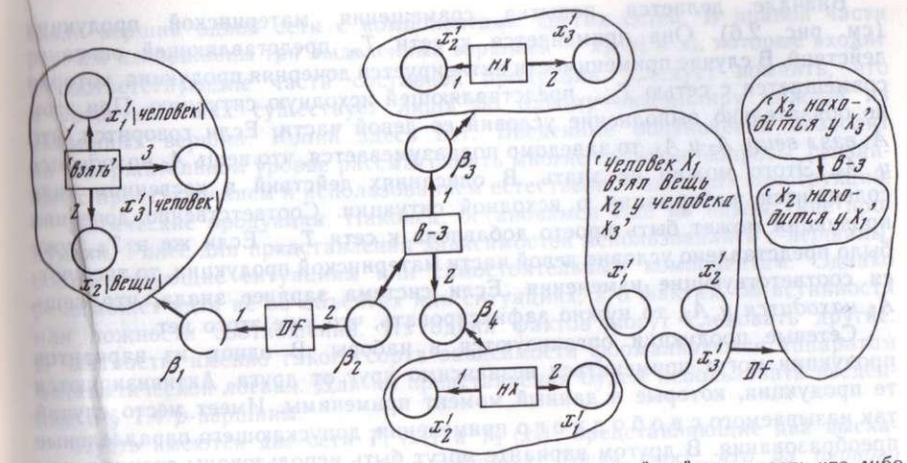


Рис. 2.6. Пример составной продукции, расшифровывающей действие *взять что-либо* (X_2) *у кого-либо* (X_3)

Аналогично следствие может иметь вид закономерности типа *вначале—затем*. Если в исходной ситуации имеет место причина, то следствием являются последовательные изменения. Подобные взаимозависимости представляются с помощью составной продукции, где материнская часть представляет отношение *причина—следствие*. Она содержит в себе дочернюю продукцию, представляющую отношения *вначале—затем*.

Составные продукции записываются следующим образом:

$$T_1(\gamma_1) \circ \langle \_\_, t, r_1, \gamma_1, \gamma_4 \rangle \circ \langle \gamma_4, t, r_2, \gamma_2, \gamma_3 \rangle \circ T_2(\gamma_2) \circ T_3(\gamma_3), \qquad (2.7)$$

или в более простом виде:

$$T_1 \underset{r_1}{\rightarrow} (T_2 \underset{r_2}{\rightarrow} T_3), \qquad (2.8)$$

где $T_1 \underset{r_2}{\rightarrow} T_3$ — есть дочерняя продукция. В общем случае у каждой материнской продукции может быть множество дочерних, организованных в набор.

Пример составной продукции изображен на рис. 2.6, где представлено следующее определение: X_1 *взял вещь* X_2 *у* X_3 *— это значит, вначале вещь* X_2 *находилась у* X_3, *а затем стала находиться у* X_1. Здесь с помощью материнской продукции представляется определение *Df*, позволяющее расшифровывать высказывания о действиях типа *взять*. Дочерняя продукция представляет изменения, которые происходят в случае выполнения действия. Такие изменения относятся к исходной ситуации. Последняя (или ее класс) может быть непосредственно указана, что представляется таким же способом, как в продукции (2.6). Другой случай, когда исходная ситуация выделяется в процессе применения составной продукции (ее совмещения с соответствующими сетями). Имеются в виду сети, представляющие как описание действий, так и исходную ситуацию. В процессе совмещения может принимать участие как материнская, так и дочерняя части. Остановимся на действии применения более подробно.

Вначале делается попытка совмещения материнской продукции (см. рис. 2.6). Она применяется к сети T_o, представляющей описание действий. В случае применимости активируется дочерняя продукция, которая совмещается с сетью $T_{\text{исх}}$, представляющей исходную ситуацию. При этом не обязательно выполнение условий ее левой части. Если говорится, что A_1 *взял вещь* A_2 *у* A_3, то заведомо подразумевается, что вещь A_2 *находилась у* A_3. Этого можно не знать. В описаниях действий в косвенном виде содержится информация о исходной ситуации. Соответственно дочерняя продукция может быть просто добавлена к сети $T_{\text{исх}}$. Если же в $T_{\text{исх}}$ уже было представлено условие левой части материнской продукции, то требуются соответствующие изменения. Если система заранее знала, что вещь A_2 *находится у* A_3, то нужно зафиксировать, что уже этого нет.

Сетевые продукции организуются в наборы. В одном из вариантов продукции могут применяться независимо друг от друга. Активизируются те продукции, которые в данный момент применимы. Имеет место случай так называемого с в о б о д н о г о применения, допускающего параллельные преобразования. В другом варианте могут быть использованы специальные с т р а т е г и и п р и м е н е н и я. Например, одной из таких стратегий является следующая. Продукции упорядочиваются. Очередную продукцию разрешается применять лишь в случае, когда не применимы все следующие. Подобных стратегий может быть множество. Они могут основываться на идеях возврата (back tracking), условного выбора и т. д. Важно отметить, что для представления стратегий перспективно использование гибких средств, которые при необходимости можно менять, корректировать извне. В дальнейшем для этого будут служить определенного вида семантические сети и графы (см. § 2.3). Графы также будут задавать и действие применения продукций (см. § 5.3). Продукции, левой и правой частями которых являются семантические графы, будут называться граф-продукциями.

Мнемонические обозначения. В дальнейшем будем широко пользоваться общепринятым естественным языком для обозначения сетей (продукций, графов). Пусть имеется сеть T_i (x_1^1), с помощью которой представлены отношения объекта X_1. Будем обозначать такую сеть через 'Π'_j, где Π_j — есть предложения ЕЯ, описывающие указанные отношения. При необходимости в таких предложениях будем особо отмечать объект X_1. Например, '*имеется умный человек* X_1' (этой записи сети соответствует рис. 1.6). Верхние запятые (апострофы) указывают на внутреннее представление, в данном случае на сеть. Подобные обозначения будем стараться использовать только в случаях, если известно, как строится сеть, представляющая смысл предложения Π_j. Это позволит рассматривать сложные конструкции без акцентирования внимания на ее простых частях.

Аналогично будем обозначать продукции. Например, продукция на рис. 2.5 записывается в виде

$$\text{'}\textit{человеку } X_1 \textit{ нравится вещь } X_2\text{'} \xrightarrow[n=c]{} \text{'}X_1 \textit{ хочет иметь } X_2\text{.'} \qquad (2.9)$$

Здесь также апострофы служат для указания на внутреннее представление — на сеть, представляющую смысл помещенного внутри них выражения. Будем изображать их, как показано на рис. 2.5 и 2.6, где части продукции представляются с помощью контуров, в которых и указано, что в них находится. При необходимости будем выделять в контурах отдельные вершины обозначенных сетей. Это необходимо для указания

связи вершин одной сети с компонентами других сетей. В правой части рис. 2.6 изображены три выделенные вершины — x_1^1, x_2^1 и x_3^1, которые входят в соответствующие части составной продукции. Следует помнить, что в представлениях существует лишь по одному экземпляру каждой из указанных вершин. Копий здесь нет. Введенные обозначения позволят на содержательном уровне рассматривать многие сложные вопросы, связанные с представлением и использованием естественно-языковых конструкций.

Логические продукции. Наконец, остановимся еще на одном типе продукций. Ранее для представления зависимостей использовались *с*-вершины, соответствующие ситуациям или самостоятельным компонентам. Однако речь может идти не об объектах или ситуациях, а о фактах, об истинности или ложности соотношений. Из одних фактов могут следовать другие. В частности, именно такого сорта зависимости формализуются аппаратом математической логики. Для их представления будем использовать введенные в § 1.4 *р*-вершины.

Пусть имеются две сети T_1 (x_1^1) и T_2 (x_2^1), представляющие два высказывания. Пусть x_1^1 и x_2^1 — их *р*-вершины. Тогда *факт, что из первого высказывания логически следует второе,* представляется следующим образом:

$$T_1\,(x_1^1) \circ T_2\,(x_2^1) \circ \langle \_\_,\ t,\ \Rightarrow,\ x_1^1,\ x_2^1,\ t \rangle, \qquad (2.10)$$

где *о*-вершина $\Rightarrow$ соответствует *логической импликации*. Сеть (2.10) будем записывать в более простом виде

$$T_1 \Rightarrow T_2 \qquad (2.11)$$

и называть логической продукцией. Такие продукции будем записывать с использованием введенных ранее мнемонических обозначений. Например,

$$\text{'}\textit{факт, что человек } X_1 \textit{ вышел из дому}\text{'} \Rightarrow \text{'}\textit{факт, что } X_1 \textit{ нет дома}\text{'} \qquad (2.12)$$

есть продукция, где левая и правая части — это сети, представляющие смысл записанных выражений. Здесь логическая зависимость отмечается словами *факт* и задается знаком $\Rightarrow$. Нетрудно видеть, что чем в (2.12) больше *н*-вершин, тем зависимость будет носить более обобщенный характер. Например, если вершину, соответствующую *дому*, заменить на *н*-вершину x_2^1, соответствующую *некоему неопределенному местонахождению*, то уже будет представлено *Факт, что человек* X_1 *вышел с некоего места* X_2, *влечет факт, что его нет в данном месте.*

Композицией логических и сетевых продукций могут представляться достаточно сложные зависимости логико-ситуационного плана. Например, продукция

$$T_3\,(z_1) \Rightarrow (T_1 \xrightarrow[r_1]{} T_2) \qquad (2.13)$$

представляет, что *изменения типа* R_1 *имеют место при выполнении условия* $\mathcal{T}_3$. При этом проверка такого условия связывается с заполнением слотов, т. е. нахождением значений *н*-вершин z_1. Полученные значения переносятся в сетевую продукцию $T_1 \xrightarrow[r_1]{} T_2$ (конечно, если в ней имеются *н*-вершины из набора z_1). В результате продукция уточняется. Продукция как бы настраивается на исходный объект. Соответствующая зависимость конкретизируется.

§ 2.2. СЕТИ, ЗАДАЮЩИЕ ЗНАЧЕНИЯ

В предыдущем параграфе упоминалось о необходимости специальных средств представления множеств перечисляемых объектов. Такие средства необходимы для записи результатов нахождения неопределенных компонент информации, для представления групповых отношений, корреляционных зависимостей и во многих других случаях [14, 24]. В данном параграфе будет начато рассмотрение таких средств, называемых *з*-сетями. Будут приведены некоторые простые примеры их использования, рассмотрены способы представления простых требований.

Многие виды деятельности человека связаны с постоянным уточнением, выявлением того, что имеется в виду в тех или иных случаях. Подобное уточнение, касающееся объектов и использующее схемы отношений, будем называть конкретизацией. Например, пусть речь идет о *друзьях братьев Ивана*. Ясно, что для их определения вначале нужно найти, какие братья имеются в виду. Пусть их оказалось множество $\{A_i, ..., A_n\}$. Тогда формируется уточненная информация, т. е. речь уже идет о *друзьях индивидуумов* $\{A_1, ..., A_n\}$. Процесс такого уточнения и есть конкретизация, в результате которой образуются уточняющие множества (в сложных случаях — семейства множеств). Для записи подобных множеств будем использовать специальные средства, называемые *з*-сетями.

В простейшем виде *з*-сеть записывается следующим образом:

$$[x_i := \{a_1, ..., a_n\}]. \tag{2.14}$$

Будем говорить, что *н*-вершина x_i имеет множество значений $\{a_1, ..., a_n\}$. С помощью *з*-сети представляется случай, когда неизвестными объектами, которым сопоставлена *н*-вершина x_i, оказались $\{A_1, ..., A_n\}$. Если в (2.14) заменить x_i на x_i^1, то будет представлено, что *объектом X_i является один из* $\{A_1, ..., A_n\}$.

Композиции с *з*-сетями. *З*-сети используются, как правило, в композиции с обычными сетями $T_1(x_i)$, с помощью которых представляются свойства и отношения объектов X_i. Композиция

$$T_1(x_i) \circ [x_i := [a_1, ..., a_n]] \tag{2.15}$$

представляет *множество объектов* $\{A_1, ..., A_n\}$, *которые связаны с другими объектами* так, как это представлено в $T_1(x_i)$. Пока мы еще не уточняем *все множество* или *некоторые его элементы*. Для такого уточнения требуются дополнительные средства (см. § 7.2).

Будем называть композиции вида (2.15) также сетями. Интерпретация таких сетей во многом зависит от представленной в T_1 информации, в частности, от указанного количества объектов. С помощью сети

$$T_1(x_i^1) \circ [x_i^1 := \{a_1, ..., a_n\}], \tag{2.16}$$

где x_i^1 соответствует одному объекту, представляется случай, когда *некий один объект множества* $\{A_1, ..., A_n\}$ (не обязательно только он) *связан с другими объектами так*, как указано в T_1. Заметим, что для проверки данного утверждения требуется перебор объектов или альтернатив с поиском подразумеваемого объекта. Вершины типа x_i^1 в ряде случаев будут задавать такие переборы. Отметим, что аналогичную роль зачастую играет и категория единственного числа в ЕЯ.

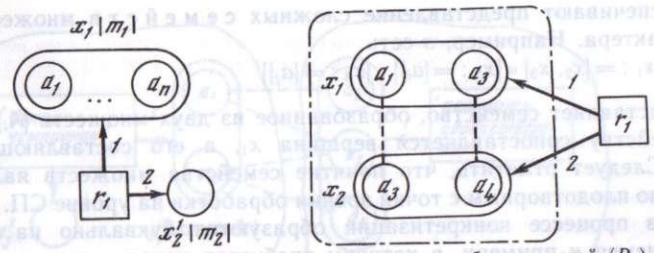


Рис. 2.7. Пример использования *з*-сетей для представления общих связей (R_1) множества объектов $\{A_1, ..., A_n\}$

Рис. 2.8. Конструкция с *з*-сетью, задающей согласованные значения пары

Будем изображать конструкции с *з*-сетями так, как показано на рис. 2.7. На нем изображена сеть, представляющая *объекты* $\{A_1, ..., A_n\}$, *относящиеся к классу* M_1 *и находящиеся в отношении* R_1 *с неким объектом класса* M_2. Если было бы указано, *что имеется в виду некий один объект из* $\{A_1, ..., A_n\}$, то на рис. 2.7 следовало бы заменить *н*-вершину x_1 на x_1^1. Итак, при изображении *з*-сетей значения *н*-вершины (множество других вершин) будем помещать внутрь изображающего ее кружочка. Как бы указывается на факт заполнения соответствующего слота, роль которого играет *н*-вершина.

Инверсные вершины, сложные значения. В дальнейшем будем использовать *з*-сети, у которых в множестве значений допускаются вершины различных типов: *о*-вершины, вершины с заданными значениями и так называемые инверсные вершины. Рассмотрим, что это такое. Будем понимать под инверсной такую вершину, которая соответствует *факту отсутствия объекта* A_i. Подобное отсутствие представляется с помощью фрагмента

$$\langle \_, t, \text{'не'}, d_1, a_i \rangle, \tag{2.17}$$

где d_1 и есть инверсная вершина. Будем обозначать ее в виде $\tilde{a}_i$. Такие вершины являются удобным средством представления разного рода отрицаний. Например, фрагмент $\langle \_, t, \text{'}$*любить*$\text{'}, a_1, \tilde{a}_2 \rangle$ представляет факт, что A_1 *любит не* A_2 (имеются в виду люди), а фрагмент $\langle \_, t, \text{'}\widetilde{\text{любить}}\text{'}, a_1, a_2 \rangle$ — что A_1 *не любит* A_2. Заметим, что последнее высказывание очень близко по смыслу к другому высказыванию *неверно* (*ложно*), *что* A_1 *любит* A_2, которое представляется в виде $\langle \_, f, \text{'}$*любить*$\text{'}, a_1, a_2 \rangle$. Такая близость определяет возможность сведения одних представлений к другим. Из приведенных примеров видно, каким образом инверсные вершины могут служить для представления разного рода смысловых оттенков.

Инверсные вершины могут служить для представления исключений, характерных для той или иной ситуации. Для этого они помещаются во множество значений какой-либо *н*-вершины. Например, сеть

$$T_1(x_1) \circ [x_1 := \{a_1, a_2, \tilde{a}_3\}]$$

представляет *свойства или отношения типа* $\mathcal{T}_1$, *которыми обладают объекты* A_1, A_2, *но не* A_3.

Рассмотрим еще один случай, когда в множестве значений одной вершины находятся другие вершины с заданными значениями. Подобные *з*-сети

обеспечивают представление сложных семейств множеств различного характера. Например, $з$-сеть

$$[x_1 := \{x_2, x_3\} \circ [x_2 := \{a_i\}] \circ [x_3 := \{a_j\}] \tag{2.18}$$

представляет семейство, образованное из двух множеств $\{A_i\}$ и $\{A_j\}$. Самому семейству сопоставляется вершина x_1, а его составляющим — x_2 и x_3.

Следует отметить, что понятие семейства множеств является чрезвычайно плодотворным с точки зрения обработки на уровне СП. Такие семейства в процессе конкретизации образуются буквально на каждом шагу. Вернемся к примеру, в котором требуется узнать, *кто является друзьями братьев Ивана*. Пусть оказалось, что у *Ивана два брата, каждый из которых имеет множество друзей, соответственно* $\{A_i\}$ и $\{A_j\}$. Подобная информация записывается в виде $з$-сети (2.18), где x_2 сопоставляется *множеству друзей одного брата*, x_3 — *другого*, а x_1 — *обоих*.

Заметим, что если в $з$-сети (2.18) заменить x_1 на $x_1^!$, то будет представлено альтернативное семейство. При этом будут иметься в виду *один из двух братьев* и соответственно *его друзья*, т. е. либо множество $\{A_i\}$, либо $\{A_j\}$. $З$-сети позволяют представлять и более сложные — многоуровневые семейства, т. е. суперсемейства, элементами которых являются семейства. Элементами последних могут быть как обычные множества, так и альтернативные. Словом, может быть представлена достаточно сложная информация. Более подробно вопросы такого представления описаны в работе [24].

Согласованные значения. Рассмотрим $з$-сети еще одного вида, у которых в левой части находится не одна $н$-вершина, а их набор: пара, тройка, ..., n-ка. Такие $з$-сети будут задавать согласованные значения наборов. Для пар они имеют следующий вид:

$$[(x_1, x_2,) := \{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}]. \tag{2.19}$$

Данная $з$-сеть представляет случай, когда под множеством пар неизвестных объектов, которым сопоставлены $н$-вершины x_1 и x_2, понимаются следующие: $(A_1, B_1), ..., (A_n, B_n)$. Другими словами, *если первым объектом является* A_1, *то вторым* — B_1 и т. д. И таких пар множество. Если в левой части $з$-сети поставить $(x_1, x_2)^1$, то будет пониматься *одна* пара из указанного множества. Подобные $з$-сети в композиции с обычными сетями будут служить для представления зависимостей. Например, сеть

$$\langle \_\_, t, r_1, x_1, x_2 \rangle \circ \langle (x_1, x_2) := \{(a_1, a_2), (a_3, a_4)\}] \tag{2.20}$$

представляет *отношение* R_1 *между двумя объектами* X_1 *и* X_2. При этом *если первым объектом является* A_1, *то вторым* — A_2 (и, наоборот), *а если первый есть* A_3, *то второй* — A_4. Эта сеть изображена на рис. 2.8 (пунктирные линии указывают на связанные пары). Такие сети могут служить для представления табличной информации, где данные столбцов связаны определенными отношениями. Например, если в (2.20) r_1 соответствует отношению *быть поставщиком детали*, то с помощью сети будет представлено A_1 *поставляет деталь* A_2, *а* A_3 — *деталь* A_4. В § 2.3 будут рассматриваться принципы представления реляционных данных и моделей.

Корреляционные продукции. В более сложном случае $з$-сети могут задавать согласованные значения n-ок $н$-вершин, которые сопоставляются связанным n-кам объектов. Подобные $з$-сети в композиции с продукциями позволяют задавать различные зависимости. В работе [14] они названы

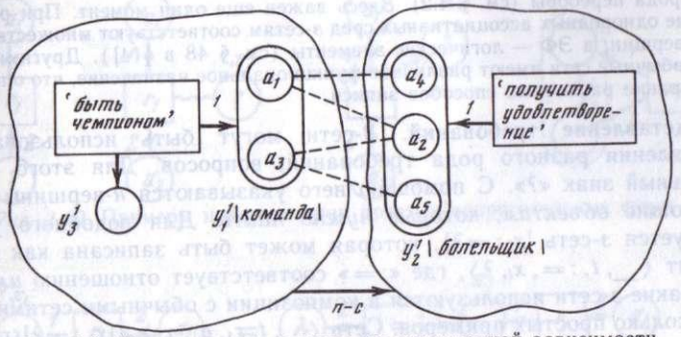


Рис. 2.9. Пример представления корреляционной зависимости

корреляционными продукциями. На рис. 2.9 приведен простой пример такой продукции. Она содержит $з$-сеть

$$[(y_1^1, y_2^1) := \{(a_1, a_2), (a_1, a_4), (a_3, a_2), (a_3, a_5)\}],$$

где через y_1^1 и y_2^1 обозначены $н$-вершины. Сама продукция представляет следующую информацию: *если команда* A_1 *станет чемпионом, то* A_2 *и* A_4 (*ее болельщики*) *получат моральное удовлетворение, а если команда* A_3, *то* A_2 *и* A_5 *получат удовлетворение* (имеется в виду, что A_2 болеет за обе команды).

Представленные зависимости могут носить более сложный характер. Может быть учтен год, когда проводился чемпионат, район или страна, в каком виде спорта и т. д. Соответственно действие *быть чемпионом* будет иметь множество семантических падежей, а вершина y_1^1 — более сложную окрестность. Болельщиков в реальном случае может быть не два, а многие тысячи. Соответственно $з$-сеть будет содержать в правой части большое количество значений — пар.

Рассмотрим еще один типичный случай. Пусть с помощью левой и правой частей продукции представлены две связанные ситуации, где одна заменяет другую. Тогда посредством $з$-сетей могут быть представлены заменяемые объекты (а также пары, тройки, ..., n-ки), которые в новой ситуации могут возникнуть в другом месте или в новом обличии. Например, *Если сегодня Иванов работает в ночь, то завтра он выходной. Если же сегодня он работает утром, то на следующий день работает в ночь*, и т. д. Подобные зависимости достаточно легко представляются с помощью корреляционных продукций, которые могут служить для прогнозирования — что будет делать Иванов, скажем, через три дня или в следующий четверг. Будет ли он работать?

Следует заметить, что $з$-сети можно достаточно легко записать с помощью обычных сетей. Например, $з$-сеть $[x_1 := \{a_1, a_2\}]$ записывается $\langle \_\_, t, :=, x_1, a_1 \rangle \circ \circ \langle \_\_, t, :=, x_1, a_2 \rangle$, где «:=» — $о$-вершина, соответствующая отношению *иметь значения*. Для записи $з$-сетей, задающих согласованные значения, конечно же, требуются более сложные сети (см. § 27 в работе [14]). Однако подобная запись неудобна. $З$-сети лучше задавать особым образом, как это было сделано ранее, расширяя допустимые синтаксические конструкции СЯ. Дело в том, что в общей организации систем с СП $з$-сети играют существенно другую роль, чем обычные ЭФ. $З$-сети в отличие от многих обычных ЭФ определяют способ идентификации вершин, задают

разного рода переборы (см. § 4.2). Здесь важен еще один момент. При реализации СП в виде однородных ассоциативных сред $з$-сетям соответствуют множества возбуждаемых вершин, а ЭФ — логические элементы (см. § 48 в [14]). Другими словами, $з$-сети и обычные сети имеют различное функциональное назначение, что оправдывает использование различных способов записи.

Представление требований. $З$-сети могут быть использованы для представления разного рода требований, вопросов. Для этого вводится специальный знак «?». С помощью него указываются $н$-вершины x_i, соответствующие *объектам, которые нужно найти*. Для подобного указания используется $з$-сеть $[x_i := ?]$, которая может быть записана как обычный фрагмент $\langle \_\_, t, :=, x_i, ? \rangle$, где «:=» соответствует отношению *иметь значение*. Такие $з$-сети используются в композиции с обычными сетями. Приведем несколько простых примеров. Сеть $\langle \_\_, t, r_1, a_1, x_1 \rangle \circ [x_1 := ?]$ представляет необходимость *нахождения объектов, связанных с* A_1 *отношением* R_1. Другая сеть $\langle \_\_, x_1^{\downarrow}, r_1, a_1, a_2 \rangle \circ [x_1^{\downarrow} := ?]$ представляет необходимость *определить, связаны ли объекты* A_1 *и* A_2 *отношением* R_1. Сеть $\langle x_1^{\downarrow}, t, r_1, a_1, a_2 \rangle \circ [x_1^{\downarrow} := ?]$ соответствует требованию *найти комплексный объект, состоящий из* A_1 *и* A_2 *с их отношением* R_1. Такие сети изображены на рис. 2.10.

Приведенные примеры в общем виде записываются $T_1(x_i) \circ [x_i := ?]$, где с помощью сети $T_1(x_i)$ могут быть представлены достаточно сложные отношения. Здесь требуется нахождение объектов X_i, что осуществляется с помощью специальной процедуры или механизма конкретизации. Последний на основе знаний (записанных в виде сетей) определяет значения $н$-вершины x_i (см. § 2.1).

В более сложных случаях требуется нахождение не одного объекта, а пары, тройки и т. д. Например, может потребоваться найти *пары объектов, связанных отношением* R_1. Для представления данного требования будем использовать сеть вида $\langle \_\_, t, r_1, x_1, x_2 \rangle \circ [(x_1, x_2) := ?]$, где с помощью $з$-сети задается необходимость нахождения согласованных пар или с о г л а с о в а н н ы х з н а ч е н и й. Указанная сеть изображена в правой части рис. 2.10, где средством указания на согласованность служит пунктирный контур.

Рассмотрим **требования** еще одного типа, в которых указывается на необходимость **проверки факта наличия** или существования объектов, связанных теми или иными отношениями. При этом в ряде случаев может быть указано (перечислено) все множество таких объектов. Подобные требования выражаются в виде *Имеется ли* (*существует ли*) *такой объект* (или *такой факт, такая ситуация*) X_1, *что ...*? Для их представления будем использовать специальный символ λ, означающий *имеется, существует*. $З$-сеть $[x_1 := \lambda]$ будет представлять, что *имеется объект* X_1. Для представления вопросительных форм типа *Имеется ли объект* X_1? будем использовать запись $[? \; x_1 := \lambda]$, которую также будем называть $з$-сетью. Например, с помощью сети вида

$$[? \; x_1 := \lambda] \circ \langle \_\_, t, r_1, x_1, a_1 \rangle \qquad (2.21)$$

представляется требование узнать, *имеются ли объекты, находящиеся в отношении* R_1 *с* A_1. Такая сеть изображена в левой части рис. 2.11, где символ λ перемещен внутрь $н$-вершины, а знак «?» вынесен вне ее.

Следует отметить, что введенная запись сводится к ранее рассмотренным конструкциям. Ее можно считать сокращенным обозначением сети, изобра-

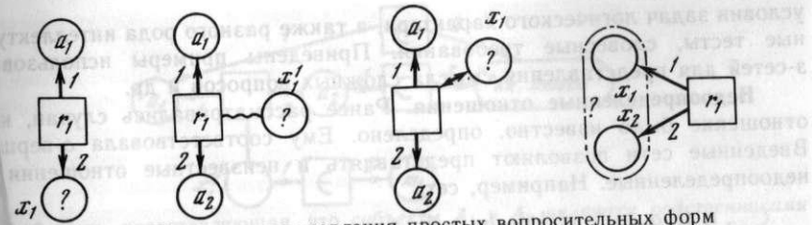


Рис. 2.10. Примеры представления простых вопросительных форм

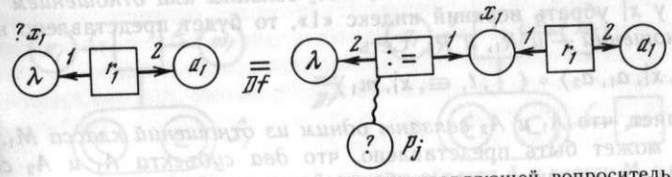


Рис. 2.11. Два способа изображения сети, представляющей вопросительную форму вида *Имеется ли объект, связанный отношением* R_1 *с* A_1?

женной в правой части рис. 2.11. В ней имеется фрагмент $\langle \_\_, p_j, :=, x_1, \lambda \rangle$, который при $[p_j := t]$ может быть записан $[x_1 := \lambda]$. В случае рис. 2.11 более правильной будет другая запись: $[? \; x_1 := \lambda]$.

Рассмотрим $з$-сети еще одного вида $[? \; x_1 := \{a_i\}]$ которые будут служить для представления вопросительных форм типа *Являются ли элементы множества* $\{A_i\}$ *объектами* X_1? В простейшем случае множество может состоять всего из одного элемента. Например, сеть

$$[? \; p_1 := t] \circ \langle \_\_, p_1, r_1, a_1, a_2 \rangle \qquad (2.22)$$

представляет требование узнать, *является ли истинным факт, заключающийся в том, что объекты* A_1 *и* A_2 *связаны отношением* R_1. С помощью сети

$$[? \; x_1 := \{a_i\}] \circ \langle \_\_, t, r_1, x_1, a_1 \rangle \qquad (2.23)$$

представлено требование узнать, *имеются ли среди* $\{A_i\}$ *такие объекты, которые находятся в отношении* R_1 *с* A_1. В общем виде у X_1 может быть множество свойств или отношений. Тогда в (2.23) вместо элементарного фрагмента будет стоять сложная сеть.

Отметим, что $[? \; x_1 := \{a_i\}]$ можно считать сокращенной записью следующей сети:

$$\langle \_\_, p_j :, =, x_1^{\downarrow}, x_k \rangle \circ [x_k := \{a_i\}] \circ [p_j := ?], \qquad (2.24)$$

т. е. они сводятся к введенным ранее конструкциям.

При изображении сетей, задающих требования, будем ставить знак «?» рядом с обозначением $н$-вершины. В остальном будем изображать их обычным способом.

Ранее рассматривались сравнительно простые примеры. Однако возможности введенных средств в плане представления различного рода требований, условий задач, зависимостей достаточно высоки. В частности, в работах [14, 24, 25] показано, что с помощью $з$-сетей могут быть представлены

условия задач логического характера, а также разного рода интеллектуальные тесты, словесные требования. Приведены примеры использования $з$-сетей для представления смысла сложных вопросов и др.

Недоопределенные отношения. Ранее рассматривались случаи, когда отношение было известно, определено. Ему соответствовала $о$-вершина. Введенные сети позволяют представлять и неизвестные отношения или недоопределенные. Например, сеть

$$\langle \_\_, t, x_1^1, a_1, a_2 \rangle \circ [x_1^1 := \{r_1, r_2\}] \tag{2.25}$$

представляет *факт, что объекты* A_1 *и* A_2 *связаны или отношением* R_1, *или* R_2. Если у x_1^1 убрать верхний индекс «1», то будет представлено *наличие обоих отношений — и* R_1, *и* R_2. Сеть

$$\langle \_\_, t, x_1^1, a_1, a_2 \rangle \circ \langle \_\_, t, \in, x_1^1, m_1 \rangle \tag{2.26}$$

представляет, что A_1 *и* A_2 *связаны одним из отношений класса* M_1. Таким способом может быть представлено, что *два субъекта* A_1 *и* A_2 *связаны родственным отношением*, что изображено на рис. 2.12. Классу родственных отношений сопоставлена $о$-вершина m_1, а имеющемуся в виду отношению — x_1^1. Наконец, рассмотрим еще один пример. Сеть

$$\langle \_\_, t, x_1^1, a_1, a_2 \rangle \circ \langle \_\_, t, x_1^1, a_3, a_4 \rangle \tag{2.27}$$

представляет, что *объекты* A_1 *и* A_2 *связаны тем же отношением, что и объекты* A_3, A_4.

Примеры иллюстрируют широкие возможности введенных средств. В более сложном случае отношения, как и объекты, могут задаваться косвенно — через другие отношения. Конечно, всегда нужно следить за местностью (арностью) отношения. Нельзя допускать, когда отношения различной местности соотносятся к одному классу или объединяются в одно множество.

Рекурсивные фрагменты. Они необходимы для представления цепочек транзитивных отношений. Последние могут рассматриваться как отдельные отношения. Например, пусть известно, что *справа от объекта A стоит некий объект, справа от которого находится B*. Мы говорим просто *B находится справа от A*, допуская, что между ними могут находиться другие объекты. Причем число таких объектов N, как правило, не фиксируется. С помощью отношения *находиться справа* как бы выражается цепочка, в которой количество звеньев не определено. Аналогично, если говорится о *части некоего объекта*, то, как правило, не уточняется, имеется ли в виду *непосредственная часть* или же *часть какой-либо части* и т. д. Предполагается возможность любого варианта.

Для представления указанных отношений введем специальные вершины и фрагменты. Пусть r_1 — $о$-вершина, соответствующая какому-либо бинарному отношению. Для представления цепочки таких отношений с количеством звеньев N будем использовать сети вида

$$[x_1 := r_1] \circ \langle \_\_, t, per, x_1, 'N' \rangle, \tag{2.28}$$

где *per* соответствует свойству *регулярности*, указывающему на возможность цепочки. $Н$-вершина x_1 сопоставляется такой цепочке. Будем записывать сеть (2.28) в более простом виде $[\hat{x}_1^N := r_1]$, подразумевая наличие второго

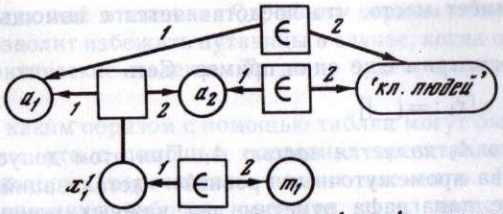


Рис. 2.12. Сеть, представляющая, что *субъекты* A_1 и A_2 *являются родственниками* ($о$-вершина m_1 соответствует *классу родственных отношений*)

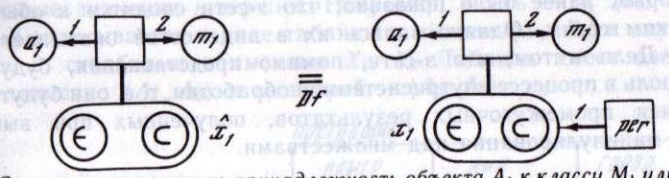


Рис. 2.13. Сети, представляющие *принадлежность объекта* A_1 *к классу* M_1 *или к одному из его подклассов*

фрагмента. Знак «$\wedge$» — указание на цепочку. Если число звеньев не фиксировано, то в (2.28) на месте 'N' будет стоять «__». Тогда сеть записывается следующим образом: $[\hat{x}_1 := r_1]$. Будем называть вершины типа $\hat{x}_1^N$, $\hat{x}_1$ и фрагменты ($з$-сети) с такими вершинами — р е к у р с и в н ы м и.

Введенные $з$-сети в композиции с обычными сетями будут служить для представления регулярных структур простого типа. Например, сеть

$$\langle \_\_, t, \hat{x}_1, a_1, a_2 \rangle \circ [\hat{x}_1 := r_1] \tag{2.29}$$

представляет, что *объекты* A_1 *и* A_2 *связаны цепочкой отношений типа* R_1 (в частном случае допускается всего одно звено в цепочке). Если r_1 соответствует отношению *находиться непосредственно слева*, то будет представлено, что A_1 *находится слева от* A_2. При этом между A_1 и A_2 допускается наличие любого числа других объектов. Если в (2.28) на место $\hat{x}_1$ подставить $\hat{x}_1^N$, то количество таких объектов будет зафиксировано числом N.

В дальнейшем будем допускать цепочки, состоящие из нескольких типов отношений. Для их представления достаточно введенных ранее средств. Например, сеть

$$\langle \_\_, t, \hat{x}_1, a_1, m_1 \rangle \circ [\hat{x}_1 := \{\in, \subset\}] \tag{2.30}$$

представляет *принадлежность объекта* A_1 *к классу* M_1. При этом такой класс может включать ($\subset$) в себя подклассы, а те — свои подклассы, для которых могут быть указаны элементы ($\in$). Фактически допускается любая иерархическая структура родовидового типа. На рис. 2.13 проиллюстрировано два способа изображения сети (2.30).

Аналогично могут представляться **комплексные отношения** типа *находиться сверху с левой стороны* и т. д. При этом будет оставаться некоторая неопределенность, т. е. допускаются различные альтернативы расположения объектов. Важно, чтобы существовала указанная цепочка. Тогда считается,

что отношение имеет место, что представляется с помощью рекурсивного фрагмента.

Наконец, рассмотрим еще один пример. Сеть

$$\langle a_1, t, \hat{x}_1, a_2, \rangle \circ [\hat{x}_1 := \{\_\}] \qquad (2.31)$$

представляет, что A_2 *является частью* A_1. При этом допускается наличие любого количества промежуточных уровней — детализаций.

В заключение параграфа отметим два важных момента. Во-первых, введенные средства и способы представления не следует считать единственно возможными. Рассматривался лишь один из приемлемых вариантов. И, во-вторых, ранее было показано, что s-сети сводятся к обычным семантическим сетям. Однако запись их в виде сетей оказывается менее удобной. Дело в том, что s-сети, помимо представления, будут играть важную роль в процессе внутрисистемной обработки, т. е. они будут служить для записи промежуточных результатов, полученных при выполнении действий манипулирования над множествами.

§ 2.3. РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ

Введенный выше язык кортежной записи семантических сетей достаточно близок к модели данных, используемых в реляционных базах [10, 47]. Ниже будут проводиться четкие параллели, которые распространяются и на методы обработки. Будут анализироваться причины трудностей, которые возникают при попытках расширения реляционных баз до баз знаний.

Таблицы-отношения. Напомним, что в реляционных моделях данные записываются в виде именованных таблиц, состоящих из именованных столбцов. Таблица, содержащая n столбцов, представляет n-местное отношение, конкретные экземпляры которого записаны в строках. На рис. 2.14 приведен пример таблицы с именем «поставщик». Таблица представляет трехместное отношение. Она содержит три столбца, имеющих имена «номер», «имя», «город». Последние называются атрибутами отношения, а данные, записанные в клетках таблицы,— значениями атрибутов. Каждая строка представляет конкретное отношение. Например, первая строка указывает, что *поставщик, которому присвоен условный номер* A1, *имеет имя Смит и проживает в Париже.*

Покажем, что введенный ранее язык кортежной записи семантических сетей сводится к реляционной модели данных. В самом деле, множество кортежей можно записать в виде таблицы-отношения. Вершины (коды), находящиеся на одной и той же позиции, поместим в один столбец. В получившейся таблице будут столбцы, которые условно можно назвать *быть кодом ситуации, быть логической составляющей, быть кодом отношения, быть первым объектом отношения* и т. д. Из такой таблицы выделим все строки, содержащие на третьей позиции (в третьем столбце) один и тот же код отношения r_i. Выделенные строки сгруппируем в собственную таблицу, из которой изымем третий столбец, а код отношения присвоим в качестве имени таблицы — R_i. Тогда будет получено множество таблиц-отношений с различными именами. В каждой из них в именах столбцов (атрибутах) укажем имя отношения: «быть кодом ситуации R_i», «быть логической составляющей отношения R_i», «быть первым объектом отношения R_i» и т. д. Это позволит избежать путаницы в случае, когда отношения имеют различные семантические падежи. Тогда и будет получена реляционная модель данных.

Рассмотрим, каким образом с помощью таблиц могут быть представлены s-сети. Напомним, что с помощью последних представляются множества объектов, для которых перечислены все их элементы. В общем случае s-сети задают значения наборов n-вершин, т. е. представляются пары, тройки, ..., n-ки (см. § 2.2). Такие наборы можно считать именами таблиц, а входящие в них n-вершины — атрибутами. В строках записываются согласованные значения n-вершин. Например, s-сеть вида $(x_1, x_2) := \{(a_1, a_2), (a_3, a_4)\}$ записывается в виде таблицы с именем (X_1, X_2) и двумя атрибутами, т. е. столбцами с именами X_1, X_2. Таблица имеет две строки: в первой записано (A_1, A_2) и во второй — (A_3, A_4).

ПОСТАВЩИК

НОМЕР	ИМЯ	ГОРОД
A1	СМИТ	ПАРИЖ
A2	СМИТ	ЛОНДОН
A3	КЛАРК	ПАРИЖ
⋮		

Рис. 2.14. Пример реляционного «отношения»

Возможность сведения одного представления к другому говорит о родстве подходов. Значит, и обработка должна осуществляться родственным способом. Ниже (в § 3.2 и 3.3) для обработки сетей будет введен операторный язык, в котором для ряда базовых операторов имеются аналоги в языке реляционной алгебры. Однако есть и существенные отличия, прежде всего связанные с ориентацией на формы ЕЯ, с необходимостью использования обобщенной информации. Отсюда вытекает потребность в средствах, выходящих за рамки реляционных методов и моделей.

Отличия в представлениях. Выше в семантические сети были введены вершины различных типов — соответствующие определенным, неопределенным объектам и отношениям, их классам. В § 2.1 были рассмотрены продукции, обеспечивающие представление динамики, причино-следственных зависимостей и т. д. Ниже (в гл. 5) будут введены средства представления разного рода гипотез, закономерностей. Подобные средства необходимы во многих случаях, например когда требуется описание изменяющихся схем поставок, когда предприятие рассматривается как сложный объект, в котором имеются взаимосвязанные части — цеха, когда объект носит динамический характер, и т. д. Здесь отдельных операционных данных, на которых основаны реляционные модели, оказывается явно недостаточно.

Итак, одно из отличий — в возможностях поддержания сложных концептуальных моделей. Еще одно важное отличие состоит в наличии согласованных представлений. Отсюда следуют возможности автоматического ввода информации на ЕЯ, что называется, автоматической разработки (конструирования) базы за счет ЕЯ. При традиционном реляционном подходе такие возможности недостаточны. Предполагаются фиксированные спо-

собы группировки данных — в рамках введенных таблиц-отношений (схем). Затрудняются типовые формы структурирования, связанные с делением на сценарии, с упорядочением по времени, по отношению *часть—целое* (завод—цех), по SUB-оси и т. д. При этом следует помнить, что каждый сценарий того или иного типа может иметь свои особенности, характерные черты. Их тоже нужно каким-то образом представлять и использовать при вводе информации — для выявления семантических ошибок. Конечно, для этой цели явно недостаточно табличных данных. Требуется развитие реляционного подхода — в направлении семантических представлений, моделей.

Напомним теперь некоторые особенности кодирования объектов и отношений в рамках систем с СП (см. § 1.2). Каждому объекту сопоставляется своя вершина семантической сети, играющая роль кода объекта. Два различных объекта всегда будут иметь различные коды, т. е. им будут сопоставлены две различные вершины. Например, если поставщиков с именем Смит много, то все они будут иметь свои коды. То же самое справедливо для городов, деталей и т. д. Их коды могут не иметь выхода на внешний язык. Пользователю совершенно не интересно знать, как закодированы данные, как они хранятся. Он хочет получить то, что ему нужно, используя свои знания. Как правило, это знания о именах поставщиков, о местах их жительства, о поставляемых деталях, схемах поставок и т. д. Отсюда необходимость четкого различения внутренних представлений (или моделей), где фигурируют внутрисистемные коды, и внешних, где объекты определяются с помощью общепринятых понятий. По этим понятиям (с учетом конструкции составленного из них предложения ЕЯ) должны каким-то образом выделяться объекты, о которых идет речь. Такое выделение — задача самой системы. Здесь следует использовать окрестностный подход.

§ 2.4. АЛГОРИТМИЧЕСКИЕ КОНСТРУКЦИИ

Данный параграф посвящен описанию принципов использования сетей для представления выражений некоторых алгоритмических языков. Аналогичные принципы могут служить для представления точных предписаний ЕЯ. Речь будет идти об организации уровня (СС), на котором может обеспечиваться компоновка алгоритмов, проверка их семантической правильности, устранение ошибок, поиск решения (по соотношениям). Такой уровень крайне необходим в «благожелательных» или концептуальных системах программирования. Рассмотрим вначале основную идею для наиболее простого случая.

Выражения (предписания) рассматриваются как наборы соотношений, которые представляются в явном виде. Например, выражение $X2=(X1+5)\times 10$ разбивается на соотношения $X3=X1+5$, $X2=X3\times 10$, которые и представляются с помощью сетей. При этом порядок нахождения значений переменных может быть не задан вовсе или же задан лишь частично. Предполагается, что система сама должна уметь достраивать порядок нужным ей образом. Например, по соотношению $X3=X1+5$ могут определяться как значения переменной $X3$, так и $X1$. Система сама должна решать, как лучше делать, учитывая характер текущей задачи.

Такой подход, видимо, впервые нашел свое воплощение в вычислительных (семантических) моделях Тыугу [45, 46], где используется специальный язык записи соотношений — УТОПИСТ. В этом языке допускаются неопределенные объекты. Для согласования данных вводятся специальные описатели типов данных. Вводятся средства записи разного рода программных соотношений. Предусматривается возможность описывать задачи, не задавая явно алгоритма его решения. Для этого вводится специальный оператор задачи. С помощью него указывается, что является аргументом, а что результатом. Выбор наилучшей стратегии вычисления или поиска поручается самой системе — так называемой инструментальной системе программирования ПРИЗ. Ее основное назначение — служить интеллектуальным партнером программисту.

Системы программирования на базе СП. Покажем, каким образом можно повысить возможности упомянутого интеллектуального партнера. Для этого вернемся к концепции систем с СП (см. § 1.2). Напомним некоторые особенности таких систем. Различается внешний язык, на котором записываются разного рода соотношения, программы, указания, и внутренний системный язык, на котором вся вводимая информация представляется в явном виде. Внутренний язык необходим не только для того, чтоб избавиться от вариабельности внешнего языка, особенностей его конструкций. Его важная роль — в обеспечении внутрисистемной обработки: проверки правильности (допустимости) алгоритма, выбора направления уточнения (вычисления) компонент. Еще одна немаловажная роль — обеспечение диалогового режима, в частности поиск ответа на запросы о уже введенных в систему компонентах алгоритма, выполнение отдельных шагов алгоритма с выдачей результатов и др. Перечисленные функции требуют разработки специального внутреннего языка, что уже отмечалось во введении. В качестве него мы будем использовать семантические сети и графы (сети с указанием направления уточнения компонент).

Сети (графы) — это чисто внутрисистемный язык, о котором пользователь не должен знать ничего. Пользовательский инструментарий — удобный ему внешний язык, который должен расширяться по мере необходимости. Система должна предоставить пользователю возможность создавать свой внешний язык. Критерий удобства пользователю предопределяет особенности такого языка — наличие сокращенных форм записи информации, широкое использование разного рода внелингвистических сведений. В частности, так ли нужно обязательно описывать типы всех данных? В ЕЯ этого нет. По объекту (данным) в большинстве случаев может быть определен его класс (тип данных). Это должна делать сама система. В тех редких случаях, когда возникают неоднозначности, они могут быть устранены организацией специального диалогового режима.

В рамках систем с СП проблема создания языка может быть решена путем накопления специальных структур, играющих роль лингвистических знаний. Они определяют процесс преобразования выражений внешних языков во внутреннее представление, т. е. в сети. В работах [14, 25] показано, что роль таких знаний могут играть наборы специальных корреляционных продукций (см. § 2.1). Напомним, что продукции — это такие же сети, которые могут формироваться и корректироваться через внешний язык. Предполагается, что в последнем имеются средства определения значения выражений этого языка (кроме самих себя). Преобразованием таких средств во внутреннее представление и обеспечивается ввод и корректировка продукций. При этом сами средства определения могут быть сделаны доста-

точно типовыми, удобными, в частности, они могут быть приближены к естественным формам — таким же способом, как и обычные выражения внешнего языка. Отметим, что в языках программирования для аналогичных целей применяются макроопределения, пользование которыми доступно лишь узкой категории системных программистов.

Наличие единых, однородных представлений перспективно с точки зрения отладки программ. Ввод частей алгоритма может осуществляться в достаточно произвольной последовательности. Более того, эти части могут быть записаны на разных внешних (алгоритмических) языках. Они преобразуются во фрагменты одного и того же языка (СЯ), которые могут стыковаться между собой достаточно простым способом. В результате формируется сеть, соответствующая всему алгоритму. При таком подходе сети играют роль языка посредника. Последний может служить основой для организации так называемой многоязыковой системы программирования, т. е. системы, допускающей ввод алгоритма, части которого записаны на разных языках. Такие части указанным ранее способом преобразуются во внутреннее представление и «стыкуются» на уровне семантических сетей.

Остановимся еще на одном важном вопросе, связанном с **семантической правильностью** или допустимостью вводимых выражений (соотношений). Ясно, что выражение может описывать законченный алгоритм, если в нем нет висячих операторов, тупиковых путей, неиспользуемых данных, если любой переменной присваиваются или уже присвоены какие-либо значения, если выполняются проверки на допустимость типов данных и т. д. Конечно, проверку таких условий или критериев можно полностью возложить на пользователя (во многих системах программирования именно так и делается). Но думается, что такой путь далеко не лучший. Здесь имеется еще одно соображение. Дело в том, что критерии семантической правильности в открытых (расширяющихся) системах сами будут меняться, например, по мере создания внешнего языка. Учесть такие изменения заранее невозможно. Отсюда единственный выход — организация автономной внутрисистемной обработки, создание специального рода системных знаний, представляющих условия, критерии.

В дальнейшем роль указанных знаний будут играть семантические графы (см. § 3.4). Они образуют внутренние представления, к которым пользователь имеет доступ через свой внешний язык. Словом, пользователю предоставляется возможность вводить критерии «правильности». Более того, допускается самостоятельное формирование системой подобных критериев, что обеспечивается путем формирования по сетям графов, называемых обязательными (см. § 7.4). Для этого вводится специальный режим обучения. Пользователь начинает играть роль учителя, который дает на вход системы правильно построенные конструкции внешнего языка. Система преобразует их во внутреннее представление и по последним строит структуры (графы), соответствующие гипотезам о правильности. Эти гипотезы могут быть выданы пользователю. После консультации с ним упомянутые структуры могут быть возведены в ранг системных знаний. Тогда они начинают играть роль решающих правил, обеспечивающих автоматическую проверку правильности последующих входных конструкций внешнего языка. В ранней работе [23] такие структуры были названы *Т*-графами. Последние есть разновидность семантических графов.

Уровни представления. Остановимся на принципах использования сетей для представления типовых алгоритмических конструкций. Этот вопрос интересен с двух точек зрения, во-первых, с точки зрения организации уровня внутрисистемной обработки (СП или языка-посредника) в новом классе концептуальных систем программирования. И, во-вторых, раз речь идет о языках и системах программирования, то возникает заманчивая идея их использования при проектировании, разработке самой системы. В частности, следует допускать случай, когда вводимые алгоритмы отображаются на внутрисистемные механизмы и приводят к пополнению, совершенствованию внутрисистемной деятельности, в частности, связанной с поиском, корректировкой и др. (см. § 3.1).

Что же нужно представлять? Известно, что любые выражения какого-либо внешнего языка — это сложный (структурный) объект, который может быть рассмотрен по-разному, с различных точек зрения, и соответственно представлен на различных уровнях. К примеру, слово *деталь* может рассматриваться, во-первых, как последовательность букв, во-вторых, как языковая конструкция, обладающая определенными характеристиками (как слово, которое находится в именительном падеже, имеет женский род и т. д.). И в-третьих, слово *деталь* имеет свой денотат или семантический эквивалент, т. е. оно называет определенную вещь или некоторого представителя указанного класса. Например, речь может идти о *конкретной детали, которая должна быть сделана или делается*. В другом контексте может говориться об *абстрактном представителе класса деталей, его свойствах*, которые в общем-то переносятся на свойства всего класса.

Соответственно с указанными способами рассмотрения будем различать **три уровня** представления слова, которые будем называть уровнями поверхностных структур (ПС), синтаксических структур (ТС) и семантических структур (СС), вкладывая в эти понятия свой смысл. Сети будут использоваться на всех уровнях. На рис. 2.15 изображены уровни представления слова *куб*. На уровне ПС представляются пространственные отношения между буквами. Конкретным экземплярам букв сопоставляются собственные вершины x_1^1, x_2^1, x_3^1. Принадлежность экземпляров к классам букв К, У и Б (соответствующие вершины обозначены через '*К*', '*У*', и '*Б*') представляется с помощью фрагментов типа $\langle \_,\ t,\ \in,\ x_1^1,\ 'K' \rangle$, $\langle \_,\ t,\ \in,\ x_2^1,\ 'У' \rangle$ и $\langle \_,\ t,\ \in,\ x_3^1,\ 'Б' \rangle$. Подобное представление позволяет избежать путаницы, возникающей при наличии нескольких однотипных букв в одном и том же слове. Такие буквы рассматриваются как различные экземпляры, относящиеся к одному и тому же классу. Последовательность экземпляров представляется с помощью фрагментов $\langle \_,\ t,\ л\text{-}п,\ x_i^1,\ x_{i+1}^1 \rangle_т$, где *л-п* — вершина, соответствующая отношению *находится слева—направо*.

На уровне ТС представляются грамматические формы или категории слова (см. рис. 2.15). При этом опять же рассматривается экземпляр слова *куб*, которому сопоставляется своя вершина x_4^1. Если в предложении имеется несколько однотипных слов (например, *куб стоит на кубе*), то им будут сопоставлены различные вершины, для которых будет указана принадлежность к одному и тому же классу *кубов*. Такая принадлежность представляется с помощью фрагмента $\langle \_,\ t,\ сл,\ x_4^1,\ 'куб' \rangle$, где *сл* — *о*-вершина, соответствующая отношению *быть словом определенного класса* (или *словоформой*), а '*куб*' — *о*-вершина, соответствующая такому классу.

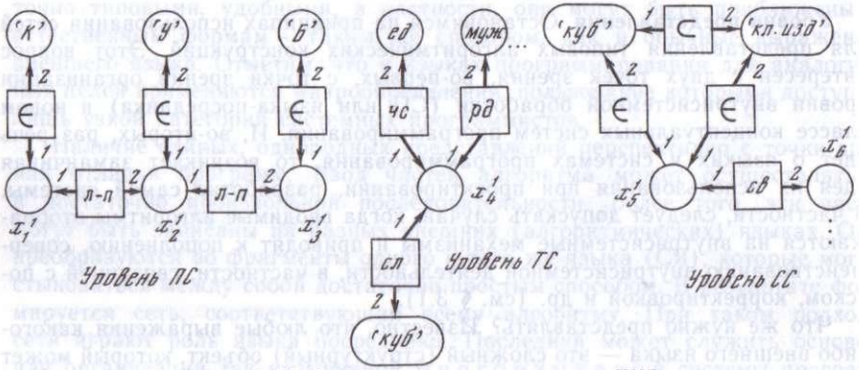


Рис. 2.15. Уровни представления слова *КУБ*

На уровне СС вводится еще одна вершина x_5^1, соответствующая денотату и фактически играющая роль кода объекта, который имеется в виду. При этом представляются уже семантические категории, в частности принадлежность к семантическим классам — *классу предметов* ('*кл. изд*'), *который включает в себя* ($\supset$) *класс кубов* ('*куб*'). На этом же уровне представляются свойства конкретного куба, его отношения и т. д.

Аналогично представляются более сложные языковые конструкции (словосочетания), отдельные предложения (выражения) внешнего языка, группы предложений (тексты). Их также следует представлять на различных уровнях. Также можно выделить три основных уровня. В частности, на уровне ПС представляются слова (их буквенный состав, пространственное расположение слов в предложении) и предложения (их словесный состав, пространственное расположение предложений в тексте). При этом более или менее самостоятельным компонентам (буквам, словам, словосочетаниям...) сопоставляются собственные вершины, которые связываются между собой фрагментами. В результате образуется достаточно сложная структура, которая сама образует собственные слои («подуровни») по отношению *часть—целое*.

На уровне ТС представляется грамматическая конструкция предложения или всего текста. К вершинам, соответствующим словам, словосочетаниям, предложениям, подсоединяются фрагменты, представляющие их грамматические категории, виды и т. д. На уровне СС представляются те отношения между реальными объектами, которые подразумеваются или имеются в виду. На нем каждой осмысленной части предложения сопоставляется собственная вершина, соответствующая денотату этой части. Например, в словосочетании *Куб стоит на кубе* имеются две таких части (два куба), связанные отношением *стоять на*. Если предложение описывает отдельную ситуацию (более сложную фигуру), которая в дальнейшем играет самостоятельную роль, то появляется третья часть. В другом словосочетании *сын брата Петра* также имеются три таких части: *Петр, его брат, сын брата*. Всем им будут сопоставлены свои вершины, соответствующие реальным (известным или неизвестным системе) людям. Такие представления мы называли согласованными.

Описанные уровни необходимы прежде всего с точки зрения организа-

ции **продукционных лингвистических процессоров** [24]. Не останавливаясь на них подробно, отметим лишь некоторые их важные особенности. Основное назначение таких процессоров — преобразование выражений внешнего языка на уровень СС. Для этого вначале строится его ПС, представляющая буквенный состав, для чего используется специальная процедура. Затем строится ПС, представляющая словесный и текстовый состав. Для этого используются наборы продукций, выявляющие самостоятельные буквосочетания, по ним — отдельные слова и словосочетания и т. д. При этом анализируется наличие пробелов, знаков препинания и других элементов, играющих важную роль в выявлении законченных конструкций.

Далее осуществляется преобразование выражений с уровня ПС на уровень СС. Для этого используется два набора продукций, определяющих два этапа преобразования сетей: с уровня ПС на уровень ТС и с уровня ТС на уровень СС. Фактически на первом этапе осуществляется морфологический и некоторые виды синтаксического анализа, а на втором — синтаксический и некоторые виды семантического анализа. Отметим, что анализ допустимости представленных соотношений — это отдельная задача.

Следует обратить внимание, что описанное деление не совсем согласуется с аналогичными понятиями лингвистики [33]. Лингвисты вкладывают в понятия ПС и СС собственное содержание, выделяют дополнительные уровни: поверхностных синтаксических структур, глубинных синтаксических структур и т. д. Причина несоответствия видится в следующем. Выбранное нами деление имеет чисто прагматические цели — построение лингвистических процессоров продукционного типа. В то же время в лингвистике уровни во многом определяют характер последующих исследований, т. е. имеют исследовательскую направленность.

Уровни алгоритмических языков. Неправильно думать, что сказанное выше относится лишь к ЕЯ. Описанные уровни необходимы и для перевода алгоритмических языков (АЯ). Последние в общем-то своими корнями уходят в ЕЯ, т. е. из ЕЯ зачастую заимствуются удобные формы (категории), которые уточняются, формализуются и переносятся в АЯ. Во многих АЯ имеются текстовые данные, разного рода идентификаторы — законченные (базовые) конструкции, состоящие из символов алфавита,— букв, цифр. Их можно считать некоторым подобием слов. Эти конструкции делятся на классы, из них составляются более сложные конструкции (таблицы, схемы и др.), с помощью которых могут выражаться разного рода соотношения. Для представления таких конструкций также оказывается удобным деление на три уровня описанного выше типа.

Например, возьмем таблицы, рассмотренные в § 2.3. На уровне ПС представляются пространственные отношения между клетками, соотнесенность клеток к строкам и столбцам, а также их содержимое, т. е. какие символы (данные) в них записаны. На уровне ТС представляются типы данных, их принадлежность атрибутам. На уровне СС представляются выражаемые отношения (см. рис. 2.15).

Те же уровни присущи ч и с л а м. Число 338 может рассматриваться как последовательность цифр (уровень ПС). Тогда оно представляется, как показано на рис. 2.16, где *о*-вершины '3' и '8' сопоставлены классам цифр, а *н*-вершины x_1^1, x_2^1 и x_3^1 — конкретным их экземплярам. С помощью фрагментов $\langle \_\_, t,$ л-п, $x_i, x_{i+1} \rangle$ представлено отношение между этими экземплярами — *находиться слева—направо* (*л-п*).

Цифры, из которых составляются числа, могут быть отнесены к собственным классам — единиц, десяток, сотен, ... Тогда к рис. 2.16 будут добавлены соответствующие фрагменты, составляющие окрестности вершин x_1^1, x_2^1 и x_3^1. Образуется уровень ТС.

В то же время число выражает нечто свое, т. е. на уровне СС каждому числу должна быть сопоставлена своя вершина. Как будет показано ниже, с помощью таких вершин и представляются разного рода арифметические соотношения. При этом окрестность вершины должна определять, что это за число. К примеру, числу 338 сопоставляется вершина γ_1 сети $\langle \gamma_1, t, б.\ чис., '3', '3', '8' \rangle$, где *б. чис.*— быть целым (трехразрядным) числом, а '3', '8' — *о*-вершины, соответствующие цифрам. На уровне СС такие цифры приобретают сущность и их различения по экземплярам (как это делалось на уровне ПС) не требуется. Будем обозначать вершину γ_1 через '338'.

Возможны и другие способы представления. Например, можно указывать разряды единиц, десятков, сотен и т. д., а также принадлежность к ним цифр. Для представления дробных чисел можно использовать специальные *о*-вершины, соответствующие фиксированной или плавающей запятой. Могут быть использованы специальные отношения *быть целой частью* (*б. цч*) и *быть дробной частью* (*б. дч*). Дробное число рассматривается как состоящее из этих двух частей. Например, числу 2,5 сопоставляется вершина d_i сети: $\langle \_\_, t, б.\ цч, d_i, '2' \rangle \circ \langle \_\_, t, б.\ дч, d_i, '5' \rangle$. Такую вершину обозначим через '2,5'.

Итак, на уровне СС любому числу сопоставляется собственная вершина. Сказанное относится и к любым другим данным, в том числе записанным в таблицах, а также имеющим вид таблиц. В последнем случае всей таблице сопоставляется собственная вершина, окрестность которой представляет структуру и содержимое таблицы. С помощью таких вершин могут быть представлены различные операции над таблицами.

Отметим, что преобразование алгоритмических конструкций с уровня ПС на уровень СС может строиться на тех же принципах, что и преобразование предложений ЕЯ. Для этого могут также использоваться наборы продукций. Только эти наборы будут значительно проще, чем для ЕЯ, так как принципы составления предложений ЕЯ значительно сложнее, чем выражений АЯ. В связи со сказанным удобно считать АЯ некоторой частью ЕЯ, его дополнением, заметно более простым, чем сам ЕЯ.

Подробнее вопросы использования сетей для организации уровней ПС и ТС рассмотрены в работах [14, 24]. Сейчас нас будет интересовать только уровень СС, т. е. способ представления семантических эквивалентов, выражаемых с помощью алгоритмических средств.

Представление арифметических выражений. Уже говорилось о том, что на уровне СС числам сопоставляются собственные вершины, знакам операций и переменным также сопоставляются свои вершины. Будем обозначать *о*-вершины, сопоставленные операциям $+$, $\times$, :, ... теми же самыми знаками. Обозначим *н*-вершины, сопоставленные переменным $X1$, $X2$, ..., через x_1^1, x_2^1, ... Каждая переменная может иметь только одно значение. Поэтому им должны сопоставляться *н*-вершины с верхним индексом «1».

С помощью перечисленных вершин могут быть представлены различные арифметические выражения. Например, соотношение $(X1+5)\times X3=X2$ представляется в виде сети рис. 2.17. На ней *н*-вершина x_3^1 соответствует

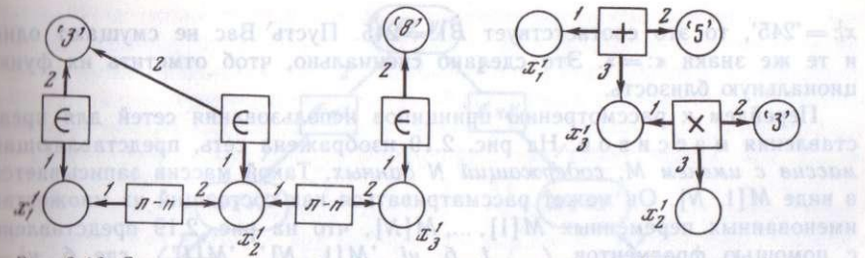


Рис. 2.16. Сеть, представляющая поверхностную структуру (ПС) числа 338
Рис. 2.17. Сеть, представляющая $(X1+5)\times 3=X2$

значению выражения $X1+5$, т. е. результату сложения. Физически представляются два соотношения $X1+5=X3$, $X3\times 3=X2$ без указания, в каком порядке следует находить значения переменных (вершин). Нетрудно видеть, на рис. 2.17 никак не представлены скобки. Это сугубо синтаксические средства, которые в рамках описываемого подхода необходимы только для правильного преобразования выражений с уровня ПС на уровень СС. То же самое можно сказать о знаке равенства значений «=», который в данном случае необходим лишь для правильной расстановки соответствующих вершин в фрагментах и кортежах.

Для представления тождеств «≡», определяющих эквивалентное преобразование выражений, используются продукции. В частности, тождество вида

$$((X1+X2)\times X3=X5)\equiv(X1\times X3+X2\times X3=X5) \qquad (2.32)$$

представляется как показано на рис. 2.18. В таком тождестве $X1$, $X2$, $X3$, $X5$— абстрактные обозначения. В зависимости от вида преобразуемого выражения на места $X1$, $X2$, $X3$ и $X5$ ставятся конкретные переменные, взятые из этого выражения. Такая подстановка реализуется в процессе применения продукции к сети T_1, представляющей преобразуемое выражение.

Тождества находят широкое распространение для преобразования выражений (соотношений) в форму, удобную с точки зрения вычисления. Такое преобразование обеспечивается соответствующим набором продукций, которые могут служить для получения решения в аналитическом виде. Система сама будет искать приемлемое решение.

Средства именования. Во многих алголоподобных языках для обозначения разного рода обособленных компонент используются имена или идентификаторы $B1$. Они могут служить для записи конкретных данных, что выражается $B1:=245$ (переменной присваивается значение 245). Такое соотношение представляется в виде $\langle \_\_, t, и.\ зн, 'B1', '245' \rangle$, где '$B1$' — вершина, соответствующая идентификатору, '245' — его значению, а *и. зн* — отношению *иметь значение*. Идентификаторы служат для обозначения конкретных переменных, входящих в какие-либо выражения. Такое вхождение представляется в следующем виде: $T_1(x_i^1) \circ \langle \_\_, t, и.\ зн, 'B1', x_i^1 \rangle$, где сеть $T_1(x_i^1)$ соответствует выражению, а x_i^1 — неизвестному (незаданному) значению $B1$. Процесс нахождения значений переменных сводится к нахождению значений *н*-вершин. Например, если было найдено

$x_i^1 := '245'$, то это соответствует $B1 := 245$. Пусть Вас не смущают одни и те же знаки «:=». Это сделано специально, чтоб отметить их функциональную близость.

Перейдем к рассмотрению принципов использования сетей для представления массивов. На рис. 2.19 изображена сеть, представляющая *массив с именем M, содержащий N данных*. Такой массив записывается в виде $M[1, N]$. Он может рассматриваться как состоящий из множества именованных переменных $M[1], ..., M[N]$, что на рис. 2.19 представлено с помощью фрагментов $\langle \_\_, t, б.чi, 'M[1, N]', 'M[J]' \rangle$, где *б. чi — быть i-й частью* ($i = 1, ..., n$). Каждая переменная имеет свое значение, которому сопоставлена собственная вершина x_i^1. Данные, которые записаны в массиве, представляются в виде значений вершин x_i^1.

Описатели массивов представляются с помощью специальных фрагментов. Общий описатель типа integer (*целый*) $M[1, N]$ представляется в виде $\langle \_\_, t, св, 'M', 'целочисл'. \rangle$, т. е. как некоторое свойство *быть целочисленным*. Однако такое свойство должно быть перенесено на содержимое массива. Для этой цели на рис. 2.19 использованы фрагменты $\langle \_\_, t, \in, x_i^1, 'кл.\ целых\ чс.' \rangle$, с помощью которых представлена *принадлежность* ($\in$) *значений, величин M[J] к классу целых чисел.*

Аналогичным образом представляются и более сложные массивы с их описателями — двумерные, ..., n-мерные. При этом возможны различные способы представления. Например, можно не указывать составные части массива, сразу представляя данные. Тогда сеть будет содержать фрагменты $\langle \_\_, t, u.\ зн\ i, 'M[1, N]', x_i^1 \rangle$ для $i = \overline{1, n}$, где *u. зн i — иметь значение i-й части*. Отметим, что выбор того или иного способа — самостоятельная задача, которая должна решаться в процессе настройки системы на пользователя.

В ряде мощных языков (типа Алгол-68) имя одной переменной может быть значением другой. Допускается возможность вычисления имен. Пусть $C1$ — имя такой переменной, значениями которой являются идентификаторы — имена других переменных, имеющих числовые значения. Подобная информация представляется как показано на рис. 2.20. Во фрагменте $\langle \_\_, t, u.\ зн, 'C1', x_1^1 \rangle x_1^1$ соответствует *значению* (*u. зн*) *переменной* $C1$, а в $\langle \_\_, t, u.\ зн, x_1^1, x_2^1 \rangle$ вершина x_2^1 соответствует *значению этого значения*.

Остановимся на **описателях типов переменных**. С помощью $\langle \_\_, t,$

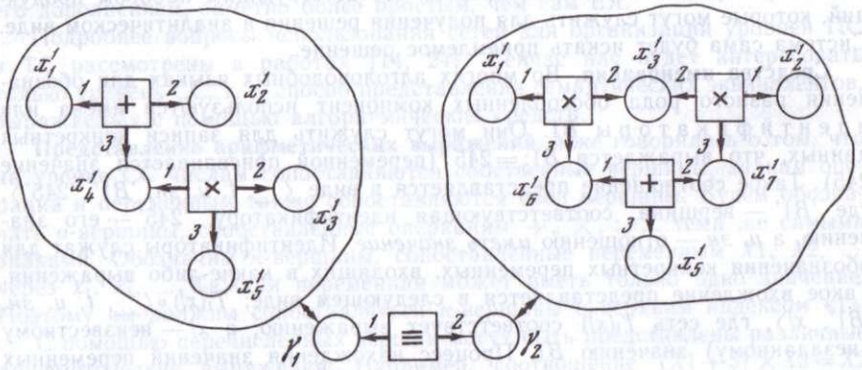


Рис. 2.18. Пример представления тождества

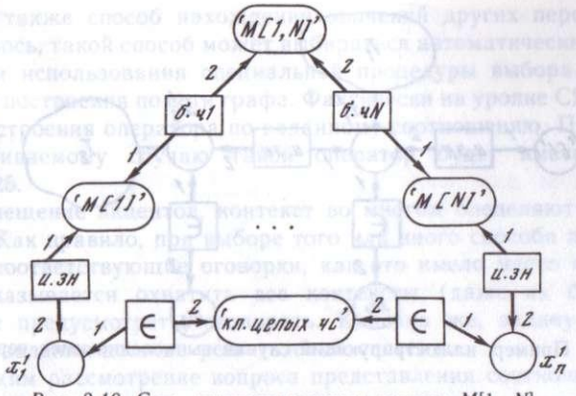


Рис. 2.19. Сеть, представляющая массив $M[1, N]$

$\in, x_1^1, m_1 \rangle$ представляется *принадлежность значений C1 к классу* M_1 (классу имен), а $\langle \_\_, t, \in, x_2^1, m_2 \rangle$ — *принадлежность значений значений к классу* M_2. Например, с помощью последнего фрагмента может быть представлено, что *имена* (X_1) *имеют числовые значения*. В свою очередь такие *значения* (X_2) *могут находиться в определенных соотношениях* $\mathcal{T}_2$ *с другими значениями*, что на рис. 2.20 изображено с помощью сети T_2 (x_2^1). Аналогично и у x_1^1 может иметься дополнительная окрестность T_1 (x_1^1), представляющая *отношения между именами и другими объектами*.

Напомним, что сеть представляет лишь соотношения без указания направления уточнения неопределенных компонент — значений $н$-вершин. Предполагается, что такое направление должно быть выбрано самой системой в зависимости от характера ее задач (от месторасположения символов типа «?»). Например, в сети рис. 2.20 допускается не только уточнение значений x_2^1 по уже найденным значениям x_1^1 (т. е. вначале находятся имена, а потом их значения), но возможен и обратный процесс — по значениям x_2^1 уточняются значения x_1^1 (ищутся имена переменных по их уже известным значениям). Имеется в виду, что вначале по T_2 находятся значения x_2^1. По ним уточняются значения x_1^1, что делает возможным дальнейшее уточнение компонент, представленных с помощью T_1. В частности, по найденным именам X_1 могут находиться другие имена X_i, связанные с ними отношениями $\mathcal{T}_1(X_1, X_i)$.

Итак, возможно различное направление уточнения. Как уже говорилось, такое направление представляется с помощью специальных средств — графов, которые строятся на базе сетей (см. § 4.1). Графы задают операции пошагового уточнения. Каждый шаг будет приводить к все более конкретной информации, представляемой с помощью $з$-сетей. К примеру, пусть было получено $[x_1^1 := 'B1']$, $[x_2^1 := '17']$. Тогда сеть рис. 2.20 будет уже представлять информацию следующего вида: $C1 := B1$, $B1 := 17$, $\mathcal{T}_1(B1)$, $\mathcal{T}_2(17)$.

Представление операторов и их соотношений. Оператору типа $F1(B1, ..., BN)$ ставится в соответствие сеть

$$\langle \gamma_1, t, 'F1', x_1^1, ..., x_n^1, x_{n+1}^1 \rangle \circ \langle \_\_, t, u.\ зн, 'B1', x_1^1 \rangle \circ ... \circ \langle \_\_, t, u.\ зн, 'BN', x_n^1 \rangle, \quad (2.33)$$

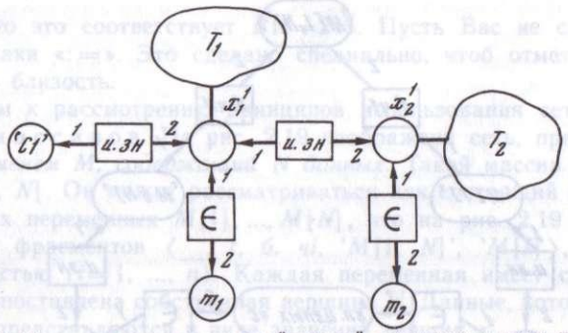


Рис. 2.20. Пример, иллюстрирующий случай вычисления имен переменных

где x_{n+1}^1 сопоставляется результату (пока неизвестному), $x_1^1, \ldots, x_n^1$ — значениям переменных $B1, \ldots, BN$, а '$F1$' — типу операции. Из таких операторов составляются соотношения. Они также могут быть представлены с помощью сетей. На рис. 2.21 изображена сеть, представляющая соотношение

$$B3=(B1+25)\times B2. \qquad (2.34)$$

В общем-то данное соотношение может интерпретироваться и представляться по-разному, что во многом определяется контекстом. На рис. 2.21 изображен один из вариантов. Представлено соотношение, которое имеет место между значениями переменных $B1$, $B2$ и $B3$. Знак «=» не представлен. Логическому значению соотношения сопоставлена p-вершина p_1 (являющаяся логическим замыканием сети T_i, см. § 1.4). Если соотношение (2.34) имеет место, то $[p_1:=t]$, если нет, то $[p_1:=f]$, если неизвестно и требуется найти, то $[p_1:=?]$.

Остановимся на некоторых других интерпретациях, определяющих свои способы представления соотношения (2.34). Если знак «=» указывает на то, какие величины должны сравниваться при проверке равенства, то в сети рис. 2.21 вместо $\langle \_, t, \text{и. зн}, \text{'}B3\text{'}, x_4^1\rangle$ будет фрагмент $\langle \_, p_1, =, x_4^1, x_5^1\rangle \circ \langle \_, t, \text{и. зн}, \text{'B3'}, x_5^1\rangle$, т. е. знаку «=» сопоставляется собственная o-вершина и фрагмент (ЭФ). В результате проверки равенства и находится логическое значение (p_1).

Пусть значение переменной $B3$ неизвестно. Тогда соотношение может иметь уже другой акцент. Оно может указывать на необходимость нахождения этих значений. Такое соотношение может быть оформлено в виде отдельного оператора, в котором выделяются аргументы и результаты: $B3:=(B1+25)\times B2$. Операторы представляются с помощью сетей (в данном случае с указанием $x_4^1:=?$) или графов со своими c-вершинами. На рис. 2.21 такая c-вершина обозначена через γ_1. C-вершина сопоставляется всему оператору, который может быть соответствующим образом обозначен, т. е. иметь свой идентификатор $R1$. Тогда на месте γ_1 будет стоять 'R_1'. Вершины типа γ_1 будут также служить для представления последовательности выполнения операторов, о чем речь будет идти ниже.

Сказанное выше справедливо и для тех случаев, когда неизвестным в (2.34) является значение $B1$ (или $B2$). Тогда с помощью T_1 представляется не только само соотношение (2.34), но и указание, что же нужно искать

$(x|:=?)$, а также способ нахождения значений других переменных. Как уже говорилось, такой способ может выбираться автоматически самой системой — путем использования специальной процедуры выбора направления обработки и построения по сети графа. Фактически на уровне СЯ реализуется действие построения оператора по заданному соотношению. Применительно к рассматриваемому случаю такой оператор будет иметь вид $B1:=B3/B2-25$.

Итак, смещение акцентов, контекст во многом определяют способ представления. Как правило, при выборе того или иного способа представления требуются соответствующие оговорки, как это имело место выше. Весьма трудным оказывается охватить все контексты (даже их большинство), все заранее предусмотреть, оговорить. Конечно же, задачу выбора способа представления должна решать сама система.

Продолжим рассмотрение вопроса представления соотношений. В соотношениях могут принимать участие различного рода **неравенства, логические связки** и т. д. На рис. 2.22 изображена сеть, представляющая логическое выражение $(A1<A2)\wedge(A1\times 2>A2)$. H-вершины p_1 и p_2 имеют тип x^1. Они соответствуют логическим значениям компонент. Например, если оказалось, что *неравенство* $A1<A2$ *имеет место*, то $[p_1:=t]$, а если *нет*, то $[p_1:=f]$. То же самое можно сказать и о другом неравенстве, которому сопоставляется $н$-вершина p_2. Вершина p_3 сопоставляется логическому значению всего выражения. Если нужно вычислить такое значение, то $[p_3:=?]$. Если заведомо известно, что выражение истинно, то $[p_3:=t]$. Если по выражению нужно оценить возможные значения переменной $A1$, то $[x_1^1:=?]$. Подобная оценка осуществляется формированием направления поиска и обработки. На сети же рис. 2.22 такое направление никак не задано, т. е. может решаться задача поиска любой из компонент.

Отметим, что каждому соотношению, как и оператору, может быть сопоставлена своя c-вершина (она формируется замыканием сети рис. 2.22) и своя p-вершина. C-вершины могут служить для выбора соотношений, обеспечивать переход от одного соотношения к другому и т. д. Соотношения могут связываться в такие же структуры, как и операторы.

Последовательность выполнения. В алголоподобных языках процесс выполнения операторов определяется порядком их написания, а также специальными операторами условного и безусловного перехода, операторами цикла и др. Рассмотрим принципы их представления (в явном виде)

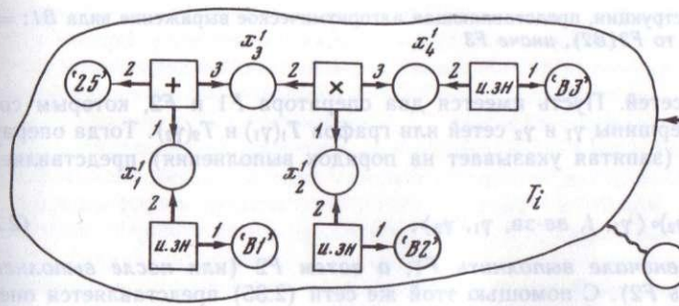


Рис. 2.21. Сеть, представляющая соотношение вида (2.34)

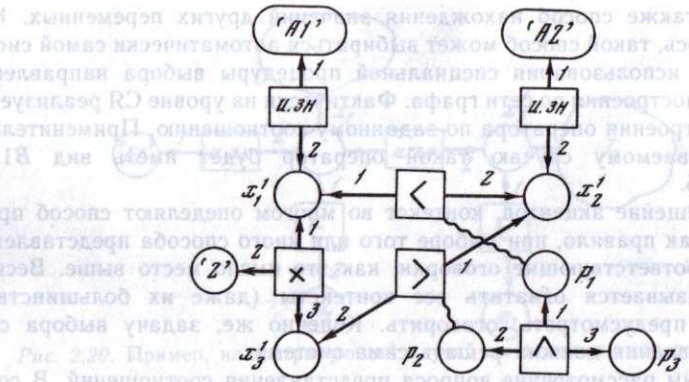


Рис. 2.22. Пример представления логического выражения

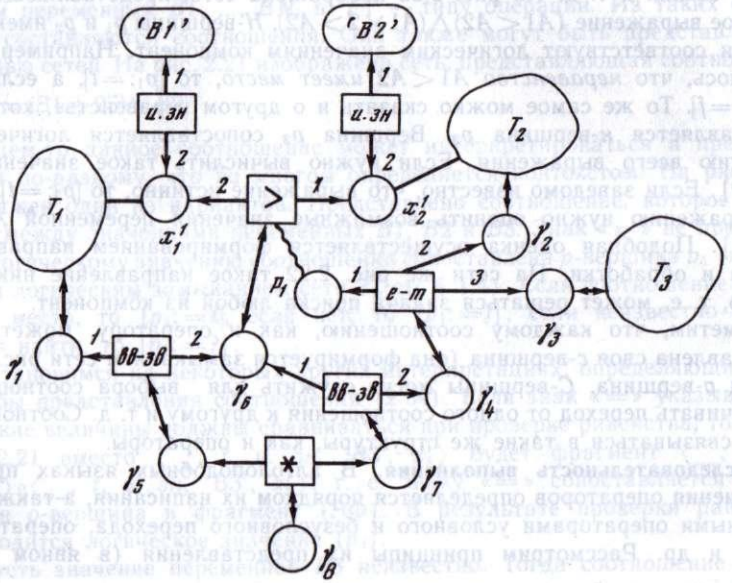


Рис. 2.23. Конструкция, представляющая алгоритмическое выражение вида $B1:=F1$, *если* $B1>B2$, *то* $F2(B2)$, *иначе* $F3$

с помощью сетей. Пусть имеется два оператора $F1$ и $F2$, которым сопоставлены c-вершины γ_1 и γ_2 сетей или графов $T_1(\gamma_1)$ и $T_2(\gamma_2)$. Тогда оператор типа $F1$, $F2$ (запятая указывает на порядок выполнения) представляется в виде сети

$$T_1(\gamma_1)\circ T_2(\gamma_2)\circ\langle\gamma_3,\ t,\ \textit{вв-зв},\ \gamma_1,\ \gamma_2\rangle, \qquad (2.35)$$

где *вв-зв* — *вначале выполнить* $F1$, *а затем* $F2$ (или *после выполнения* $F1$ *выполнить* $F2$). С помощью этой же сети (2.35) представляется оператор безусловного перехода типа $F1$ *go to* $F2$. Фактически оба приведенных

выше оператора означают одно и то же, что предопределяет однотипность их представлений.

Оператор **условного перехода** типа *если* $P1$, *то* $F1$, *иначе* $F2$ представляется с помощью сети

$$T_1(\gamma_1)\circ T_2(\gamma_2)\circ\langle\gamma_3,\ t,\ \textit{е-т},\ p_1,\ \gamma_1,\ \gamma_2\rangle, \qquad (2.36)$$

где *е-т* — *о*-вершина, соответствующая отношению *если, то*. Данная сеть означает, что *если имеет место условие* $P1$, *то выполняется оператор* $F1$, *иначе* $F2$.

Для иллюстрации принципов представления более сложных операторов (составленных из упомянутых ранее) приведем пример. Рис. 2.23 иллюстрирует способ представления следующего выражения. *Вначале нужно выполнить оператор* $F1$ *и найти значение* $B1$. *Затем проверить неравенство* $B1>B2$, *и если оно имеет место, то выполнить оператор* $F2$, *зависящий от аргумента* $B2$, *иначе* — $F3$. На рис. 2.23 операторы $F1$, $F2$ и $F3$ представляются с помощью T_1, T_2 и T_3. Каждому оператору сопоставляется своя c-вершина — γ_1, γ_2 и γ_3, а оператору проверки неравенства — γ_6. Порядок выполнения операторов представляется с помощью фрагментов вида (2.35) и (2.36). При этом указывается, что *после проверки неравенства управление передается на оператор условного перехода*, которому сопоставляется своя c-вершина γ_4. Последний обеспечивает передачу управления на один из операторов — $F2$ или $F3$, которым сопоставлены c-вершины γ_2 и γ_3. Такая передача осуществляется в зависимости от значения $н$-вершины p_1, т. е. от результата проверки.

Отметим, что в рамках введенных средств обеспечивается так называемый принцип постоянной интеграции — когда любому выражению или алгоритму может быть сопоставлена своя вершина. В частности, рассмотренному только что выражению сопоставляется c-вершина γ_8. С помощью нее может быть представлена взаимосвязь данного выражения с другими выражениями, т. е. порядок выполнения, условные зависимости и т. д. Например, если к рис. 2.23 добавить фрагмент $\langle\gamma_{10},\ t,\ \textit{е-т},\ p_2,\ \gamma_8,\ \gamma_9\rangle$, то это будет означать, что *указанное выражение должно выполняться, если только имеет место условие* $P2$. Выражению с таким условием сопоставляется уже другая c-вершина — γ_{10}, с помощью которой может быть представлено *когда нужно проверять это условие* и т. д.

ГЛАВА 3

ПРИНЦИПЫ ОБРАБОТКИ

В главе речь будет идти о перспективной схеме логической обработки, в рамках которой находят свое воплощение многие классические подходы, в частности методология моделей Робинсона, типовые методы организации вычислений и др. При этом будут учитываться следующие факторы. Во-первых, совершенствование знаний предполагает постоянное расширение функциональных возможностей внутрисистемных механизмов («умений»), обеспечивающих их использование. Любые новые виды или структуры знаний будут мертвым грузом, если не знать, что с ними делать, как использовать. Во-вторых, то, что происходит на уровне знаний, должно отображаться на этот же уровень. Должно обеспечиваться постоянное накопление разного рода «эвристик», связанных с удачными акциями и стратегиями внутрисистемной обработки, В-третьих, системные знания должны иметь выход вовне. Соответствующие структуры должны отображаться на «органы», связывающие систему с пользователем или с предметной областью (ПО). В свою очередь все, что происходит в ПО, должно отражаться на уровне системных знаний и учитываться системой. И, в-четвертых, система должна уметь работать с «подручными инструментами» (вычислительными устройствами), допускающими реализацию некоторых функций в автономном режиме, т. е. без постоянного обращения к системным знаниям. В перспективе возможности такой реализации должны возрастать, что предполагает перенос некоторых видов внутрисистемной деятельности на уровень внешних устройств.

Далее будут рассматриваться средства обработки. Будет показана необходимость в направленной обработке, которая должна умело сочетаться с декларативной компонентой знаний. В связи с этим будет введено новое понятие семантического графа, играющего важную роль в плане представления и обработки естественно-языковых форм.

§ 3.1. ДВУХУРОВНЕВАЯ СХЕМА ОБРАБОТКИ

Будем различать **два вида умений**. Первые связаны с работой внешних устройств и служат для организации взаимодействия с ПО. Вторые предполагают оперирование над знаниями и служат для ответа на запросы, различных умозрительных построений и т. д. С точки зрения унификации заманчиво предположить, что такие умения имеют одну основу. Оперирование над знаниями осуществляется с помощью специальных «органов» (механизмов), для которых такие знания являются как бы внешней средой. На схеме, изображенной на рис. 3.1, имеется среда H, которая с помощью органов воздействия F и восприятия V обеспечивает взаимодействие системы с ПО. Предполагается, что среда H может работать в автономном режиме, в частности, может настраиваться на тот или иной способ воздействия. Допускается возможность «проб» с их отражением на уровне H, выделением полезных структур (например, приводящих к целевым ситуациям), обобщением и интеграцией. В дальнейшем такие структуры служат для целенаправленного взаимодействия с ПО.

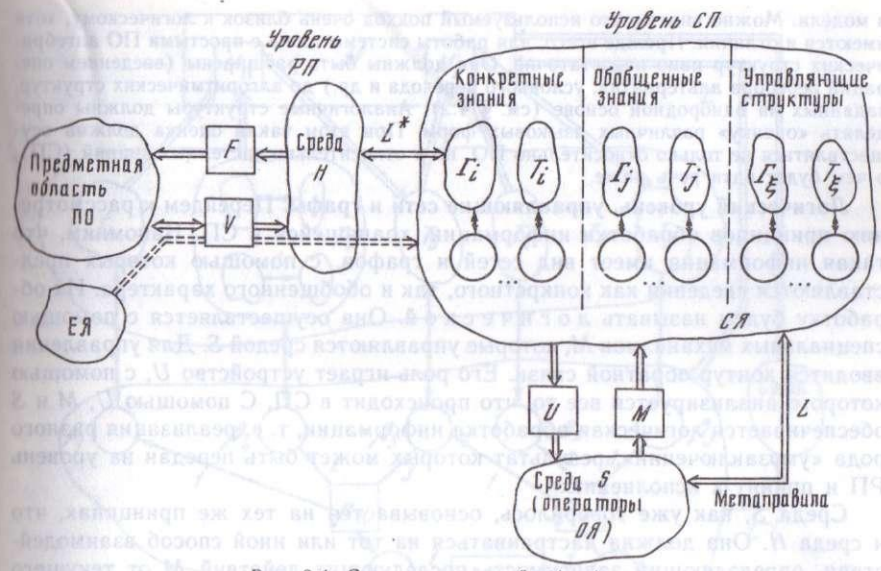


Рис. 3.1. Схема логической обработки

Аналогом подобных сред являются персептроны Френка Розенблата. В них, правда, отсутствуют органы воздействия и средства интеграции. Но все остальное имеется. Имеется среда, в которой отражается образ и с помощью которой обеспечивается выделение полезных структур (что осуществляется изменением весовых коэффициентов A-элементов). Задана целевая ситуация — в виде элементов, соответствующих образам. Один из таких элементов должен быть выделен в процессе распознавания. Другие аналоги — пирамидальные сети [7], нейроподобные схемы, в частности основанные на системе усиления — торможения [1], разного рода стимул-реактивные модели. Для реализации сложных стратегий с учетом обучения автором были предложены так называемые отражающие среды, которые во многом напоминают СЯ, но реализуют свои функции на уровне внутренней структуры [23].

Среда H образует так называемую р е ф л е к т о р н у ю п а м я т ь (РП). Все, что происходит в H, может отображаться (L^*) на уровень СП, т. е. быть представленным с помощью конструкций СЯ. Например, если некоторое действие F_i над объектом A_1 из ПО привело к восприятию объекта A_2, то в СП формируется фрагмент $\langle -, t, f_i, a_1, a_2 \rangle$. Допускается и обратное отображение, необходимое для управления работой H со стороны СП. Для обеспечения управления в СП должны быть специальные средства, определяющие направление и способ обработки. Роль таких средств будут играть семантические графы, частным случаем которых являются введенные ранее сети.

Понятия логики. Следует отметить, что обсуждаемая схема взаимодействия двух уровней традиционна для области математической логики, где отображение L^* получило название и н т е р п р е т а ц и и языка (в нашем случае — СЯ). Это же отображение вместе с предметной областью (ПО) и операциями (F) названо м о д е л ь ю или алгебраической структурой [21]. Аналогом действий формирования фрагментов типа $\langle -, t, f_i, a_1, a_2 \rangle$ является так называемая оценка истинности формулы $F_i(A_1) = A_2$

в модели. Можно видеть, что используемый подход очень близок к логическому, хотя имеются и отличия. Прежде всего, для работы системы даже с простыми ПО алгебраических структур явно недостаточно. Они должны быть расширены (введением операций перебора альтернатив, условного перехода и др.) до алгоритмических структур, заданных на однородной основе (см. § 4.2). Аналогичные структуры должны определять «оценку» различных языковых форм. При этом такая оценка должна осуществляться не только относительно ПО, но и относительно системных знаний (СП), о чем будет идти речь ниже.

Логический уровень, управляющие сети и графы. Перейдем к рассмотрению принципов обработки информации, хранящейся в СП. Напомним, что такая информация имеет вид сетей и графов, с помощью которых представляются сведения как конкретного, так и обобщенного характера. Их обработку будем называть л о г и ч е с к о й. Она осуществляется с помощью специальных механизмов M, которые управляются средой S. Для управления вводится контур обратной связи. Его роль играет устройство U, с помощью которого анализируется все то, что происходит в СП. С помощью U, M и S обеспечивается логическая обработка информации, т. е. реализация разного рода «умозаключений», результат которых может быть передан на уровень РП и принят к исполнению.

Среда S, как уже говорилось, основывается на тех же принципах, что и среда H. Она должна настраиваться на тот или иной способ взаимодействия, определяющий зависимость последующих действий M от текущего результата. Здесь также необходимы механизмы, обеспечивающие случайные воздействия с выделением полезных структур (реализующих сложные стратегии). Должны иметься средства интеграции, обеспечивающие простые способы инициирования сложных стратегий, и т. д.

Среда S связана с СП не только через устройства U и F, но и специальными средствами отображения L (см. рис. 3.1). Все, что происходит в среде S, может быть отражено на уровне СП. Пусть каким-то образом (последовательным использованием сетей или графов, находящихся в СП) системе удалось правильно решить задачу или ответить на запрос. Ясно, что ход или стратегия такого решения должен быть где-то запомнен и в дальнейшем распространен на другие («родственные») задачи. Конечно, в идеале подобные действия должны осуществляться за счет внутреннего функционирования среды S. Но есть и другой вариант, когда действия переносятся на уровень знаний (СП), соответствующим образом обрабатываются (корректируются, обобщаются) и потом уже используются для управления. В последнем варианте запоминание и управление осуществляются на базе СП. В СП формируется сеть (или граф), представляющая удачную стратегию (см. § 2.4). Будем называть такую сеть (граф) у п р а в л я ю щ е й. Ее использование для управления предполагает наличие специального отображения L на операторы алгоритмического языка или на среду, как это было описано выше.

Отображение L допускается в обе стороны, управляющие структуры (графы) могут отображаться на среду S, вызывая выполнение действий по логической обработке информации, т. е. такая обработка может быть сделана управляемой со стороны СП. Нетрудно видеть, что в схеме рис. 3.1 можно выделить контур так называемого командного управления, принятого в вычислительных машинах. Напомним, что в ЭВМ и команды и данные хранятся в одной и той же памяти, что делает возможным обработку команд как данных. То же самое имеет место в схеме рис. 3.1. Только роль команд

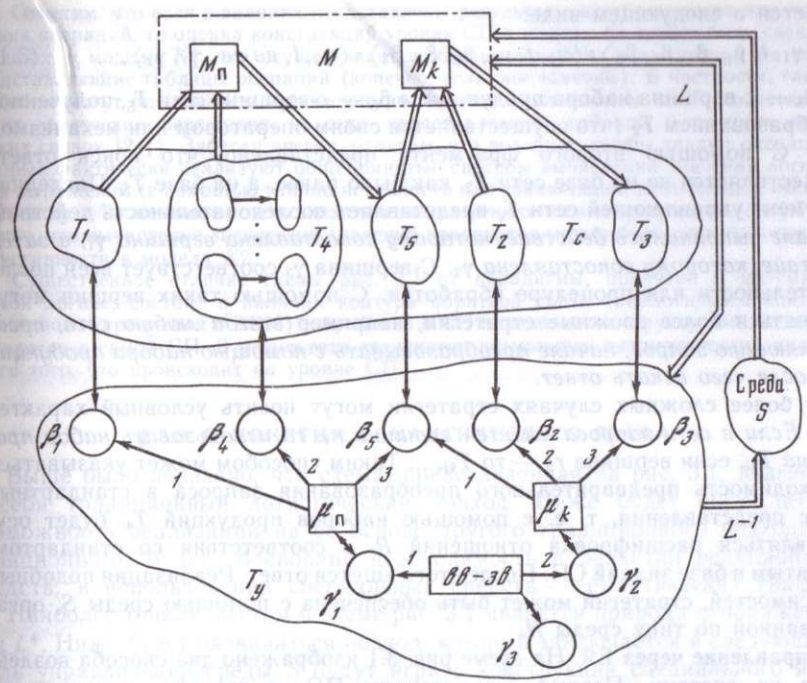


Рис. 3.2. Пример управляющей сети (T_y), представляющей акции внутрисистемной обработки (T_c)

и данных играют структуры СП, а роль устройства управления (с регистром команд) — среда S. При этом, как указывалось выше, предполагаются дополнительные функциональные возможности, что требует развития схемы командного управления. В частности, предполагаются действия обобщения структур и данных, возможность многоаспектного использования последних.

Остановимся на **принципах представления стратегий** в виде управляющих сетей. Последние составляются из c-вершин обычных сетей, используемых в процессе поиска ответа или решения. Например, пусть в СП имеется сеть T_1, представляющая запрос, а также сеть T_2, представляющая соотношение между известными системе объектами. Ответ на запрос находится внутрисистемной процедурой, называемой механизмом к о н к р е т и з а ц и и. Пусть эта процедура оформлена как отдельный оператор M_k. Если в результате его использования была получена сеть T_3, представляющая удачный ответ, то в СП может быть сформирован фрагмент $\langle \_\_, t, \mu_k, \beta_1, \beta_2, \beta_3 \rangle$, представляющий этот факт. Здесь β_1, β_2 и β_3 — c-вершины сетей T_1, T_2 и T_3, а μ_k — вершина, соответствующая процедуре M_k.

Рассмотрим более сложный пример (см. рис. 3.2). Пусть для удачного ответа потребовалось предварительное преобразование сети T_1 в стандартный базис, для чего был использован механизм $M_п$ и набор продукций T_4. Тогда формируется (L) более сложная управляющая сеть (T_y), которая запи-

сывается в следующем виде:

$$\langle \gamma_1, t, \mu_п, \beta_1, \beta_4, \beta_5 \rangle \circ \langle \gamma_2, t, \mu_k, \beta_5, \beta_2, \beta_3 \rangle \circ \langle \gamma_3, t, \textit{вв-зв}, \gamma_1, \gamma_2 \rangle, \qquad (3.1)$$

где β_4 — c-вершина набора продукций, а β_5 — c-вершина сети T_5, полученной преобразованием T_2 (что осуществляется своим оператором или механизмом $M_п$). С помощью второго фрагмента представлено, что поиск ответа осуществляется не на базе сети T_2, как было ранее, а на базе T_5. Последний фрагмент управляющей сети T_y представляет последовательность действий: *вначале выполнилось действие, которому сопоставлена вершина* γ_1, *а затем действие, которому сопоставлена* γ_2. C-вершина γ_3 соответствует всей последовательности или процедуре обработки. С помощью таких вершин могут задаваться более сложные стратегии, например *всегда любую сеть, представляющую запрос, начале преобразовывать с помощью набора продукций* T_4, *после чего искать ответ.*

В более сложных случаях стратегии могут носить условный характер типа *Если в сети запроса имеется вершина* r_1, *то использовать набор продукций* T_{4i}, если вершина r_2 — *то* T_{4j}, ... Таким способом может указываться необходимость предварительного преобразования запроса в стандартный базис представления, т. е. с помощью наборов продукций T_{4i} будет осуществляться расшифровка отношений R_i в соответствии со стандартом, принятым в базе знаний СП. После этого ищется ответ. Реализация подобных зависимостей, стратегий может быть обеспечена с помощью среды S, организованной по типу среды H.

Управление через ЕЯ. На схеме рис. 3.1 изображено два способа воздействия на систему. Первый — со стороны ПО, и второй — путем ввода предложений внешнего языка, в общем случае — ЕЯ. Естественно-языковое взаимодействие отмечено пунктирной линией. Показано, что такое взаимодействие осуществляется по тем же каналам, что и связь с ПО. Только в СП формируются другого сорта структуры — поверхностные, которые путем использования соответствующих лингвистических знаний преобразуются в семантические структуры, т. е. в сети, графы соответствующего уровня.

В перспективе необходимо, чтобы внутрисистемная обработка любой информации была управляемой через ЕЯ, чтобы любому предложению ЕЯ находилось место в СП. Конечно, в системе должно иметься и **базовое ядро**, не подверженное изменениям. В нашем случае таким ядром являются механизмы U, M и среда S. Конструкция этих механизмов и природа среды должны определяться сущностью физических элементов, выбранных в качестве реализующей основы.

Парадигма логики. Чтобы подчеркнуть достаточную универсальность схемы рис. 3.1 с точки зрения логической обработки, вернемся к проведению аналогий с понятиями математической логики. Важная особенность схемы — что в ее рамках допускаются две интерпретации. Выше была рассмотрена первая интерпретация, которая задается отображением L^* на ПО. Такая интерпретация определяет свою модель, которую обозначим через $K1$. Вторая интерпретация задается другим отображением L, которое вместе со структурами СП (знаниями) и операциями M (над знаниями) образует собственную модель $K2$. При этом допускается оценка истинности конструкций уровня СП (сетей, графов) как в модели $K1$, так и $K2$. Например, графы и сети, представляющие алгебраические выражения, оцениваются в модели $K1$, а графы и сети, представляющие гипотезы обобщенного характера,— в модели $K2$. В первом случае оценка сводится к выполению соответствующих операций, задаваемых выражениями, а в последнем — к проверке правильности гипотез относительно системных знаний.

Отметим, что если в знаниях представлены результаты алгебраических или логических операций, то оценка конструкций уровня СП в модели $K1$ может быть сведена к оценке в модели $K2$, что осуществляется наложением этих конструкций на сети, представляющие таблицы операций (конечно, если они конечны). В частности, таким способом могут оцениваться логические выражения (наложением сетей, представляющих логические выражения, на сети, представляющие таблицы истинности логических связок [24]). Забегая вперед, отметим, что подобные графы задают операции, которые фактически реализуют общепринятые способы вычисления значений логических выражений (с помощью таблиц истинности для конъюнкции, дизъюнкции, импликации и др.). То же самое можно сказать и о выражениях логики предикатов (см. § 7.3). Важно, что вычисление логических значений предикатов может быть сведено к оценке их истинности в модели $K2$.

Существенное отличие схемы рис. 3.1 от парадигмы, принятой в математической логике, состоит в наличии контура обратной связи (аналогичного контуру командного управления в ЭВМ): обработка в СП определяется структурами, которые находятся в самой СП. В результате становится возможным автоматический анализ всего того, что происходит на уровне СП.

§ 3.2. О ПОНЯТИИ СЕМАНТИЧЕСКОГО ГРАФА

Выше было показано, что схема, представленная на рис. 3.1, включает в себя традиционный логистический подход. Ниже будет обсуждаться возможность реализации на ее основе другого традиционного подхода — реляционного. Будет обоснована необходимость в развитии реляционных средств, в использовании специальных средств — семантических графов.

Наиболее тонким местом в схеме рис. 3.1 является природа отображений L и L^*. Ниже будет развиваться подход, который заключается в следующем. Роль управляющей среды S будут играть конструкции специального языка — ОЯ, а роль механизмов M — его базовые операторы. Для реализации отображения L будут использованы наборы специальных метаправил, обеспечивающих построение по любому графу оператора ОЯ (см. § 4.1). Выполнение последнего над другой сетью или графом приводит к получению требуемого результата.

Описанная процедура обработки в какой-то степени заимствована из области реляционных баз данных [10, 47]. В § 2.3 было показано, что табличный способ представления данных, используемый в реляционных базах, весьма близок к кортежному. Отсюда, естественна имеющая место общность схем или принципов обработки. Напомним, что в реляционных базах ответ на запрос предполагает его предварительное преобразование в оператор реляционной алгебры, т. е. в наборы операторов над таблицами-отношениями (типа *соединение* таблиц, их *деление* и др.). Ответ находится путем выполнения таких операторов.

При переходе от табличных данных к структурам описания схема существенно не изменяется. Также на базе запроса (его семантической структуры) формируются операторы. Только последние имеют несколько другое назначение. Это операторы, реализующие процедуру «хождения» по ребрам сети от одних вершин к другим с целью направленного уточнения информации, имеющейся в запросе. Для уточнения используются операторы ассоциативного поиска, групповые операторы над множествами. Имеются также операторы, обеспечивающие анализ результатов уточнения. С их помощью осуществляется использование не только конкретных данных, но и разного рода обобщенной информации, образующей системные знания. В этом и состоит значительное отличие развиваемого здесь подхода.

Особенности естественной обработки. Принятая схема привлекательна прежде всего с точки зрения обработки естественно-языковой информации. Во-первых, в ее рамках достаточно легко может быть учтена операционная семантика выражений естественного языка (ЕЯ). В ЕЯ имеются слова и отдельные конструкции сугубо операционного характера, т. е. требующие соответствующих действий, в том числе над внутренними представлениями. Например, предложения *Это все неправда, Как раз все наоборот, Не выиграл, а проиграл* требуют изменения представлений. С подобными предложениями должны быть связаны операторы изменения. Такая связь предполагается в обсуждаемой схеме.

Во-вторых, как уже говорилось, в ЕЯ зачастую указывается желательное направление обработки входной информации. Такое направление определяется конструкцией предложения (его темой и ремой), порядком изложения, а также специальными словами типа *каждый, какие-либо, существует, только* и т. д. Последние не имеют денотатов, т. е. эквивалентов в предметной области. Они задают некоторые способы уточнения, проверки (а в ряде случаев — порождения) информации, имеющейся в предложении. Выясняется, что с ними должны быть связаны соответствующие операции, что учтено в рамках выбранной схемы. При этом операции лучше связывать не с самими естественно-языковыми конструкциями, а с их семантическими эквивалентами или структурами, на которые должны отображаться конструкции. Об этом говорилось в § 1.2 при обсуждении систем с СП.

Итак, обсуждаемая схема перспективна с точки зрения возможности алгоритмическими средствами обеспечить направленную обработку, прямо или косвенно задаваемую с помощью предложений ЕЯ. В перспективе желательно, чтоб обработка была согласованной с естественными представлениями. Система должна уметь следовать ходу рассуждений человека, говорящего о том, как бы он искал ответ на запрос, решал задачу и т. д. Словом, человек должен иметь возможность указывать (в явном виде) на вариант поиска, стратегию решения и т. д. Система должна уметь конкретизировать и обобщать такие указания. Это легче делать, если каждому акту указания соответствует свой оператор, т. е. имеет место согласованность. Ясно, что количество таких операторов, их вид должны определяться характером естественных рассуждений. Задача минимизации здесь не ставится. Сколько нужно операторов, столько и берется. Если какого-либо оператора нет, то его нужно создать, т. е. возникает некоторая разновидность задачи обучения, требующей специальной организации.

Все сказанное приводит к необходиости специальных средств «поддержания» логического уровня. В § 1.2 говорилось, что внутренние представления систем, допускающих ввод естественно-языковой информации, должны носить структурный характер. Должен быть организован уровень семантических структур. Конечно, здесь явно недостаточно табличных данных, используемых в реляционных базах. Необходимы средства с более широкими изобразительными возможностями. Соответственно для задания операционной семантики требуются специальные языки, выходящие за рамки реляционной алгебры (см. § 4.1).

Стрелки порядка, графы. В качестве основы внутренних представлений нами были выбраны семантические сети. Почему именно сети? Напомним (см. гл. 1), что с помощью последних обеспечивается однородное и согласованное представление запросов, а также утверждений конкретного и обобщенного характера. Поэтому не требуется специальных языков записи запросов (типа реляционного исчисления), как это имеет место в реляционных базах. Допускается формулировка запросов и утверждений на внешнем языке, который может быть максимально приближен к естественному. Выражения такого языка могут быть автоматически преобразованы во внутрисистемные представления — сети. Системные знания — это тоже сети. В результате становится возможным широкое использование принципа сопоставления по образцу, развитого для случая сетей. При этом допускается сопоставление обобщенных структур с учетом сложных зависимостей между ситуациями, между сценариями и их конкретными проявлениями (воплощениями). Возникают дополнительные возможности использования связанных знаний структурного (конкретного и обобщенного) характера, проверки гипотез с помощью системных знаний и др.

Для обеспечения подобной обработки, как уже говорилось, требуются специальные средства указания направления поиска или способа проверки. Конечно, для этой цели могут использоваться сети, представляющие предписания и разного рода алгоритмические выражения (см. § 2.4). Но подобный способ представления неудобен, так как приводит к значительному усложнению конструкций. В дальнейшем в качестве средств указания направления поиска и проверок будем использовать с т р е л к и п о р я д к а «↦». Каждая такая стрелка исходит из вершины связи какого-либо фрагмента сети и направлена к *н*-вершине, значения которой следует искать по данному фрагменту. Если стрелка ↦ направлена к *о*-вершине или к вершине с заданными значениями, то это означает необходимость проверок найденных значений. Сеть с такими стрелками образует новый формальный объект, который называется семантическим г р а ф о м. Граф представляет информацию с указанием, какие части информационного материала должны быть использованы для нахождения тех или иных неопределенных компонент и какие проверки следует осуществлять над найденными компонентами.

Важнейшая роль графов — обеспечение более точного представления входной информации на уровне семантических структур. Давно подмечено, что какие-то элементы смысла задаются конструкцией предложений ЕЯ, что такие конструкции во многом определяют характер обработки, от которой зависит степень читабельности предложения. Например, в вопросе *Кто является другом брата Ивана?* ясно, что вначале нужно найти брата, а затем друга. Изменим конструкцию предложения: *Братом некоторого человека является Иван. Кто является другом этого человека?* Нетрудно видеть, что вопрос получился менее понятным.

Объяснить подобное явление можно, пользуясь понятием графа. Граф определяется конструкцией предложения. В связи с этим двум указанным вопросам будут сопоставлены различные графы. Они будут отличаться направленностью стрелок ↦. Во втором вопросе акценты расставлены не совсем правильно. Такой вопрос представляется графом, в котором стрелки порядка расставлены не лучшим образом. Граф задает операции более высокой степени сложности. Стрелки как бы сбивают с толку. Приходится ставить вопрос иначе, изменяя или уточняя его конструкцию, а стало быть, и направление стрелок. Этим и можно объяснить невысокую степень «читабельности» — явление, широко распространенное на практике. Итак, предложениям различных конструкций сопоставляются различные графы. Последние задают собственные операции обработки, которые также во

многом определяют сложность или степень читабельности самого предложения.

И, наконец, пользуясь понятием семантического графа, можно объяснить еще одно явление: смысловую близость многих вопросов, гипотез, предположений, сообщений и т. д. Например, вопросу *Кто брат Ивана?* очень близким является сообщение *У Ивана есть брат* или предложение *У Ивана должен быть брат*. Подобные предложения представляются с помощью весьма «близких» графов, задающих один и тот же способ поиска. Будем считать, что это и определяет их смысловую близость, которая зачастую делает невозможным четкое отличие сообщения от предложения или вопроса. Подобное явление будем объяснять, сводя его к случаю использования одних и тех же графов.

Активные и пассивные знания. В схеме рис. 3.1 по характеру использования можно разделить структуры СП (знания) на два типа. Одни через отображение L вызывают формирование операторов ОЯ. Другие — это наборы структур, над которыми такие операторы выполняются. Будем называть первые структуры — активными, а вторые — пассивными. Активные структуры инициируют деятельность системы: ее поисковую активность, а также активность, связанную с проверками, преобразованиями и др. Их роль будут играть графы, представляющие разного рода вопросы, гипотезы, определения и др. Пассивные структуры — это своего рода материал, на котором осуществляются действия поиска, проверки и др. Их роль будут играть сети и графы.

Структуры, в которых система уверена и которые в дальнейшем использует в своей деятельности, будем называть знаниями. Соответственно будем различать активные и пассивные знания.

Например, утверждение *У Ивана должен быть брат* представляется в виде графа, который может играть роль активных знаний. Такой граф инициирует задаваемые им операции в случае, если на вход системы поступает описание родственных связей Ивана. В результате подобные описания могут быть дополнены или признаны неверными.

Следует отметить, что введенные понятия активный—пассивный определяются характером использования структур, который может видоизменяться в процессе деятельности. Например, в процессе обучения системы на основе пассивных знаний и множества каких-либо объектов могут формироваться графы, представляющие гипотезы о классах объектов, т. е. активные структуры. В общем случае любая часть пассивных знаний может быть сделана активной. Справедливо и обратное. Операции, задаваемые каким-либо графом, могут выполняться над любыми другими графами, в которых не будут учитываться стрелки $\mapsto$. Такие графы будут уже играть роль пассивных структур или знаний.

В заключение параграфа следует отметить, что понятие графа, которое будет использоваться в дальнейшем, пока еще не является устоявшимся. Во многих работах под семантическими графами понимаются обычные семантические сети. Часто сеть определяется как ориентированный граф. В дальнейшем будем следовать введенной ранее трактовки этого термина.

§ 3.3. МЕТОДЫ ПОИСКА НА СТРУКТУРАХ

Для многих запросов можно предложить множество вариантов поиска ответа. С усложнением запроса, как правило, количество таких вариантов возрастает. В процессе формализации или представления запросов все множество вариантов должно быть сохранено. Тогда повышаются возможности в плане автоматического выбора наилучшего варианта. Почему язык реляционного исчисления считается предпочтительней для записи запросов по сравнению с языком реляционной алгебры? [47]. Прежде всего потому, что в процессе построения выражений реляционной алгебры заведомо фиксируется и направление поиска, обработки. При использовании реляционного исчисления определение эффективного поиска предоставляется специальному редуктору (компилятору). В то же время, как станет ясно ниже, язык реляционного исчисления значительно уступает языку семантических сетей по количеству сохраняемых вариантов поиска и уточнения. При использовании сетей в силу их согласованности с ЕЯ сохраняются практически все варианты так называемой естественной обработки, т. е. задаваемые характером естественно-языковых описаний. Выбор таких вариантов, определение эффективного направления поиска сводятся к построению графов, т. е. к формированию стрелок $\mapsto$, что осуществляется в автоматическом режиме. При такой постановке сам язык графов, конечно же, должен, по возможности, задавать все многообразие способов поиска и структурной обработки, в том числе и чисто гипотетических. В связи со сказанным, прежде чем переходить к детальному изложению языка семантических графов, следует остановиться на наиболее типовых способах обработки.

Варианты поиска. Начнем с примера. Пусть требуется *найти того, кто имеет друга, живущего в г. Хабаровске и работает на одном заводе с братом этого друга*. Один из вариантов поиска связан с перебором хабаровчан. Для каждого из них делается попытка найти друга, который работает на одном заводе с братом этого хабаровчанина. Естественно, для нахождения привлекаются соответствующие знания.

Второй вариант связан с перебором заводов. Для каждого из них делается попытка найти работника, являющегося другом хабаровчанина, брат которого работает на этом же заводе.

Еще один вариант, основанный на так называемом методе групповой обработки, заключается в последовательном уточнении неопределенных компонент с использованием групповых операций. В этом (третьем) варианте выбирается все множество хабаровчан. Далее находится полное множество их друзей, т. е. включающее в себя всех, кто дружит хотя бы с каким-либо хабаровчанином. После чего находятся все заводы, на которых эти друзья работают. По множеству заводов находятся все их работники и далее — полное множество братьев этих работников. С помощью последних уточняется первое множество — среди хабаровчан остаются только те, которые входят в последнее множество братьев (т. е. которые являются братьями работников найденных заводов). Итак, имеет место цикл, т. е. множество хабаровчан уточнялось через это же множество. Если полученное множество уменьшилось, то можно еще раз пройтись по циклу, т. е. заново переуточнить все множества. Подобное переуточнение можно осуществлять и другим способом — двигаясь в обратную сторону, т. е. по уточ-

ненному множеству хабаровчан находятся их братья и сослуживцы последних и т. д.

Для уточнения можно использовать еще один вариант групповой обработки, при котором допускаются параллельные ветви. Вначале находится множество хабаровчан, далее — множество их друзей и параллельно — множество сослуживцев их братьев. Ищутся общие элементы двух полученных множеств, которые объявляются результатом. Здесь используются только групповые операции и нет циклов. Хотя, как будет показано в § 4.4, возможны шумы, т. е. элементы, не удовлетворяющие запросу. Для их устранения нужны переборы, как имело место в первом варианте.

В общем случае может быть использован и вариант о б р а т н о г о п о и с к а, напоминающий метод обратного доказательства теорем [37]. Он заключается в том, что делаются предположения, которые затем проверяются. Если требуется найти некоторого человека, который ... , то делаются предположения типа *А что если этим человеком является* A_1? И далее по информации запроса предложение проверяется, т. е. выбираются друзья A_1 и т. д. Конечно, количество таких предположений может оказаться значительным.

Приведенные примеры далеко не исчерпывают все варианты поиска, число которых определяется степенью структурированности запроса (удельным числом взаимосвязанных компонент). Как было показано, с каждым таким вариантом связаны собственные операции поиска. Нетрудно видеть, что эти варианты различаются не только по количеству шагов, но и по количеству промежуточных элементов — множеств, которые на каждом шаге должны где-то храниться и в дальнейшем уточняться. Имеются в виду множества хабаровчан, заводов и т. д., найденных с привлечением соответствующих знаний. Шаги отличаются по степени сложности или трудоемкости. Например, у операций проверок одна сложность, у групповых операций поиска — другая, у теоретико-множественных — третья и т. д. Все это определяет общую трудоемкость процедуры поиска (в знаниях) и уточнения компонент запроса. Такая трудоемкость у всех вариантов будет различной.

Выбор вариантов. При разработке алгоритмов выбора вариантов следует отметить два подхода. Первый из них связан с построением универсальных алгоритмов, когда в каждом конкретном случае не ставится задача выбора наилучшего варианта. Важно, чтобы по входной сети выбирался неплохой вариант, а соответствующие типовые процедуры обхода цепей и контуров, нахождения и уточнения компонент были применимы для любой сети. Стратегии такого обхода делаются едиными. Не важно, что в ряде случаев обработка будет вестись далеко не лучшим способом. Считается, что при использовании достаточно быстродействующих ЭВМ всегда можно уложиться в допустимый интервал времени, отведенный на поиск. Важно, чтобы правила поиска были наглядными (чтобы можно было чисто умозрительно убедиться в их универсальности), а алгоритмы — простыми (чтобы меньше программировать) как например в языке *PROLOG*.

Второй подход предполагает анализ структуры входной сети, соответствующей запросу, с выбором наилучшего или **экстремального направления поиска** и уточнения компонент. В каждом конкретном случае выбирается собственное направление. Этот подход в последние годы получил широкое распространение в распознавании образов, где выбор алгоритма распознавания для каждого класса объектов рассматривается с точки зрения мини-

мизации функционала качества алгоритма. Такие алгоритмы получили название экстремальных. Казалось бы, распознавание образов и ответ на запросы — это различные задачи. Но в самом деле в рамках систем с СП это одна и та же задача. Распознавание сводится к ответу на запросы, представляющие особенности того или иного образа. Например, распознавание буквы А сводится к ответу на запросы типа: *Имеются ли в данном объекте три прямые линии, такие, что первые две образуют острый угол, а третья пересекает их*? (см. § 7.5). При этом поиск ответа осуществляется на основе входного описания образа.

Подход, связанный с построением экстремальных алгоритмов, весьма перспективен и при решении других задач, в частности, связанных с организацией поиска и проверок. Здесь также во многих случаях разумно вначале подумать, как искать, после чего начинать искать.

Вернемся к обсуждаемому ранее примеру. Ясно, что если заводов значительно меньше, чем работающих хабаровчан, то лучше перебирать заводы. Если и заводов чрезмерно много, то можно воспользоваться групповыми операциями поиска, которые обеспечивают поиск одних множеств на базе других. Например, по множеству хабаровчан находится все множество их друзей и т. д., как это имело место в третьем варианте. Отметим, что при использовании специальной ассоциативной памяти подобные операции могут быть реализованы достаточно быстро и эффективно [14, 23], т. е. сложность каждого шага может быть сделана сравнительно невысокой. Конечно, если такой памяти нет, то поиск все равно будет связан с переборами. Словом, оценка вариантов во многом определяется видом реализующего устройства.

В дальнейшем будем следовать второму подходу. Он предполагает использование (в рамках внутреннего языка) специальных средств указания направления поиска и также алгоритмов формирования такого направления. Роль средств будут играть стрелки порядка, формируемые на фрагментах семантической сети, а роль алгоритмов — специальные алгоритмы формирования порядка.

Случаи обработки. Будем различать три случая обработки: первый, когда в запросе направление задается однозначно, второй, когда направление задается лишь частично, и третий, когда направление никак не задается. Соответственно в первом случае запрос представляется в виде графа, у которого на всех фрагментах сформированы стрелки $\mapsto$, во втором — не на всех, и в третьем — таких стрелок нет, т. е. запрос представляется в виде сети. Конечно, всегда можно перейти от одного случая к другому, например, вычеркиванием всех стрелок обеспечивается переход от первого случая к третьему, а формированием — обратный переход. Эти же случаи имеют место при вводе сообщений, проверке гипотез.

Будем связывать с каждым таким случаем свои методы обработки. В первом случае предполагается уточнение алгоритма поиска — с помощью точного указания порядка нахождения всех неопределенных компонент. В третьем случае имеет место полный произвол. С ним будем связывать метод хаотического поиска. Во втором случае имеют место некоторые промежуточные методы поиска, когда допускается некоторый произвол, но есть и определенные шаги. Алгоритмы формирования стрелок порядка имеют целью устранения такого произвола. Остановимся более подробно на упомянутых методах обработки.

Метод хаотического поиска. Он основан на использовании групповых операций обработки, которая ведется ненаправленным образом. Подобная обработка заключается в хаотических попытках уточнения одних компонент запроса на базе других. Вначале в качестве исходных берутся элементы классов или полные объемы используемых понятий. Так, в обсуждаемом выше примере предполагается, что *тем человеком, которого требуется найти, а также его другом и братом друга может быть любой индивидуум*, т. е. берется все множество людей. Далее делаются попытки уточнения. Попытки будут результативными, если они приводят к уменьшению объемов соответствующих множеств. Например, результативными будут попытки выделения хабаровчан, выделения людей, являющихся братьями других людей и т. д. Работа заканчивается, если в течение некоторого (достаточно большого) числа шагов не оказалось ни одной результативной попытки.

Особенность метода — в отсутствии четкого деления на аргументы и результаты. В зависимости от направления поиска, которое может быть выбрано достаточно произвольным, каждая компонента может играть роль как аргумента (т. е. быть используемой для уточнения), так и результата (т. е. уточняемой).

Один из недостатков описанного метода — в большом числе требуемых шагов или попыток. Конечно, их количество может быть уменьшено за счет использования специальных эвристических критериев. Например, выбор отношений, по которым уточняются множества, может осуществляться с учетом удельного числа их неопределенных компонент, объема множеств и т. д. Такой выбор уже делает обработку более направленной. Чем больше подобных эвристических критериев используется, тем менее хаотической будет обработка.

Другой недостаток метода — ограниченные возможности обработки обобщенной информации. Групповых операций поиска и уточнения множеств ($\cap$) недостаточно для ответа на запросы, содержащие слова-кванторы и др. (см. § 7.3).

Достоинство метода — отсутствие необходимости в специальных алгоритмах формирования порядка, обладающих достаточно высокой степенью сложности.

Указанный метод положен в основу формализма [36], где заданные (неокончательные) значения вершин названы недоопределенными множествами, а сети с такими вершинами — недоопределенными моделями. Работа системы заключается в попытках уточнения таких множеств на основе заданной схемы отношений. Для этого используются знания об отношениях между конкретными объектами и соответствующая процедура сопоставления.

Некоторая модификация метода, как уже говорилось в § 2.4, применена в вычислительных (семантических) моделях Тыугу [46], где порядок вычисления очередной переменной заранее не задается и определяется уже вычисленными значениями. При этом каждое отношение (операция) может интерпретироваться по-разному. Например, если по соотношению $A_1 + X_2 = X_1$ уточняется компонента X_1, то используется операция сложения, а если X_2, то — вычитания. В общем случае операции или вычислительные функции могут сопоставляться более сложным соотношениям, например квадратным уравнениям. Конечно, такие функции будут иметь более сложный вид.

Метод смещения граней. Интересным представляется сочетание только что описанного метода с интервальной математикой, развиваемое А. С. Нариньяни [36]. Каждая числовая переменная ограничивается сверху и снизу, т. е. вводятся верхняя и нижняя грани, которыми являются числа. Вычислительная модель или схема отношений (представляющая арифметические операции) используется для смещения таких граней, уточнения ограничений. При этом верхняя грань смещается вниз, а нижняя — вверх. Отношения, которые в каждом конкретном случае используются для смещения граней, выбираются по ходу вычисления.

Аналогичный метод может быть распространен на случай невычислительных (концептуальных) моделей, представляющих нечисловые зависимости. В таких моделях с помощью отношений связываются множества. В качестве верхней и нижней граней, ограничивающих каждую неопределенную компоненту, используются также множества. Верхняя грань — это супермножество, заведомо включающее в себя все априори известные (приемлемые) варианты уточнения компоненты [1]. Нижняя грань — множество, элементы которого известны как правильные варианты уточнения. Например, множество друзей некоего человека ограничивается указанием, кто из людей может быть другом (верхняя грань) и кто, как уже известно, является им (нижняя грань). При этом считается, что известно далеко не все. Обработка заключается в последовательном смещении верхней и нижней граней всех имеющихся неопределенных компонент. Верхние грани, как уже говорилось, должны смещаться вниз, т. е. количество элементов супермножеств должно постоянно уменьшаться, а нижние грани — вверх. В этих двух случаях попытки будут считаться результативными.

Следует отметить, что разработка принципов смещения граней — довольно непростая задача. Операции по смещению обладают своими особенностями. В ряде случаев для смещения верхних граней одних компонент должны использоваться нижние грани других и наоборот. Конечно, объем множеств, представляющих верхние грани, может оказаться значительным, что может существенно затруднить обработку.

Метод согласованного уточнения. Он основан на независимом нахождении согласованных n-ок объектов и их последующем уточнении. Вначале из запроса выделяются все используемые в нем («базовые») отношения. По ним (путем поиска в знаниях) независимо находятся неопределенные компоненты или же их согласованные пары, тройки и т. д. Например, отдельно находится множество *хабаровчан*, множество пар *друзей*, пар братьев, а также пар *человек — место его работы*.

Затем найденные множества уточняются, исходя из информации запроса, т. е. схемы отношений. Эта схема определяет множества, которые соответствуют одним и тем же объектам. Такие множества специальным образом «стыкуются» — из них выделяются подмножества, имеющие одну и ту же проекцию на соответствующие координаты. Например, выделяются только пары друзей, таких, что один из них является элементом множества хабаровчан и присутствует в паре, составленной из братьев, а другой входит в левую часть хотя бы одной пары *человек—место его работы*. Другими словами, один человек должен быть хабаровчанином и иметь брата, а его друг — где-то работать.

Аналогичным образом выделяются и другие пары (в общем случае — n-ки). Итак, уточнение множеств осуществляется на основе их «стыковки».

[1] Здесь рассматривается один из подходов. В другом подходе верхняя грань — множество неприемлемых вариантов. В процессе уточнения такое множество должно постоянно расти.

Подобная операция в реляционных базах данных получила название с о е д и н е н и я. И здесь тоже уточнять можно различными способами. В частности, «стыкующиеся» множества могут выбираться случайно. Делается попытка их уточнения. Если некоторое число таких попыток оказывается неудачным, то работа заканчивается. Другой способ уточнения — последовательный, когда за каждый проход множества уточняются по одному разу: вначале берется и уточняется одно множество, по нему — другое, стыкующееся с ним, и т. д. Как показывает опыт, при умелой организации такого уточнения оказывается достаточно двух проходов — за второй проход множества переуточняются, но уже в обратном направлении (они берутся в обратном порядке). В результате остаются только такие n-ки, которые соответствуют информации запроса. На основе их формируется ответ.

Описанный метод использован в ряде отечественных и зарубежных разработок. Он был реализован в первой версии системы активного диалога, разрабатываемой в ВЦ ДВНЦ АН СССР. Один из его недостатков — в необходимости выборок больших объемов. Если в запросе в явном виде не указаны объекты какого-либо отношения, то из системных знаний требуется выборка всех конкретных воплощений данного отношения. А если не указано и само отношение — то всех системных знаний. Другой недостаток — в ограниченных возможностях обработки обобщенной информации.

Достоинство метода — в возможности параллельного поиска и выборки фрагментов из знаний, представляющих известные системе отношения между конкретными объектами. При последовательной организации таких знаний все необходимые фрагменты выбираются за один цикл просмотра. По таким фрагментам находятся избыточные множества согласованных пар, ..., n-ок, которые затем уточняются. Еще одно достоинство — в достаточной универсальности, т. е. метод может служить для ответа на весьма сложные запросы конкретного характера.

В дальнейшем, как уже говорилось, основное внимание будем уделять направленной обработке, как наиболее универсальной. В ряде случаев без такой обработки не обойтись, например, когда требуется *найти общих друзей всех братьев некоего человека*, проверить, *имеются ли друзья у каждого из братьев* и т. д. Итак, первоочередной будем считать задачу выбора средств для обеспечения направленной обработки. Средства будут вводиться в рамках языка семантических графов.

§ 3.4. СЕМАНТИЧЕСКИЕ ГРАФЫ ПРОСТОГО ВИДА

В данном параграфе начинается описание семантических графов — средств, представляющих структурную информацию с указанием направления ее обработки. Ниже речь будет идти лишь об определенном виде обработки, связанном с поиском, уточнением информации и выполнением разного рода проверок. Будут вводиться графы, соответствующие объектам и их наборам с указанием способа их выделения, а также графы, представляющие разного рода вопросы, утверждения, и указывающие, как искать ответ или проверять утверждение, в каком порядке находить неопределенные компоненты и т. д. Предполагается, что для нахождения и проверок используются системные знания пассивного типа (см. § 3.2). Далее будут введены понятия смешанного, полного и приведенного графов, различающихся степенью

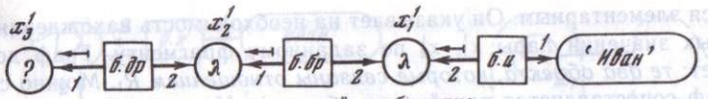


Рис. 3.3. Граф, представляющий требование *найти друга брата Ивана*

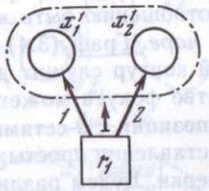


Рис. 3.4. Граф, сопоставленный *паре объектов, связанных отношением* R_1

определенности или законченности указаний о направлении обработки. Вначале остановимся на графах, соответствующих объектам и их наборам.

Элементарные графы. Фокус и область. Напомним, что семантический граф — это та же сеть, на которой с помощью специальных стрелок $\mapsto$ указано, какие фрагменты следует использовать для нахождения значений тех или иных n-вершин. Словом, представляется, какая информация должна использоваться для уточнения тех или иных компонент. На рис. 3.3 изображен семантический граф, соответствующий простому вопросу: *Кто является другом брата Ивана*? При этом *Ивану* (пока неизвестно какому) сопоставлена n-вершина x_1^1, *его брату* — x_2^1, а *другу* — x_3^1. С помощью o-вершин *б. и, б. бр, б. др* представляются отношения соответственно *быть именем, быть братом* и *быть другом*. Стрелки типа $\mapsto$, которые были названы стрелками п о р я д к а, указывают, что для нахождения значений n-вершины x_1^1 следует использовать фрагмент $\langle \_\_, t, б.\ и, 'Иван', x_1^1 \rangle$, а для нахождения значений x_2^1 и x_3^1 — свои фрагменты. Будем записывать подобное указание (для вершины x_1^1) в виде

$$x_1^1 \perp \langle \_\_, t, б.\ и, 'Иван', x_1^1 \rangle, \tag{3.2}$$

где x_1^1 называется ф о к у с о м графа, фрагмент — его о б л а с т ь ю, а знак $\perp$ — связкой типа *тот, который*. Данный граф дословно означает: *тот объект* X_1, *который имеет имя Иван*. Будем говорить, что граф с о п о с т а в л я е т с я такому объекту. Если в графе рис. 3.3 заменить n-вершины x_2^1 и x_3^1 на x_2 и x_3, то он будет сопоставлен *друзьями братьев Ивана*. А если зменить только x_2^1 на x_2, то граф будет сопоставлен *другу братьев Ивана* (имеется в виду — общему другу). Нетрудно видеть, что с помощью введенных изобразительных средств обеспечиваются представления, согласованные с естественно-языковыми формами. Для простых предложений и представления будут достаточно простыми.

Область графа используется для нахождения значений n-вершины, являющейся его фокусом. Пусть $T_1^o(x_1)$ — элементарный фрагмент (ЭФ) с n-вершиной x_1. Назовем конструкцию

$$x_1 \perp T_1^o(x_1) \tag{3.3}$$

э л е м е н т а р н ы м графом. В случае пустой области, т. е. отсутствия фрагмента $T_1^o(x_1)$, такой граф вырождается в отдельную вершину x_1.

В дальнейшем будем понимать под элементарными и такие графы, у которых фокусами являются наборы n-вершин, а область — есть ЭФ. Например, граф

$$(x_1^1, x_2^1) \perp \langle \_\_, t, r_1, x_1^1, x_2^1 \rangle \tag{3.4}$$

является элементарным. Он указывает на необходимость нахождения согласованных значений пары x_1^1, x_2^1 по заданному фрагменту. Граф дословно означает: *те два объекта, которые связаны отношением* R_1. Можно сказать, что граф сопоставляется такой паре объектов. Например, если r_1 соответствует отношению *быть мужем — женой*, то граф будет сопоставлен *супружеской паре*. Граф (3.4) изображается, как показано на рис. 3.4, где пунктирный контур служит для указания на согласованность. В общем случае в качестве фокуса может быть не пара, а n-ка n-вершин.

Композиции с з-сетями. Символ λ. Ниже будут рассматриваться средства представления простых утверждений, высказываний с указанием способа их проверки. Будем различать три типовые конструкции элементарных графов. Первый тип, когда граф сопоставляется неопределенному объекту или набору объектов, что было рассмотрено выше. Второй тип, когда с помощью графа представляется, что данный объект или набор имеет место. И, наконец, третий тип, когда с помощью графа представляется, что это за объект или набор, т. е. дается один или множество вариантов его уточнения. Графы последних двух типов представляют высказывания, утверждения. Остановимся вначале на рассмотрении конструкций в т о р о г о типа.

Для указания на факт существования объекта X_1 будем использовать символ λ и конструкцию следующего вида:

$$[x_1 := \lambda] \circ [x_1 \perp T_1^0(x_1)]. \qquad (3.5)$$

которая означает *Имеются объекты, обладающие заданными свойствами* $\mathcal{T}_1$. Символ λ представляет факт наличия, существования и означает *имеется, есть*. Будем называть конструкцию (3.5) э л е м е н т а р н ы м графом и записывать ее в более простом виде

$$[x_1 \perp T_1^0(x_1)].$$

считая, что к ней всегда может быть добавлен фрагмент $[x_1 := \lambda]$ (если объект упоминается и нет специальных оговорок, значит, он есть). Будем изображать введенные графы размещением символа λ внутрь вершины x_1.

Рассмотрим пример. Граф

$$[x_1^1 \perp \langle \_,\ t,\ б,\ и,\ \text{'Иван'},\ x_1^1 \rangle] \qquad (3.6)$$

означает, что *имеется некий Иван* и изображается с помощью символа λ (см. рис. 3.3). Ясно, что если спрашивается о *друге брата Ивана*, то заведомо подразумевается, что *имеется такой Иван и его брат*, что на рис. 3.3 показано с помощью второго символа λ.

Для указания на факт отсутствия объектов X_1 (если имеются такого сорта оговорки) будем использовать символ $\tilde{\lambda}$ и конструкцию вида

$$[x_1 \perp T_1^0(x_1)] \circ [x_1 := \tilde{\lambda}], \qquad (3.7)$$

также называемую элементарным графом. Она означает *отсутствие объектов со свойством* $\mathcal{T}_1$. К графу (3.7) уже не может быть добавлен фрагмент $[x_1 := \lambda]$, так как представлено отрицание. При изображении подобных графов символ $\tilde{\lambda}$, означающий *не имеется*, также будем помещать внутрь n-вершины x_1. Например, если на рис. 3.3 внутри n-вершины x_3^1 вместо символа ? поставить $\tilde{\lambda}$, то получившийся граф будет означать *у брата Ивана нет друзей*, т. е. уже будет представлено некоторое негативное предположение или утверждение.

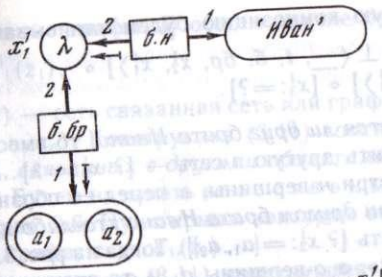


Рис. 3.5

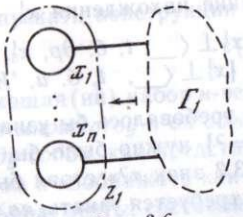


Рис. 3.6

Рис. 3.5. Граф, означающий, что *братом Ивана является кто-либо из* $\{A_1, A_2\}$

Рис. 3.6. Общий вид составного графа

Рис. 3.7. Граф, сопоставленный *человеку* X_1, *который является хозяином (б. хоз) собаки* X_2, *укусившей (укус) его друга (б. др)* X_3

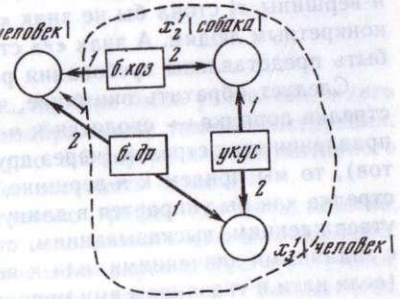


Рис. 3.7

Для указания на факт, что *объектами* X_1 *со свойством* $\mathcal{T}_1$ *является множество* $\{A_1, ..., A_n\}$, будем использовать конструкции

$$[x_1 \perp T_1^0(x_1)] \circ [x_1 := \{a_1, ..., a_n\}]. \qquad (3.8)$$

Подобные композиции элементарных графов с з-сетями будем называть также элементарными графами. Если в (3.8) на место x_1 поставить x_1^1, то граф будет означать, что *объектом* X_1 *со свойствами* $\mathcal{T}_1$ *является один из элементов множества* $\{A_1, ..., A_n\}$. Имеют место альтернативы. Если множество одноэлементное, то образуется граф

$$[x_1^1 \perp T_1^0(x_1^1)] \circ [x_1^1 := a_i], \qquad (3.9)$$

означающий, что *объект со свойством* $\mathcal{T}_1$ — *есть* A_i. Будем изображать графы вида (3.8) обычным способом. Например, граф рис. 3.5 будет означать, что *братом Ивана является либо субъект* A_1, *либо* A_2. Если такие субъекты определяются своими характеристиками (по имени, возрасту ...), то вершины a_1 и a_2 будут фокусами соответствующих графов, которые добавляются к рис. 3.5.

Композиции графов. Представление требований. Элементарные графы могут объединяться друг с другом, что будем называть композицией. В результате образуются более сложные конструкции, которые также будем называть графами (но уже не элементарными). С помощью композиций представляется направление уточнения множества неопределенных компонент. При этом может быть указано, что это за компоненты, представлены факты их наличия или отсутствия.

Например, одна такая композиция была рассмотрена на рис. 3.3, где указано, как искать значения трех n-вершин x_1^1, x_2^1, x_3^1, и представлено тре-

бование нахождения x_3^1. Подобную композицию будем записывать в виде

$$[x_3^1 \perp \langle \_\_,\ t,\ б.\ др,\ x_3^1,\ x_2^1\rangle] \circ [x_2^1 \perp \langle \_\_,\ t,\ б.\ бр,\ x_2^1,\ x_1^1\rangle] \circ$$
$$\circ\ [x_1^1 \perp \langle \_\_,\ t,\ б.\ и,\ 'Иван',\ x_1^1\rangle] \circ [x_3^1 := ?]. \qquad (3.10)$$

Если требовалось бы узнать, *имеется ли друг брата Ивана*, то вместо з-сети $[x_3^1 := ?]$ нужно было бы подставить другую з-сеть — $[?\ x_3^1 := \lambda]$. Тогда на рис. 3.3 знак «?» стоял бы не внутри *н*-вершины, а перед ее обозначением. Если требуется узнать, *являются ли другом брата Ивана кто-либо из* $\{A_1, A_2\}$, то вместо $[x_3^1 := ?]$ следует поставить $[?\ x_3^1 := \{a_1, a_2\}]$. Тогда на рис. 3.3 внутри *н*-вершины x_3^1 стоял бы не знак «?», а *о*-вершины a_1 и a_2, соответствующие конкретным людям. А знак «?» стоял бы перед x_3^1. С помощью графов могут быть представлены требования различного характера.

Следует обратить внимание, что в графах, соответствующих вопросам, стрелки порядка $\mapsto$ сходятся к *н*-вершинам со знаком «?». Если идти в направлении этих стрелок (через другие *н*-вершины и вершины связи фрагментов), то мы придем к *н*-вершине, отмеченной этим знаком «?». Последняя стрелка как бы упирается в данную *н*-вершину. В графах, соответствующих утверждениям, высказываниям, стрелки порядка сходятся или к вершинам с заданными значениями, или к вершинам со знаком λ. Последняя стрелка (если идти в указанном ими направлении) упирается в одну из этих вершин, направлена к ней. Таким способом представляется оттенок наличия. Если последняя стрелка направлена к вершине $x_i := a_1$, то представляется *факт наличия объекта* A_1, что *соответствующие отношения характерны именно для объекта* A_1. Например, если на рис. 3.3 внутрь вершины x_3^1 поместить *о*-вершину a_1, убрав знак «?», то будет представлено, что *другом брата Ивана является субъект* A_1. Если внутрь вершины x_3^1 поместить символ λ, то будет представлено утверждение, что *Имеется некий друг брата Ивана*. В обоих случаях последняя стрелка порядка будет направлена к одной из перечисленных ранее вершин.

Итак, графы, соответствующие требованиям, вопросам, отличаются от графов, соответствующих утверждениям, высказываниям, лишь по типу вершин, к которым направлена последняя стрелка порядка $\mapsto$. Отсюда, изменением такой вершины и ее типа, как было показано выше, можно легко из вопросов делать утверждения и наоборот. Это очень важно, так как проверка правильности утверждений сводится к ответу на запросы. Например, пусть имеется граф $\Gamma_1(x_i) \circ [x_i := a_1]$ с фокусом x_i. Представлено утверждение об объекте X_i. В таком графе стрелки сходятся к вершине x_i. Тогда для проверки утверждения можно переделать этот граф следующим образом: $\Gamma_i(x_i) \circ [?x_i := a_1]$. Полученная конструкция будет представлять вопрос. Далее можно воспользоваться типовыми процедурами поиска ответа на запросы. Ответ типа *да* означает правильность утверждения.

В общем случае утверждение может быть представлено с помощью сети T_1. Тогда для проверки утверждения в T_1 выбирается одна из *о*-вершин — a_k. Формируется сеть $T_1(x_i) \circ [?x_i := a_k]$. По ней строится граф. Она заменяется на x_i и ищется ответ. Из T_1 может быть выбрана и любая ее *н*-вершина — x_j. Тогда формируется сеть $T_1(x_j) \circ [?x_j := \lambda]$. По ней строится граф, по которому ищется ответ. Такого сорта формирование соответствует случаю, когда, например, для проверки утверждения *Имеется друг брата Ивана* формируется вопрос *Имеется ли у Ивана брат, у которого есть друг?* Ответ *Да* говорит о правильности утверждения.

Составные графы. В общем случае будем допускать конструкции вида

$$z_1 \perp \Gamma_1(z_1), \qquad (3.11)$$

где $\Gamma_1(z_1)$ — есть связанная сеть или граф, содержащая(ий) набор *н*-вершин $z_1 = (x_1, ..., x_n)$. Конструкция (3.11) соответствует *множеству п-ок объектов* $X_1, ..., X_n$, *связанных с другими объектами и между собой*, как представлено в $\Gamma_1(z_1)$. Будем называть ее с о с т а в н ы м графом и изображать как показано на рис. 3.6. Пунктирным контуром очерчены фрагменты, составляющие область графа, т. е. $\Gamma_1(z_1)$.

Производная от (3.11) конструкция

$$[z_1 \perp \Gamma_1(z_1)] \circ [z_1 := \lambda]$$

означает, что *имеется п-ка объектов* $X_1, ..., X_n$. Как было условлено, з-сеть $[z_1 := \lambda]$ может подразумеваться. Для изображения такой конструкции, также называемой составным графом, на рис. 3.6 внутри штрихпунктирного контура, изображающего z_1, следовало бы поставить символ λ.

На рис. 3.7 изображен пример составного графа, где представлено, что *для нахождения человека* X_1 *следует использовать всю информацию: что* X_1 *является хозяином собаки* X_2, *что последня укусила* X_3 *и что* X_3 *является другом* X_1. Фактически с помощью графа на рис. 3.7 указывается, что для нахождения значений следует использовать все три фрагмента сети (образующих некоторое «сетевое пространство»). Однако не указывается, в каком порядке нужно брать эти фрагменты, в какой последовательности искать значения *н*-вершин x_2^1 и x_3^1. Только указывается, что значения *н*-вершины x_3^1 следует искать в последнюю очередь, т. е. стрелка порядка направлена к x_3^1, «сходится» к ней. Если на фрагментах области графа (см. рис. 3.7) проставить стрелки порядка, то получится также составной граф, но в некотором смысле уточненный. Будет указано, в каком порядке и что искать.

Смешанный граф. Под ним будем понимать композицию сетей и элементарных графов. Смешанные графы будут служить для представления разного рода утверждений, требований, в которых отсутствует указание на то, как использовать те или иные фрагменты. У смешанного графа не от всех вершин связи исходят стрелки порядка $\mapsto$. Другими словами, направление обработки определяется неполностью. Допускается некоторый произвол, варианты. Устранение произвола или выбор варианта заключается в доформировании стрелок порядка, что приведет к уточнению представленной информации.

Примеры. На рис. 3.8 изображен смешанный граф, представляющий высказывание *Король Франции лысый*. При этом вершина x_1^1 соответствует *королю Франции* (предполагается, что *такой король существует*), а *лысый* рассматривается как свойство. Такой граф записывается в виде

$$\langle \_\_,\ t,\ св,\ x_1^1,\ 'лысый'\rangle \circ [x_1^1 \perp \langle \_\_,\ t,\ б.\ кор,\ x_1^1,\ 'Франция'\rangle], \qquad (3.12)$$

где *св — иметь свойство*, а *б. кор — быть королем*. Естественно было бы уточнить предыдущее высказывание следующим образом: *это истинный факт, что король Франции лысый*, что представляется в виде обычного графа

$$[x_2^1 := t] \circ [x_2^1 \perp \langle \_\_, x_2^1,\ св,\ x_1^1,\ 'лысый'\rangle] \circ [x_1^1 \perp \langle \_\_,\ t,\ б.\ кор,\ x_1^1,\ 'Франция'\rangle]. \qquad (3.13)$$

Здесь вершина x_2^1 соответствует логической составляющей утверждения X_1 *лысый*. Возможны и другие способы уточнения. Например, высказывание типа *Свойство, которым обладает король Франции, есть лысость* представляется следующим образом:

$$[x_3^1 := \text{'лысый'}] \circ [x_3^1 \perp \langle \_\_,\ t,\ св,\ x_1^1,\ x_3^1 \rangle] \circ [x_1^1 \perp \langle \_\_,\ t,\ б.\ кор,\ x_1^1,\ \text{'Франция'} \rangle]. \qquad (3.14)$$

Подобные графы изображаются таким же способом, как показано на рис. 3.8, с тем отличием, что добавляются стрелки порядка, идущие от вершины связи (помеченной звездочкой). В случае (3.13) эта стрелка направляется к *о*-вершине t (обозначенной через x_2^1), а в случае (3.14) — к *о*-вершине *'лысый'* (обозначенной x_3^1). Отсюда видно, что любое расположение стрелок порядка связано со своим вариантом уточнения информации. При этом могут представляться различные ее смысловые оттенки.

Приведем еще несколько примеров. На рис. 3.9 изображен смешанный граф, представляющий требование узнать, *имеется ли такой человек* X_1, *который является другом* X_2 *и братом* X_3. При этом X_2 *является мужем* X_3. Граф записывается в виде

$$[x_1^1 \| человек \perp \langle \_\_\ t,\ б.\ др,\ x_1^1,\ x_2^1 \ | человек \rangle] \circ [x_1^1 \perp \langle \_\_,\ t,\ б.\ бр,\ x_3^1 | человек \rangle \circ \langle \_\_,\ t,\ б.м\text{-}ж,\ x_2^1,\ x_3^1 \rangle \circ [?x_1^1 := \lambda]. \qquad (3.15)$$

В данном графе в соответствии с введенными в § 1.4 обозначениями использована конструкция $x_1^1 \| человек |$ означающая, что X_1 *относится к классу людей*. То же самое можно было представить в расшифрованном виде $x_1 \perp \langle \_\_,\ t,\ \in,\ x_1,\ \text{'кл. людей'} \rangle$.

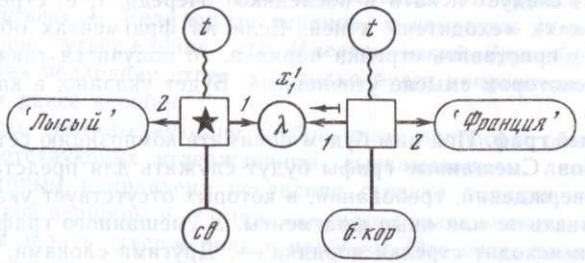


Рис. 3.8. Смешанный граф, представляющий факт, что *король Франции лысый*

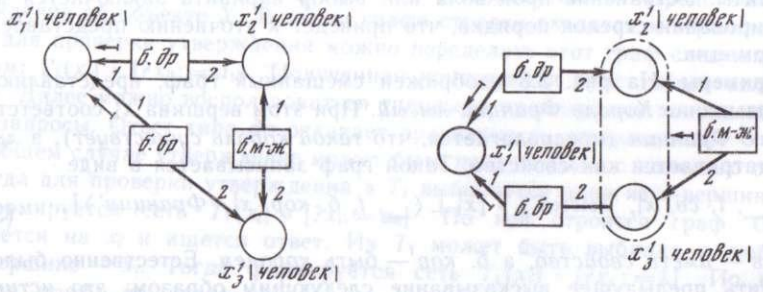


Рис. 3.9. Смешанный граф, представляющий родственно-дружеские связи между тремя индивидуумами X_1, X_2 и X_3

Рис. 3.10. Пример приведенного графа

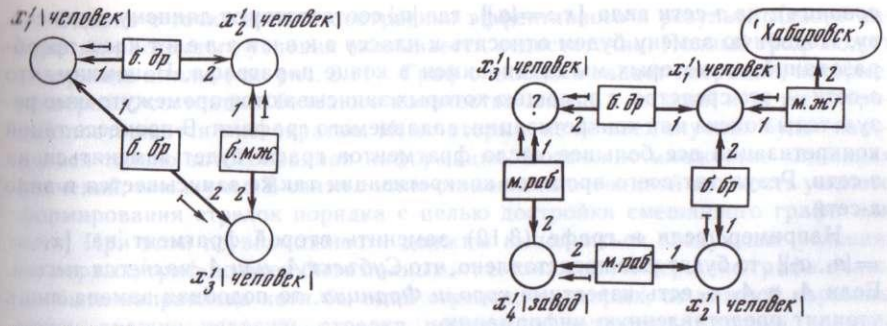


Рис. 3.11. Пример графа, содержащего цикл

Рис. 3.12. Граф, представляющий требование *найти человека, являющегося другом хабаровчанина и сослуживцем брата этого друга*

Отметим, что в графе (3.15) не указано, как находить значения x_2^1 и x_3^1. Это можно сделать согласованной конкретизацией, т. е. построив другой граф, изображенный на рис. 3.10. Последний записывается в виде композиции (3.15), в которой вместо $\langle \_\_,\ t,\ б.\ м\text{-}ж,\ x_2^1,\ x_3^1 \rangle$ будет $[(x_2^1,\ x_3^1) \perp \langle \_\_,\ t,\ б.\ м\text{-}ж,\ x_2^1,\ x_3^1 \rangle]$.

Другой граф

$$[x_1^1 | человек \perp \langle \_\_, t, б.\ др, x_1^1, x_2^1 \rangle] \circ [x_2^1 | человек \perp \langle \_\_, t, б.\ м\text{-}ж, x_2^1, x_3^1 \rangle] \circ \circ [x_3^1 | человек \perp \langle \_\_,\ t,\ б.\ бр,\ x_1^1,\ x_3^1 \rangle] \circ [x_1^1 := ?] \qquad (3.16)$$

представляет требование *найти, кто является другом мужа своей сестры*. Такой граф показан на рис. 3.11. В нем стрелки порядка расположены так, что образуется ц и к л. Отметим, что граф с циклом может быть построен только по такой сети, в которой имеются к о н т у р ы, т. е. возможны замкнутые пути. В таких сетях существует хотя бы одна такая *н*-вершина, от которой, двигаясь по ребрам через *н*-вершины и вершины связи, можно придти к ней самой.

Введенные графы позволяют реализовать достаточно сложные методы **структурного поиска**, что будет подробно рассматриваться в § 4.4. Сейчас только проиллюстрируем некоторые возможности на примере, который уже обсуждался в § 3.3. На рис. 3.12 изображен граф, представляющий требование *нахождения того, кто имеет друга, живущего в Хабаровске, и работает на одном заводе с братом этого друга*. При этом отношение *иметь место жительства* представлено с помощью *м. жт*, а *иметь место работы* — с помощью *м. рб*. Напомним, что расположение стрелок $\mapsto$, т. е. вид графа, мы связывали с операциями по обработке. Говорили, что граф задает такие операции. В графе рис. 3.12 стрелки расположены так, что они задают вариант обработки, рассмотренный в § 3.3 первым. Предлагаем читателю самому убедиться, что изменением направленности стрелок может быть получен как второй, так и третий варианты (в третьем варианте имеется цикл).

Будем допускать в смешанных графах замену фрагментов $[x_1 \perp T_1(x_1)]$, сопоставленных *множеству объектов* (и представляющих *факт их сущест-*

вования), на *з*-сети вида $[x_1 := \{a_i\}]$, где $\{a_i\}$ соответствуют данному множеству. Подобную замену будем относить к классу эквивалентных преобразований, на которых мы остановимся в конце параграфа. Напомним, что *з*-сети — это средства, с помощью которых записываются промежуточные результаты поиска или конкретизации, задаваемого графами. В процессе такой конкретизации все большее число фрагментов графа будет заменяться на *з*-сети. Результат всего процесса конкретизации также записывается в виде *з*-сети.

Например, если в графе (3.12) заменить второй фрагмент на $[x_1 := \{a_1, a_2\}]$, то будет уже представлено, что *Субъект* A_1 *или* A_2 *является лысым*. Если A_1 и A_2 — есть известные *короли Франции*, то подобная замена лишь уточнит представленную информацию.

В общем случае будем допускать замену фрагментов $[z_1 \perp T_1(z_1)]$ на $[z_1 := \{w_i\}]$, где z_1 — набор *н*-вершин, распространяя все сказанное выше на наборы.

Усложним понятие смешанного графа. Будем называть смешанно-составным такой граф, в множество фрагментов которого входят как сети, так и составные графы. Словом, допускается композиция сетей, элементарных графов и составных графов. У таких графов отсутствует указание на то, как использовать те или иные фрагменты. Причем последние могут входить в области, для которых указаны уточняемые по ним значения *н*-вершин. Это наиболее общий случай.

Реализация способов обработки. В § 3.3 были описаны три случая обработки. Введенные конструкции СЯ (сети, графы) обеспечивают их реализацию. Отметим, что далеко не каждая конструкция СЯ задает операции, которые имеют вид законченного алгоритма. Чем больше в такой конструкции стрелок порядка $\mapsto$, тем в большей степени будут определенными операции. В обычной семантической сети таких стрелок вообще нет. Допускается полный произвол. Случайным образом выбираются *н*-вершины и связанные с ними фрагменты, по которым уточняются значения этих *н*-вершин. Смешанные же графы задают некоторую комбинацию из описанных только что операций случайного выбора, а также операций направленного уточнения. Последние задаются фрагментами, из вершин связи которых исходят стрелки порядка. С помощью таких операций находятся или уточняются значения *н*-вершин, в которые входят (направлены) стрелки порядка. Все другие фрагменты выбираются и используются «хаотическим» способом, что приводит к уточнению их *н*-вершин. При этом если у фрагмента имеется несколько *н*-вершин, то выбор той из них, которая уточняется, также осуществляется случайным образом.

Как отмечалось в § 3.3, один из способов обработки предполагает предварительное уточнение направления поиска. Вначале сеть или смешанный граф достраивается до графа, задающего операции алгоритмического характера. Устраняется всякий произвол. Однозначно указывается, что и как следует уточнять. Графы, задающие такие операции, получили название приведенных. Рассмотрим их особенности.

Полные и приведенные графы. Для построения законченного алгоритма требуется, по крайней мере, чтобы в каждую *н*-вершину (связанной сети) входили стрелки порядка. Данное условие легло в основу понятия полного графа. В процессе обработки информации каждый смешанный граф должен быть достроен до полного (см. рис. 3.10).

Однако с точки зрения построения эффективного и результативного алгоритма введенного условия оказывается недостаточно. Полный граф может содержать циклы (см. рис. 3.11). Граф с циклами задает операции, которые, во-первых, состоят из большого количества шагов, т. е. одни и те же операции повторяются многократно. И, во-вторых, не во всех случаях будет находиться только затребованная информация. Возможно множество «лишних» значений, т. е. шумы. В связи с этим становится очевидной важность умелого формирования стрелок порядка с целью достройки смешанного графа или сети. При этом по возможности должны выполняться следующие условия: во-первых, граф должен быть полным, т. е. к каждой *н*-вершине графа должна быть направлена хотя бы одна стрелка, во-вторых, от каждой вершины связи должна исходить стрелка порядка и, в-третьих, стрелки должны сходиться к *н*-вершине, соответствующей затребованной компоненте. Перечисленные условия могут быть выполнены только на связанных графах, т. е. из которых при удалении стрелок порядка получаются связанные сети. Такие условия легли в основу понятия приведенного графа. В последнем используются все фрагменты, по которым находятся все неопределенные составляющие и исключается возможность зацикливания. Приведенные графы задают операции, носящие характер законченных и результативных алгоритмов.

Приведенные графы строятся с помощью специальных процедур — формирования стрелок порядка. Еще раз отметим, что один из основных критериев при построении приведенных графов состоит в минимизации сложности задаваемых ими операций. При этом весьма желательно исключение сложных операций перебора. Хотя в ряде случаев для этого приходится изменять конструкцию графа, направленность стрелок порядка.

Теоретико-множественные аналоги. Аналогом элементарных графов $x_1 \perp T_1^0(x_1)$ являются математические конструкции вида $\{X_1/R_1^0(X_1)\}$, где $R_1^0(X_1)$ — базовый предикат, область истинности которого определяет множество X_1. Сеть $T_1^0(x_1)$ представляет этот предикат. Аналогом графов $[x_1 \perp T_1(x_1)]$ является эта же конструкция, к которой добавляется условие $X_1 \neq \varnothing$. Аналогом графов $z_1 \perp T_1(z_1)$ является конструкция $\{(X_1, ..., X_n)/R_1(X_1, ..., X_n)\}$, где R_1 может быть достаточно сложным предикатным выражением. Например, аналогом $x_1 \perp T_1(x_1) \circ T_2(x_1)$ является $\{X_1/R_1(X_1) \wedge R_2(X_1)\}$, где R_1 и R_2 представляются с помощью T_1 и T_2.

Аналогом *з*-сетей являются обычные множества, а также альтернативные множества и их семейства. Аналогом композиции графов являются наборы, составленные из упоминавшихся конструкций. Так, аналогом графа $[x_1 \perp T_1^0(x_1, x_2)] \circ [x_2 \perp T_2^0(x_2)]$ является $\{X_1/R_1(X_1, X_2)\}$, $\{X_2/R_2(X_2)\}$ с условиями $X_1 \neq \varnothing$ и $X_2 \neq \varnothing$. Аналогом другого графа $[x_1 \perp T_1^0(x_1)] \circ [x_1 \perp T_2^0(x_1)]$ является конструкция

$\{X_1^*/R_1(X_1^*)\}$, $\{X_1^{**}/R_2(X_1^{**})\}$, $X_1 = X_1^* \cap X_1^{**}$,

где X_1^* и X_1^{**} — различные множества. Здесь к набору добавляется теоретико-множественное соотношение.

Аналогом составного графа вида $x_1 \perp \Gamma_1(x_1)$, где Γ_1 — также граф, является конструкция $\{X_1/G_1(X_1)\}$, где $G_1(X_1)$ — есть алгоритмическое или полуалгоритмическое выражение, задаваемое введенными ранее средствами. Такое выражение представляется в виде $\Gamma_1(x_1)$. С помощью него может быть проверена принадлежность любого элемента к множеству X_1.

Аналогом смешанных графов являются более сложные конструкции, в которые включаются интерпретации сетей. В связи с этим рассмотрим вначале, что является аналогом сетей. Если сеть составлена из фрагментов $\langle \_\_, t, r_i, ... \rangle$, то аналогом является конструкция, которая представляет собой конъюнкцию базовых предикатов $\bigwedge_i R_i(...)$. Если в сети имеется фрагмент $\langle \_\_, f, r_i, ... \rangle$, то в конъюнкции на месте $R_i(...)$ ставится $\overline{\ }R_i(...)$, т. е. отрицание. В случае фрагментов $\langle \gamma_i, t, r_i, ... \rangle$ аналоги уже не могут быть записаны в языке логики предикатов. Требуется расширенный язык логи-

ки, в который должны быть введены средства именования предикатных выражений $D_j \overline{=}_{Dj} R_i(...)$. Символы D_j вводятся в множество констант и сопоставляются ситуациям. Допускается подстановка этих символов на аргументные места предикатов. Подобные выражения и будут аналогами фрагментов, у которых первое место занято вершиной γ_j. Аналогом фрагментов $\langle \_\_, p_\xi, r_i, ... \rangle$, где p_ξ — *н*-вершина, является выражение $R_i(...) = P_\xi$, где P_ξ — пропозициональная переменная. Аналогом сетей, состоящих из перечисленных фрагментов, является конъюнкция описанных выражений. Аналог смешанного графа получается добавлением к такой конъюнкции введенных ранее конструкций, определяющих множества.

Наличие аналогов показательно с двух точек зрения. Во-первых, язык графов был введен как средство согласованного представления естественной (семантической) информации. Логика предикатов и язык теории множеств также возникли как средство формализации естественных представлений и рассуждений определенного характера. Отсюда и источник аналогов. И, во-вторых, становится ясным, что каждому графу можно сопоставить строгую математическую модель. Тогда ввод описаний и их преобразование во внутрисистемное представление (СП) можно трактовать как задачу автоматической формализации описания с построением математической модели. А построение по сети графа можно трактовать как задачу автоматического преобразования предикатных конструкций в теоретико-множественные. Собственно, с учетом этих задач и осуществлялся выбор внутрисистемных средств (см. § 1.1).

ГЛАВА 4

ОПЕРАЦИИ МАНИПУЛИРОВАНИЯ НА СТРУКТУРАХ

Обычно под средствами манипулирования данными понимают группу алгебраических операций, осуществляющих их объединение, пересечение, дополнение. В ряде случаев такие операции распространяются для работы с таблицами, традиционными графами [10, 47]. Вводятся новые операции. Такой подход принят в реляционных базах данных, где для целей манипулирования разработан специальный язык реляционной алгебры. Запрос вначале формализуется, записывается в языке реляционного исчисления и затем редуцируется (преобразуется) в выражение реляционной алгебры. Таким способом осуществляется обработка.

Описанный подход представляется весьма перспективным. Но для обработки естественно-языковой информации с учетом операционных оттенков требуется его развитие. Во-первых, для представления запросов лучше использовать средства, согласованные с естественно-языковыми формами и конструкциями. В качестве таких средств будем использовать введенный ранее язык семантических графов, обеспечивающий представление не только запросов, но и разного рода гипотез, утверждений. И, во-вторых, для обработки лучше использовать группу операторов, которые можно непосредственно связывать с естественно-языковыми формами. Будем называть их операторами е с т е с т в е н н о й обработки. Тогда делается возможной последовательное восприятие естественно-языковой информации с выделением самостоятельных конструкций (форм), по которым путем использования связанных с ними операторов тут же могут быть уточнены имеющиеся в виду объекты, отношения.

Конечно, среди операторов естественной обработки должны иметься такие, которые обеспечивают работу над множествами. Должны иметься и аналоги операторов реляционной алгебры, которые в ряде случаев позво-

ляют обеспечить естественную обработку. Но множество таких операторов должно быть расширено. Помимо указанных, также должны иметься специальные операторы перебора вариантов (с формированием альтернатив, выделением адекватных элементов и др.), проверки и др. Они необходимы для работы с естественно-языковыми конструкциями, содержащими категории единственного числа, имеющими оттенки долженствования и т. д. В данной главе будет рассматриваться группа таких операторов, которые будем связывать не с естественно-языковыми формами, а с их семантическими эквивалентами. Роль последних будут играть семантические графы. Графы задают операции естественной обработки, выполнение которых над сетями приводит к нахождению, уточнению, преобразованию информации. Например, операции, которые задаются графом, представляющим запрос, выполняются над сетью-знаниями. В результате находится ответ на запрос.

Ниже будет введено лишь подмножество операторов, необходимых в основном для поиска и уточнения сравнительно простой информации, выражаемой с помощью языковых форм, не содержащих слов-кванторов типа *каждый*, *какие-либо*, *только* и др. Другое подмножество будет рассматриваться в гл. 7, посвященной средствам представления обобщенной информации. Операторы, обеспечивающие разного рода изменения, преобразования, будут введены в гл. 5, посвященной вопросам представления зависимостей, оттенков долженствования и др.

§ 4.1. ПРИНЦИПЫ УЧЕТА ОПЕРАЦИОННЫХ ОТТЕНКОВ

Ранее говорилось, что многие выражения имеют операционный оттенок, для представления которого был введен язык семантических графов. Оттенок будет учитываться сопоставлением графам операций манипулирования или обработки. Как уже отмечалось, граф з а д а е т такие операции. При рассмотрении графов подобные операции в какой-то степени оговаривались. Сейчас нас будут интересовать вопросы формализации операционных оттенков. Какой язык необходимым для записи операторов? Можно ли использовать для этой цели существующие языки и в какой степени?

Будем различать **операционные оттенки двух видов.** Первый — когда указывается направление поиска, проверок. Такой оттенок имеют запросы, высказывания. И второй вид — когда указываются изменения, которые имеют место или должны иметь место. Подобный оттенок характерен для определений, а также для гипотез, утверждений, выражаемых с высокой долей уверенности (см. § 5.2).

В связи со сказанным будем различать две группы операторов. Первая обеспечивает поиск неопределенных компонент представленной информации, выполнение разного рода проверок, переборы вариантов и т. д. Вторая группа обеспечивает изменение компонент и отдельных конструкций СЯ — изъятие, добавление, замену. Первая группа операторов задается графами, которые были введены в § 3.4. Такие операции приводят к уточнению информации, представленной с помощью самих же графов. Естественно, операционные оттенки вопроса служат для указания направления уточнения его же компонент. Вторая группа операторов задается специальными графами, представляющими гипотезы, разного рода определения, зависимости. Такие графы будут введены в гл. 5. Задаваемые ими операции воздействуют

на другие структуры, которые могут предполагаться или быть специальным образом указаны. Например, *Ты сказал неправильно: не Иванов поставляет товар, а Петров.* Здесь указывается на необходимость изменения высказывания. В общем случае (если нет специальных указаний) может предполагаться изменение других структур. Например, утверждения типа *Вначале имело место* $\mathcal{T}_1$, *затем* $\mathcal{T}_2$, предполагают изменение исходной ситуации, утверждение A_1 — *лжец* — изменение внутрисистемных знаний.

Операторный язык. Каким же требованиям должны удовлетворять обе группы операторов? Прежде всего, их аргументами и результатами (в том числе промежуточными) должны быть правильные конструкции СЯ. Это необходимо для того, чтобы в любой момент можно было остановить выполнение операций, задаваемых тем или иным графом. В результате должна оставаться правильная конструкция. Дело в том, что часто по тем или иным причинам обработка не может быть закончена. Например, пусть при поиске ответа на запрос оказалось, что о некоторых свойствах системе ничего не известно. Тогда выполнение операций должно быть приостановлено, но то, что уже было получено, должно сохраняться. Результат выполнения предыдущих операций должен приводить к правильной конструкции СЯ — уточненному графу. Сказанное относится и ко второй группе операторов. Любые изменения, задаваемые соответствующими графами, должны приводить к формированию правильных конструкций СЯ.

Какими же должны быть сами операторы? При восприятии многих видов естественно-языковой информации как бы напрашивается тот или иной способ обработки. Например, в предложении *Именно Петр является братом Ивана* как бы предполагается необходимость нахождения братьев Ивана и проверки, что среди них имеется Петр. Для этого естественно использование операторов поиска объектов по заданному отношению и проверки. В вопросе *Кто друзья братьев Ивана*? предполагается нахождение братьев, а затем — друзей. Сравните с вопросом *Кто друг братьев Ивана*? Предполагается выделение общего друга. Возьмем еще один вопрос — *Кто друзья брата Ивана*? Если братьев несколько, то хочется переспросить *Какого брата*? Иначе требуется их перебор с нахождением для каждого из них множества друзей.

Отсюда видно, что незначительное изменение языковых категорий может приводить к изменению характера обработки используемых операторов. Продолжая рассматривать примеры, можно постараться выделить более или менее полный набор операторов так называемой естественной обработки. Ясно, что в рамках систем с СП должны обеспечиваться запоминание и выполнение таких операторов. В дальнейшем для этой цели будем использовать специальный алгоритмический язык, который будем называть операторным языком (ОЯ). Впервые такой язык был рассмотрен в работе [24], а позднее развит в [14].

Базовый интерпретатор [1]. Как показал опыт, сразу построить какой-либо алгоритм, соотносящий графам операторы ОЯ, не представляется возможным. В ЕЯ имеется большое количество форм, несущих специфические операционные оттенки. Учесть все их оказалось достаточно трудно. Поэтому

[1] Разработка интеллектуальной системы во многом сводится к построению базового интерпретатора. Это ядро системы, от которого зависят ее возможности в плане постоянного совершенствования своих умений — внутрисистемных акций.

система должна быть «обучающейся» в том смысле, что должна иметься возможность постоянного дополнения форм и соотнесения каждой новой формы с соответствующими операторами. Последние должны конструироваться системой за счет входной информации, т. е. специального вида пояснений, определений. Рассмотрим более подробно, что это за информация. Тогда станет яснее, каким же должно быть устройство соотнесения, т. е. базовый интерпретатор.

Будем различать **три случая конструирования и соотнесения.** Первый определяется пояснениями типа *Как найти друга брата Ивана? Вначале нужно найти брата, а затем его друга.* Определяющая часть имеет вид предписания, которое представляется с помощью сети, а последняя отображается на операторы ОЯ (см. § 2.4 и 3.1). По определяющей части строится оператор, который и соотносится с вопросом. Далее следует процесс обобщения. Конкретный вопрос обобщается до языковой формы, а в операторе соответствующие величины заменяются на переменные *Какая разница, что искать (брата, сестру, мать...)* и *у кого искать (у Ивана, Петра...)*? Принцип поиска остается тем же.

Пояснения особенно необходимы в сложных случаях, когда вопрос носит характер задачи со многими неизвестными. С их помощью задается методика, способ решения задачи, используемые при этом операторы. Вводятся и новые операторы, необходимые для решения. Такого сорта пояснения должны учитываться системой, т. е. отображаться на ОЯ. Отсюда следуют жесткие требования к ОЯ. Дело в том, что некоторые формы пояснений (предписаний на ЕЯ) заимствованы из существующих алгоритмических языков. Ясно, что возможности ОЯ должны быть не ниже. В частности, ОЯ должен допускать различные способы управления. В нем должны быть операторы условного перехода, цикла и многие другие.

Второй способ конструирования и соотнесения предполагает запоминание удачных акций в процессе ответа на запросы, проверки правильности сообщений и др. Пусть вопрос имеет достаточно типовую форму и каким-то образом был построен соответствующий оператор. Тогда последний может быть непосредственно связан с этой формой. Полученная пара обобщается (см. выше). По ней любому вопросу аналогичного типа сразу может быть соотнесен оператор.

В данном случае предполагается, что имеется некий механизм автоматического конструирования операторов в процессе последовательного анализа вопросов, сообщений. Каким он должен быть? Здесь следует учитывать составной характер типовых предложений ЕЯ. В них могут быть части со своими операционными оттенками. Например, части (компоненты форм) могут указывать на объекты, а операционные оттенки — как находить эти объекты. Части могут входить в более сложные конструкции, с помощью которых указывается, что делать с найденными объектами: перебирать, проверять и т. д. Естественный процесс понимания заключается в последовательном выявлении таких частей, нахождении их семантических эквивалентов с переходом к анализу все более сложных конструкций. Такому процессу должны быть сопоставлены действия последовательного конструирования оператора ОЯ. Важную роль здесь играет метод суперпозиции. Например, в процессе анализа предложения *Дочь Петра является женой сына Ивана* должны последовательно конструироваться операторы нахождения дочери Петра (обозначим такой оператор через Q_1),

сына Ивана Q_2. Суперпозицией составляется более сложный оператор Q_3 (Q_1, Q_2), где Q_3 осуществляет проверку отношения *быть женой.*

Аналогичное конструирование требуется в процессе восприятия системой специальных определений, т. е. когда одна форма ЕЯ определяется через другую. По последней должен быть сконструирован оператор ОЯ. Он и соотносится с первой формой. Например, возьмем предложение *проверить, что каждый элемент множества* M_1 *обладает свойством* R_1*, можно, проверив, что первый элемент обладает этим свойством, затем второй элемент и т. д.* По второй части данного предложения строится оператор ОЯ, который соотносится с первой частью. Далее предполагается умение системы обобщать предложение до соответствующей языковой формы и умение продолжить начатую последовательность, как это имело место в последнем примере. Конечно, в ряде случаев определения уже могут носить обобщенный характер, например *каждый — и тот, и другой, и третий* [22]. Для начатой последовательности может быть указано, где она заканчивается. Здесь соответствующих механизмов может не потребоваться.

Наконец, еще один способ конструирования операторов связан с так называемым **обучением по примерам**. Системе дается множество вопросов и ответов на них. Она сама должна построить те операторы, которые позволяют находить ответы. Спрашивается, как должна быть устроена подобная система? Здесь оказывается весьма важной последовательность предоставляемых примеров: пар вопрос—ответ. Процесс обучения должен идти от простого к сложному. Видимо, через аналогичного сорта обучение прошел каждый из нас. Начинается с простого: *Кто твой дядя? Дядя Петя — вот он, и дядя Гриша...* По таким примерам обучаются умению искать, проверять и т. д. Аналогичными возможностями в перспективе должна обладать и система, в которой по примерам должно обеспечиваться формирование операторов поиска, проверки и т. д.

Понятие семантической конструкции. Для автоматического конструирования операторов (базовым интерпретатором) требуются специальные средства настройки, расширения возможностей. Они должны иметь вид пар, состоящих из левой и правой частей и связывающих предложения ЕЯ с операторами естественной обработки. (В дальнейшем такие пары будут называться м е т а п р а в и л а м и.) Какие средства формализации здесь следует использовать? Пока остановимся на принципах формализации левых частей.

Прежде всего операторы должны связываться не с группами слов или отдельными предложениями (которых неизмеримо много), а их ф о р м а м и, сохраняющими операционный оттенок классов родственных предложений. Об этом уже говорилось выше. В таких формах некоторые места могут быть не заняты. Могут быть указаны (или не указаны) группы слов, стоящих на этих местах. Пусть $X_i^н$ $R_\xi^н$ $X_j^н$ — одна их таких форм, где через $X_i^н$, $X_j^н$, $R_\xi^н$ обозначены достаточно произвольные группы слов, называющие объекты X_i, X_j и отношения R_ξ. Порядок расположения слов в такой форме фиксирован. В одном из вариантов с каждой формой может быть связана своя группа операторов, в данном случае проверяющая наличие отношения. Тогда построение оператора по исходному предложению будет сводиться к последовательному выявлению форм.

Указанный вариант, конечно, лучше, чем вариант, в котором операторы связываются с отдельными предложениями. Но и эквивалентных форм,

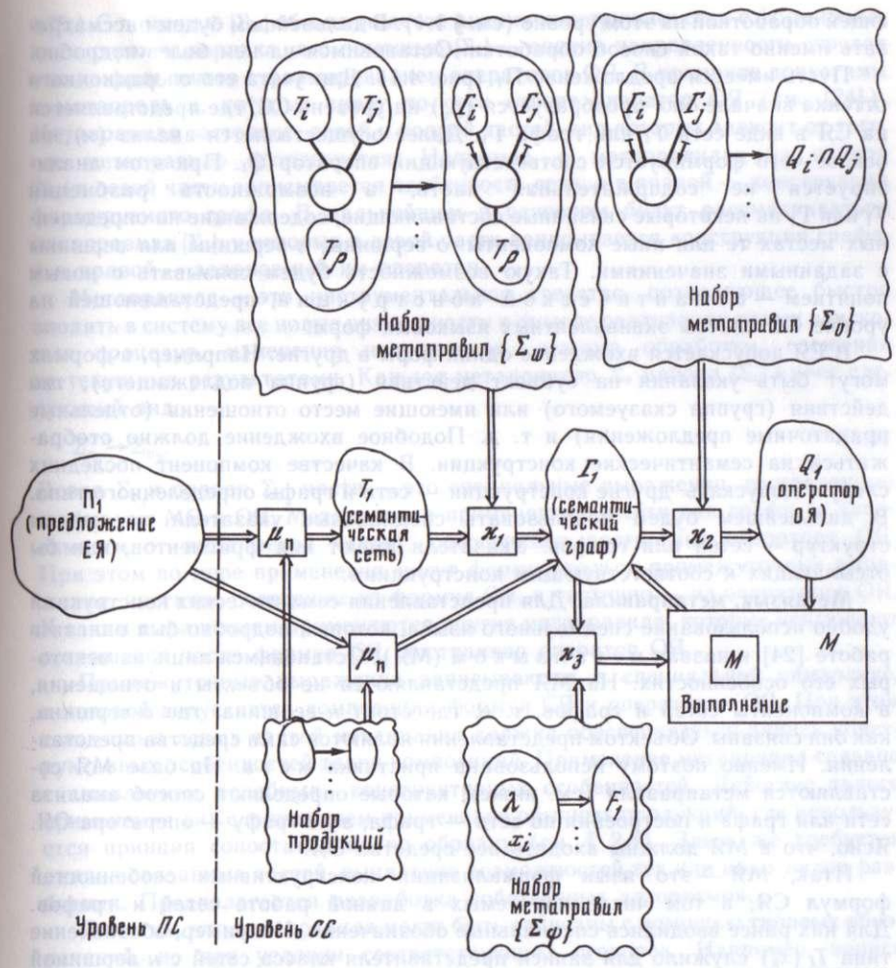


Рис. 4.1. Схема обработки, основанная на учете операционных оттенков языковых форм

выражающих одно и то же, может быть значительное количество. Все они должны связываться с одной и той же группой операторов. Например, если речь идет об анализе предложений русского языка, то, помимо формы $X_i^н$ $R_\xi^н$ $X_j^н$, потребуется еще пять форм, полученных из предыдущей перестановкой символов (в русском языке во многих случаях перестановка слов сохраняет смысл). Имеются и другие свойства инвариантности, связанные с возможностью замены слов на синонимы, выражения свойств с помощью причастных и деепричастных оборотов и т. д. Одно и то же можно выразить по-разному, с помощью различных конструкций ЕЯ. И чем сложнее предложение, тем таких конструкций будет больше. Отсюда — необходимость предварительного формирования семантических структур (СС) с последу-

ющей обработкой на этом уровне (см. § 1.1). В дальнейшем будем рассматривать именно такой способ обработки. Остановимся на нем более подробно.

Пусть имеется предложение $П_1$ (рис. 4.1). Для учета его операционного оттенка вначале оно преобразуется ($\mu_п$) на уровень СС, где представляется на СЯ в виде сети T_1 или графа $Г_1$. Далее осуществляется анализ ($\varkappa$), на основе чего формируется соответствующий оператор Q_1. При этом анализируется не содержательная часть, а возможность разбиения T_1 или $Г_1$ на некоторые связанные составляющие, содержащие на определенных местах те или иные компоненты: *о*-вершины, *н*-вершины или вершины с заданными значениями. Такую возможность будем связывать с новым понятием — семантической конструкции, представляющей на уровне СС классы эквивалентных языковых форм.

В ЕЯ допускается вхождение одних форм в другие. Например, в формах могут быть указания на субъект действия (группа подлежащего), тип действия (группа сказуемого) или имеющие место отношения (отдельные придаточные предложения) и т. д. Подобное вхождение должно отображаться на семантические конструкции. В качестве компонент последних следует допускать другие конструкции — сети и графы определенного типа. В дальнейшем будем использовать специальные указатели на классы структур — сетей или графов. Указатели имеют вид фрагментов, как бы отсылающих к соответствующим конструкциям.

Метаязык, метаправила. Для представления семантических конструкций удобно использование специального языка, который подробно был описан в работе [24] и назван метаязыком (МЯ). Остановимся лишь на некоторых его особенностях. На МЯ представляются не объекты и отношения, а компоненты сетей и графов, т. е. где стоит *н*-вершина, где *о*-вершина, как они связаны. Объектом представления являются сами средства представления. Именно поэтому использована приставка мета. На базе МЯ составляются метаправила (см. ниже), которые определяют способ анализа сети или графа и построения по сети — графа, а по графу — оператора ОЯ. Ясно, что в МЯ должны входить все средства ОЯ.

Итак, МЯ — это язык формализации конструктивных особенностей формул СЯ, в том числе изучаемых в данной работе сетей и графов. Для них ранее вводились специальные обозначения. Например, обозначение типа $T_i\ (x_k)$ служило для записи представителя класса сетей с *н*-вершиной x_k. При этом индексы i и k указывали на то, что представитель и *н*-вершина не фиксируются, т. е. могут быть достаточно произвольными. Обозначение типа $T_i(x_k) \circ T_j(x_k)$ использовалось для указания достаточно произвольного разбиения сети на две части с общей *н*-вершиной x_k, которая конкретно не указывается. В ряде случаев с помощью соответствующих оговорок указывались особенности T_i и T_j, например, что это связанные сети. МЯ обеспечивает представление подобной информации. МЯ имеет свою собственную символику, включающую в себя символику СЯ. Но есть и специфические средства, характерные для описаний на метауровне. В дальнейшем, чтобы не запутать читателя, будем стараться избегать этой специальной символики, пользуясь типовыми обозначениями. Но всегда следует помнить о возможности их формализации.

Метаправила строятся на основе МЯ и ОЯ и необходимы для преобразования сетей в графы, а графов — в операторы ОЯ. Используются сменные наборы метаправил, которые на рис. 4.1 обозначены через $\{\Sigma_w\}$

и $\{\Sigma_v\}$. С помощью $\{\Sigma_w\}$ обеспечивается анализ конструкции сети с формированием стрелок порядка, а с помощью $\{\Sigma_v\}$ — анализ направленности стрелок и вида фрагментов с формированием операторов ОЯ. В принципе допустимы и метаправила, которые сразу по сети строят оператор ОЯ (см. [24]). Метаправила состоят из левой и правой частей, вид которых зависит от того, где метаправило используется. Например, в метаправилах из набора $\{\Sigma_w\}$ в левой части записывается особенность сети, а в правой — конструкция формируемого графа. В дальнейшем в основном будут рассматриваться метаправила $\{\Sigma_v\}$, у которых в левой части записывается конструкция графа, а в правой — задаваемый им оператор.

Метаправила — это инструментальное средство, позволяющее быстро вводить в систему все новые возможности в смысле реализации новых поисковых процедур, изменения направления, режима обработки, слежения за текущими результатами. Каждое метаправило Σ_v набора $\{\Sigma_v\}$ имеет следующий вид:

$$\Sigma_{v_1} \rightarrow \Sigma_{v_2}. \tag{4.1}$$

Левая Σ_{v_1} и правая Σ_{v_2} части — это специальные выражения, составленные из формул МЯ и ОЯ. Метаправила применяются к сети или графу (в зависимости от вида левой части), обеспечивая построение оператора ОЯ. При этом по мере применения могут формироваться промежуточные выражения, частично состоящие из формул СЯ, а частично — из операторов ОЯ. К таким выражениям применяются другие метаправила, которые исключают оставшиеся части формул СЯ, достраивая оператор ОЯ.

Промежуточные выражения записываются в специальной символике, в которой допускается композиция формул СЯ и операторов ОЯ. При этом с помощью левых частей метаправил должна обеспечиваться запись конструктивных особенностей таких композиций. Применение метаправил связано с выявлением подобных конструктивных особенностей. Действие такого применения ($\varkappa$) сложнее, чем применение обычных продукций, где используется принцип сопоставления по образцу (см. § 2.1). Здесь же требуется анализ связанных частей, выявление возможностей тех или иных видов разбиения. Предполагается разработка собственных алгоритмов.

Отметим, что метаправила могут быть записаны с помощью **типовых обозначений**, но при условии соответствующих оговорок. Например, запись

$$Г_i(x_1) \circ Т_\zeta(x_1,\ x_2) \circ Г_j(x_2) \rightarrow Q_\zeta(Q_i,\ Q_j) \tag{4.2}$$

есть метаправило, в левой части которого записаны конструктивные особенности графа: он должен состоять из подграфов $Г_i(x_1)$ и $Г_j(x_2)$, связанных фрагментом $Т_\zeta(x_1,\ x_2)$. При этом предполагается, что по $Г_i(x_1)$ и $Г_j(x_2)$ могут быть построены задаваемые ими операторы ОЯ — Q_i, Q_j (с помощью других метаправил). Данное метаправило применяется к исходному графу Г. И если он имеет указанные конструктивные особенности, то строится суперпозиция $Q_\zeta(Q_i,\ Q_j)$, где фрагменту $Т_\zeta$ сопоставляется оператор Q_ζ.

На рис. 4.1, в его верхней части, изображен пример другого метаправила, входящего в набор $\{\Sigma_v\}$. С помощью него выявляются *н*-вершины, к которым сходятся стрелки порядка, и формируется оператор теоретико-множественного пересечения.

В дальнейшем для упрощения изложения и некоторые операторы ОЯ будем стараться описывать на ЕЯ, избегая излишней символики. Вместо

стрелки → будем пользоваться словом «задает», т. е. будем говорить, что «та или иная конструкция задает операции по обработке». Хотя еще раз следует отметить, что в принципе возможна полная формализация метаправил и процесса их применения (см. [24]).

Будем различать метаправила по характеру их работы. Ранее рассматривались метаправила, которые строят оператор ОЯ. Предполагается, что имеется специальное устройство M, которое может «прочитать» этот оператор и выполнить соответствующие действия, т. е. включить в работу внутрисистемные механизмы. Чтобы исключить промежуточное звено, перспективно использование правил другого типа, которые на рис. 4.1 обозначены через $\{\Sigma_\psi\}$. Они являются аналогом так называемых а к т - п р о д у к ц и й, обеспечивающих непосредственное **управление действиями**. Такие метаправила имеют вид

$$\Sigma_{\psi_1} \Rightarrow \Sigma_{\psi_2}, \qquad (4.3)$$

где левая часть Σ_{ψ_1} определяет конструкцию графа, задает способ выделения его частей, а правая часть Σ_{ψ_2} — способ непосредственного воздействия, т. е. сигналы, которые должны поступать на устройство M.

Отметим, что метаправила (4.3) необходимы не только для учета операционных оттенков информации, но и для организации самого процесса обработки. Такие метаправила обеспечивают слежение за ходом обработки и осуществляют прерывание, если чего-либо не получается. Например, метаправило $[x_i := \lambda] \Rightarrow F_1$, изображенное на рис. 4.1, может служить для анализа хода выполнения поисковой процедуры. Пусть такая процедура приводит к последовательному уточнению значений $и$-вершин графа Γ_1. Указанное метаправило будет следить за непустотой множеств таких значений. И как только хотя бы у одной из $и$-вершин оказалось пустое множество значений, то обеспечивается прерывание с выполнением действий — F_1, например приводящих к формированию встречных вопросов.

Следует отметить два существенных момента. Во-первых, действия, связанные с прерыванием, могут зависеть от характера решаемой задачи. В одних случаях допускаются встречные вопросы, в других — нет. Настройка на ту или иную задачу осуществляется с помощью сменных наборов метаправил. И, во-вторых, метаправила рассмотренного вида предполагают более сложные виды анализа, чем типовые конструкции акт-продукции $P_1 \Rightarrow F_1$ (их часто называют просто продукциями), где P_1 — декларативная компонента, а F_1 — процедуральная. Подобные продукции основываются на обычном сопоставлении по образцу. Таким образом, в частности, устроены логико-трансформационные правила [31]. В нашем случае предполагается анализ конструктивных особенностей, т. е. сложные виды сопоставления.

Режимы компиляции и интерпретации. Как уже говорилось, метаправила объединяются в наборы. При этом может быть указан или не указан порядок их применения. Если указаний нет, то имеет место с в о б о д н о е применение. Метаправила играют как бы роль «демонов», которые смотрят за объектом (сетью, графом). Каждый «демон» следит за своей частью. И как только выполняются определенные условия, «демон» тут же активизируется, выполняя соответствующие действия.

Конечно, в случае свободного применения метаправила должны быть независимыми. При их параллельном срабатывании не должно быть коллизий следующего типа — когда срабатывание одних метаправил изменит

части сети (графа), входящие в условие срабатывания других — параллельных. Это не всегда удается обеспечить.

Другой случай — когда метаправила упорядочиваются. Наиболее типовой является следующая процедура их применения — очередное метаправило применяется только в том случае, если не применимы все предыдущие.

Будем различать два способа организации процедуры уточнения информации, которые будем называть режимами компиляции и интерпретации. В режиме к о м п и л я ц и и используются метаправила (4.1) вида $\Sigma_{v_1} \rightarrow \Sigma_{v_2}$, которые последовательно применяются к графу Γ_1, обеспечивая построение все более законченного оператора ОЯ. И только когда оператор полностью построен, он выполняется, обеспечивая нахождение значений $и$-вершин.

Режим и н т е р п р е т а ц и и реализуется с помощью метаправил (4.3) вида $\Sigma_{\psi_1} \Rightarrow \Sigma_{\psi_2}$. Операторы выполняются по мере их построения, вызывая уточнение значений $и$-вершин из Γ_1, формирование соответствующих $з$-сетей. При этом $з$-сети замещают фрагменты графа Γ_1. Например, пусть граф Γ_1 имеет вид $\Gamma_i \circ \Gamma_j$ и пусть применением метаправила по Γ_i был построен оператор, выполнение которого привело к формированию $з$-сети $[z_1 := \eta]$. Тогда формируется более простой граф $[z_1 := \eta] \circ \Gamma_j$ и к нему уже начинают применять метаправила. В результате будет получаться все более простой граф, который в конце концов будет преобразован в $з$-сеть.

Оба описанных режима обладают своими достоинствами и недостатками. В режиме интерпретации реализуются естественные виды анализа, заключающиеся в попытках нахождения семантических эквивалентов в процессе выявления компонент. Например, если говорится *о брате Петра*, то сразу строится граф и делается попытка нахождения вершины, соответствующей такому брату. И если вершина найдена, то она замещает свой граф. Фактически находится семантический эквивалент, который в дальнейшем и имеется в виду. В то же время в режиме компиляции легче обеспечить оптимизацию процедуры поиска. Если построенный оператор оказался достаточно сложным, то граф может быть переформирован, направление поиска изменено.

Метасети. В заключение параграфа хочется обратить внимание на один интересный момент. Сети и графы вводились как средства согласованного представления информации, выраженной на ЕЯ. При описании сетей (их синтаксиса, семантики, особенностей) также был использован ЕЯ. Стало быть, должна иметься возможность представления того, что касается описания сетей, с помощью самих же сетей. В частности, описание конструктивных особенностей сетей (и соответствующие выражения МЯ) должны представляться с помощью сетей, т. е. на СЯ. Будем называть последние метасетями, подчеркивая более высокий уровень представления. Рассмотрим некоторые особенности языка метасетей.

Язык м е т а с е т е й (МСЯ) имеет такой же вид, как и семантический язык (СЯ). МСЯ основан на синтаксисе СЯ. В МСЯ также имеются $о$-вершины и $и$-вершины, из которых обычным способом составляются фрагменты, а из последних — сети. Они записываются в виде формул. Только средства МСЯ соответствуют совершенно другим объектам. В МСЯ вводятся $о$-вершины, соответствующие классам вершин СЯ, т. е. классу $и$-вершин сопоставляется своя $о$-вершина (β), классу $о$-вершин — своя (α). При этом представляется деление последних на подклассы, соответствующие объектам и отношениям. Далее вводятся $о$-вершины (τ_i), соответствующие классам фрагментов СЯ определенного вида T_i, а также $о$-вершины (ρ_j), представляющие различные виды связи между ними. В МСЯ допускаются $и$-вершины (χ), соответствующие некоторым нераспознанным компонентам или фрагментам СЯ. Например, формула МСЯ

$$\langle \chi_3^1,\ t,\ \rho_1,\ \chi_1^1\,|\tau_1|,\ \chi_2^1\,|\tau_2|\rangle \qquad (4.4)$$

представляет *связь типа* ρ_1 между фрагментом СЯ класса τ_1 и фрагментом класса τ_2. Самим фрагментом СЯ сопоставлены *н*-вершины χ_1 и χ_2, а образованной ими более сложной конструкции — *н*-вершина χ_3. Под связью типа ρ_1, например, может пониматься, что *фрагменты СЯ содержат общую н-вершину или множество н-вершин, общие части* и т. д. Каждому типу связи сопоставляется своя *о*-вершина МСЯ, представляющая определенное отношение между компонентами. Конструкции простого вида представляются с помощью отдельных фрагментов, а более сложного вида — с помощью множества фрагментов, их композиции. Например, пусть ρ_0 — *о*-вершина, соответствующая отношению *принадлежности вершины к фрагменту*. Тогда формула МСЯ

$$\langle \_\_,\ t,\ \rho_0,\ \chi_3^1\,|\beta|,\ \chi_i^1\,|\tau_1|\rangle \circ \langle \_\_,\ t,\ \rho_0,\ \chi_3^1\,|\beta|,\ \chi_2^1\,|\tau_2|\rangle \qquad (4.5)$$

представляет, что *два фрагмента СЯ содержат одну и ту же н-вершину*, которой сопоставлена $\chi_3^1\,|\beta|$.

Итак, формулы МСЯ строятся точно таким же образом, как и формулы СЯ. Только в МСЯ имеется собственный набор *н*-вершин и *о*-вершин, из которых составляются свои формулы. В эти наборы должны быть включены *о*-вершины и *н*-вершины СЯ, т. е. МСЯ является расширением СЯ. Дело в том, что при описании разного рода разбиений некоторые компоненты могут быть строго фиксированы. Например, может быть указано, что *две связанные части содержат общую о-вершину* — a_i. Для представления подобного сорта разбиений (семантических конструкций) в МСЯ должна быть вершина a_i. Возникают формулы МСЯ и с теми и с другими вершинами. Например, формула МЯ $\langle \_\_,\ t,\ \rho_0,\ a_1,\ \chi_i^1\,|\tau_1|\rangle$ представляет, что *фрагмент класса* τ_1 *содержит о-вершину* a_1. Сказанное относится и к фрагментам. Фрагменты СЯ должны быть включены в МСЯ.

Отметим, что в общем случае МСЯ может быть дополнен средствами, необходимыми для представления операторов ОЯ, метаправил и т. д. По-видимому, имеется принципиальная возможность такого расширения МСЯ, при котором обеспечивается представление всего, что касается самого МСЯ. Могут быть представлены и рассуждения о самих рассуждениях. Здесь возникает благотворная почва для разного рода неразрешимых проблем, для анализа причин неразрешимости, «анатомии» приводящих к ним рассуждений. Этот вопрос будет рассмотрен ниже (см. § 7.4).

§ 4.2. ОПЕРАЦИИ АССОЦИИРОВАНИЯ И ПРОВЕРКИ

В данном параграфе будут рассматриваться операторы, необходимые для ответа на простые вопросы и анализа правильности сообщений. Будут введены операторы группового поиска, названные операциями ассоциирования, и операторы проверки. Будут проводиться аналогии с языком реляционной алгебры.

Операция ассоциирования. Она близка к операции сопоставления по образцу и является одной из важнейших в процессе обработки семантической информации. Например, словосочетания типа *поставщики товаров в города* A_1 *и* A_2, *друзья братьев Петра* и др. означают множества, которые должны определяться по их отношениям с другими множествами. *Поставщики* должны выделяться по отношению *поставлять товар*, связывающему их с множеством $\{A_1, A_2\}$. *Друзья* — это множество людей, выделяемое по отношению *быть другом*, связывающему их с другим множеством — *братьев Петра*. Выделение одних множеств, используя их отношения с другими множествами, и обеспечивается операцией ассоциирования, у которой можно найти много аналогов в существующих языках манипулирования данными [47].

Простейшая операция ассоциирования задается конструкциями вида $x_i \perp T_1^2(x_i)$, т. е. элементарными графами, областями которых являются кортежи (ЭФ) с *н*-вершиной x_i. Операция выполняется над другой сетью —

T_2, к примеру, представляющей системные знания. В T_2 ищутся «аналогичные» (сопоставимые) кортежи, в которых на месте x_i могут стоять достаточно произвольные элементы. Из найденных кортежей выбираются эти элементы (вершины) и подставляются в качестве значения x_i. Итак, результатом операции ассоциирования является сформированная *з*-сеть, задающая значения x_i. В общем случае в кортеже T_1^o может быть несколько *н*-вершин. Тогда находятся их согласованные значения.

Например, граф $x_1 \perp \langle \_\_,\ t,\ б.\ и,\ 'Иван',\ x_1\rangle$, сопоставленный *субъектам с именем (б. и) Иван*, задает операции следующего вида. В T_2 ищутся все кортежи, содержащие на первом месте произвольную вершину (на что указывает символ «__»), на втором месте — *о*-вершину t, на третьем — *о*-вершину *б. и* и на четвертом — *'Иван'*. Из найденных кортежей выбираются вершины, стоящие на пятом месте, т. е. на месте x_1. Они и присваиваются в качестве значений x_1.

Будем записывать описанную операцию в виде

$$\langle \_\_,\ t,\ б.\ и,\ 'Иван',\ \uparrow x_1\rangle\ /T_2. \qquad (4.6)$$

Аргументами операции являются *о*-вершины кортежа, а результатом — найденные значения *н*-вершины x_1. Так, если найдено множество $\{a_i\}$, то результат может быть записан в виде *з*-сети $[x_1 := \{a_i\}]$ или же в виде соотношения

$$\langle \_\_,\ t,\ б.\ и,\ 'Иван',\ \uparrow x_1\rangle\ /\ T_2 = \{a_i\}. \qquad (4.7)$$

Стрелка ($\uparrow$) указывает на *н*-вершину, для которой ищутся значения.

Рассмотрим еще один пример. Граф $(x_1,\ x_2) \perp \langle \_\_,\ t,\ и,\ цн,\ x_1,\ x_2\rangle$, где *и. цн. — иметь цену* задает операцию нахождения согласованных значений пары $(x_1,\ x_2)$, соответствующей *изделию — его цене*. Такая операция сводится к поиску в T_2 кортежей, в которых на втором месте стоит *о*-вершина t, а на третьем — *и. цн.* Из найденных кортежей выбираются *о*-вершины, стоящие на четвертом и пятом местах. Они и присваиваются в качестве значений $(x_1,\ x_2)$. Подобная операция записывается в виде

$$\langle \_\_,\ t,\ и.\ цн,\ \uparrow x_1,\ \uparrow x_2\rangle\ /\ T_2. \qquad (4.8)$$

Аналогичным образом может осуществляться поиск троек, ..., *n*-ок. Для *n*-ок операция ассоциирования записывается следующим образом:

$$T_1^o\ (\uparrow z_1)\ /\ T_2, \qquad (4.9)$$

где T_1^o — есть ЭФ, содержащий набор *н*-вершин $z_1 = (x_1, ..., x_n)$. Такая операция также сводится к поиску в T_2 кортежей, из соответствующих мест выбираются *n*-ки и присваиваются в качестве значений $(x_1, ..., x_n)$.

Будем допускать **групповой поиск**, выполняемый оператором ассоциирования. Такой поиск задается графами, содержащими *з*-сети. Рассмотрим в качестве примера граф рис. 4.2, который записывается в виде

$$x_2 \perp \langle \_\_,\ t,\ и.\ цн,\ x_1,\ x_2\rangle \circ [x_1 := \{a_i\}]. \qquad (4.10)$$

Граф соответствует *множеству цен изделий* $\{A_i\}$. Он задает следующие операции. Из T_2 выбираются все «сопоставимые» кортежи, т. е. те, в которых на первом месте может стоять любая вершина, на втором — t, на третьем — *и. цн* и на четвертом — одна из вершин, входящих в $\{a_i\}$. Из выбранных кортежей выделяются вершины, стоящие на пятом месте. Они и

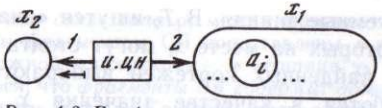


Рис. 4.2. Граф, сопоставленный *множеству цен изделий*

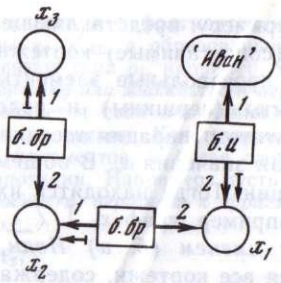


Рис. 4.3. Граф, сопоставленный *друзьям братьев людей по имени Иван*

присваиваются в качестве значения x_2. Последние соответствуют *множеству известных системе цен* (представленных в T_2) без указания, к какому конкретному изделию из $\{A_i\}$ они относятся. Будем записывать подобную «групповую» операцию в виде

$$\langle \_\_,\ t,\ и.\ цн,\ x_1,\ \uparrow x_2 \rangle \circ \{x_1 := \{a_i\}\} / T_2. \qquad (4.11)$$

Нетрудно распространить введенную операцию на случаи, когда в фрагменте (образце) имеются вершины с заданными согласованными значениями, когда ищутся значения наборов $н$-вершин и т. д. Подобная операция реализуется методом группового поиска, т. е. за один просмотр выбираются все требуемые кортежи из T_2.

Из операций ассоциирования путем их **суперпозиции** составляются более сложные операции. Рассмотрим в качестве примера граф рис. 4.3, где *б. др — быть другом*, *б. бр — быть братом*, *б. и — быть именем*. Граф сопоставляется *друзьям братьев людей с именем Иван*. Он задает операцию, являющуюся суперпозицией трех операторов. С помощью них последовательно находятся значения $н$-вершин: вначале x_1, затем по найденным значениям x_1 ищутся значения x_2, а по последним — значения x_3. Подобную операцию можно записать в следующем виде:

$$\langle \_\_,\ t,\ б.\ и,\ 'Иван',\ \uparrow x_1 \rangle / T_2,\ \langle \_\_,\ t,\ б.\ бр,\ \uparrow x_2,\ x_1 \rangle / T_2,$$
$$\langle \_\_,\ t,\ б.\ др,\ \uparrow x_3,\ x_2 \rangle / T_2, \qquad (4.12)$$

где T_2 — знания о родственных отношениях, а запятая между операторами служит для указания их последовательного выполнения. Словом, вначале ищутся все, кто имеет имя *Иван*. Затем по найденному множеству X_1 ищется множество их братьев X_2. После этого по множеству братьев находятся все их друзья X_3. Второй и третий операторы реализуют групповые методы поиска.

Операция свободного ассоциирования. Она необходима для поиска по отношениям, в которых роли (семантические падежи) некоторых объектов не указаны. Способ представления таких отношений рассматривался в конце § 1.4. Напомним, что для представления использовались фрагменты с непомеченными (непронумерованными) ребрами типа →. В записи таких фрагментов перед вершинами, соответствующими объектам с незафиксированной ролью, ставились знаки двоеточия. Считается, что места подобного сорта вершин в кортежах точно не зафиксированы — допускаются перестановки. Например, в фрагменте $\langle \_\_,\ t,\ r_1,\ :a_1,\ a_2,\ :a_3 \rangle$ $о$-вершина a_1 может быть переставлена местами с a_3. При этом для ряда отношений (которые однозначно определяют объекты со свободными ролями) двоеточия для простоты не ставились. Среди них следует отметить отношение типа *часть—целое*, которое представляется в виде $\langle d_1,\ t,\ \_\_,\ a_1,\ ...,\ a_k \rangle$, а также отношение *иметь множество значений*, что представляется в виде $з$-сети или же в расшифрованном виде — $\langle \_\_,\ t,\ :=,\ x_1,\ a_1,\ ...,\ a_k \rangle$. Все объекты $(A_1, ..., A_k)$ таких отношений играют одну и ту же роль. Допускаются всевозможные перестановки, характерные для элементов множеств.

Далее, важное место занимают отношения R_1, обладающие свойством симметричности — типа *быть братом*, *находиться слева-направо* и т. д. Бинарные отношения такого типа представляются в виде $\langle \_\_,\ t,\ r_1,\ d_1,\ d_2 \rangle$ (двоеточия не ставятся, считается, что они всегда могут быть однозначно расставлены, что осуществляется перед тем, как выполнять поиск).

Учитывать возможность перестановок объектов можно несколькими способами, например преобразованием представлений (см. § 2.1). Однако это не самый лучший способ, так как предполагаются достаточно сложные операции манипулирования структурами (см. § 4.3), а число перестановок может быть значительным. Другой способ, когда соответствующим графам сопоставляются комплексные операции, составленные из базисных. Например, считается, что граф $x_1^{\downarrow} \perp \langle \_\_,\ t,\ +,\ :'5',\ :'3',\ x_1^{\downarrow} \rangle$ задает операцию следующего вида:

$$\langle \_\_,\ t,\ +,\ '5',\ '3',\ \uparrow x_1^{\downarrow} \rangle / T_2 \cup \langle \_\_,\ t,\ +,\ '3',\ '5',\ \uparrow x_1^{\downarrow} \rangle / T_2, \qquad (4.13)$$

т. е. находятся два множества значений и объединяются. Фактически возможность перестановок, размножения (с учетом перестановок) учитывается на уровне ОЯ, где последовательность операций типа (4.13), задаваемых тем или иным фрагментом, может быть скомпонована в виде отдельного оператора.

И, наконец, еще один способ, когда возможность перестановок учитывается в процессе самого сопоставления кортежей, т. е. реализуется в рамках специальной базовой операции. Будем называть ее операцией с в о б о д н о г о а с с о ц и и р о в а н и я. Она задается графами вида $x_i \perp T_1\ (x_i)$, где в T_1 имеются ребра типа → без номеров (а в записи T_1 — двоеточия). Операции выполняются над другой сетью — T_2, в которой ищутся сопоставимые кортежи. В последних на месте x_i могут стоять достаточно произвольные элементы и допускаются перестановки вершин, помеченных двоеточиями. Например, граф $x_1 \perp \langle \_\_,\ t,\ б.\ бр,\ :a_1,\ :x_1 \rangle$, сопоставленный *брату субъекта* A_1, задает операцию следующего вида. В T_2 ищутся все кортежи, содержащие на первом месте произвольную вершину (на что указывает символ «__»), на втором — $о$-вершину t, на третьем — *б. бр*, а на четвертом или пятом — a_1. Из найденных кортежей выбираются вершины, стоящие соответственно на пятом или четвертом месте, и присваиваются в качестве значения x_1.

Другой граф $x_1 \perp \langle a_1,\ t,\ \_\_,\ :x_1 \rangle$, сопоставленный *частям объекта* A_1, задает операцию выбора из T_2 всех кортежей вида $\langle a_1,\ t,\ \_\_,\ ...,\ a_j,\ ... \rangle$, где a_1 может стоять на любом месте, начиная с четвертого. Далее присваиваются значения $[x_1 := \{a_j\}]$.

Двоеточиями могут быть отмечены и вершины из T_2. Тогда сопоставление будет сводиться к перекрестному отождествлению (идентификации) вершин с выбором из T_2 требуемых фрагментов.

Принципы поиска с использованием родовидовых и теоретико-множественных отношений. Выше рассматривался частный случай операции ассоциирования, предполагающий, что в T_2 представлены свойства и отношения индивидуальных объектов. Более сложные случаи возникают, если представлены свойства множеств, классов. Оказывается, что здесь также может быть использована операция ассоциирования, дополненная специальными средствами.

Пусть граф Γ_1 имеет вид $x_1 \perp \langle \_\_, t, x_1, a_1 \rangle$, т. е. сопоставляется *свойствам объекта* A_1. Пусть в T_2 имеются фрагменты $\langle \_\_, t, \in, a_1, m_1 \rangle$, $\langle \_\_, t, r_1, m_1 \rangle$, т. е. представлено, что $A_1 \in M_1$ *и что объекты класса* M_1 *обладают свойством* R_1. Ясно, что по кортежу $\langle \_\_, t, x_1, a_1 \rangle$ должен быть выбран (из T_2) кортеж $\langle \_\_, t, r_1, m_1 \rangle$, т. е. свойства классов должны р а с п р о с т р а н я т ь с я на их конкретных представителей. Только тогда у объекта A_1 может быть найдено свойство R_1.

Итак, использование знаний, представляющих принадлежность к классам, сводится к описанному распространению. Оно может осуществляться двумя способами. Первый из них — и д е н т и ф и к а ц и е й вершин из T_1 и T_2, управляемой с помощью специальных правил. Второй основан на размножении вершин из T_1.

Остановимся вначале на первом способе. Будем считать, что если одна вершина (a_i) идентифицируется с другой (a_j), то при сопоставлении a_j рассматривается, как будто это есть a_i. Так, в предыдущем примере отмеченное распространение свойств может быть обеспечено путем идентификации вершины a_1 с m_1. Тогда фрагменты $\langle \_\_, t, x_1, a_1 \rangle$ и $\langle \_\_, t, r_1, m_1 \rangle$ будут сопоставимыми. При использовании операции ассоциирования будет получено $[x_1 : = r_1]$, что соответствует нахождению требуемых свойств.

В общем случае в T_2 могут быть представлены групповые соотношения (что осуществляется с помощью $з$-сетей см. § 2.2) и родовидовые связи (см. § 1.4). Использование подобной информации также обеспечивается путем идентификации. Вершина a_1 из Γ_1 должна идентифицироваться с вершиной $x_i := \{..., a_1, ...\}$ из T_2. Если имеет место включение множеств $M_1 \subset M_2$, что представляется $\langle \_\_, t, \subset, m_1, m_2 \rangle$, то вершина m_1 из Γ_1 должна идентифицироваться с вершиной m_2 из T_2. В результате соответствующие фрагменты делаются сопоставимыми. Обеспечивается распространение свойств множеств X_i на индивидуальные объекты A_1, а свойств классов M_2 на подклассы M_1.

Для идентификации вершин используются специальные наборы п р а в и л, описанные в работах [24, 25]. Отметим, что такие правила определяются прежде всего выбранными способами представления обобщенной информации. Допускаются сменные наборы правил. Это необходимо в случае, когда в системе предусматривается расширение средств представления, повышение изобразительных возможностей. Использование новых средств сводится к пополнению набора правил.

Рассмотрим второй способ использования обобщенной информации, основанный на **размножении вершин.** Вернемся к предыдущему примеру. Пусть имеется граф $x_1 \perp \langle \_\_, t, x_1, a_1 \rangle$. Нетрудно видеть, что если добавить к a_1 вершину m_1, сформировав граф $x_1 \perp \langle \_\_, t, x_1, x_2 \rangle \circ [x_2 : = \{a_1, m_1\}]$, то можно обычным способом найти требуемую информацию $[x_1 : = r_1]$. Это может быть сделано с помощью операции группового ассоциирования. Здесь как бы имеет место «размножение» вершин, т. е.

из a_1 получилось $\{a_1, m_1\}$. Аналогичным размножением можно обеспечить распространение свойств классов на подклассы и т. д.— все, что решалось средствами идентификации. В дальнейшем в качестве «инструмента» размножения будем использовать обязательные графы, которые будут рассмотрены в § 5.2. Для этой цели они используют знания о родовидовых связях, отношениях.

Непосредственным аналогом операции ассоциирования является оператор реляционной алгебры, называемый делением (*DIVIDE*).

Операция проверки непустоты. Она необходима для работы с вопросительными формами типа *Имеются ли объекты* X_1, *которые...*?, например, *Имеются ли друзья у брата Ивана*?. Предполагается, что после нахождения соответствующих множеств требуется проверка их пустоты или непустоты. В зависимости от этого формируется результат. Подобные формы представляются в виде графа

$$[x_1 \perp T_1(x_1)] \circ [?x_1 : = \lambda], \tag{4.14}$$

где сама $з$-сеть является сокращенной записью конструкции (см. § 2.2):

$$\langle \_\_, p_1, : = , x_1, \lambda \rangle \circ [p_1 : = ?]. \tag{4.15}$$

Первая часть из (4.14) задает операции нахождения значений x_1 (пусть $[x_1 : = \{a_i\}]$), а вторая — проверки $\{a_i\} \neq \varnothing$. Если проверка удовлетворяется, то формируется $[p_1 : = t]$ *(да, имеется)*, что приведет к $[x_1 : = \lambda]$, а если не удовлетворяется, то $[p_1 : = f]$ *(нет, не имеется)*, что может быть записано в другом виде — $[x_1 : = \tilde{\lambda}]$. Итак, в результате выполнения операции $н$-вершине p_1 присваиваются значения, уточняется $з$-сеть $[?x_1 : = \lambda]$.

Операция сравнения. Согласие или несогласие. Рассмотрим утвердительную форму типа *Имеются объекты* X_1, *которые обладают свойствами* $\mathcal{T}_1$ *(или связаны отношениями* $\mathcal{T}_1$ *с другими объектами)*. Напомним, что такая форма представляется в виде графа, у которого стрелки порядка $\mapsto$ сходятся к вершинам с символом λ. Граф записывается

$$[x_1 \perp T_1(x_1)] \circ [x_1 : = \lambda]. \tag{4.16}$$

Как и в предыдущем случае, предполагается, что вначале нужно найти объекты X_1 и проверить их непустоту. В то же время уже подразумевается результат — непустота должна иметь место, на что указывает $[x_1 : = \lambda]$. Задается дополнительная операция сравнения того, что будет получено (т. е. либо $[x_1 : = \lambda]$, либо $[x_1 : = \tilde{\lambda}]$), с тем что утверждается (λ). Результатом сравнения является бинарная величина, указывающая на совпадение ($\lambda = \lambda$) или несовпадение ($\lambda \neq \tilde{\lambda}$). Роль такой величины может играть $р$-вершина (p_i) сети T_1 (см. § 1.4), которой присваивается $[p_i : = t]$ *(да, правильно)* или $[p_i : = ?]$ *(не знаю, правильно ли)*. Аналогичную роль может играть $с$-вершина сети T_1, к которой приформировываются соответствующие фрагменты. Здесь возможно множество приемлемых вариантов.

Важно отметить, что результат выполнения операции каким-то образом изменяет (дополняет) саму сеть (4.16). За этими изменениями следят специальные демоны (метаправила), инициирующие последующие действия. Такие же демоны следят за режимом работы, учитывают вид поступившего вопроса или сообщения и т. д. Их функции достаточно разнообразны. Например, если $\mathcal{T}_1$ — есть некоторое высказывание сомни-

тельного характера и система уверена в полноте своих знаний T_2, то совпадение будет означать согласие с этой информацией. Никаких дополнительных действий не требуется. Несовпадение будет означать несогласие, что может приводить к встречным вопросам, в связи с чем должны включаться в работу свои демоны. Если $\mathscr{T}_1$ — высказывание, поступившее от достоверных источников, и система не уверена в полноте своих знаний, то несовпадение может означать, что системе что-либо неизвестно, что в $\mathscr{T}_1$ содержится новая для нее информация. Срабатывают другие демоны. В высказываниях может акцентироваться внимание на то, что является новым. Такое высказывание может иметь вид гипотезы, закономерности, которую нужно проверить или использовать и т. д. Все эти случаи должны быть учтены и приводить к дифференцированным реакциям.

Форма негативного характера *Не имеется* (*не существует*) *объектов* X_1, *которые* ... $\mathscr{T}_1$ представляется в виде графа

$$[x_1 \perp T_1(x_1)] \circ [x_1 : = \tilde{\lambda}], \tag{4.17}$$

в изображении которого стрелки порядка сходятся к вершине x_1 с символом $\tilde{\lambda}$. Последний фрагмент указывает, что должна иметь место пустота найденных значений x_1. Здесь также задается дополнительная операция сравнения того, что будет получено ($\tilde{\lambda}$, λ), уже с негативной компонентой ($\tilde{\lambda}$), т. е. совпадение будет иметь место в случае, если найденное множество оказалось пустым ($\tilde{\lambda} = \tilde{\lambda}$). Результат операции представляется, как было указано ранее для графа (4.10). Аналогичным образом инициируются и последующие действия.

Теоретико-множественные операции проверки (включения, непустого пересечения). Они необходимы для работы с утвердительными формами типа *Объектом, обладающим свойствами* $\mathscr{T}_1$, *является* A_i или *Объектами со свойствами (отношениями)* $\mathscr{T}_1$ *являются* $\{A_\xi\}$. Напомним, что первая форма представляется в виде графа

$$[x_1 \perp T_1(x_1)] \circ [x_1 : = a_i], \tag{4.18}$$

в изображении которого стрелки порядка $\mapsto$ сходятся к o-вершине a_i. Последний фрагмент задает проверку вхождения ($\in$, $\subset$) o-вершины a_i в множество найденных значений x_1. Результат имеет тот же характер, что и у операции сравнения, т. е. представляется совпадение или несовпадение, что изменяет сеть T_1.

Вторая форма представляется в виде графа

$$[x_1 \perp T_1 (x_1)] \circ [x_1 : = \{a_\xi\}], \tag{4.19}$$

в изображении которого стрелки порядка сходятся к вершине x_1. Последний фрагмент задает операцию проверки вхождения $\{a_\xi\}$ во множество найденных значений x_1. Пусть по T_1 было найдено $x_1 : = \{a_j\}$. Тогда в случае $\{a_\xi\} \subset \{a_j\}$ формируется величина, указывающая на совпадение, а в случае $\{a_\xi\} \not\subset \{a_j\}$ — на несовпадение.

Для представления форм типа *Объектами со свойствами* $\mathscr{T}_1$ *являются* $\{A_\xi\}$, *но не* A_k в § 1.3 были введены инверсные вершины $\tilde{a}_k$. Если для вершин $\{a_\xi\}$ проверяется вхождение ($\subset$), то для вершин $\tilde{a}_k$ проверяется факт их отсутствия в множестве найденных значений x_1, т. е. $a_k \notin \{a_j\}$.

При выполнении обеих проверок вырабатывается бинарная величина, указывающая на совпадение.

Достаточно интересными являются формы типа *Объекты со свойствами* $\mathscr{T}_1$ *— есть некоторые из* $\{A_\xi\}$. Для их представления в работе [14] были использованы s-сети, задающие так называемые приближенные значения — $[x: \approx \{a_\xi\}]$, где знак $:\approx$ указывает на приближенность. Тогда сама форма представляется в виде графа

$$[x_1 \perp T_1(x_1)] \circ [x_1 : \approx \{a_\xi\}], \tag{4.20}$$

который задает операцию проверки непустого пересечения множества $\{a_\xi\}$ и множества найденных значений $x_1: = \{a_j\}$. Если $\{a_\xi\} \cap \{a_j\} \neq \varnothing$, то вырабатывается величина, указывающая на совпадение, иначе — на несовпадение. Введенные операции будут играть важную роль в процессе обработки некоторых видов обобщенной информации (см. гл. 7).

§ 4.3. СРЕДСТВА МАНИПУЛИРОВАНИЯ НАД МНОЖЕСТВАМИ И НАБОРАМИ

Во многих сообщениях, высказываниях, вопросах речь идет о множествах объектов. Понимание сообщения, выделение соответствующих объектов, поиск ответа на запросы предполагает соответствующие действия манипулирования над множествами. Речь может идти не только об отдельных объектах, но и об их наборах (парах, тройках, ..., n-ках), из которых составляются другие наборы, выделяются элементы и т. д. В простейшем случае такие действия сводятся к операциям и понятиям, используемым в языке теории множеств. Имеются в виду операции $\cap$, $\cup$, $/$, понятия произведения множеств, их проекции на координатные оси и т. д.

Однако во многих случаях использование только категорий языка теории множеств, выражение в нем всевозможных способов манипулирования над наборами оказывается не совсем удобным. Некоторые такие способы выражаются достаточно сложным образом. Рациональнее использование специальных наборов операторов. Видимо, именно по этой причине был создан язык **реляционной алгебры** и создаются все новые языки, находящие применение в банках и базах данных. Напомним, что операторы языка реляционной алгебры работают над таблицами. В общем-то они могут служить для реализации некоторых видов естественной обработки. В связи с этим требуется их распространение на случай работы с конструкциями введенного ранее вида — сетями. В начале параграфа будет рассматриваться такое распространение. Будут введены операторы над наборами, для многих из которых имеются аналоги в языке реляционной алгебры.

Однако языка реляционной алгебры в плане реализации естественных видов обработки явно недостаточно. При работе с семантическими сетями (в том числе s-сетями) требуются действия, выходящие за рамки этого языка. Например, когда речь идет о единичном объекте, то может быть не ясным, что это за объект. Возникают альтернативы, которые порождают другие альтернативы, с которыми связаны соответствующие действия. Здесь требуются специальные операторы перебора с формированием альтернатив, отсевом неприемлемых вариантов и др. Ниже будут рассматриваться операторы, которые могут непосредственно быть связаны с рядом типовых

форм ЕЯ, т. е. служить для выражения их операционной семантики. Напомним, что в соответствии с ранее принятой методикой обработки такая связь осуществляется путем предварительного формирования соответствующих семантических графов, которые, как мы говорили, задают операции обработки.

Операция соединения. Она необходима для составления из заданных наборов общего набора. Например, из наборов *изделие — его цена* и *изделие — место изготовления — поставщик* составляется общий набор *изделие — его цена — место изготовления — поставщик*. В реляционной алгебре имеется ее непосредственный аналог с таким же названием — соединение (JOIN). Только в нашем случае операция выполняется не над таблицами. Ее аргументами являются множества найденных или заданных значений *н*-вершин, а в общем случае наборов *н*-вершин, соответствующих парам, тройкам, ..., *п*-кам объектов. Результатом является сформированное множество значений одной *н*-вершины или набора *н*-вершин.

Частным случаем операции соединения является операция теоретико-множественного п е р е с е ч е н и я ($\cap$), которая задается конструкциями вида

$$x_1 \perp \Gamma_i(x_1) \circ \Gamma_j(x_1), \tag{4.21}$$

где Γ_i и Γ_j — графы, у которых стрелки порядка сходятся к одной и той же вершине x_1. Сами графы задают собственные операции — нахождения двух множеств значений *н*-вершины x_1. Далее ищутся общие элементы этих множеств, которые и присваиваются в качестве окончательного (результирующего) значения x_1. Если по графам Γ_i и Γ_j были найдены значения $[x_1 := \{a_i\}]$ и $[x_1 := \{a_j\}]$, то они могут быть подставлены в (4.21) на места Γ_i и Γ_j. Получившаяся конструкция задает операцию формирования результирующего значения $[x_1 := \{a_i\} \cap \{a_j\}]$.

Отметим, что если областями графов Γ_i и Γ_j являются элементарные фрагменты, то конструкция (4.21) вырождается в граф вида

$$x_1 \perp T_i^3(x_1) \circ T_j^3(x_1), \tag{4.22}$$

который задает операцию

$$T_i^3(\uparrow x_1)/T_2 \cap T_j^3(\uparrow x_1)/T_2. \tag{4.23}$$

Результат присваивается в качестве значения x_1. Пример такого графа изображен на рис. 4.4. Он задает операции, выполнение которых приведет к тому, что вначале будут найдены *все, кто являются лысыми,* затем *все короли Франции,* а затем (путем теоретико-множественного пересечения) — *все те, кто являются лысыми и королями Франции.*

Для указания необходимости дополнительной проверки непустоты результата пересечения в § 3.4 была введена конструкция вида

$$[x_1 \perp \Gamma_i(x_1) \circ \Gamma_j(x_2)], \tag{4.24}$$

к которой, как указывалось, может быть добавлена *з*-сеть $[x_1 := \lambda]$. Такой граф изображается, как показано на рис. 4.5. В частности, если внутри *н*-вершины x_1 графа рис. 4.4 поставить символ λ, то он будет задавать указанные проверки. Соответствующие операции и вызываемые ими действия были рассмотрены в предыдущем параграфе.

Нетрудно видеть, что рассмотренные операции согласуются с простейши-

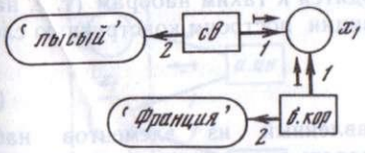


Рис. 4.4

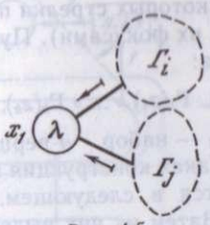


Рис. 4.5

Рис. 4.4. Граф, сопоставленный *лысым королям Франции*

Рис. 4.5. Граф, задающий операции поиска двух множеств с выявлением их общих элементов и проверкой непустоты

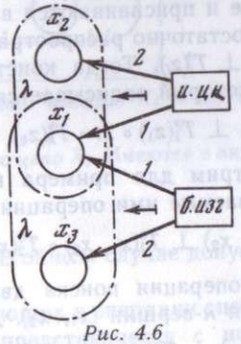


Рис. 4.6

Рис. 4.6. Составной граф, сопоставленный тройкам *изделие* X_1 *— его цена* X_2 *— место изготовления* X_3

ми теоретико-множественными аналогами. Однако если элементами (данными) являются наборы, то требуются более сложные действия, реализуемые операцией соединения. Рассмотрим конструкции, которые задают эти действия.

Операция соединения общего вида задается графами, у которых стрелки порядка сходятся к набору *н*-вершин. Напомним, что такой набор изображается с помощью штрихпунктирного контура. Начнем с примера, который приведен на рис. 4.6. *О*-вершина *и. цн* соответствует отношению *иметь цену,* а *б. изг — быть изготовленным.* Такой граф записывается в виде

$$(x_1, x_2, x_3) \perp \langle \_, t, \text{и. цн}, x_1\, x_2 \rangle \circ \langle \_, t, \text{б. изг}, x_1, x_3 \rangle. \tag{4.25}$$

Он представляет следующие тройки: изделие (X_1) — его цена (X_2) — место изготовления (X_3). Граф задает операцию поиска двух множеств пар, представляющих *изделие — его цену* и *изделие — место изготовления.* Из этих множеств выбираются пары (из каждого по одной), в которых на первом месте находится одна и та же *о*-вершина. Такие пары представляют *цену* и *место изготовления одного и того же изделия.* Из них составляются тройки.

Аналогом такого выбора и составления в реляционной алгебре является соединение (JOIN) двух таблиц — отошений по указанному атрибуту. Применительно к предыдущему примеру имеется в виду соединение таблиц, в которых записаны пары *изделие—цена* и *изделие — место изготовления,* по атрибуту *изделие.* Обозначим такие таблицы через X_1, X_2 и X_1, X_3 (см. § 2.3). Будем считать X_1, X_2 и X_3 именами их столбцов (атрибутами), т. е. первая таблица имеет атрибуты X_1 и X_2, а вторая — X_1 и X_3. Тогда операцию можно записать следующим образом: JOIN (X_1, X_2) *и* (X_1, X_3) *по* X_1.

Перейдем к рассмотрению графов общего вида, задающих операцию соединения. Пусть z_i для $i = \overline{1, k}$ есть наборы *н*-вершин, а $\Gamma_i(z_i)$ —

графы, у которых стрелки порядка сходятся к таким наборам (т. е. наборы являются их фокусами). Путем композиции построим конструкцию следующего вида:

$$z_{k+1} \perp \Gamma_1(z_1) \circ \ldots \circ \Gamma_k(z_k), \tag{4.26}$$

где z_{k+1} — набор n-вершин, составленный из элементов наборов $z_1, \ldots, z_k$. Такая конструкция и будет задавать операцию соединения, которая заключается в следующем. Вначале ищутся значения наборов z_i для $i = \overline{1, k}$. Затем из них выделяются цепочки, содержащие общие элементы, которые и присваиваются в качестве значения z_{k+1}.

В достаточно распространенном случае графы $\Gamma_i(z_i)$ для $i = \overline{1, k}$ имеют вид $z_i \perp T_i^3(z_i)$. Тогда конструкция (4.26) вырождается в более простой граф, который записывается следующим образом:

$$z_{k+1} \perp T_1^3(z_1) \circ \ldots \circ T_k^3(z_k). \tag{4.27}$$

Рассмотрим для примера несколько частных видов конструкции (4.27) и задаваемые ими операции. Граф вида

$$(x_1, x_2) \perp T_1^3(x_1, x_2) \circ T_2^3(x_1, x_2) \tag{4.28}$$

задает операция поиска двух множеств пар, образующих согласование значения n-вершин (x_1, x_2). Далее выделяются общие пары. Другой граф

$$(x_1, x_2) \perp T_1^3(x_1, x_2) \circ T_2^3(x_2) \tag{4.29}$$

задает операции поиска двух множеств пар, образующих согласование множества вершин. Далее выделяются только такие пары, у которых на втором месте стоит элемент последнего множества.

Операция сцепления. Она необходима для ограничения множеств пар, ..., n-ок объектов в соответствии с условием равенства их компонент. Например, с помощью такой операции в множествах пар *изделие — поставщик* и *поставщик — его статус* могут быть выделены такие подмножества, в которых речь идет об *одних и тех же поставщиках*. Аргументы у операции сцепления такие же, как и у предыдущей операции соединения. Аргументами являются значения нескольких наборов n-вершин. Но результат другой. Если предыдущая операция вызывала объединение наборов в один, то операция сцепления так и оставляет эти наборы. Изменяются лишь значения наборов n-вершин.

Операция сцепления задается графами, у которых стрелки порядка сходятся к различным наборам n-вершин. Пример такого графа изображен на рис. 4.7, где имеется два штрихпунктирных контура. Такой граф будем записывать в виде

$$[(x_1, x_2) \perp \langle \_\_, t, \text{и. цн}, x_1, x_2 \rangle] \circ [(x_1, x_3) \perp \langle \_\_, t, \text{б. изг}, x_1, x_3 \rangle]. \tag{4.30}$$

Он, так же как и граф рис. 4.6, задает операции выбора двух множеств пар. Однако в отличие от предыдущего случая тройки не формируются. Так и остается два множества пар: *изделие* X_1 *— его цена* X_2 и *изделие* X_1 *— место изготовления* X_3, но представляющие характеристики одних и тех же изделий.

Сцепление — это операция над множеством в n-мерном пространстве, заключающаяся в выделении из них таких подмножеств, которые имеют одну и ту же проекцию на указанные координаты. В рассмотренном

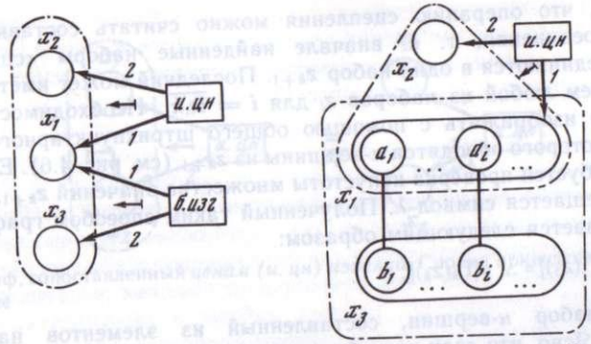


Рис. 4.7. Граф, сопоставленный двум множествам пар: *изделие* X_1 *— его цена* X_2 *и это же изделие — место его изготовления* X_3

Рис. 4.8. Граф, сопоставленный парам *изделие* X_1 *— его цена* X_2. Имеются в виду лишь *изделия* (A_i), *присутствующие в множестве* $\{(A_i, B_i)\}$

ранее примере такой координатой является X_1. В общем случае допускается несколько подобных координат.

В частном случае одно из множеств, участвующих в операции сцепления, может быть уже найдено или задано, что представляется с помощью 3-сети. Например, граф

$$[(x_1, x_2) \perp \langle \_\_, t, \text{и. цн}, x_1, x_2 \rangle] \circ [(x_1, x_3) := \{(a_i, b_i)\}], \tag{4.31}$$

изображенный на рис. 4.8, задает операцию поиска пар *изделие — его цена* для изделий из множества $\{A_i\}$, а также операцию изъятия из множества $\{(A_i, B_i)\}$ таких пар, у которых для A_i не задана цена. В результате такого изъятия в двух множествах останутся только такие пары, которые имеют одну и ту же проекцию на первую координату.

Рассмотренный выше граф может быть обобщен до следующей композиции элементарных графов:

$$[z_1 \perp T_1^3(z_1)] \circ \ldots \circ [z_k \perp T_k^3(z_k)], \tag{4.32}$$

которая задает операции формирования значений наборов $z_1, \ldots, z_k$ n-вершин в соответствии с требованием обеспечения одинаковых проекций: если любые два набора z_i и z_j $(i, j = \overline{1, k})$ имеют общие n-вершины, то проекции множеств их значений на соответствующие координаты должны быть одинаковыми.

В общем случае операция сцепления задается композицией следующего вида:

$$[\Gamma_1(z_1)] \circ \ldots \circ [\Gamma_k(z_k)], \tag{4.33}$$

где $\Gamma_i(z_i)$ для $i = \overline{1, k}$ — графы, фокусами которых являются наборы n-вершин z_i. В результате выполнения операции формируются значения этих наборов — в соответствии с требованием обеспечения одинаковых проекций: если любые два набора z_i и z_j (где $i, j = \overline{1, k}$) имеют общие n-вершины, то проекции множеств их значений на соответствующие координаты должны быть одинаковыми.

Заметим, что операцию сцепления можно считать составной частью операции соединения, т. е. вначале найденные наборы «сцепляются», а затем объединяются в один набор z_{k+1}. Последний может иметь большую местность, чем любой из наборов z_i для $i = \overline{1, k}$. Необходимость объединения будем изображать с помощью общего штрихпунктирного контура, с помощью которого обводятся $н$-вершины из z_{k+1} (см. рис. 4.6). Если дополнительно требуется проверка непустоты множества значений z_{k+1}, то внутрь контура помещается символ λ. Полученный таким способом граф в общем виде записывается следующим образом:

$$[z_{k+1} \perp [\Gamma_1(z_1)] \circ \ldots \circ (\Gamma_k(z_k)]], \tag{4.34}$$

где z_{k+1} — набор $н$-вершин, составленный из элементов наборов z_i, для $i = \overline{1, k}$. Ясно, что если хотя бы одна вершина из каждого z_i присутствует в z_{k+1}, то непустота значений z_{k+1} будет иметь место только в случае непустоты значений всех z_i. Если выполнена проверка непустоты значений z_{k+1}, то аналогичные проверки заведомо будут выполнены для всех z_i. Поэтому в (4.34) внутренние квадратные скобки, указывающие на последние проверки, особой роли не играют и могут быть опущены. В результате будет получена конструкция типа (4.26), задающая операцию соединения.

Операция проекции. Она необходима для выделения отдельных компонент из известных пар, ..., n-ок объектов. Аргументом этой операции является набор (z_1) $н$-вершин с множеством заданных значений, а результатом — проекция этого множества на определенные координаты. Результат проекции присваивается в качестве значений поднабора $н$-вершин (z_2), входящих в z_1. Подобного сорта операция задается конструкцией вида

$$z_2 \perp [z_1 : = \{w_i\}], \tag{4.35}$$

где все $н$-вершины из z_2 входят в набор z_1. С помощью указанной операции обеспечивается формирование значений отдельных $н$-вершин x_1, где $i = \overline{1, k}$, по заданным значениям набора $z_1 = (x_1, \ldots, x_k)$. В общем случае обеспечивается формирование значений поднаборов z_2.

Отметим, что аналогичные операции (к которым добавляется операция ассоциирования) задаются конструкцией вида

$$z_2 \perp [z_1 \perp T_1(z_1)]. \tag{4.36}$$

Если при выполнении операции ассоциирования результатом является $T_1(\uparrow z_1)/T_2 = \{w_i\}$, то получается конструкция (4.35).

Рассмотрим в качестве примера граф рис. 4.9, который записывается следующим образом:

$$x_2 \perp (x_1, x_2) \perp \langle \_\_, t, \text{и. цн}, x_1, x_2 \rangle. \tag{4.37}$$

Граф задает поиск множества пар, представляющих *изделие — его цена*. Из этого множества выделяются элементы, представляющие только *цены*. Результатом операции будет множество $о$-вершин, соответствующих *ценам всевозможных изделий* (естественно, если такие цены заданы и представлены в системных знаниях T_2).

Будем допускать случаи, когда операция проекции не задается в явном

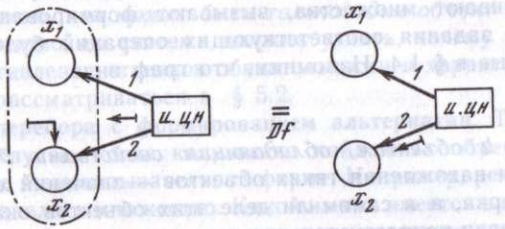


Рис. 4.9. Граф, сопоставленный *ценам (и. цн) изделий*. Справа приведено более простое изображение

виде (специальными фрагментами, о которых говорилось ранее), а предполагается, т. е. задается в рамках других операций конструкциями определенного типа. Справа на рис. 4.9 изображен граф, задающий операции поиска цен. При этом имеются в виду цены изделий, т. е. предполагается нахождение множества пар *изделие — цена* с выделением *цен*. Такой граф записывается следующим образом:

$$x_2 \perp \langle \_\_, t, \text{и. цн}, x_1, x_2 \rangle.$$

Множества таких графов могут быть обобщены до конструкции вида

$$z_2 \perp T_1^3(z_1), \tag{4.38}$$

где z_2 — поднабор $н$-вершин набора z_1. Нетрудно убедиться, что имеет место следующее правило эквивалентного преобразования

$$z_2 \perp [z_1 \perp T_1^3(z_1)] \equiv z_2 \perp T_1^3(z_1), \tag{4.39}$$

где с помощью левой и правой частей задаются одни и те же операции.

С помощью введенных операций сцепления и проекции могут быть выражены достаточно сложные операции над наборами с выделением отдельных элементов. Рассмотрим пример одной из таких операций, которая необходима для работы с корреляционными продукциями (см. § 2.2). Напомним, что в процессе применения таких продукций требовалось, чтобы по найденному значению $н$-вершины x_1 и заданным согласованным значениям пары (x_1, x_2) обеспечивался выбор значения x_2. Например, если известны пары *поставщик—товар*, то по любому *поставщику* всегда можно было бы указать на поставленный *товар*. Подобного сорта действия задаются конструкциями следующего вида:

$$x_2^1 \perp [(x_1^1, x_2^1) : = \{(,), \ldots, (,)\}] \circ [x_1^1 \perp T_1(x_1^1)]. \tag{4.40}$$

Вначале по свойствам $\mathcal{T}_1$ находится объект X_1. Затем, пользуясь операцией сцепления, выбирается одна из пар, входящих в множество значений (x_1^1, x_2^1). Из этой пары, выполняя операции проекции, выделяется второй элемент, который и присваивается в качестве значения x_2^1.

Непосредственным аналогом операции проекции в реляционной алгебре является одноименная операция *PROJECT*. В частности, операция, задаваемая графом $x_2 \perp [(x_1, x_2) : = \{w_i\}]$, записывается следующим образом:

PROJECT (X_1, X_2) *на* X_2,

где (X_1, X_2) — имя таблицы с атрибутами X_1 и X_2. В этой таблице записаны данные $\{W_i\}$.

Операция проверки количества. Количества в ЕЯ играют существенную роль — ограничивают множества, вызывают формирование альтернатив, переборы. Для задания соответствующих операций будем использовать графы, введенные в § 1.4. Напомним, что граф вида

$$x_1^k \perp T_1(x_1^k) \tag{4.41}$$

сопоставляется *k объектам, обладающим свойствами* $\mathcal{T}_1$. Такой граф задает операции нахождения таких объектов — значений x_1^k. Далее осуществляется проверка, а в самом ли деле этих объектов оказалось k штук. Результат проверки используется для организации последующих действий. Их характер может быть достаточно разнообразным и зависеть от вида представленной информации, режима работы и ряда других факторов, что может быть учтено с помощью сменного набора метаправил (см. § 4.1). Рассмотрим некоторые такие действия в простейшем случае, когда $k = 1$. Граф вида

$$x_1^1 \perp T_1(x_1^1) \tag{4.42}$$

сопоставляется *обекту, обладающему свойствами* $\mathcal{T}_1$. Граф задает операции нахождения множества значений $н$-вершины x_1^1 с проверкой, равно ли число элементов в этом множестве единице. Если проверка удовлетворяется, то никаких дополнительных действий не требуется. Например, если фрагмент (4.42) входит в граф более сложного вида, то может быть продолжено выполнение операций, задаваемых другими фрагментами. Если же число элементов в множестве оказалось больше одного, то требуются дополнительные операции. Какими они могут быть?

Пусть имеется в виду конкретный объект X_1 и системе разрешается переспрашивать. Тогда может быть организована внешняя активность для уточнения, какой объект имеется в виду. В ряде случаев такое уточнение может осуществляться за счет внутренней активности, т. е. на базе использования специальных знаний, в которых представлено, *что может быть*, а *чего нет* (см. § 5.2).

Если система не имеет возможности переспрашивания, то должен быть организован перебор вариантов с формированием альтернатив (соответствующий оператор будет рассмотрен чуть ниже).

Например, пусть во входном сообщении упоминается *некий друг брата Ивана*, что представляется, как показано на рис. 3.3. Пусть известно, о каком Иване идет речь, а братьев, как выяснилось, у него множество. Тогда естественно узнать, какой брат имеется в виду, т. е. система должна перехватывать инициативу, формулируя вопрос с требованием уточнения. Если уточнение по каким-либо причинам невозможно (например, не у кого спросить), то нужно перебирать этих братьев и для каждого из них находить друзей. Возникают альтернативы. Более того, если у каждого брата множество друзей, а имеется в виду один из них, то и такие множества будут иметь альтернативный характер. Возникает два уровня альтернатив. Будет сформирована конструкция, представляющая, что *если имеется в виду брат такой-то* (A_1), *то указанным другом является или* B_{11}, *или* B_{12}, ..., *если имеется в виду другой брат* (A_2), *то указанным другом является или* B_{21}, *или* B_{22}, ... Для представления таких конструкций могут служить $з$-сети (см. ниже).

Наконец, если граф вида (4.42) представляет оттенки обязательности,

то требуются несколько другого сорта действия. Например, пусть с помощью графа представлено, что *может быть только один объект* X_1. Если же их оказалось множество, то система должна узнать, почему так получилось. Средства представления оттенков обязательности и характер последующих реакций будут рассматриваться в § 5.2.

Операторы перебора с формированием альтернатив. Такие операторы необходимы в случае, когда количество найденных объектов превышает число, указанное в высказывательной форме. Например, речь идет об одном объекте, а их оказалось множество. Какой из них имеется в виду? Если уточнение по каким-либо причинам невозможно, то возникают альтернативы. При этом возможно множество уровней таких альтернатив, что иллюстрировалось на предыдущем примере.

Оператор перебора с формированием альтернатив в простом случае задается конструкциями (графами), представляющими высказывание о единичном объекте X_1. При этом последний может принимать участие в процессе нахождения или уточнения других объектов X_2. Подобная конструкция может быть записана следующим образом:

$$\Gamma_1(x_1^1) \circ \Gamma_2(x_1^1, x_2), \tag{4.43}$$

где x_1^1 является фокусом графа Γ_1, а x_2 — фокусом графа Γ_2. Первый граф задает операции нахождения множеств значений x_1, например, *множества братьев Ивана*. Пусть оказалось $x_1^1 := \{a_i\}$, где $i = \overline{1, k}$. Тогда берется каждый элемент a_i этого множества и находится множество значений x_2, для чего выполняются операции, задаваемые $\Gamma_2(x_1^1, x_2) \circ [x_1^1 := a_i]$. Пусть в результате выполнения оказалось $x_2 := \{b_{ij}\}$, т. е. *у каждого брата несколько друзей*. Тогда и формируется альтернативное множество. При этом будем различать два варианта формирования. Первый — когда это множество никак не связывается с альтернативами объекта X_1. Такое множество имеет следующий вид:

$$x_2 := \{\{b_{1j}\}, \{b_{2j}\}, \ldots, \{b_{kj}\}\}^1, \tag{4.44}$$

т. е. представляется, что *объектами* X_2 *является какое-либо одно из множеств среди* $\{B_{1j}\}$, $\{B_{2j}\}$, ... Последние множества никак не связываются с объектами $\{A_i\}$. Например, указывается, что *братьями, которые имеются в виду, являются или те, или другие, или третьи*. Отметим, что аналогом конструкций (4.44) является традиционное понятие с е м е й с т в а множеств, распространенное на случай наличия альтернатив.

В общем случае в графе (4.43) может иметься несколько $н$-вершин вида x^1, т. е. соответствующих единичным объектам. Тогда будет сформировано множество, представляющее несколько уровней альтернатив. Причем, если в графе (4.43) отсутствуют контуры (замкнутые пути по ребрам через $н$-вершины и вершины связи), то число таких уровней будет равно числу $н$-вершин вида x^1. Если все вершины графа имеют только такой вид, то возникает любопытная возможность использования **групповых методов для поиска альтернатив.** Остановимся на такой возможности применительно к конструкции (4.43).

Пусть в графе (4.43) $н$-вершина x_2 имеет вид x_2^1, т. е. также соответствует единичному объекту, а других $н$-вершин в этом графе нет. Тогда сначала, пользуясь введенными ранее операциями ассоциирования, соединения и др., находится множество значений x_2. Пусть оказалось $x_2 := \{b_{ij}\}$. Тогда просто

формируется $x_2^1 := \{b_{ij}\}^1$. Дело в том, что если имеют место альтернативы альтернатив, то все это может быть как бы вынесено на одну плоскость, один уровень, т. е. быть представлено как единое множество альтернатив. Но если в графе имеются контуры, то такой способ нахождения неприемлем. Возможно появление лишних элементов (см. § 4.4).

Более подробно синтаксис подобного сорта **многоуровневых множеств**, эквивалентные преобразования, способы работы с ними и примеры использования рассмотрены в работе [24]. Там же в рамках синтаксиса ОЯ введены операторы перебора с формированием, объединением и пересечением альтернатив. Первый из них может быть записан следующим образом:

$$\Pi(x_1, \{x_2\}^1)[Q_2],$$

где Q_2 — операции, задаваемые графом Γ_2; П — указывает на необходимость перебора значений x_1, а $\{x_2\}^1$ — формирования альтернатив на базе найденных значений x_2. В упомянутой работе рассмотрены метаправила, позволяющие по конструкциям вида (4.43) строить соответствующие операторы ОЯ.

Остановимся на другом варианте формирования — когда найденные множества связываются с порождающими их элементами. Например, указывается вначале *один брат и его друзья*, затем *другой* и т. д. Формируются согласованные значения особого вида

$$(x_1^1, x_2) := \{(a_1, \{b_{1j}\}), (a_2, \{b_{2j}\}), \ldots, (a_k, \{b_{kj}\})\}. \qquad (4.45)$$

Данная $з$-сеть представляет, что *с объектом* A_1 (выделенным по свойствам $\mathcal{T}_1$) *связаны* (отношениями $\mathcal{T}_2$) *объекты* $\{B_{1j}\}$, *с объектом* A_2 — $\{B_{2j}\}$ и т. д.

Особо будем выделять случай, когда все $н$-вершины графа (4.43) соответствуют единичным объектам, т. е. имеют вид x^1. Тогда делается возможным формирование согласованных значений обычного типа (см. § 2.2). Например, если в графе (4.43) имеются только две $н$-вершины (x_1^1, x_2^1) и обе из них соответствуют единичным объектам, то формируются следующие согласованные значения:

$$(x_1^1, x_2^1) := \{(a_1, b_{11}), (a_1, b_{12}), \ldots, (a_2, b_{21}), (a_2, b_{22}), \ldots\}^1, \qquad (4.46)$$

т. е. представляются все альтернативы, которые как бы выносятся на один уровень. Например, если речь идет о *друзьях брата конкретного человека*, то с помощью $з$-сети (4.46) представляются *все варианты, связанные с наличием множеств братьев и друзей*. При этом для *каждого брата* (A_i) отдельно представляется *его друг* (B_{ij}).

Следует отметить, что если следовать концепциям работы [24], то возможна более емкая запись $з$-сети (4.46):

$$(x_1^1, x_2^1) := \{(a_1, \{b_{1j}\}^1), (a_2, \{b_{2j}\}^1), \ldots\}^1,$$

где в явном виде представлены два уровня альтернатив. Однако использование таких согласованных значений связано со своими особенностями. Возникает необходимость в специальных правилах эквивалентного преобразования.

Случай, когда все $н$-вершины графа сопоставляются единичным объектам, будет играть важную роль в действиях **применения сетевых продукций** (см. § 2.1 и 5.3). Напомним, что продукции представляют закономерности, которые должны быть настроены на реальный объект или ситуацию. Такая настройка связана с формированием согласованных значений $н$-вершин (заполнением слотов), для чего в дальнейшем будет использоваться введенный ранее оператор. Если имеется несколько вариантов настройки закономерности, то возникают альтернативы. Оператор обеспечивает выявление и формирование всех таких альтернатив, из которых в дальнейшем выбирается одна. Собственно она и определяет последующие преобразования. Если допускается повторное применение одной и той же продукции, то выбранная альтернатива должна быть каким-то образом отмечена или просто изъята из множества, чтоб исключить возможность одних и тех же преобразований.

Существенный недостаток описанной процедуры связан с необходимостью предварительного формирования всех альтернатив. Хотя в ряде случаев может потребоваться нахождение лишь одной из них, к примеру, когда имеет место однократное применение продукции или речь идет о существовании объектов. Тогда рациональней использование специального метода, основанного на последовательном нахождении множеств значений, перебором элементов с возвратом. Вначале находится множество значений первой $н$-вершины. Берется один элемент и по нему находится множество значений второй. Если оно не пусто, то берется первый его элемент и ищется множество значений следующей $н$-вершины и т. д. Если же множество пусто, то происходит возврат к первому множеству и берется второй его элемент. Используется для последовательного анализа вариантов с возвратом, когда вариант оказался непригодным. При этом формируются согласованные значения $н$-вершин. Более подробно такой метод в некоторой модификации будет рассмотрен в § 4.4. Хотя для его реализации в ряде случаев удобно иметь самостоятельный оператор или встроенный механизм, как это имеет место в системе *PROLOG* [59].

Перебор с отсевом вариантов. Это типовая операция, которая необходима для работы с естественно-языковыми формами вида *те* X_1, *для которых справедливо* ... (описывается, что нужно сделать с X_1 и что при этом получить). Каждая такая форма представляется с помощью графа, областью которого является другой граф, задающий проверку на объектность. Например, граф

$$x_1\,|\textit{люди}|\ \perp [x_2 := \text{'лысый'}] \circ [x_2 \perp \langle \_, t, \textit{св}, x_2, x_1^1 \rangle] \qquad (4.47)$$

соответствует *людям, таким, что их свойством является лысость*. Граф задает операцию перебора людей с выполнением проверок, которые задаются его правой частью. Если проверка для очередной $о$-вершины (соответствующей тому или иному человеку) не выполняется, то $о$-вершина исключается из множества. В результате остается множество нужных вершин. Если в (4.47) заменить x_1 на x_1^1, то граф будет задавать дополнительную операцию проверки количества элементов в получившемся множестве. Граф с фокусом x_1^1 сопоставляется *отдельному человеку, такому, что его свойством является лысость*. Такой граф изображен как показано в левой части рис. 4.10, где пунктирный контур служит для выделения области графа, а одна из стрелок порядка указывает, что на основе этой области находятся значения x_1^1.

В дальнейшем будем использовать различные модификации введенной ранее конструкции, задающей операции перебора с отсевом вариантов.

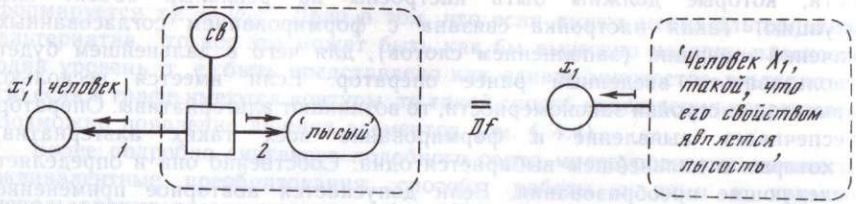


Рис. 4.10. Граф (и его мнемоническое изображение), задающий операцию перебора с отсевом вариантов

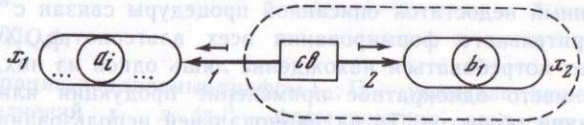


Рис. 4.11. Граф, соответствующий *тем объектам из* $\{A_i\}$, *одним из свойств которых является* B_1

Полное множество таких вариантов может быть представлено с помощью графа или с помощью *з*-сети (рис. 4.11). Такой граф записывается в виде

$$[x_1 := \{a_i\}] \circ [x_1 \perp [x_2 := b_1]] \circ [x_2 \perp \langle \_\_, t, св, x_1^1, x_2 \rangle]. \qquad (4.48)$$

Он соответствует *объектам из* $\{A_i\}$, *таким, что их свойством является* B_1. Граф задает операцию перебора элементов $\{a_i\}$ с выполнением проверок и их забраковкой. Остается подмножество из $\{a_i\}$. Оно и присваивается в качестве результирующего значения x_1.

Конечно, введенные конструкции весьма неудобны. В простых случаях ими лучше не пользоваться. Гораздо понятнее вместо *Люди, такие, что их свойством является лысость* сказать просто *лысые люди*. Последним сопоставляется более простой граф

$$x_1 \; |люди| \perp \langle \_\_, t, св, x_1, 'лысый' \rangle.$$

Однако в сложных случаях перебор может оказаться просто необходимым. Например, если требуется *найти человека, все друзья которого имеют по одному сыну*. Характерные примеры и способы их представления рассмотрены в работе [14].

Рекурсивные графы. Под ними будем понимать графы с рекурсивными фрагментами. Такие графы сопоставляются *объектам, связанным с каким-либо объектом цепочкой отношений (с незафиксированным или зафиксированным числом звеньев)*. Например, граф

$$x_2 \perp \langle a_1, t, \hat{x}_1, :x_2 \rangle \circ [\hat{x}_1 := \_\_] \qquad (4.49)$$

сопоставляется *всевозможным частям объекта* A_1, в том числе *непосредственным частям, частям этих частей* и т. д. Двоеточие перед x_2 указывает на достаточное произвольное месторасположение в кортежах вершин, соответствующих упомянутым частям (имеются в виду места, начиная с четвертого). Знак $\wedge$ у вершины x_1 указывает на рекурсию (см. § 2.2). Если в (4.49) заменить x_2 на x_2^1, то граф будет сопоставлен *одной части или компоненте*. Будем считать, что это первая непосредственная часть A_1.

Другой рекурсивный граф

$$x_2 \perp \langle \_\_, t, \hat{x}_1, x_2, a_1 \rangle \circ [\hat{x}_1 := \{\in, \subset\}] \qquad (4.50)$$

сопоставляется *всем классам и подклассам, включающим в себя объект* A_1. Захватывается вся цепочка родовидовых связей, исходящая от A_1, в соответствующей иерархической структуре.

Рекурсивные графы задают специального вида циклические операции, предполагающие подстановку получаемых результатов на аргументные места. В качестве примера рассмотрим граф

$$x_2 \perp \langle \_\_, t, \hat{x}_1, x_2, a_1 \rangle \circ [x_1 := \{л\text{-}п\}], \qquad (4.51)$$

соответствующий *объектам, которые находятся слева (л-п) от* A_1. Он задает следующие операции. Вначале на базе фрагмента $\langle \_\_, t, л\text{-}п, x_2, a_1 \rangle$ находятся значения x_2, для чего используется групповая операция ассоциирования. Найденные значения *н*-вершины x_2 подставляются на место a_1, после чего снова выполняется указанная операция... И так до тех пор, пока процесс не установится, т. е. очередное найденное множество значений x_2 не будет совпадать с тем, что было найдено ранее.

§ 4.4. РЕАЛИЗАЦИЯ ПОИСКОВЫХ СТРАТЕГИЙ

В данном параграфе будут рассматриваться конструкции графов, задающих различные стратегии структурного поиска и обработки. Будут иллюстрироваться широкие возможности графов не только для представления операционных оттенков информации, но и для реализации описанных в § 3.3 методик обработки. Такая реализация обеспечивается формированием стрелок порядка различной направленности и различного типа. Поэтому в каждом конкретном случае становится возможным выбор наилучшего направления обработки или наилучшей методики, что может осуществляться самой системой. В том числе имеются в виду методики, основанные на искусственных приемах, зачастую трудновоспринимаемых и трудновыражаемых средствами ЕЯ.

Метод хаотического поиска. Операции, обеспечивающие реализацию метода, задаются сетью, в которой стрелки $\mapsto$ отсутствуют вовсе. Обработка заключается в том, что из сети T_1, представляющей запрос, в случайном порядке берутся фрагменты (ЭФ), с помощью которых делается попытка уточнения значений их *н*-вершин. При этом используются операции ассоциирования и соединения (см. § 4.2), т. е. имеет место групповая обработка, где множества значений одних *н*-вершин служат для уточнения других. Работа заканчивается, если в течение некоторого времени не было ни одной результативной попытки, приводящей к такому уточнению.

Метод полухаотического поиска задается смешанными графами. Напомним, что в таких графах не для всех фрагментов указывается, каким образом они должны использоваться для нахождения или уточнения *н*-вершин. Для некоторых фрагментов подобного указания нет. Из каждого такого фрагмента случайным образом выделяются *н*-вершины, значения которых уточняются с помощью данного фрагмента. Подобная процедура продолжается до тех пор, пока возможно уточнение. Указанный метод используется в том случае, если во входной информации (в запросе)

направление обработки задается лишь частично и в дальнейшем оно недоопределяется.

С точки зрения оптимизации более интересными представляются случаи, связанные с предварительным уточнением или доопределением направления обработки. Устраняется всякий произвол, после чего осуществляется сама обработка. Вначале на сети (или смешанном графе) формируются (доформировываются) стрелки порядка $\mapsto$. При этом используются специальные **эвристические критерии.** В частности, стрелки $\mapsto$ расставляются таким способом, чтоб в первую очередь осуществлялся поиск по «нетипичным» вершинам (т. е. имеющим слабую окрестность в сети-знаниях) и фрагментам (т. е. имеющим как можно большее количество нетипичных вершин). Таким способом формируется полный или приведенный граф. Задаваемые им операции выполняются над сетью-знаниями, что и приводит к нахождению требуемых компонент. Эвристические критерии в некотором смысле гарантируют небольшие объемы получаемых множеств, в связи с чем обработка будет более эффективной.

В процессе подобной обработки возможны различные варианты доопределения, приводящие к реализации той или иной методики (см. § 3.2). Остановимся на некоторых из них.

Метод групповой направленной обработки. Он реализуется за счет использования групповых операций ассоциирования и операторов над множествами, см. § 4.3. Каждому элементарному фрагменту со стрелкой $\mapsto$ сопоставляется операция ассоциирования, приводящая к нахождению множеств. Если стрелки $\mapsto$ сходятся, то используется операция $\cap$ над найденными множествами. Например, граф на рис. 4.4 задает две операции ассоциирования с применением операции $\cap$ для выделения общих элементов найденных множеств. Заметим, что здесь операций перебора нет.

Конечно, описанными методиками, связанными с использованием только групповых операторов и операторов над множествами, следует пользоваться с большой осторожностью. Далеко не во всех случаях может быть найдено точное множество значений n-вершин $x_i{:}=?$, удовлетворяющих информации запроса. Сравнительно часто могут возникать лишние («неадекватные») элементы, т. е. «шумы», для устранения которых не удается обойтись без переборов. Некоторые примеры будут рассмотрены чуть ниже.

При каких же условиях, для какого сорта сетей или графов можно ограничиться только операторами групповой обработки, обойтись без переборов? Во-первых, сеть, представляющая запрос, не должна содержать контуров, т. е. замкнутых путей, проходящих по ребрам через n-вершины и вершины связи. Во-вторых, стрелки порядка $\mapsto$ должны исходить из каждой вершины связи, быть направленными к одной из n-вершин и все вместе сходиться к n-вершине $x_i{:}=?$. Эти два условия определяют тип графа. Граф должен представлять собой композицию элементарных графов. Он не может быть составным. Стрелки порядка должны образовывать дерево с корнем — n-вершиной $x_i{:}=?$. При выполнении этих условий удается избавиться от переборов, в том числе при наличии вершин типа x^1. Для поиска требуются только операции ассоциирования и пересечения ($\cap$). При этом число шагов их выполнения не будет превышать удвоенного числа элементарных фрагментов сети-запроса. Если учитывать, что последнее число имеет порядок $\approx$ 10—30, то отсюда число шагов

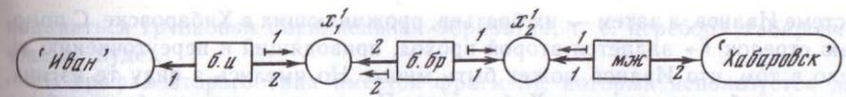


Рис. 4.12. Граф, задающий операции поиска ($\overset{1}{\mapsto}$) *братьев людей по имени Иван, живущих в Хабаровске.* Далее уточняется ($\overset{3}{\mapsto}$), *какие люди имеют таких братьев*

можно оценить $\approx$ 20—60, т. е. поиск может быть выполнен быстро. Конечно, сказанное справедливо, если имеются быстрые физические устройства или процессоры, обеспечивающие выполнение операций ассоциирования и пересечения.

В качестве примера обратимся к рис. 3.3, где изображен граф, удовлетворяющий перечисленным условиям. Все значения n-вершины x_3^1, соответствующей *другу брата Ивана,* могут быть найдены за три шага путем последовательного выполнения операций ассоциирования. Число шагов будет равно числу фрагментов графа (сказанное имеет место в том случае, когда не требуется операции $\cap$). Если полученных значений x_3^1 оказалось более одного, то должно быть сформировано альтернативное множество. Оно будет определять ответ *Друг брата Ивана есть кто-то из* {...}. Конечно, таким способом нельзя получить ответ типа *если имеется в виду Федор* (являющийся братом Ивана), *то другом* (про которого спрашивали) *есть кто-то из ..., если имеется в виду Василий, то ...* и т. д. Такой ответ предполагает использование специальных операторов перебора (см. § 4.3).

Метод групповой циклической обработки. Пусть вопрос представлен в виде сети, имеющей контура. Тогда в ряде случаев также удается обойтись без операторов перебора. Можно воспользоваться методом последовательного приближения, который применительно к семантическим сетям реализуется следующим образом. На базе сети строится граф, содержащий циклы (см. § 3.4). Подобного сорта графы задают суперпозицию операций ассоциирования (в ряде случаев к ним добавляются операторы над множествами) с рекурсивным обращением. Например, граф, изображенный на рис. 3.11, задает последовательное выполнение трех операций ассоциирования, где аргумент первого из них уточняется в результате выполнения последнего, второго — в результате выполнения первого, а третьего — в результате выполнения второго. Имеет место обход по циклу, образованному стрелками $\mapsto$. При этом каждый виток обхода — это и есть шаг последовательного приближения. Процесс заканчивается, если очередной шаг приближения не привел к уточнению значений x_1^1, x_2^1 и x_3^1.

В общем случае возможно множество контуров, предполагающих организацию специальных обходов. При этом должны захватываться все фрагменты сети, что осуществляется последовательным переходом с одних контуров на другие.

Иногда возникает необходимость многократного использования одного и того же фрагмента, но для уточнения различных его компонент. Требуется различать «проходы», осуществляемые в различном направлении. Для этого могут использоваться специальным образом отмеченные стрелки. На рис. 4.12 с помощью стрелок $\overset{1}{\mapsto}$ задается один проход, приводящий к уточнению значений x_1^1 и x_2^1. Находится множество всех известных

системе Иванов, а затем — их братьев, проживающих в Хабаровске. С помощью стрелок $\overset{2}{\mapsto}$ задается второй проход, приводящий к переуточнению $x^{\downarrow}_{1}$. Дело в том, что Иванов может быть много. Но имелись в виду те из них, у которых братья живут в Хабаровске. После того, как эти братья были найдены, легко определить, о каких Иванах шла речь, что осуществляется за счет второго прохода.

Использование метода последовательных приближений может значительно ускорить обработку, если исключить операции перебора. Однако в ряде случаев становятся возможными, как уже говорилось, побочные эффекты — лишние или неадекватные элементы. Рассмотрим в качестве примера граф на рис. 3.12. Чтобы избавиться от переборов, изменим расположение стрелок $\mapsto$. Направим верхнюю стрелку $\mapsto$ не к вершине $x^{\downarrow}_{3}$, а к $x^{\downarrow}_{1}$. В результате образуется цикл. Граф с циклом будет задавать процедуру последовательного приближения. При этом будут находиться требуемые люди X_3, т. е. удовлетворяющие условию запроса. Но, помимо этого, раз за разом будут выявляться такие *два человека* X_3 (соответствующие *о*-вершины будут присваиваться в качестве значения $x^{\downarrow}_{3}$), *где первый имеет друга, брат которого является сослуживцем второго, а второй имеет другого друга, брат которого является сослуживцем первого.* Нетрудно видеть, что ни для первого, ни для второго не удовлетворяются оба условия запроса, в котором предполагается, что *требуемый человек должен иметь друга (хабаровчанина) и работать с братом этого же друга.* Появляется пара лишних элементов.

Интересным является **метод групповой обработки** с так называемыми **параллельными** ветвями, см. § 3.3. В графе допускаются параллельные пути, образованные стрелками $\mapsto$ и определяющие варианты независимого уточнения компонент — по одному и тому же найденному множеству. Подобного сорта графы представляют предложения ЕЯ, в которых имеются специального вида операционные оттенки, выражаемые с помощью оговорок типа *не обязательно тот же самый, могут быть связаны с различными.* Такие графы и задают указанную методику.

В дальнейшем будем различать графы (с параллельными путями) двух типов. В графах первого типа множество найденных значений какой-либо *н*-вершины используется в двух или нескольких вариантах уточнения множеств значений других *н*-вершин, а в конечном счете и *н*-вершины типа x_i: = ?. Пусть в графе рис. 3.12 два пути по стрелкам $\mapsto$ сделаны независимыми. Тогда уже будет представлен запрос следующего вида: *Кто является другом хабаровчанина и работает вместе с братом некоего хабаровчанина, не обязательно первого*? Сравните с тем, что было представлено на рис. 3.12. Имеется существенное отличие, определяемое оговоркой. Поиск ответа на такой запрос осуществляется следующим образом: по множеству хабаровчан находятся все их друзья и параллельно — сослуживцы всех их братьев. Далее выделяются общие элементы ($\cap$). Используются уже другие операции, чем задаваемые графом рис. 3.12.

Для изображения графов первого типа требуются специальные средства указания параллельных ветвей. Их нужно отличать от обычных контуров и проходов. Для такого указания будем использовать меченные стрелки вида $1\mapsto$. Например, если на рис. 3.12 такие стрелки поставить на места стрелок $\mapsto$, исходящих из вершин связи с отметками *б. др* и *б. бр*, то будет

задаваться групповая параллельная обработка, т. е. перебора хабаровчан уже не будет.

В графах второго типа имеется фрагмент, который используется для нахождения согласованных значений пары или набора *н*-вершин, а последние используются в параллельных вариантах уточнения. Подобные варианты также будем указывать меченными стрелками вида $1\mapsto$. При этом возможны и стрелки $2\mapsto$ — для указания на множество других параллельных ветвей, отличных от первых.

Следует отметить, что использовать указанный метод нужно с большой осторожностью. В самом деле, пусть он применяется для реализации требования, представленного на рис. 3.12, где стрелки не помечены. Тогда будут найдены и такие люди (X_3), которые *являются друзьями одних хабаровчан, но работают вместе с братьями других.* Отсюда становится ясной природа лишних элементов.

Методы согласованного уточнения и обратного поиска. Напомним, что метод согласованного уточнения предполагает два этапа. На первом осуществляется независимое нахождение по каждому свойству или отношению, упомянутому в запросе, всех связанных с ними неопределенных компонент, а на втором этапе — отбор требуемых элементов, см. § 3.3. Подобная обработка задается графами, состоящими только из фрагментов $[z_i \perp T^{\circ}_{j}(z_i)]$ (где — z_i — все множество *н*-вершин из T°_{j}) и *з*-сетей z_{ξ} : = {...}. В таких графах стрелки порядка исходят из вершины связи каждого элементарного фрагмента (ЭФ) и направлены к шрихпунктирной окружности, ограничивающей все *н*-вершины данного фрагмента (простейшие примеры таких графов см. на рис. 4.7 и 4.8). Обработка, задаваемая подобными графами, сводится к нахождению полных наборов согласованных значений и их усечению с помощью операций сцепления. Последние выполняются многократно до тех пор, пока возможно усечение.

Остановимся на принципах реализации метода о б р а т н о г о поиска. Напомним, что этот метод задается графами, представляющими формы вида *те X, для которых справедливо* $\mathscr{T}_1$. Пример такого графа был изображен на рис. 4.11, где имеется один контур, обведенный пунктирной линией. В общем случае допускается множество таких контуров, где один может быть «вложен» в другой, т. е. находиться внутри него. На рис. 4.13 изображен граф, имеющий три вложенных контура. Он задает три операции перебора (с отсевом вариантов), с помощью которых обеспечивается нахождение *друга брата Ивана.* Если сказать более точно, то ищется *тот человек, который имеет друга, имеющего брата, который имеет имя Иван.* Конечно, предложение весьма трудно воспринимается, что связано со сложностью задаваемых операций. Рассмотрим последние более подробно.

Поиск имеет характер так называемых обратных рассуждений типа *А что если таким человеком является* A_i. *Посмотрим, есть ли у него друзья. Вот они —* $\{B_{i1}, B_{i2}\}$. *Возьмем одного из них —* B_{i1} *и посмотрим, есть ли у него брат. Нет, тогда берем другого...* Подобные рассуждения сводятся к поиску соответствующих множеств и перебору их элементов. Применительно к графу на рис. 4.12 будут выполнены следующие операции. Вначале (на основе знаний) выбираются все известные системе люди — X_3. Берется один из них — A_i. Находится множество его друзей. Если оно не пусто, то берется первый из них. Для него находится множество братьев. Если множество не пусто, то берется первый из братьев и проверяется, имеет ли он

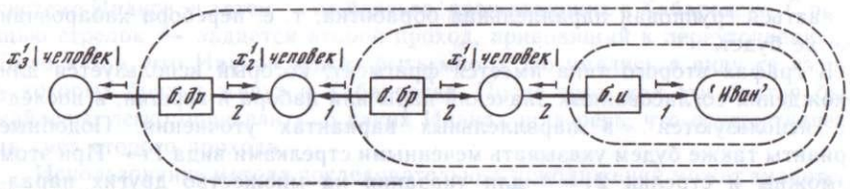


Рис. 4.13. Пример использования метода обратного поиска для нахождения *друга брата Ивана*

имя Иван. Если да, то выбранный человек A_i отвечает информации запроса. Если нет, то берется второй элемент из множества братьев и т. д. Если все братья перебраны и ни для кого не были выполнены проверки, то берется второй элемент множества друзей и описанные операции повторяются заново. Если все друзья перебраны и ни для кого ни в каком варианте не были выполнены проверки, то A_i считается неадекватным и удаляется из множества людей X_3. Берется второй человек — A_j, и все повторяется заново. То же самое делается, если у A_i нет друзей, т. е. порожденное множество оказалось пустым. Вообще, если на каком-либо шаге получено пустое множество, то соответствующий вариант бракуется, осуществляется возврат и переход к другому варианту. Результатом работы всех операций, задаваемых графом рис. 4.13, является множество оставшихся значений *н*-вершины x_3^1. В этом отличие описанных операций от тех, которые рассматривались в § 4.3, когда речь шла о способах выявления одной приемлемой альтернативы среди имеющихся. Кроме того, там предполагался поиск согласованных значений.

Конечно, пользоваться методом обратного поиска следует только в том случае, если множества элементов и соответственно вариантов могут быть ограничены до умеренных величин. Если в систему заложены знания о чрезвычайно большом количестве людей, то их перебор недопустим. Переборы и использование идеи возврата хороши в том случае, когда в запросе или сообщении речь идет об уникальных объектах, указывается на принадлежность к классам, имеющим малое число элементов (представителей), или на принадлежность к аналогичным множествам. Более того, отношения, о которых идет речь в запросе, должны быть «нетипичными», т. е. быть представленными в знаниях в ограниченном количестве экземпляров. Тогда есть гарантия, что в процессе поиска по отношениям объем множеств не будет сильно возрастать. Количество переборов будет оставаться умеренным.

Например, возьмем запрос *С объектом* A_1 *или* A_2 *связан отношением* R_1 *объект* X_1, *который находится в отношении* R_2 *с объектом, обладающим свойствами* $\mathcal{T}_2$?. На первом шаге возможны только допущения *Пусть этим объектом является* A_1 и *Пусть* — A_2. И в том и в другом случае находится множество X_1, относительно которого также делаются допущения. Объем полученных множеств во многом будет зависеть от «типичности» отношения R_1, что влияет на количество допущений и переборов.

Сильные ограничения на множество допущений накладываются запросами вида *Правильно ли, что ..., Существует ли такой* X_1, *что ...*, и др. Такие запросы представляются в виде сетей, у которых значения *н*-вершин

типа p_i: = ? ограничены всего двумя элементами — $\{t, f\}$. При переборе (на первом шаге) имеют место всего две альтернативы. В некоторых случаях и на значения других *н*-вершин могут быть наложены аналогичные ограничения. Например, пусть требуется *найти* P_1, *для которого справедливо* $(P_1 \vee O) \wedge 1 = 1$. Пропозициональной переменной P_1, а также результату операции $\vee$ сопоставляются *н*-вершины p_1^1: = $\{t, f\}$ и p_2^1: = $\{t, f\}$. Тогда метод допущений и обратного поиска будет достаточно эффективным.

В общем случае метод обратного поиска может сочетаться с прямым поиском. Вначале выбираются фрагменты, представляющие уникальные объекты и нетипичные отношения. По ним делаются попытки уточнить неопределенные компоненты, т. е. значения *н*-вершин. И если уточняющие множества оказались небольшими по объему, то используется метод обратных рассуждений. Соответствующим образом переформировываются стрелки порядка и осуществляются переборы.

Конечно, метод обратных рассуждений с возвратом, используемый в поисковых стратегиях, значительно у́же, чем применяемый при доказательстве теорем. При поиске имеется схема отношений. Неизвестными являются лишь компоненты данной схемы. При доказательстве такой схемы нет. Приходится подыскивать ее варианты, т. е. из каких одних истинных утверждений могут следовать другие. Такие варианты должны выбираться в соответствии с теоремой, которую требуется доказать. При возвратах должны делаться попытки использования новых теорем или правил вывода. Для их выбора требуются свои эвристики. Возникает как бы еще один уровень неопределенности.

ГЛАВА 5

СРЕДСТВА АКТИВНОГО ИЗМЕНЕНИЯ СТРУКТУР

Типовыми задачами информационных, распознающих и других систем является, во-первых, пополнение и корректировка входной информации, в частности выявление недостающих компонентов, признаков, во-вторых, накопление и корректировка знаний за счет входных сообщений, описаний и, в-третьих, так называемое преобразование представлений. К последним относятся и эквивалентные преобразования. Как будет показано ниже, эти задачи имеют родственный характер. Они обладают одними и теми же особенностями. Отсюда вытекает необходимость их решения в рамках единого подхода и единых средств. В качестве последних будет развиваться язык семантических графов. Остановимся на особенностях упомянутых задач, способах их решения.

При **пополнении входных сообщений** большую роль играют разного рода знания. Будем различать два способа пополнения — за счет пассивных и за счет активных знаний. Первый способ предполагает пополнение в процессе идентификации (отождествления компонент запроса или сообщения и знаний), поиска и конкретизации (уточнения неопределенных компонент запроса или сообщения). Например, пусть на вход системы поступило сообщение, в котором речь идет о *некоем субъекте со свойством* R_1. Пусть в процессе поиска выяснилось, что таким субъектом является A_1.

Тогда все, что известно об A_1, может быть автоматически перенесено в сообщение. Подобное пополнение сводится к специальной идентификации вершин-объектов с вершинами-классами с переносом окрестностей (см. § 4.2). В процессе поиска сообщение будет как бы обрастать все новыми связями, которые делают возможным уточнение все новых компонент. Указанный процесс делает возможным понимание связанных текстов, где допускаются разного рода умолчания, эллиптические конструкции и др. [39].

Ниже в основном будет развиваться другой способ пополнения — за счет активных знаний, определяющих, что в тех или иных случаях *может быть* и что *должно быть*. Они, как правило, носят условный характер: *если сказано A, то должно быть и B*. В дальнейшем подобного сорта знания будут называться о б я з а т е л ь н ы м и. Они играют ту же роль, что и белые кровяные тельца в системе жизнедеятельности человека. При поступлении «инородного» тела (входного сообщения, вопроса) они как бы активизируются и воздействуют, изменяя (дополняя, корректируя сообщения) или полностью уничтожая его (указывая на невозможность входной информации). Если же тело оказалось не инородным (сообщение согласовано с информацией знаний), то оно остается в системе жизнедеятельности (сообщение переносится в знания). Но есть существенное отличие. Функции белых кровяных телец фиксированы генетической структурой. В то же время обязательные знания могут сами постоянно изменяться, пополняться за счет входных сообщений. В результате будут изменяться анализируемые и синтезируемые компоненты. Обеспечиваются дополнительные возможности в плане анализа и синтеза все новых сообщений.

Аналогичные действия необходимы и в процессе **накопления знаний.** Предполагается анализ наличия той или иной информации в знаниях и синтез, связанный с формированием новых (недостающих) компонент. Последние должны быть каким-то образом привязаны к уже имеющимся компонентам (вершинам). Здесь роль активных структур играют входные сообщения, которые воздействуют на знания, изменяя их.

И, наконец, к описанным задачам близко примыкают задачи **преобразования представлений,** где также требуются действия анализа и синтеза, например, выявление наличия одних отношений и формирование других. Во многих случаях учесть все такие действия не представляется возможным. Способы анализа и синтеза должны постоянно изменяться, в связи с чем возникает необходимость в использовании специального вида «обязательных» знаний, роль которых могут играть сетевые продукции (см. § 2.1). Их применение приводит к активному воздействию на входную информацию с ее преобразованием.

Сравнительно часто бывает трудно отличить высказывательные формы, претендующие на роль знаний, от конкретных высказываний, несущих новую информацию. Это еще раз подчеркивает родство, близость упомянутых задач. Хотя имеются и отличия. Например, возьмем высказывания *У дома должна быть крыша.* Если это входное высказывание, то оно может относиться к конкретному дому. Тогда оно будет нести информацию о наличии у этого дома крыши, что должно быть представлено в знаниях. Это же высказывание может быть прочитано другим способом. *Если упоминается дом, то должна быть и крыша.* Имеется в виду достаточно

произвольный дом, т. е. некий абстрактный объект. Подобного сорта высказывания и отображаются на обязательные знания, которые будут играть важную роль в процессе пополнения и корректировки входных сообщений.

Разумно допустить, что родство задач предполагает необходимость единой методики их решения. При этом важную роль в процессе такого решения должны играть способы анализа и воздействия. Они во многом определяются видом высказывательных форм, имеющейся в них акцентацией. Во входных сообщениях с помощью акцентации (тем, рем) обращается внимание на то, что является новым. В обязательных знаниях такая акцентация имеет вид указаний, чего может не доставать во входных сообщениях и что должно быть. Более того, может быть указано, как искать и то и другое, в какой последовательности уточнять неопределенные компоненты. Сказанное относится и к задаче преобразования представлений, где акцентация должна определять саму процедуру анализа и синтеза. Сетевых продукций здесь оказывается явно недостаточно. В связи с этим будет использоваться язык семантических графов, который будет развиваться в направлении обеспечения согласованного представления высказывательных форм, предполагающих активное воздействие. Будут введены графы, которые задают соответствующие операции «изменения». Их выполнение и обеспечивает необходимые виды анализа и синтеза. Таким способом будет учитываться операционная семантика высказывательных форм.

В данной главе будут рассматриваться графы, необходимые для представления высказываний лишь конкретного характера, т. е. не содержащих слов-кванторов типа *все, только, некоторые* и др., а также нечетких понятий. Принципы представлений высказываний типа *У каждого дома имеется крыша, У некоторых домов нет балконов* и др. будут рассматриваться в гл. 7. Ниже основное внимание будет уделено так называемым активным графам, т. е. связанным с функциями воздействия и учитывающим имеющийся акцент. Будут введены понятия обязательных знаний и граф-продукций. Последние являются расширением понятия сетевой продукции (см. § 2.1). С графами и граф-продукциями будут связываться операции, определяющие характер обработки, способ использования высказываний, определений, пояснений.

§ 5.1. ПРЕДСТАВЛЕНИЕ ВЫСКАЗЫВАНИЙ С АКЦЕНТАЦИЕЙ НОВИЗНЫ

В данном параграфе будет развиваться язык представления высказывательных форм, несущих новую информацию. Будут рассматриваться семантические графы, задающие не только операции поиска, но и воздействия для изменения внутрисистемных знаний.

Об оттенках новизны. В предыдущих главах рассматривались операционные оттенки, связанные с поиском, уточнением и проверкой информации. Такой оттенок имеют вопросы, а также высказывания, если основной целью является проверка, правильны ли они, содержится ли соответствующая информация в знаниях. Для их представления с учетом имеющих место акцентов были введены различные конструкции семантических графов. Говорилось о том, что они задают операции поиска и проверки. Например, вопрос представляется в виде графа, задающего операции, выполнение которых над знаниями приводит к нахождению значений *и*-вершин графа, а стало быть, и неопределенных компонент вопроса.

Заметим, что если система учитывала бы только такие операционные оттенки, то она могла бы лишь отвечать на запросы и реагировать на высказывания следующим образом: *да, это так* или *это мне известно* (если проверки удовлетворяются) и *нет, этого я не знаю* (если проверки не удовлетворяются или не могут быть найдены какие-либо компоненты). Конечно, это была бы крайне ограниченная система, не способная гибко реагировать на входные сообщения.

В то же время многие высказывания имеют целью передать собеседнику новую информацию, изменить его представление о том или ином объекте, ситуации. Фактически имеет место подтекст — *собеседник должен знать все то, о чем говорится. И если он этого не знает, то нужно сделать так, чтобы он знал*. Передача новой информации может делаться в прямой форме (в виде указания на существование объектов или на их свойства) или же в условной форме (в виде таких же указаний, которые справедливы лишь при выполнении определенных условий).

Остановимся вначале на **прямой форме**. Выясняется, что, как правило, акцент делается на тех компонентах, которые (по мнению автора высказывания) интересны для собеседника, ранее ему не были известны. Например, возьмем высказывание *Объектом со свойством* $\mathcal{T}_1$ *точно является* (или *должен быть*) A_1, адресованное субъекту B_1. Предполагается, что ранее (с какой-то вероятностью) B_1 это не было известно. Подтекст здесь следующий — B_1 должен в своих представлениях найти объекты со свойством $\mathcal{T}_1$ и проверить, что среди них есть A_1, а если нет, то изменить представления, сформировав A_1 с указанными свойствами. Следует обратить внимание, что несколько другой акцент имеет высказывание *Объект* A_1 *(точно) обладает свойством* $\mathcal{T}_1$. Предполагается нахождение свойств у объекта A_1 и проверки, что среди них есть $\mathcal{T}_1$, а если нет, то формирование этого свойства.

На представление подобной акцентации был ориентирован язык семантических графов. Для указания на наличие новой информации, ниже будут введены специальные средства, означающие долженствование (предполагается выполненность проверок), необходимость соответствующих изменений во внутрисистемных представлениях. Такие средства являются атрибутом активных структур или знаний (см. § 3.2). Они будут служить для указания на фрагменты, представляющие новую информацию. На фрагментах как бы ставятся метки. При этом в ряде высказываний акцентация может присутствовать в неявной форме. Новой может быть любая компонента. Тогда необходим режим работы, когда метки на фрагментах проставляются самой системой, что осуществляется в процессе выявления причин отсутствия компонент, нерезультативных реакций. Ниже будет описан такой режим.

Конечно, далеко не всегда наличие новой информации следует связывать с изменениями во внутрисистемных представлениях. Здесь большую роль играет, от кого поступило высказывание и даже о чем идет речь и в какой предметной области. В математических текстах новыми могут быть свойства абстрактных объектов, которые нужно вывести, а не просто изменить знания, представив данное свойство. Сейчас рассмотрим наиболее простой случай, когда высказывание заведомо поступает от достоверных источников и имеет целью воздействие на собеседника (систему) для изменения его представления. При этом речь будет идти лишь о высказываниях, представляющих свойства и отношения конкретных объектов.

Знак обязательности $\square_\phi$. **Понятие активного графа**. С помощью высказываний могут выражаться различные оттенки обязательности, долженствования. Например, могут указываться обязанности — *Он должен сделать работу в срок*. Ниже будет рассматриваться несколько другой оттенок, когда долженствование используется с тем, чтобы придать высказыванию уверенность, обязательность выраженных соотношений (*Он должен знать, что...*). Для представления таких высказываний будем использовать графы со специальными фрагментами. Последние будут служить для указания способа воздействия на пассивные структуры (в частности, внутрисистемные знания) для их изменения.

В дальнейшем, чтобы отличать одни структуры (активные графы) от других (пассивных), будем обозначать их *н*-вершины и *с*-вершины по-разному. Для обозначения *н*-вершин активных графов будем пользоваться символами y (в отличие от символов x в пассивных структурах), а *с*-вершин — символами β (в отличие от символов γ).

Напомним, что в общем случае граф, представляющий высказывательную форму, имеет вид композиции $\Gamma_\xi \circ \Phi_i$, где Γ_ξ сопоставляется неопределенным компонентам, указывается, как их находить. Часть Φ_i представляет собой *з*-сеть или множество *з*-сетей, отражающих наличие тех или иных соотношений. Указывается на факт существования или несуществования объектов в той или иной ситуации или предметной области, представленной с помощью собственной сети T. Может указываться и на соотнесенность объектов к множествам. Соответствующие *з*-сети и задают операции проверки.

Пусть $[y_i := \eta]$ — *з*-сеть, входящая в часть Φ_i. Обозначим через $[\square_\phi y_i := \eta]$ *з*-сеть, задающую не только проверки, но и указывающую на необходимость изменения сети T (если проверки на ней не удовлетворяются). Знак $\square_\phi$ будет означать д о л ж е н с т в о в а н и е, а индекс «ф» — указывать на активность, связанную с формированием (в конце параграфа будут рассматриваться и другие виды внутрисистемной активности). Вся *з*-сеть со знаком $\square_\phi$ будет означать *должны быть представления типа* η *об объекте* Y_i, *нужно сделать так, чтоб* Y_i *был представлен, как это указано в* η.

Будем называть в е р ш и н ы (и *з*-сети) со знаками $\square_\phi$ о б я з а т е л ь н ы м и. Имеется в виду обязательность представленных ими соотношений и выполнимость задаваемых проверок. Г р а ф ы с такими вершинами будем называть а к т и в н ы м и. Предполагается активность, связанная с изменением знаний. При изображении активных графов будем ставить знак $\square_\phi$ впереди *н*-вершин y_i, т. е. тех, которые соответствуют новым объектам и связям. К таким *н*-вершинам сходятся стрелки порядка. Рассмотрим несколько простых примеров. Граф

$$[\square_\phi y_1^1 := a_2] \circ [y_1^1 \perp \langle \_, t, r_1, a_1, y_1^1 \rangle] \tag{5.1}$$

означает, что *отношением* R_1 *с объектом* A_1 *связан именно объект* A_2 (и если в знаниях этого раньше не было представлено, то нужно соответствующим образом изменить их). Другой граф

$$[\square_\phi y_1^1 := \{a_2, a_3\}] \circ [y_1 \perp \langle \_, t, r_1, y_1^1, a_1 \rangle] \tag{5.2}$$

означает, что *с* A_1 *отношением* R_1 *связаны и* A_2 *и* A_3 *и что это точно известно*. Такой граф изображен на рис. 5.1.

В активных графах будем допускать инверсные вершины. Пусть $\tilde{a}_3$ — одна такая вершина, которая соответствует *отрицанию объекта* A_3.

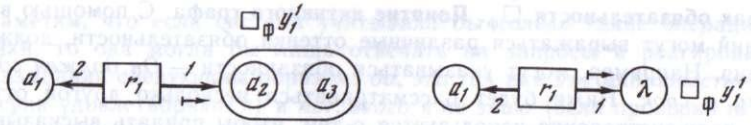


Рис. 5.1. Активный граф, представляющий, что *объект* A_1 *точно связан отношением* R_1 *с* A_2 *и* A_3

Рис. 5.2. Пример представления факта *существования* (λ) *объекта, связанного с* A_1 *отношением* R_1

Тогда граф

$$[\square_\phi y|:=\{a_2, \tilde{a}_3\}] \circ [y| \perp \langle \_\_, t, r_1, y|, a_1 \rangle \quad (5.3)$$

будет означать, что *отношением* R_1 *с* A_1 *связан не объект* A_3, *а* A_2. Речь идет о достаточно типовой форме, часто встречающейся в ЕЯ. Предполагается, что в знаниях ранее было представлено отношение R_1 между объектами A_3 и A_1. Такие представления должны быть изменены. Должна быть представлена связь объекта A_1 уже с A_2.

Наконец, рассмотрим еще один типовой пример. Граф

$$[\square_\phi y|:=\lambda] \circ [y| \perp \langle \_\_, t, r_1, y|, a_1 \rangle] \quad (5.4)$$

означает, что *точно имеется объект* Y_1, *связанный отношением* R_1 *с* A_1. И если ранее этого не было известно, то представления должны быть соответствующим образом дополнены. Такой граф изображен на рис. 5.2.

Оператор формирования фрагментов. Ясно, что изменение, модификация знаний должны осуществляться в процессе поступления сообщений, высказываний. Для представления последних будем использовать активные графы с вершинами, помеченными знаками $\square_\phi$. Будем считать, что фрагменты с такими вершинами задают специальную операцию формирования, которая выполняется над сетью-знаниями T и вызывает ее изменение.

Прежде чем перейти к описанию операции формирования, хочется отметить, что операторы добавления, изъятия и замены характерны для языков манипулирования данными [10, 47]. В настоящее время предложено множество разновидностей таких операторов — применительно к различным языкам описания данных. Однако они не применимы для работы с данными структурного характера, имеющими вид введенных ранее семантических сетей. Здесь возникают свои **особенности**. Остановимся на некоторых из них.

Чтобы дополнить семантическую сеть, необходимо, во-первых, указать, к каким вершинам привязываются новые фрагменты, и, во-вторых, знать, что это за фрагменты. Следует помнить, что многие вершины сети не имеют своих уникальных имен и могут быть выделены только по окрестности, т. е. по их связям с другими вершинами. Далее, дополнение фрагментов должно сопровождаться указанием на существование соответствующих отношений, истинность информации. Для этого должна быть выделена *с*-вершина или *р*-вершина фрагмента. *С*-вершинам (в кортежах они стоят на первом месте) должны присваиваться значения λ, а *р*-вершинам (они стоят на втором месте) — *t*.

При изъятии фрагментов также необходимо выделение их *с*-вершин. Если *с*-вершина не имеет своего уникального «имени», то должна быть указана ее окрестность, которой может являться сам фрагмент со всеми его вершинами. Требуется предварительное выделение последних. По ним уже выделяется *с*-вершина, что равносильно выделению самого фрагмента. Аналогичную роль могут играть и *р*-вершины.

Следует отметить, что если речь идет о системах представления знаний, то одного действия изъятия (как физического уничтожения фрагментов, полного «забывания» фактов) явно недостаточно. Требуются более тонкие действия, связанные с модификацией фрагментов. Система должна знать и то, что имело место в прошлом, и то, что имеет место в настоящем. Здесь физическое уничтожение просто недопустимо. Конечно, далеко не всегда необходимо учитывать то, что имело место в прошлом. Здесь необходимы специальные действия м е т к и фрагментов. Если мы имели метки, то в дальнейшем фрагмент при определенных видах внутрисистемной обработки просто не будет учитываться. Соответствующие соотношения не будут использоваться. Проставление меток может осуществляться различными способами, в частности присвоением *с*-вершине фрагмента значения $[\gamma:=\tilde{\lambda}]$, т. е. представляется факт *несуществования соответствующего отношения*, или присвоением *р*-вершине значения $[p:=f]$, т. е. представляется *ложность*.

И, наконец, еще один важный момент связан с возможностью интегрированных представлений, когда какая-либо ситуация рассматривается как единая. Все, что входит в данную ситуацию, рассматривается как ее части. Тогда любое изменение ситуации должно приводить и к изменению отношения *часть — целое*. Подобного сорта изменения должны учитываться в процессе формирования новых фрагментов. Пусть ситуации сопоставлена своя вершина. Ее окрестностью будут фрагменты, представляющие отношение *часть — целое* (см. § 1.4). Тогда при вводе новой информации должны каким-то образом изменяться и эти фрагменты, т. е. осуществляться перезамыкание.

Перечисленные особенности делают невозможным использование типовых операторов, нашедших распространение в реляционных и других языках Требуются собственные средства. Их роль будут играть активные графы. Фрагменты с вершинами, помеченными знаками $\square_\phi$, будут задавать операции формирования, выполнение которых над сетью-знаниями и будет приводить к необходимым изменениям.

Остановимся на общих соображениях, позволяющих прояснить **функции активных графов**. Последние как бы играют роль клише, которое заполняется (сколь это возможно) материалом знаний — взятыми оттуда «деталями». Имеет место нечто подобное заполнению слотов фреймов. Не только фиксируется порядок заполнения. Далее клише автоматически дополняется. Вводятся новые «детали», которые определяются незаполненными звеньями клише. Введенные детали привязываются к тем, которые участвовали в заполнении. В результате получается цельное изделие, которое переносится в знания. Если необходимо, то осуществляется их некоторая перекомпоновка, связанная с интегрированными представлениями. Обратите внимание, что изделие будет привязано к имеющимся в знаниях «деталям». Подобного сорта действия и осуществляются оператором формирования.

Оператор формирования включается в работу, если не выполнены проверки, задаваемые соответствующими фрагментами графа (с вершинами, помеченными $\square_\phi$). На базе последних и формируются новые фрагменты,

которые пополняют сеть-знания, изменяют ее. При этом допускается случай, когда в формировании участвуют группы связанных фрагментов. Пусть при выполнении операций, задаваемых очередным фрагментом графа, было получено пустое множество значений какой-либо *н*-вершины y_i. Ничего не известно о соответствующем объекте Y_i. Тогда нужно дополнить знания, введя представления об этом объекте. Это осуществляется формированием «резервного» значения *н*-вершины y_i и нового фрагмента. Таким способом могут формироваться цепочки фрагментов, представляющих связь промежуточных объектов типа Y_i с основным, на котором и акцентируется внимание в высказывании. Могут возникать и альтернативы, например, когда речь идет об одном объекте Y_i, а их оказалось несколько. Какой из них имеется в виду? Все это учитывается оператором формирования. Для его описания остановимся вначале на частных случаях, задаваемых графами сравнительно простого вида.

Пусть $\Gamma_1(y_i^1)$ — граф, задающий способ нахождения значений *н*-вершины y_i^1, а $T_2(y_1, y_i^1)$ — фрагмент с вершинами y_i^1 и y_1. Рассмотрим конструкцию вида

$$\Gamma_1(y_i^1) \cdot [y_1 \perp T_2(y_1, y_i^1)] \circ [\square_\phi y_1 := \{a_\xi\}], \tag{5.5}$$

представляющую, что *объектами* Y_1, *находящимися в отношении* $\mathcal{T}_2$ *с объектом* Y_i, *точно являются* $\{A_\xi\}$ *и что* Y_i *— объект со свойствами* $\mathcal{T}_1$. Пусть в результате выполнения операций, задаваемых Γ_1, было найдено лишь одно значение y_i^1. Тогда конструкция (5.5) задает операцию нахождения значений y_1 и выполнения проверок (что задается $[y_1 := \{a_\xi\}]$). И если проверки не выполнимы, то на базе $T_2(y_1, y_i^1)$ формируются новые фрагменты, которые пополняют сеть-знания. Такое формирование сводится к замене в T_2 *н*-вершины y_i^1 на ее значение, а *н*-вершины y_1 — на каждое из $\{a_\xi\}$, не присутствующее в множестве найденных значений y_1. В результате формируется множество фрагментов. Они и пополняют сеть-знание.

Например, граф вида (5.1) задает операцию формирования фрагмента $\langle \_\_, t, r_1, a_1, a_2 \rangle$, а граф рис. 5.1 — формирования двух фрагментов $\langle \_\_, t, r_1, a_1, a_2 \rangle$ и $\langle \_\_, t, r_1, a_1, a_3 \rangle$. Они вводятся в знания.

Рассмотрим конструкцию другого вида:

$$\Gamma_1(y_i^1) \circ [y_1 \perp T_2(y_i^1, y_i^1)] \circ [\square_\phi y_i^1 := \lambda], \tag{5.6}$$

представляющую факт *существования объекта* Y_1. Она задает те же операции поиска, что и (5.5). Но проверяется непустота найденных значений y_i^1. И если проверка не выполнима, то формируется «резервное» значение y_i^1. В качестве такого значения берется *о*-вершина, ранее не входившая в сеть-знания. Далее в $T_2(y_i^1, y_i^1)$ все *н*-вершины заменяются на их значения, а полученный фрагмент дополняет сеть-знания. Например, граф рис. 5.2 задает операцию формирования «резервной» вершины d_k, а на базе нее — фрагмента $\langle \_\_, t, r_1, d_k, a_1 \rangle$.

Если в (5.6) y_i^1 является *с*-вершиной фрагмента $T_2(y_i^1, y_i^1)$, то просто формируется фрагмент. Он получается на базе T_2 путем описанных ранее замещений. В качестве примера рассмотрим типовую конструкцию вида

$$[y_2^1 \perp T_1(y_2^1)] \circ \langle \beta_1, t, r_1, y_2^1, y_3^1 \rangle \circ [\square_\phi \beta_1 := \lambda] \circ [y_3^1 \perp T_2(y_3^1)], \tag{5.7}$$

где вершина y_1 из (5.6) обозначена через β_1. Представляется *обязательность отношения* R_1 *между объектами со свойствами* $\mathcal{T}_1$ *и* $\mathcal{T}_2$. Стрелки порядка

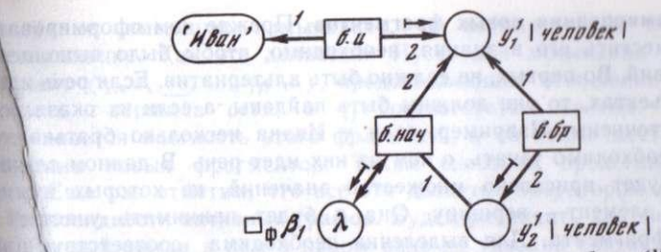


Рис. 5.3. Пример активного графа, представляющего, что *брат* (Y_2) *некоего Ивана* (Y_1) *точно является его же начальником*

должны быть направлены к *с*-вершине $\square_\phi \beta_1$. Если указанное отношение R_1 не представлено в знаниях, то предполагается формирование соответствующего фрагмента. Конечно, такое формирование возможно лишь в случае, если найдены уникальные значения y_2^1 и y_3^1, т. е. им присвоено только по одному значению. Тогда формируется кортеж, составленный из этих значений. При этом сами *н*-вершины никак не будут присутствовать в знаниях.

Перейдем к рассмотрению конструкций с инверсными вершинами. Заменим в (5.7) *з*-сеть $[\square_\phi \beta_1 := \lambda]$ на $[\square_\phi \beta_1 := \tilde{\lambda}]$. Тогда будут представлены сведения об *отсутствии отношения* R_1 *между двумя объектами*. Получившаяся конструкция будет задавать следующие операции. Вначале ищутся значения y_2^1 и y_3^1. Затем на базе «клише» $\langle \beta_1, t, r_1, y_2^1, y_3^1 \rangle$, которое накладывается на сеть-знания, будет осуществляться поиск в этой сети идентичных фрагментов. И если такие фрагменты имеются, то их *с*-вершинам (γ_i) присваиваются значения $[\gamma_i := \tilde{\lambda}]$, т. е. объявляется о *не существовании отношения* R_1. (В другом варианте фрагменты просто изымаются, т. е. имеет место случай полного «забывания».) Если же такой фрагмент не найден, то никаких действий, связанных с модификацией знаний, не осуществляется.

Пусть в (5.5) на месте $\{a_\xi\}$ стоит инверсная вершина $\tilde{a}_\xi$. Тогда будет представлено, что *объектом* Y_1 *не может быть* A_ξ. Задается операция поиска соответствующего фрагмента в знаниях. И если он есть, то его *с*-вершине присваивается значение $\tilde{\lambda}$ (в другом варианте фрагмент просто изымается). Например, граф (5.3) задает операции поиска фрагмента $\langle \_\_, t, r_1, a_3, a_1 \rangle$. И если последний имеется, то формируется $\langle \gamma_j, t, r_1, a_3, a_1 \rangle \circ [\gamma_j := \tilde{\lambda}]$.

Рассмотрим несколько содержательных примеров. На рис. 5.3 изображен активный граф, представляющий, что *Брат Ивана точно является его начальником*. *О*-вершина *б. нач.* соответствует отношению *быть начальником*, а *б. бр.— быть братом*. Предполагается, что собеседник знает, о каком *Иване* и каком *брате* идет речь. Соответственно в знаниях *Т* могут быть найдены значения *н*-вершин y_1^1 и y_2^1. Пусть после выполнения операций, задаваемых фрагментами графа рис. 5.3, оказалось $y_1^1 := a_1$, $y_2^1 := a_2$. Тогда проверяется наличие в *Т* фрагмента $\langle \gamma_\xi, t, б.\ нач, a_2, a_1 \rangle$. Если такового нет, т. е. не выполняется соотношение $[\beta_1 := \lambda]$, то фрагмент формируется. Следует еще раз обратить внимание, что фрагмент компонуется из *о*-вершин графа рис. 5.3 и значений *н*-вершин y_1^1 и y_2^1, которые находятся на базе *Т*. Это определяет привязку формируемой структуры к вершинам из *Т*, т. е. к тому, что было известно ранее. Сами *н*-вершины y_1^1 и y_2^1 не будут присутствовать в знаниях *Т*.

Условия формирования новых фрагментов. Прежде чем сформировать фрагмент и поместить его в знания, необходимо, чтобы было выполнено множество условий. Во-первых, не должно быть альтернатив. Если речь идет о единичных объектах, то они должны быть найдены, а если их оказалось несколько, то уточнены. Например, если у Ивана несколько братьев, то прежде всего необходимо узнать, о ком из них идет речь. В данном случае $н$-вершине y_1^1 будет присвоено множество значений, из которых нужно выделить один элемент — вершину. Она и будет принимать участие в формировании фрагмента. Для выделения необходима соответствующая системная активность — внешняя (выражающаяся в виде переспрашивания) или внутренняя (имеются в виду внутрисистемные действия для отделения допустимых значений от недопустимых, для чего могут быть использованы обязательные знания (см. § 5.3)).

Во-вторых, должны быть исчерпаны все возможности поиска, в том числе связанного с эквивалентными преобразованиями, элементами логического вывода. Поиск может быть нерезультативным, к примеру, по той причине, что в высказывании имеют место категории, отношения, которые в знаниях представлены другим способом, в другом базисе. Например, пусть речь идет о *друге тестя,* а в знаниях представлены лишь отношения типа *быть отцом, быть мужем* и др. Тогда требуется преобразование представлений. Если для какого-либо фрагмента графа отсутствует аналог (сопоставимый фрагмент) в знаниях, то необходимо попытаться заменить этот фрагмент на эквивалентные конструкции, используя их для поиска. И только если все возможности исчерпаны, следует делать вывод о наличии новой информации.

В-третьих, необходимо пропустить новый фрагмент через соответствующие семантические фильтры, т. е. узнать, не противоречит ли она тому, что имелось ранее. Например, пусть системе заведомо известно, что *Иван* (который имеется в виду) *является единственным ребенком в семье.* Тогда высказывание, что *друг его брата точно женат* должно приводить к более сложным действиям, связанным как с внешней активностью (переспрашиванием, утверждением, что такого не может быть, и др.), так и с внутренними действиями (изменением представлений, в том числе обобщенного характера). Подобного сорта обработка связана со специальными задачами, необходимыми с точки зрения построения систем активного диалога, см. [14]. В дальнейшем будут рассматриваться лишь некоторые из них.

И, наконец, в-четвертых, необходимо учитывать возможность интегрированных представлений. Напомним, что каждый комплексный объект или ситуация могут рассматриваться как единое целое. Тогда ему сопоставляется своя вершина. Ей является $с$-вершина, полученная в результате замыкания фрагментов (см. § 1.4), т. е. представляются части объекта или ситуации. Такие $с$-вершины будут находиться в знаниях. Каждый новый фрагмент должен быть правильным образом «упакован» в подобного сорта представления. Должно быть дополнительно представлено, что новые объекты или отношения являются частью какой-либо известной ситуации. Процедура замыкания как бы должна сработать заново — с учетом нового фрагмента.

Будем считать, что описанные действия входят в «компетенцию» оператора формирования. Если новый фрагмент вставляется в упакованную

сеть (т. е. имеющую замыкание), то автоматически осуществляется перезамыкание. Как это делается? Пусть для замыкания использовался фрагмент $\langle \gamma_i, t, \_\_, \ldots, \gamma_j, \ldots \rangle$, представляющий отношение *часть—целое*, γ_i — есть $с$-вершина сети, а γ_j соответствуют частям. Тогда просто увеличивается местность этого фрагмента, а на новые места помещаются $с$-вершины новых фрагментов. Таким способом представляются новые части. Еще раз отметим, что все это делается оператором формирования.

Разновидности активных графов. Будем различать два типа высказываний. Первый, когда в нем в явном виде не указана ситуация (или сценарий), к которому относится передаваемая информация. Такая ситуация должна быть каким-то образом выделена в процессе поиска, сопоставления. Именно такие высказывания в основном рассматривались выше. И второй тип, когда ситуация указана в явном виде с помощью отдельных слов типа *за грибами, на пикник* и др. (фиксируется сценарий). Подобное указание, естественно, должно быть представлено в активном графе. На рис. 5.4 изображен пример такого графа, где вершина a_3 соответствует комплексной ситуации, β_1 — есть $с$-вершина фрагмента, представляющего новую информацию. Имеется также рекурсивный фрагмент, представляющий, что *новое отношение* R_1 *с его объектами* A_1 *и* A_2 *относится к ситуации* A_3 (*является ее частью*). Граф рис. 5.4 задает операцию формирования фрагментов, представляющих не только отношение R_1, но и *часть—целое*. Причем последнее отношение может быть представлено путем увеличения местности фрагмента $\langle a_3, t, \_\_, \ldots \rangle$, имеющегося в знаниях. На вновь созданное место ставится β_1.

Перейдем к рассмотрению других разновидностей активных графов. Интересным является случай, когда описываются два объекта (Y_1 и Y_3) и специально указывается (акцентируется внимание), что *это одно и то же.* Например, говорится, что *у Ивана есть брат и имеется начальник. Так это один и тот же человек.* Предполагается, что ранее собеседник думал иначе. Подобная акцентация должна представляться в виде графа, задающего операции отождествления вершин (входящих в знания), их слияния. Для этого будем использовать фрагменты следующего вида: $\langle \beta_1, t, =, y_2^1, y_3^1 \rangle$. Пример графа с таким фрагментом изображен на рис. 5.5. Граф задает операцию нахождения значений $н$-вершин y_2^1 и y_3^1 и формирования фрагмента, представляющего равенство ($=$) таких значений. Пусть в процессе поиска было получено $y_2^1 := d_1$ и $y_3^1 := d_2$, где d_1 соответствует *начальнику Ивана,* а d_2 — *брату Ивана.* Тогда формируется $\langle \_\_, t, =, d_1, d_2 \rangle$. В общем случае такое формирование можно совместить с операцией слияния вершин d_1 и d_2 в одну — d_1. Все, что ранее было известно о начальнике и брате, объединяется и переносится на одного человека.

Заметим, что для представления необходимости отождествления объектов могут быть использованы и $з$-сети $[\square_ф y_2^1 := y_3^1]$. Тогда в случае применения графа вершина d_1 заменяется на резервную $н$-вершину x_ξ^1 и формируется $[x_\xi^1 := d_2]$. *Субъект (брат Ивана)* как бы уточняет, кто же *является начальником,* что представляется путем присвоения значений.

С помощью знаков $\square_ф$ обеспечивается представление достаточно сложных указаний. Остановимся на случае, когда отмечается единственность, уникальность объектов, о которых идет речь. Пусть $\Gamma_1(y_1)$ — граф, сопоставленный таким объектам. Тогда для представления того, что *имеется* (или

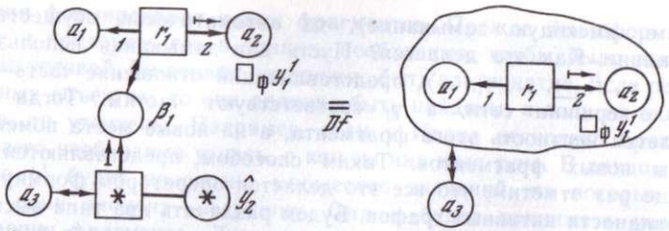


Рис. 5.4. Активный граф, представляющий, что *в рамках ситуации* A_3 *объект* A_1 *точно связан отношением* R_1 *с объектом* A_2

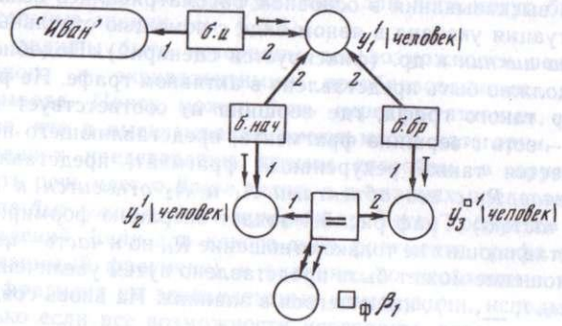


Рис. 5.5. Пример, иллюстрирующий представление, что *у Ивана есть только один брат и что начальник некоего Ивана и его брат — это одно и то же лицо*

должно иметься) *только один такой объект* Y_1, используется конструкция

$$\Gamma_1(y_1) \circ [\square_\phi \gamma_1 \perp \langle \gamma_1, t, \textit{кол}, y_1, \text{'1'} \rangle], \tag{5.8}$$

где *кол* — иметь количество (см. § 1.3). Последний фрагмент задает свои собственные проверки и действия (связанные с внутренней и внешней активностью), которые выполняются по ходу поисковых операций, задаваемых $\Gamma_1(y_1)$. Будем обозначать вершину y_1, входящую в данный фрагмент, через $y_1^{\square 1}$. При этом сам фрагмент будем опускать, подразумевая, что он всегда может быть восстановлен по вершинам типа $y_1^{\square 1}$. В соответствии со сказанным, конструкция (5.8) записывается в более простом виде — $\Gamma_1(y_1^{\square 1})$. Например, граф

$$[\square_\phi y_1^{\square 1} := a_2] \circ [y_1^1 \perp \langle \_\_, t, r_1, y_1^1, a_1 \rangle \tag{5.9}$$

представляет, что *отношением* R_1 *с* A_1 *связан только один объект* — A_2. Все остальные связи должны быть объявлены несуществующими или разрушены.

В каждом активном графе может быть не один, а несколько символов типа $\square_\phi$. С помощью подобных графов представляется информация, имеющая множественную акцентацию. Предполагается множество проверок, которые выполняются в процессе поиска. Например, на рис. 5.5 изображен граф, представляющий основную акцентацию, связанную с отождествлением субъектов. В то же время акцентируется внимание и на уникальность

объекта Y_3 — *брата Ивана*. Если системе известно множество таких братьев, то предполагается активность, связанная с переспрашиванием или проверкой допустимости.

Нечеткость в восприятии акцентации, новизны. Ранее рассматривался в некотором смысле идеальный случай, когда в высказывательной форме четко указывается, что является новым и что нужно изменять. И такие указания в определенном смысле согласованы со знаниями собеседника, т. е. кому адресовано высказывание. То, что подразумевается, он на самом деле знает, а то, на чем акцентируется внимание, для него является новым. Например, при восприятии высказывания, представленного на рис. 5.3, собеседнику было ясно, о каком *Иване* и *его брате* идет речь. Соответственно y_1^1 и y_2^1 получили непустые значения. Новым являлось отношение *быть начальником*.

Подобный случай является сравнительно редким. Значительно чаще упомянутой согласованности не удается добиться. В какой-то степени остается неопределенным, что же может быть новым. Например, возьмем высказывание *друг брата Ивана женат*. Акцент сделан на свойство *быть женатым*. Однако собеседнику ничего может быть не известно, что *у Ивана есть брат*, а *у того есть друг*. И то и другое может быть для него новым. Каждое высказывание может нести сразу много нового, требующего изменения представлений. Если заведомо известно, что все в высказывании достоверно, то оно должно быть представлено с помощью графа, у которого к каждой *н*-вершине y_j добавляется $[\square_\phi y_i := \lambda]$ и все *з*-сети имеют вид $[\square_\phi y_j := \eta]$. Словом, должны выполняться все проверки. На основе материала знаний нужно попытаться конкретизировать весь граф (заполнить клише), а если это не так, то доформировать недостающие структуры (дополнить клише, выделив полученное таким способом изделие) и поместить их (его) в знания.

Будем называть подобные графы также а к т и в н ы м и, обозначая их в виде $\square_\phi \Gamma_i$. При изображении будем ставить знак $\square_\phi$ впереди *с*-вершины сети T_i (последняя получается из Γ_i изъятием стрелок порядка). Допускается, что такие стрелки могут быть расставлены неправильно. Их можно вовсе не учитывать, что записывается в виде $\square_\phi T_i$.

Итак, графы $\square_\phi \Gamma_i$ и сети $\square_\phi T_i$ будут соответствовать высказываниям, автор которых плохо себе представляет, что знает или не знает его собеседник. Он акцентирует внимание на одном, а собеседник (система) в качестве нового может воспринимать совершенно другое. Этому случаю соответствует режим работы системы, связанный с постоянным изменением направления поиска (см. § 3.2). Если системе что-либо не известно, то делается попытка уточнить компоненты неизвестной части, пользуясь всей совокупностью передаваемых сведений, в том числе и тем, на чем акцентируется внимание.

Например, возьмем высказывание *Друг брата Ивана — чемпион края по толканию ядра*. Пусть известно, о каком брате идет речь, но ничего неизвестно про наличие у него друга. Тогда делается попытка найти этого друга, пользуясь знаниями о чемпионах. И если таковой может быть найден (он известен), то принимается к сведению новая информация. Фактически осуществляется переиначивание высказывания в другое — *Другом брата Ивана является чемпион края по толканию ядра*. Изменяется направление уточнения неопределенных компонент.

Процедура изменения направления поиска при внутрисистемной обработке реализуется следующим образом. Пусть в графе Γ_i оказался фрагмент, у которого не найдено аналогов (сопоставимых фрагментов) в знаниях T. Соответственно у одной или нескольких его $н$-вершин оказалось пустое множество значений. Тогда выделяется данный фрагмент, его $н$-вершины, а на других фрагментах графа стрелки порядка переформировываются таким образом, чтоб они были направлены к этим $н$-вершинам. При этом вовлекаются все новые фрагменты. И когда будут найдены значения всех $н$-вершин первого фрагмента, то на базе них в знаниях T формируется новый фрагмент. Если же значения каких-либо $н$-вершин оказались все еще пустыми, то им присваиваются «резервные» значения, соответствующие новым объектам. Система как бы сама в зависимости от результатов поиска просматривает варианты расстановки знаков $\square_ф$, выбирая наиболее подходящий и действуя в соответствии с ним. По ходу действий формируются вершины и фрагменты, соответствующие новым объектам и отношениям.

Описанная методика является наиболее гибкой в смысле восприятия новой информации. Она может быть распространена на случай, когда в высказывании упоминается о множестве новых объектов и фактов. Последние представляются с помощью фрагментов графа. Для них будут отсутствовать сопоставимые фрагменты в знаниях T. Поиск значений их $н$-вершин осуществляется аналогичным образом. Конечно, по ходу поисковых операций должна организовываться и системная активность. Обратимся к предыдущему примеру. Если чемпионами в разные годы были различные люди и неясно, о ком же из них идет речь, то можно просто переспросить или же чисто логически постараться выделить, кто мог быть другом, а кто нет. Как уже говорилось, в процессе такого выделения большую роль играют обобщенные знания о классах объектов — людях (см. гл. 7). Естественно, требуются и конкретные знания о возрасте, характере, местожительстве и др.

Факультативные вершины. Если высказывание достоверно, то все, что в нем упоминается, имеет место, существует. Именно этот случай рассматривался ранее. Если какая-либо $н$-вершина, к которой направлена стрелка $\mapsto$, не была отмечена знаком $\square_ф$, то такой знак просто предполагался (см. рис. 5.3 и 5.5). Однако в некоторых высказываниях могут иметь место оттенки сомнения, для представления которых будем использовать знак $\Diamond$. Отмеченные этим знаком $н$-вершины (будем ставить $\Diamond$ впереди них) назовем ф а к у л ь т а т и в н ы м и. Они соответствуют тому, что *может быть*. Варианты возможного будем представлять с помощью $з$-сетей. Например, $з$-сеть $[\Diamond y|: = \{a_1, a_2\}]$ будет означать *Возможно, объектом* Y_1 (о котором идет речь) *является* A_1 *или* A_2, а $з$-сеть $[\Diamond y|: = \lambda]$ — *Возможно, имеется такой объект* Y_1. Подобные $з$-сети будем использовать в композиции с графами. Например, граф

$$[\Diamond y|: = a_2] \circ [y| \perp \langle \_, t, r_1, a_1, y| \rangle] \qquad (5.10)$$

означает, что *отношением* R_1 *с объектом* A_1 *связан объект, которым, возможно* ($\Diamond$), *является* A_2.

Будем допускать активные графы, в которых одни $н$-вершины отмечены знаками $\square_ф$, а другие — $\Diamond$. Например, на рис. 5.6 изображен граф, представляющий, что, *Возможно, у Ивана есть брат, (но если это так,*

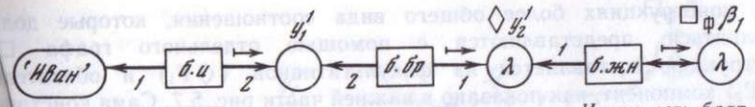


Рис. 5.6. Активный граф, представляющий, что, *возможно, у Ивана есть брат, тогда он женат*

то) последний точно женат. В скобках записано то, что дается по умолчанию. *О*-вершина *б. жн* графа соответствует свойству *быть женатым*. Вершина $\Diamond y_2^1$, соответствующая *брату*, является факультативной, а вершина $\square_ф \beta_1$ — обязательной. Вершина y_1^1, соответствующая *Ивану*, никак не отмечена. В случае представления высказывательных форм такие вершины являются обязательными, т. е. перед ними должен быть проставлен знак $\square_ф$.

Будем считать, что граф с факультативными вершинами задает те же операции, что и рассмотренные ранее активные графы, лишь с небольшим отличием. Если не найдено значений факультативных вершин, то никакого формирования не производится и последующие операции не выполняются. Например, если при выполнении операций, задаваемых графом рис. 5.6, не было найдено значений y_2^1 (о брате Ивана ничего не известно), то действия заканчиваются. Свойство *быть женатым* не анализируется. Соответствующий фрагмент не ищется в знаниях.

Рассмотрим еще один пример. Проставим на рис. 5.3 знаки $\Diamond$ перед $н$-вершинами y_1^1 и y_2^1. Тогда уже будет представлено следующее: *Возможно, что имеется Иван, и возможно, что у него есть брат. Тогда последний точно является начальником Ивана.*

Бросается в глаза некоторая искусственность примеров. Высказывательные формы, в которых в чем-то выражается сомнение, а что-то известно точно (и второе определяется через первое), крайне редки. Соответствующие графы нам будут нужны в основном для других целей — для представления специального вида знаний, называемых обязательными (см. § 5.2). При использовании таких знаний предполагается, что что-то может быть, а что-то может и не быть. Но если есть, то это должно приводить к соответствующей внутрисистемной активности. Об этом будет говориться чуть ниже.

Обозначим граф, у которого все $н$-вершины являются факультативными, через $\Diamond\Gamma_1$. Стрелки порядка направлены только к вершинам, отмеченным знаком $\Diamond$. Во всем выражается сомнение. Любой объект может быть а может и не быть. Будем называть г р а ф $\Diamond\Gamma_1$ ф а к у л ь т а т и в н ы м. Факультативные графы будут служить как компоненты более сложных конструкций. Рассмотрим наиболее типовую из них.

Пусть $\Diamond\Gamma_1(y_1)$ — граф, у которого имеется только одна не факультативная $н$-вершина y_1. Тогда с помощью

$$[\square_ф y_1: = \eta] \circ [y_1 \perp \Diamond\Gamma_1(y_1)] \qquad (5.11)$$

представляется оттенок *возможности соотношений типа* $\mathcal{T}_1$, указывается, как выявлять их. Но если соотношения имеют место, то *должны выполняться другие соотношения,* в данном случае представленные в виде $[\square_ф y_1: = \eta]$. Конструкция (5.11) изображается, как показано в верхней части рис. 5.7. Собственно она и иллюстрировалась на предыдущих примерах, в частности см. рис. 5.4.

В конструкциях более общего вида соотношения, которые должны выполняться, представляются с помощью отдельного графа $\square_\phi \Gamma_2$. Конструкция составляется из факультативной ($\Diamond \Gamma_1$) и обязательной ($\square_\phi \Gamma_2$) компонент, как показано в нижней части рис. 5.7. Сама конструкция записывается в виде $\Diamond \Gamma_1 \circ \square_\phi \Gamma_2$. При этом графы $\Diamond \Gamma_1$ и $\square_\phi \Gamma_2$ могут быть связаны через набор факультативных вершин $\Diamond y_1^1, \ldots, \Diamond y_n^1$.

Условные зависимости, графы. Нетрудно видеть, что высказывания, для представления которых были использованы факультативные вершины, очень близки к условным формам, фактически выражают одно и то же. Например, сказать *Возможно у Ивана есть брат, последний женат* — одно и то же, что *Если у Ивана есть брат, то последний точно женат*. Как и в предыдущем случае, предполагается, что брат может быть (о нем может быть что-либо известно), а может и не быть.

Для представления условных зависимостей в § 2.1 были введены специальные вершины $\Rightarrow$, соответствующие *логической импликации*, и фрагменты. Будем называть активные графы с подобными фрагментами условными. На рис. 5.8 изображен условный граф наиболее общего вида. С помощью него представляется случай, когда *факты наличия одних соотношений* $\mathcal{T}_1$ *связываются* ($\Rightarrow$) *с фактами наличия других* — $\mathcal{T}_2$. При этом p_1 и p_2 являются так называемыми p-вершинами (см. § 1.4). Они сопоставляются логическим значениям высказываний о соотношениях $\mathcal{T}_1$ и $\mathcal{T}_2$.

Граф Γ_1 задает способ уточнения компонент из $\mathcal{T}_1$ (нахождения значений $н$-вершин y_i^1), а также способ оценки факта истинности или ложности того, что касается соотношений $\mathcal{T}_1$ (вершине p_1 присваивается значение t или f). Естественно, такая оценка возможна только в случае, если найдены непустые значения y_i^1. Граф $\square_\phi \Gamma_2$ определяет способ проверки того, что (как указано в высказывании) должно иметь место, а если проверки не выполнимы, то задаются операции формирования соответствующих фрагментов. Ясно, что проверки с формированием имеют смысл только при условии истинности соотношений $\mathcal{T}_2$, представлен-

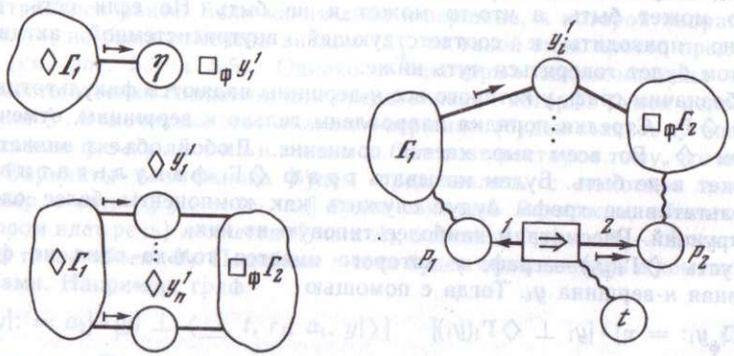


Рис. 5.7. Графы с факультативными и обязательными вершинами

Рис. 5.8. Условный граф, представляющий, что *если имеют место соотношения* $\mathcal{T}_1$, *то должны иметь место и соотношения* $\mathcal{T}_2$

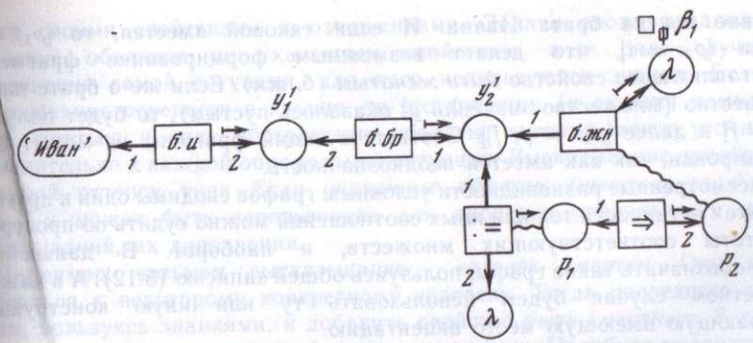


Рис. 5.9. Условный граф, представляющий, что *если у Ивана есть брат, то последний, точно, женат*

ных с помощью Γ_2, т. е. когда $[p_2 := t]$. А это возможно лишь при $[p_1 := t]$. Подобная зависимость реализуется путем выполнения операций, задаваемых подграфом $p_2 \perp \langle \_\_, t, \Rightarrow, p_1, p_2, t \rangle$, над сетью, представляющей *таблицы истинности логических связок* (см. § 4.2). Упомянутый подграф входит в конструкцию рис. 5.8, которую будем записывать в следующем виде:

$$\Gamma_1 \Rightarrow \square_\phi \Gamma_2. \qquad (5.12)$$

Будем допускать возможность проставления знаков $\square_\phi$ перед p-вершинами, соответствующими фактами истинности или ложности соотношений. Например, на рис. 5.8 знак $\square_\phi$ может быть поставлен перед p-вершиной p_2. Тогда будет представлено, что *должны иметь место все соотношения, входящие в* $\mathcal{T}_2$.

Заметим, что частным видом рассмотренных условных графов является конструкция $T_1 \Rightarrow T_2$, введенная в § 1.4 и названная логической продукцией. Она может быть получена из графа рис. 5.8 изъятием всех стрелок порядка. В результате будет представлена информация без какого-либо указания направления поиска.

Ранее рассматривались принципы представления высказываний, выражающих связь ($\Rightarrow$) фактов наличия соотношений между объектами ПО. В общем случае в высказываниях может говориться о фактах существования объектов, указываться, каким может быть или должен быть тот или иной объект. Подобные указания могут связываться в **сложные условные зависимости.** Для их представления требуются графы, содержащие $з$-сети и задающие соответствующие операции проверки. С результатом выполнения последних могут быть связаны другие факты или же действия выполнения других проверок. Для представления подобных зависимостей будем использовать $з$-сети, которые будем записывать в «расшифрованном» виде, т. е. в виде фрагментов с p-вершинами.

Рассмотрим вначале графы, представляющие *факт наличия (истинности) соотношений* $\mathcal{T}_2$ *при условии существования указанных или специальным образом выделенных объектов.* Пример такого графа изображен на рис. 5.9. С помощью него представляется, что *если у Ивана есть брат* (что-либо известно о нем), *то этот брат женат.* Граф задает

действие поиска брата Ивана. И если таковой имеется, то $[p_1: = t]$, откуда $[p_2: = t]$, что делает возможным формирование фрагмента, представляющего свойство *быть женатым (б. жн)*. Если же о брате ничего не известно (множество значений y_2^1 оказалось пустым), то будет получено $[p_1: = f]$ и далее $[p_2: = \{t, f\}]$. Соответствующий фрагмент не может быть сформирован, так как имеется неоднозначность.

Рассмотренные разновидности условных графов сводимы один к другому. О фактах истинности тех или иных соотношений можно судить по проверкам непустоты соответствующих множеств, и наоборот. В дальнейшем будем обозначать такие графы, пользуясь общей записью (5.12). А в каждом конкретном случае будем использовать ту или иную конструкцию, отражающую имеющую место акцентацию.

В заключение параграфа отметим, что возможность различным способом, с помощью различных категорий и высказывательных форм выражать одно и то же является важным свойством ЕЯ. Такое свойство может быть учтено с помощью языка семантических графов, которые очень чувствительны к способам выражения. Формам, основанным на различных категориях, конструкциях, но выражающих одно и то же, сопоставляются различные семантические графы, но задающим эквивалентные операции — с одинаковыми результатами. Если же формы имеют оттенки, с помощью которых делается отличной выражаемая ими суть, то соответствующие графы будут задавать различные операции — с различными результатами.

Обратимся к условному высказыванию, представленному на рис. 5.9. Предполагается, что брата может и не быть. Вершины y_2^1, β_1 могут не иметь значений. Сравните с другим высказыванием: *Брат Ивана женат*. Это уже безусловная конструкция, несущая свой операционный оттенок. Предполагается наличие брата. Все вершины соответствующего графа должны получить непустые значения. Если этого не произошло, то нужно что-то предпринимать. Условия выполнения операций и последующие действия сильно отличаются.

§ 5.2. ОБЯЗАТЕЛЬНЫЕ ЗНАНИЯ ПРОСТОГО ВИДА

В данном параграфе будут рассматриваться семантические графы, представляющие достоверную информацию и образующие внутрисистемные знания специального типа — «обязательные». Будут описаны принципы их использования для проверки правильности входных высказываний, для ответа на запросы и др. При этом речь будет идти о знаниях сравнительно простого вида, т. е. представляющих «бескванторную» информацию (предполагается отсутствие слов-кванторов типа *каждый, только, некоторые* и др.).

О переносе высказываний в знания. Как указывалось в предыдущем параграфе, долженствование может относиться не только к знаниям (что *следует знать*), но и ко входной информации (что в ней *должно быть*). Если высказывание заведомо достоверно, то оно может быть перенесено в знания. При этом если не уточнять компоненты высказывания, то при переносе изменяется его акцент, становится возможным использование этого высказывания для проверки других высказываний.

Пусть на вход системы поступило высказывание, касающееся определенной ситуации. В нем упоминаются конкретные объекты, характеризующиеся своими свойствами и отношениями. Задача ввода предполагает уточнение объектов по указанным свойствам, а в ряде случаев — и уточнение самой ситуации, для чего используются знания. Если же высказывание перенести в знания, то его функции (и акцент) изменятся. Будут иметься в виду абстрактные объекты, которые могут уточняться применительно к каждой описываемой ситуации. Высказывание приобретает условный оттенок типа *Если указанные объекты имеются, то должно быть...* и может быть использовано для проверки правильности других высказываний, их дополнения.

Например, возьмем высказывание *человек смертен*. Оно может относиться к некоторому конкретному человеку. Тогда последнего нужно найти, пользуясь знаниями, и добавить свойство *быть смертным*. В другом варианте речь может идти об абстрактном понятии. Подобное высказывание и начинает играть роль специальных знаний. При этом приобретается условный оттенок типа *Если говорится о каком-либо человеке, то к нему может быть добавлено свойство «быть смертным»*. Обратите внимание, что здесь нет слов-кванторов типа *каждый*.

Возьмем другое высказывание: *Брат Ивана женат*. Если перенести его в знания, не конкретизировав, о каком Иване и брате идет речь, то соответствующие понятия становятся абстрактными. Высказывание приобретает оттенок *Если имеется некий Иван и у него есть брат, то последний женат*. Подобного сорта высказывания могут быть использованы для уточнения и пополнения других высказываний.

В дальнейшем будем допускать возможность переноса активных графов, представляющих те или иные высказывания, во внутрисистемные знания. Тогда операции, задаваемые этими графами, могут выполняться над любой сетью (или графом), представляющей какое-либо другое высказывание — входное или уже перенесенное в знания. Таким образом обеспечивается анализ и пополнение сети.

Обязательные графы. Особенности задаваемых операций. Будем называть активные графы, перенесенные в знания, о б я з а т е л ь н ы м и. Имеется в виду обязательность (для других структур) представленных в них соотношений и задаваемых ими проверок. Как уже говорилось, при переносе активного графа происходит некоторая его модификация. Изменяются оттенки представленной информации и задаваемые операции. В чем же заключаются такие изменения, чем отличаются активные и обязательные графы?

Во-первых, в обязательном графе все вершины, не отмеченные знаком $\square_\phi$, следует пометить знаком $\Diamond$. Вершины, которые не являются обязательными, делаются факультативными, т. е. несущими условный оттенок. Представляется уже не то, что *есть* (в конкретной описываемой ситуации), а то, что *может быть, а может и не быть*, (в зависимости от того, с какой ситуацией соотносится высказывание). То же самое относится и к фрагментам, если специальным образом не отмечена их обязательность. Напомним, что обязательность отношений представляется с помощью $[\square_\phi \beta_i: = \lambda]$, где β_1 — c-вершины соответствующих фрагментов. Во всех других случаях отношения будут подразумеваться, носить условный оттенок.

Во-вторых, операции, задаваемые обязательным графом, могут выполняться многократно. Допускается многократное применение графа

как к различным (входным) сетям, так и к одной и той же сети. Дело в том, что с помощью графа может указываться на обязательность соотношений, имеющих место для единичных объектов, а с помощью сети — представляться множества объектов. Чтобы обеспечить пополнение свойств у каждого объекта такого множества и требуется многократное применение графа.

И в-третьих, **оператор формирования** фрагментов, который задается обязательным графом, несколько изменяет свои функции. Результатом выполнения оператора, задаваемого активным графом (см. § 5.1), было изменение сети-знания. В ней формировались новые *о*-вершины и фрагменты. Более того, если в знаниях не было какой-либо из «обязательных» вершин, то она формировалась со своим фрагментом, т. е. из обязательности наличия той или иной новой компоненты следовала обязательность всех ее отношений. При использовании обязательных графов вносится явно выраженный условный оттенок, о чем говорилось ранее. Из обязательности наличия объекта может и не следовать обязательность его соотношений. Последние могут составлять условие.

Далее, следует учитывать тот факт, что в ряде случаев формирование фрагмента с представлением новой информации может быть сведено к присвоению значений. Например, пусть во входном сообщении говорится о *некоем человеке со свойством* $\mathcal{T}_1$, что представлено в виде сети $\mathcal{T}_1(x_1^1)$. Пусть на основе обязательных знаний было получено, что *таким человеком является (должен быть) субъект* A_1. Подобная информация может быть представлена путем формирования и добавления *з*-сети. В результате получится $T_1(x_1^1) \circ [x_1^1 := a_1]$. Будем считать, что такие действия осуществляются оператором формирования, задаваемого обязательным графом.

К примеру, пусть $\Gamma_1(y_1^1)$ — обязательный граф, с фокусом y_1^1 (к нему сходятся стрелки порядка). Тогда конструкция

$$\Gamma_1(y_1^1) \circ [\square_\phi y_1^1 := \eta] \tag{5.13}$$

задает операции нахождения значений *н*-вершины y_1^1. Эти операции выполняются над сетью T, представляющей какое-либо высказывание или являющейся фрагментом знания. Если при выполнении оказалось $y_1^1 := x_k^1$, т. е. значением y_1^1 является *н*-вершина из T, то формируется *з*-сеть $[x_k^1 := \eta]$, которая добавляется к T. Если же множество значений y_1^1 оказалось пустым, т. е. $y_1^1 := \tilde{\lambda}$, то граф считается неприменимым и никакого формирования не осуществляется. В этом отличие рассмотренных действий от тех, которые задавались конструкцией (5.6). Отметим, что формирование новых фрагментов в T задается только обязательными графами вида

$$\Gamma_1(\beta_1) \circ [\square_\phi \beta_1 := \lambda], \tag{5.14}$$

где β_1 — *с*-вершина какого-либо фрагмента из Γ_1. Собственно на базе него и формируется новый фрагмент в T, как это было описано в § 5.1.

С помощью оператора формирования учитывается и возможность интегрированных представлений, о чем говорилось в § 5.1. Каждое входное высказывание, описываемая ситуация или фрагмент знаний может рассматриваться как единое целое. Последнему сопоставляется отдельная вершина, т. е. *с*-вершина. При пополнении высказывания, например, путем указания на новый объект, должно дополнительно представляться,

что последний является частью этого единого целого. Необходимо перезамыкание (см. § 1.4). Будем считать, что подобного сорта действия входят в компетенцию указанного оператора.

Функции обязательных знаний. Активные графы, перенесенные в знания, были названы обязательными. Они соответствуют достоверным высказываниям. Множество обязательных графов образуют знания, которые также будем называть о б я з а т е л ь н ы м и. Представляются достоверные высказывания, с помощью которых можно изменять, пополнять другие высказывания.

Важная роль обязательных графов и знаний — служить в качестве с е м а н т и ч е с к и х ф и л ь т р о в, пропускающих через себя информацию, указывающих на их правильность, допустимость, а если нужно, то дополняющих. При этом от места знака $\square_\phi$ в графе зависит, что может или должно быть дополнено. Такими знаками отмечаются вершины и фрагменты, соответствующие недостающим частям, которые, например, подразумеваются во входной информации, как бы даются по умолчанию. Система должна уметь их восстанавливать, что осуществляется выполнением операций, задаваемых графами. Другие вершины и фрагменты последних представляют условную часть. Операции формирования и пополнения выполняются, только если входная информация удовлетворяет указанному условию. При этом в процессе пополнения должна учитываться возможность интегрированных представлений.

Другая не менее важная роль обязательных знаний — обеспечивать выбор пригодных вариантов в задачах ответа на запросы, уточнения высказываний. Пусть запрос представлен с помощью сети $T_{вх}$. Тогда путем выполнения над $T_{вх}$ операций, задаваемых тем или иным обязательным графом, находятся значения *н*-вершин, т. е. фиксируются *з*-сети, которые добавляются к $T_{вх}$. Информация запроса как бы уточняется «насильственным» образом. Для этого используются знания о том, что *должно быть*, а чего *не может быть*. Отсеиваются неприемлемые варианты. Остаются только те, которые удовлетворяют обязательным знаниям.

Еще одна из функций обязательных знаний связана с **оценкой истинности** или ложности высказываний. С помощью обязательных графов могут представляться заведомо истинные высказывания — аксиомы, доказанные утверждения, теоремы. Задаваемые ими операции выполняются над сетями или графами, представляющими другие высказывания — недоказанные утверждения, теоремы, которые нужно доказать. В результате *н*-вершинам, соответствующим логическим составляющим, как бы насильственным образом присваиваются значения (t, f). Таким способом может быть четко установлен факт истинности или ложности высказываний. При этом по ходу дела могут уточняться и другие компоненты.

Наконец, обязательные графы могут служить для «размножения» вершин, необходимого для использования некоторых видов обобщенной информации. Размножением обеспечивается перенесение свойств классов на отдельные элементы и т. д. (см. § 5.1).

Проиллюстрируем некоторые из описанных функций на примерах. На рис. 5.10 изображен обязательный граф, представляющий *Человек смертен*. *Н*-вершина y_1^1 соответствует некоему *абстрактному человеку*, который может быть конкретизирован для каждой конкретной ситуации в отдель-

ности. Это осуществляется выполнением операций, задаваемых графом, над соответствующей сетью. Если оказалось, что значения y_1^1 не пусты, т. е. при описании ситуации упоминается какой-либо человек, то формируется фрагмент, представляющий свойство *быть смертным* (*б. см*). Если *н*-вершина y_1^1 получила множество значений (в описываемой ситуации принимает участие множество людей), то выбирается одно из них и формируется соответствующий фрагмент. При повторном применении графа будет выбрано другое значение и т. д. Так, всем людям может быть приписано свойство *быть смертным*. Если значения *н*-вершины y_1^1 оказались пустыми, то выполнение последующих операций просто прекращается. Перед y_1^1 подразумевается знак $\Diamond$, который всегда может быть проставлен.

Обязательный граф рис. 5.10 может быть использован для ответа на запросы типа *Обладает ли некий конкретный человек* (*например, Сократ*) *свойством смертности*? Путем выполнения операций, задаваемых графом, над сетью $T_{вх}$, представляющей запрос, обеспечивается насильственное формирование значений *н*-вершин из $T_{вх}$, в том числе *н*-вершины $[\gamma_i{:=}?]$. Будет получено $[\beta_1{:=}\gamma_i]$, что вызовет $[\gamma_i{:=}\lambda]$, т. е. ответом будет *обладает, существует такое свойство.*

Заметим, что если спрашивается не о наличии или отсутствии чего-либо, а об истинности или ложности, то для получения ответа требуются обязательные графы несколько другого типа. Знаком $\square_\phi$ должна быть отмечена *р*-вершина p_1 (см. рис. 5.10). К ней должна быть направлена стрелка порядка и указано значение *t*. Полученный таким способом граф записывается следующим образом:

$$[\square_\phi p_1{:=}t] \circ [p_1 \perp \langle \_\_, p_1, \textit{б. см}, y_1^1 \rangle] \circ [y_1^1 \perp \langle \_\_, t, \in, y_1^1, \text{'кл. людей'} \rangle]. \quad (5.15)$$

С помощью него обеспечивается уточнение логических значений высказываний.

Рассмотрим еще два примера. На рис. 5.11 изображен обязательный граф, представляющий, что *если речь идет о некоем человеке и некоем*

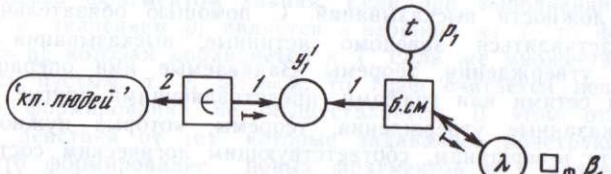


Рис. 5.10. Обязательный граф, представляющий *Человек смертен*

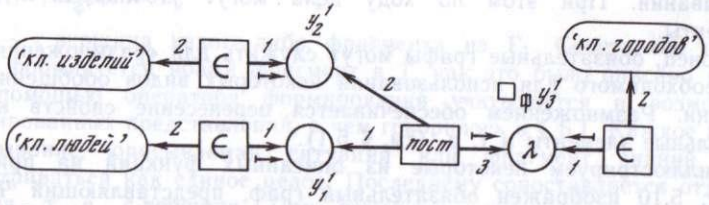


Рис. 5.11. Обязательный граф, представляющий, что *если имеется некий человек* Y_1, *поставляющий некое изделие* Y_2, *то должен быть город* Y_3, *куда он поставляет*

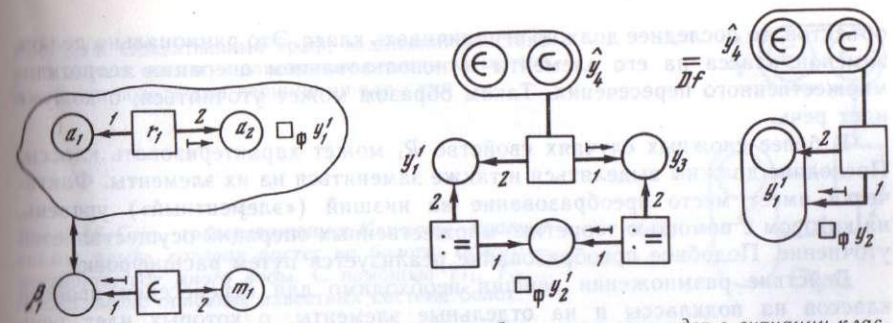


Рис. 5.12. Обязательный граф, представляющий, что *если речь идет о ситуации класса* M_1, *то с объектом* A_1 *отношением* R_1 *должен быть связан именно объект* A_2

Рис. 5.13. Обязательный граф, задающий действие р а с ш и ф р о в к и — добавления к вершинам-классам вершин-подклассов и вершин-элементов. В правой части приведено сокращенное изображение

изделии и говорится, что человек поставляет это изделие, то должен иметься и город, куда он поставляет. *Н*-вершины y_1^1 и y_2^1 (по умолчанию) являются факультативными. Самому городу сопоставлена вершина y_3^1, которая $[\square_\phi y_3^1{:=}\lambda]$. Если заменить данную *з*-сеть на $[\square_\phi\ y_3^{\square 1}{:=}\lambda]$, то дополнительно будет представлено, что *должен иметься только один город.* Операции, задаваемые графом рис. 5.11, могут быть выполнены над любой сетью, представляющей описание каких-либо поставок. В результате обеспечивается его пополнение — сведениями о наличии города поставки.

С помощью обязательных графов могут представляться достоверные высказывания, относящиеся лишь к информации определенного вида или к ситуации определенного класса. На рис. 5.12 показан пример. Изображен обязательный граф, представляющий, что *в ситуации класса* M_1 *при наличии объекта* A_1 *должен быть и объект* A_2, *связанный с первым отношением* R_1. Операции, задаваемые этим графом, выполняются над сетью, представляющей описываемую ситуацию. И если выполнены упомянутые условия (*н*-вершине β_1 будет присвоено значение, которым является *с*-вершина, соответствующая ситуации), то осуществляется действие формирования.

Действия расшифровки и размножения. Они необходимы для обеспечения результативного поиска в случае, когда в знаниях представлены свойства и отношения классов объектов. Напомним, что для представления высказываний о классах могут использоваться как активные, так и пассивные структуры (см. § 3.2). Например, рассмотренное ранее высказывание *человек смертен* может быть представлено в виде пассивной структуры, т. е. фрагмента $\langle \_\_, t, \textit{б. см}, \text{'кл. людей'} \rangle$. Использование таких структур предполагает упомянутые действия.

Действие расшифровки заключается в замене класса на его элементы, что на уровне внутрисистемных представлений сводится к замене вершин. Такая замена необходима в тех случаях, когда допускается уточнение неопределенных компонент указанием на классы. Например, пусть спрашивается *Кто смертен*? В качестве ответа может быть указан весь класс людей. Пусть речь идет о некоторых смертных существах с дополнительным свойством R_1. Если по свойству R_1 может быть выделено множество

объектов, то последнее должно ограничивать класс. Это рационально делать заменой класса на его элементы и использованием операции теоретико-множественного пересечения. Таким образом может уточняться, о ком же идет речь.

В более сложных случаях свойство R_1 может характеризовать классы. Последние должны выделяться и также заменяться на их элементы. Фактически имеет место преобразование на низший («элементный») уровень, на котором с помощью теоретико-множественных операций осуществляется уточнение. Подобное преобразование реализуется путем расшифровки.

Действие размножения вершин необходимо для перенесения свойств классов на подклассы и на отдельные элементы, о которых идет речь в запросах или сообщениях. В § 4.2 указывалось, что в процессе внутрисистемной обработки это может быть сделано заменой *о*-вершин сети, представляющей запрос или сообщение, на множества: *о*-вершина, соответствующая конкретному объекту, заменяется на множество, в которое входит она сама, и на *о*-вершины, соответствующие ее классам. Например, если говорится о некоем *Иване*, то рассматривается уже случай *Иван — мужчина* или *Иван — мужчина — человек*. Все, что справедливо для класса мужчин и класса людей, делается справедливым и для Ивана. Подобное распространение свойств, которое в логике реализуется с помощью правила индивидуализации, может осуществляться размножением вершин.

Действия расшифровки и размножения «поддерживаются» системными знаниями, имеющими вид обязательных графов. Рассмотрим вначале обязательный граф, задающий **операции расшифровки.** Он изображен на рис. 5.13. В нем имеется рекурсивный фрагмент, который необходим для использования цепочек родовидовых связей ($\in$, $\subset$), объединенных в иерархические структуры. Пусть такие связи представлены с помощью сети $T_{рв}$. Другая сеть — $T_{вх}$ представляет какой-либо запрос или высказывание. Тогда расшифровка уточняемых компонент осуществляется путем применения графа рис. 5.13 к композиции $T_{вх} \circ T_{рв}$. И если в $T_{вх}$ имеется хотя бы одна вершина m_i, соответствующая классу (т. е. входящая в фрагменты $\langle \_\_, t, \in, d_j, m_i \rangle$ или $\langle \_\_, t, \subset, d_j, m_i \rangle$), то граф делается применимым. В результате в $T_{рв}$ будут найдены все вершины типа d_j, а в $T_{вх}$ на базе m_i будет сформирована резервная *н*-вершина x_ξ со значениями $x_\xi := \{..., d_j, ..., m_i\}$. Вершина m_i окажется среди множества других значений x_ξ. Если m_i уже была среди множества значений какой-либо *н*-вершины (x_k), к примеру, она представляла результат уточнения какого-либо объекта, то формирование резервной *н*-вершины x_ξ не требуется. Просто значения x_k будут дополнены вершинами-элементами или вершинами-подклассами типа d_j.

Рассмотрим пример. Пусть требуется поиск *объектов* x_1, *обладающих как свойством* R_1, *так и* R_2. В процессе поиска оказалось, что свойством R_1 обладает объект A_1 и все объекты класса M_1, а свойством R_2 — объекты A_2 и A_3, т. е. были получены значения $[x_1 := \{a_1, m_1\}]$, $[x_1^* := \{a_2, a_3\}]$. Тогда попытка выделения общего элемента будет безуспешной: $\{a_1, m_1\} \cap \{a_2, a_3\} = \varnothing$. В этом случае возникает необходимость использования родовидовых связей, что обеспечивается графом рис. 5.13. Пусть в знаниях $T_{рв}$ представлено $\langle \_\_, t, \in, a_3, m_1 \rangle \circ \langle \_\_, t, \in, a_4, m_1 \rangle$, т. е. *класс* M_1 *состоит всего из двух элементов*. Тогда выполнением операций, задаваемых графом, будет получено $[x_1 := \{a_1, a_3, a_4\}]$. В результате $\{a_1, a_3, a_4\} \cap \{a_2, a_3\} = a_3$,

Рис. 5.14. Обязательный граф, задающий действие размножения — дополнения вершин-элементов и вершин-подклассов вершинами-классами

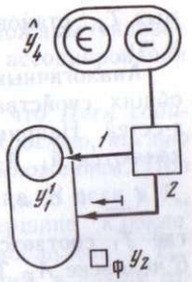
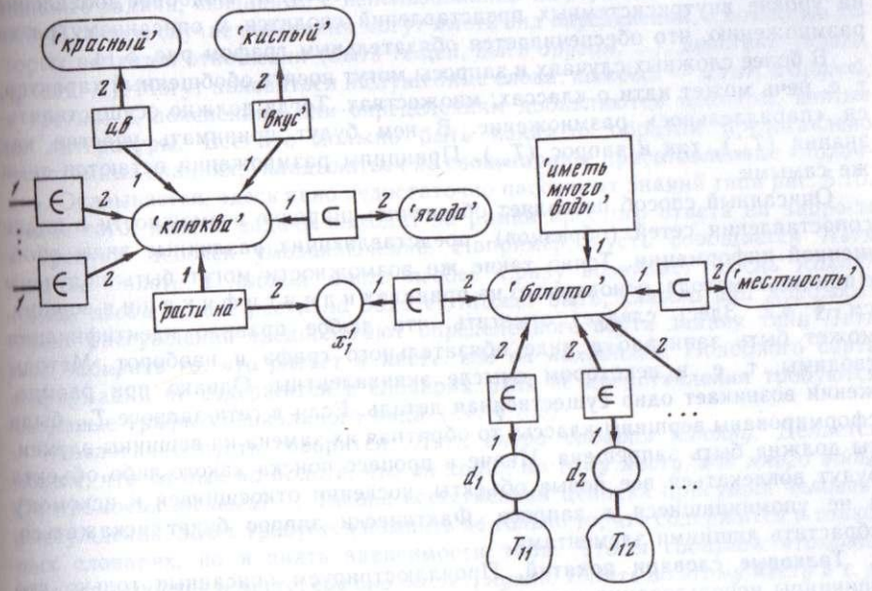


Рис. 5.15. Сеть, представляющая *Клюква — красная кислая ягода, которая растет на болоте*; *Болото — местность, где много воды*. С помощью T_{11}, T_{12}, ... представлено описание известных системе болот



т. е. будет выделен общий элемент a_3. В данном случае формирование резервной *н*-вершины не требовалось..

Нетрудно видеть, что действия расшифровки основаны на специальном виде размножения, когда вершины-классы дополняются вершинами элементами. Представления сводятся к элементному уровню. Другой случай размножения основан на **обратных заменах**, которые, как говорилось ранее, необходимы для использования знаний о свойствах классов в процессе ответа на запросы. Размножение сводится к дополнению вершин-элементов (в сети-запросе) вершинами-классами. Подобные действия задаются обязательным графом вида рис. 5.14. Он мало чем отличается от графа рис. 5.13. Только изменилась нумерация ребер у рекурсивного фрагмента. Граф рис. 5. 14 также применяется к композиции $T_{вх} \circ T_{рв}$. И если в сети $T_{вх}$ имеется вершина d_i, а в $T_{рв}$ представлено $\langle \_\_, t, \in, d_i, m_j \rangle$ или $\langle \_\_, t, \subset, d_i, m_j \rangle$, то обеспечивается формирование $[x_\xi := \{..., m_j, ..., d_i\}]$. Резервная вершина x_ξ как бы замещает d_i и пополняется другими значениями. В результате при выполнении последующих шагов поиска, задаваемых

мых $T_{вх}$, становится возможным использование всех связей, которые есть у m_j.

Аналогичным размножением обеспечивается использование знаний об общих свойствах и отношениях множеств, представленных с помощью з-сетей. Например, пусть в знаниях факт, что *людьми с именем Иван являются* A_1, A_2 *и* A_3, представлен в виде

$$\langle \_\_,\ t,\ б.\ и,\ 'Иван',\ x_i \rangle \circ [x_i := [a_1,\ a_2,\ a_3]] \circ T_1(x_i), \qquad (5.16)$$

где T_1 соответствует их общим свойствам. Пусть в запросе речь идет о человеке A_1. Тогда, чтобы стало возможным использование упомянутых свойств, к $\mathbf{A}_1$ должен быть как бы добавлен элемент X_i. Подобное добавление на уровне внутрисистемных представлений сводится к описанному ранее размножению, что обеспечивается обязательным графом рис. 5.14.

В более сложных случаях и запросы могут носить обобщенный характер, т. е. речь может идти о классах, множествах. Тогда должно осуществляться «параллельное» размножение. В нем будут принимать участие как знания ($T_{рв}$), так и запрос ($T_{вх}$). Принципы размножения остаются теми же самыми.

Описанный способ позволяет обеспечить широкие возможности в плане сопоставления сетей (образцов), представляющих различные виды обобщенной информации. Точно такие же возможности могут быть получены с помощью метода, основанного на правилах идентификации вершин, см. § 4.2. Здесь следует отметить, что любое правило идентификации может быть записано в виде обязательного графа и наоборот. Методы сводимы, т. е. в некотором смысле эквивалентны. Однако при размножении возникает одна существенная деталь. Если в сети-запросе $T_{вх}$ были сформированы вершины-классы, то обратная их замена на вершины-элементы должна быть запрещена. Иначе в процесс поиска какого-либо объекта будут вовлекаться все новые объекты, косвенно относящиеся к искомому и не упоминавшиеся в запросе. Фактически запрос будет искажаться, обрастать лишними элементами.

Толковые словари понятий. Проиллюстрируем описанные только что принципы использования родовидовых связей на реальных примерах. Для этого обратимся к информации толковых словарей, с помощью которых вводятся и поясняются понятия. Для представления пояснений могут использоваться сети. На рис. 5.15 изображена сеть, представляющая: *Клюква — красная, кислая ягода, которая растет на болоте*, а также и *Болото — местность, где много воды* (см. [22]). Каждому понятию сопоставлена своя вершина-класс. У классов могут быть конкретные элементы — представители, что на рис. 5.15 показано для случая *болот*. Такая информация уже не содержится в толковых словарях и накапливается в процессе опыта. Пояснения словарей как бы составляют костяк, который в дальнейшем обрастает все новыми связями — «мясом».

Сеть рис. 5.15 может быть использована для ответа на запросы типа: *Является ли клюква красной ягодой?*, *Где она растет?* и т. д. Запрос представляется в виде $T_{вх}$. При этом может иметься в виду *конкретная клюква, которая растет на «нашем» болоте*, и т. д. Ей в $T_{вх}$ сопоставляется своя вершина x_i, которая $\langle \_\_,\ t,\ \in,\ x_i,\ 'клюква' \rangle$. В процессе поиска ответа осуществляется размножение. На базе вершин x_i будет сформировано $[x_i := ['клюква',\ 'ягода']]$. В результате сеть-запрос $T_{вх}$ приводится

к тому же виду, что и сеть рис. 5.15. Становится возможным типовое сопоставление, реализуемое с помощью операций ассоциирования (см. § 4.2). Ответ получается обычным способом.

Аналогично, если в каком-либо сообщении говорится, что *Петя собирал клюкву*, то сразу становится ясным, что он *собирал красную, кислую ягоду*. Подобное добавление свойств обеспечивается размножением. При этом следует помнить, что вершина *'клюква'* в знаниях одна. И если в $T_{вх}$ было сформировано $[x_i := ['клюква',\ 'ягода']]$, то по вершине *'клюква'* в знаниях может быть выделена вся ее окрестность и добавлена к $T_{вх}$.

Выше рассматривались лишь некоторые виды пояснений, а также отдельные задачи, связанные с использованием информации толковых словарей. В общем случае пояснения могут иметь вид **определений**, с помощью которых вводятся отношения (*быть тещей, быть другом*, ...), действия (*взять, купить*, ...). Могут поясняться и служебные слова: *каждый — и тот, и другой, и третий*. К пояснениям или определениям добавляются примеры, взятые из литературы. Все это должно быть каким-то образом представлено в системных знаниях, укладываться на свои, заранее приготовленные «полочки». Оказывается, здесь явно недостаточно пассивных знаний типа рис. 5.15. Более того, многие задачи выходят за рамки простого ответа на запросы и требуют цепочек умозаключений. Например, пусть сообщается *Петя был на болоте и набрал много ягоды*. Сразу возникает мысль *Какой? Какая ягода может расти на болоте? Может быть, клюква или морошка?* Такие рассуждения предполагают определенного сорта знания типа *Петя мог набирать то, что растет в месте, где он находился*. Подобного сорта информация не содержится в словарях. Для ее представления требуются активные графы специального вида (см. § 5.3).

Аналогично пусть говорится: *Петя вчера собирал клюкву*. Делается ясным, что *он был на болоте*, что *он ходил по тому месту, где много воды. Не промочил ли ноги?* ... Такова естественная цепочка присущих человеку рассуждений. Здесь требуется помнить не только то, что содержится в толковых словарях, но и знать зависимости типа: *Чтобы собирать что-либо, нужно находится в месте, где оно растет, нужно ходить по этому месту* и т. д. Для представления таких знаний весьма перспективно использование аппарата продукций — для обеспечения направленного преобразования (см. § 5.3).

Следует отметить чрезвычайную важность проблемы ввода информации толковых словарей с ее последующим использованием в различных задачах. Как должна быть устроена система, которую можно было бы заставить прочитать, например, все, что есть в словаре Даля? И за счет этого она смогла бы повысить свой интеллектуальный уровень — в плане понимания вопросов, сообщений, ответа на них, реализации умозаключений и др. Решение данной проблемы позволит поставить на качественно новый уровень саму процедуру общения человека с машиной (системой). Обучать последнюю только в режиме непосредственного диалога с человеком может оказаться чересчур утомительным для последнего. Интересно общаться с собеседником, который уже что-то знает и умеет. Более того, решение указанной проблемы позволило бы ближе подойти к вопросам обучения, где часто используются типовые пояснения, как и в словарях.

Реализация элементов логического вывода. Естественный процесс присущих человеку рассуждений предполагает постоянное уточнение информа-

ции — неопределенных объектов, соотношений. Один из частных случаев такого уточнения, формализуемый средствами математической логики, сводится к нахождению логических значений высказываний. Считается, что в этом состоит суть задачи **доказательства теорем**. Как указывалось ранее, для уточнения могут быть использованы обязательные графы. Приведем примеры, иллюстрирующие возможности графов в смысле реализации процедуры доказательства.

На рис. 5.16 изображен обязательный граф, представляющий типовую аксиому языка исчисления высказываний $A \Rightarrow (B \Rightarrow A)$. Пропозициональным переменным A и B сопоставлены $и$-вершины y_1^1 и y_2^1. Логическому значению всего выражения сопоставлена $и$-вершина y_4^1, а его части $(B \Rightarrow A)$ — $и$-вершина y_3^1. Операции, задаваемые таким графом, могут быть выполнены над любой сетью, представляющей то или иное логическое выражение $\mathcal{T}$. И если в последнее входит представленная на рис. 5.16 конструкция (является ее составной частью), то обеспечивается «насильственное» формирование логического значения выражения, которое объявляется истинным. Соответствующей $и$-вершине присваивается значение $[p_i := t]$.

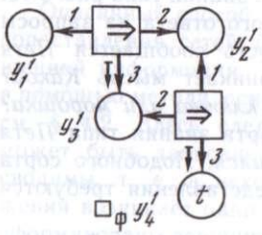


Рис. 5. 16. Обязательный граф, представляющий одну из схем аксиом языка исчисления высказываний, т. е. $A \Rightarrow (B \Rightarrow A)$

Обратите внимание, что на рис. 5.16 сами обозначения A и B никак не представлены. В общем-то такие обозначения носят вспомогательный характер. Когда речь идет о схемах аксиом, то на местах A и B допускаются любые символы или выражения. Соответственно в графе рис. 5.16 $и$-вершины y_1^1 и y_2^1 могут принимать достаточно произвольные значения. Среди последних могут быть вершины, соответствующие пропозициональным переменным, а также логическим значениям каких-либо высказываний. Важно, чтобы выполнялись представленные соотношения. Тогда обязательный граф делается применимым. Словом, граф, представляющий схему аксиом, играет как бы роль «свободно заполняемого» клише.

Выше были рассмотрены принципы использования одной из аксиом — так называемой т а в т о л о г и и, т. е. истинной при всех логических значениях A и B. Аналогично представляются и другие схемы аксиом языка исчисления высказываний. В общем случае они дополняются аксиомами содержательного характера. Классическая задача математической логики сводится к выводу из таких аксиом истинных высказываний — теорем. При этом доказанные теоремы участвуют в последующих шагах вывода. Для вывода используются специальные правила, в частности модус поненс и др.

Данная задача на базе введенных средств решается следующим образом. Аксиомы представляются с помощью обязательных графов. Допускается выполнение операций, задаваемых одними графами, над другими (в последних временно не учитываются стрелки порядка). В результате

обеспечивается постоянное уточнение логических значений, представленных в графах. Само правило м о д у с п о н е н с может быть реализовано с помощью обязательного графа вида

$$[\square_\phi y_1^1 := t] \circ [y_1^1 \perp \langle \_, \Rightarrow, t, y_1^1, t \rangle]. \qquad (5.17)$$

Выполнение операций, задаваемых этим графом, над сетями, представляющими высказывание, приведет к следующему. Пусть высказывание имеет вид $A \Rightarrow B$ и известно, что оно истинно, а также, что истинно A. Тогда объявляется истинным и B. Соответствующей вершине сети присваивается значение t.

Следует отметить, что формализм семантических графов по своим функциональным возможностям не уступает формализму языка исчисления высказываний (а также функционального исчисления первого и второго порядков, см. § 7.3). При этом допускается не только уточнение логических значений, но и любых других компонент, в частности значений предметных переменных. Такого сорта уточнение присуще математическим рассуждениям. Более того, уточнение осуществляется направленным (алгоритмическим) способом, что является существенным преимуществом формализма семантических графов.

К настоящему времени для аналогичных целей развиты аппараты так называемых динамических и алгоритмических логик. Однако их задачи никак не связаны с автономной внутрисистемной обработкой. Эти аппараты являются лишь инструментом исследования. Их конструкции ориентированы на возможности человека, который записывает информацию на специальном языке, осуществляет действия оперирования и др. Задачи представления и обработки решаются за счет умений специально подготовленного человека.

Введенные средства позволяют реализовать **два способа уточнения неопределенных компонент.** Один из них предполагает использование обязательных графов $\{\square\Gamma_i\}$. Пусть неопределенные компоненты представлены с помощью сети $T_{вх}$. Тогда над последней выполняются операции, задаваемые $\square\Gamma_i$, что приводит к насильственному присвоению значений $и$-вершинам из $T_{вх}$. Графы как бы играют роль демонов, которые активно воздействуют на $T_{вх}$. Каждый демон может в любое время пробовать свои силы. Но включается в работу только при выполнении определенных условий. Собственно, аналогичный процесс реализуется в математической логике, где аксиомы и правила вывода играют роль описанных демонов.

Второй способ уточнения — на основе пассивных знаний, представляющих известные соотношения. На базе сети $T_{вх}$ строится граф, определяющий направление уточнения. Задаваемые им операции выполняются над знаниями, что и приводит к уточнению.

Каждый из указанных способов обладает своими преимуществами и недостатками. Реализация многих математических рассуждений предполагает использование и того, и другого способа. Пользоваться только обязательными графами нерационально. Известные соотношения между конкретными объектами лучше представлять в виде пассивных знаний. Дело в том, что при использовании обязательных графов уточнение возможно только в одну сторону — одной или нескольких специально выделенных компонент. В то же время многие соотношения справедливы вне зависимости от направления уточнения. Например, возьмем соотношение $A_1 R_1 A_2$,

которое представляется в виде $\langle \_\_, t, r_1, a_1, a_2 \rangle$. Пользуясь им и зная, что первым объектом отношения R_1 является A_1, можно уточнить второй объект. Зная, что вторым объектом является A_2, можно уточнить первый объект. Зная оба объекта можно уточнить вид отношения. Такое уточнение рационально осуществлять, представляя соотношения в виде пассивных знаний. Если же использовать обязательные графы, то потребуется их значительное количество — для каждого случая уточнения свой граф.

Сказанное справедливо и для логических соотношений. Таблицы истинности для логических связок лучше представлять в виде пассивных знаний (T), как показано на рис. 5.17 для импликации. Тогда, к примеру, если известна истинность посылок A и B, то легко сделать вывод об истинности высказывания $A \Rightarrow B$. Если известна истинность посылки B и выражения $A \Rightarrow B$, то можно постараться уточнить логическое значение A и т. д. Никаких ограничений здесь нет. Подобное уточнение осуществляется методом сопоставления по образцу, в данном случае оператором ассоциирования. При этом то, что уточняется, представляется в виде $T_{вх}$. Один из примеров уточнения проиллюстрирован на рис. 5.17. В общем случае с помощью сетей $T_{вх}$ могут быть представлены любые выражения языка исчисления высказываний. Их уточнение (нахождение значений пропозициональных переменных) возможно путем использования пассивных знаний (см. [14] § 29).

В то же время, одних пассивных знаний может оказаться недостаточно. В ряде случаев для уточнения требуется использование соотношений, содержащих свои неопределенные компоненты и справедливые лишь при определенных способах их уточнения. Для их представления и использования был введен аппарат обязательных графов. В процессе реализации математических рассуждений требуются оба способа уточнения. При попытках компенсировать один из них за счет другого возникают существенные трудности конструктивного характера.

Об организации обязательных знаний. Напомним, что обязательные знания — это множество графов $\{\square\Gamma_i\}$, представляющих, что *должно быть*

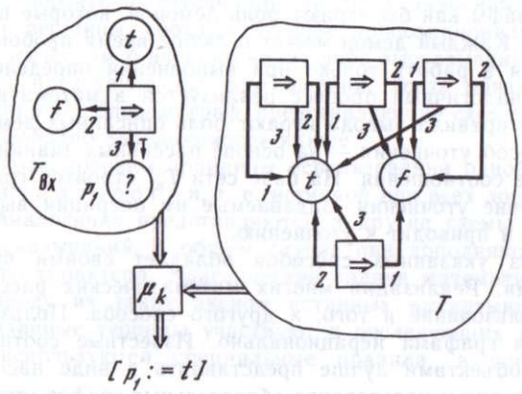


Рис. 5.17. Случай использования пассивных знаний для нахождения логического значения выражения $A \Rightarrow B$, где A — истинно, а B — ложно. Сеть T представляет таблицу истинности для импликации ($\Rightarrow$)

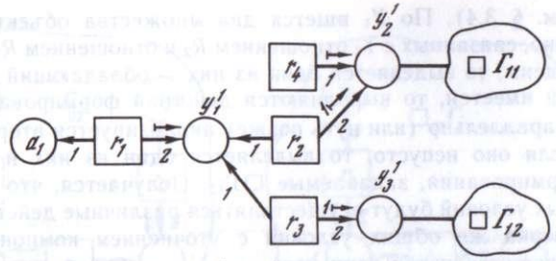


Рис. 5.18. Обязательный граф с параллельными «ветвями»

в тех или иных случаях. Считается, что в различных графах используются непересекающиеся множества $н$-вершин, образующих связанные конструкции (см. § 1.4). Каждый граф представляет самостоятельное утверждение. Пусть на вход системы поступило высказывание, которое было представлено в виде сети $T_{вх}$. Как говорилось ранее, графы играют роль «демонов», которые следят за $T_{вх}$, пробуют свои силы на этой сети. Как же должны быть организованы такие «демоны», как включаться в работу, активироваться? Последовательный процесс, когда вначале активируется один «демон», затем другой и т. д. крайне не эффективен. Факторы, за которыми следят «демоны», условия, при которых допустимо их воздействие на $T_{вх}$, могут сильно различаться друг от друга. Многим «демонам» заведомо не под силу воздействие в данном конкретном случае. И часто об этом можно судить по характеру самого высказывания — по объектам и отношениям, о которых идет речь. По идее $T_{вх}$ должно каким-то образом инициировать деятельность тех «демонов», которые могут быть активированы в каждом конкретном случае. Как это сделать?

Здесь возможно множество вариантов. Рассмотрим некоторые из них. Один из вариантов организации заключается в объединении обязательных графов («демонов») в комплексные структуры — укрупненного типа. «Демоны» как бы объединяются в коалиции, чтобы легче было следить за $T_{вх}$. Одному поручается следить за одним фактором, другому — за другим и т. д. И если какой-либо фактор имеет место, то соответствующий «демон» активирует другие, которые выполняют или функции слежения, или воздействия.

Словом, если факультативные части двух или множества графов содержат общие фрагменты, с которых начинается поиск, то лучше объединить последние в один фрагмент. От него сделать разветвления, направив стрелки порядка $\mapsto$ к различающимся фрагментам. Получается иерархическая структура (к о м п л е к с н ы й граф), задающая поиск по тем или иным фрагментам. В зависимости от результата будет осуществляться переход к соответствующей ветви, принадлежащей одному или множеству графов из $\{\square\Gamma_i\}$.

Пример обязательного графа указанного (комплексного) типа изображен на рис. 5.18. Граф получается в результате объединения двух обязательных графов, содержащих общий фрагмент $\langle \_\_, t, r_1, a_1, y_1 \rangle$. Граф задает операции поиска объекта (Y_1), связанного с A_1 отношением R_1. Далее поиск идет по двум параллельным ветвям, на что указывают стрелки

типа $\mapsto$ (см. § 3.4). По Y_1 ищется два множества объектов (Y_2 и Y_3) соответственно, связанных с Y_1 отношением R_2 и отношением R_3. Если первые объекты найдены, то выделяется один из них — обладающий свойством R_4. Если таковой имеется, то выполняются действия формирования, задаваемые $\square\Gamma_{11}$. Параллельно (или чуть позже) анализируется второе множество объектов. Если оно непусто, то выделяется один из них и выполняются действия формирования, задаваемые $\square\Gamma_{12}$. Получается, что при выполнении различных условий будут осуществляться различные действия формирования. Проверка же общих условий с уточнением компонент (поиском множеств) осуществляется один раз.

В общем случае могут объединяться множества графов. Тогда будет иметь место сложная иерархическая структура, составляющая граф комплексного типа. Нетрудно видеть, что операции, задаваемые такими графами, аналогичны тестовым алгоритмам, нашедшим применение в распознавании образов [44]. Отметим, что задача автоматического построения тестовых алгоритмов может решаться формированием обязательных графов (комплексного типа) по сетям, представляющим обучающую выборку, или же по другим графам, представляющим известные решающие правила.

Другой вариант организации обязательных графов $\{\square\Gamma_i\}$ основан на идее активизации их вершин и фрагментов за счет входной информации. Имеют место действия, напоминающие процесс возбуждения тока в одних электрических цепях — за счет тока, пропущенного через другую цепь. Конечно, такие цепи должны иметь общее электромагнитное поле. Остановимся на способе реализации идеи.

Выделение обязательных графов осуществляется в процессе «восприятия» высказывания, его «осмысления». Высказывание представляется в виде сети $T_{вх}$, на которой формируются стрелки порядка, что может осуществляться в процессе восприятия высказывания. Получается граф $\Gamma_{вх}$, задающий свои операции. Последние выполняются над $\{\square\Gamma_i\}$, у которых в данном случае условные части играют роль пассивных структур. В результате $н$-вершинам из $\Gamma_{вх}$ присваиваются множества значений. Среди них будут и $н$-вершины из $\{\square\Gamma_i\}$. На основе специального анализа таких $н$-вершин и осуществляется выбор графов, допустимых к применению. Здесь, образно говоря, входное высказывание само активирует демонов, которые в данном случае могут испробовать свои силы.

В описываемом варианте процесс выбора и активации может быть организован несколькими способами. Первый из них, когда операции, задаваемые графом $\Gamma_{вх}$, выполняются последовательно с выделением все меньшего числа графов, допустимых к применению. Вначале лишь начинается выполнение этих операций. В результате находятся множества значений $н$-вершин из $\Gamma_{вх}$. Далее анализируется входимость в эти множества $н$-вершин из $\{\square\Gamma_i\}$. Выделяются те графы $\square\Gamma_i$, которые имеют своих представителей в каждом из этих множеств, т. е. в любом из них содержится хотя бы одна их $н$-вершина. Анализируется число выделенных графов. И если их оказалось допустимое количество, то они применяются к $T_{вх}$. Иначе процесс выполнения операций продолжается. Захватывается все большее количество фрагментов из $\Gamma_{вх}$. В результате уменьшается число графов, выделяемых из $\{\square\Gamma_i\}$. Когда их число оказывается в допустимых пределах, то осуществляется их применение к $T_{вх}$. Выполняются действия проверки оставшихся условных частей $\square\Gamma_i$ и формирования. Итак, два

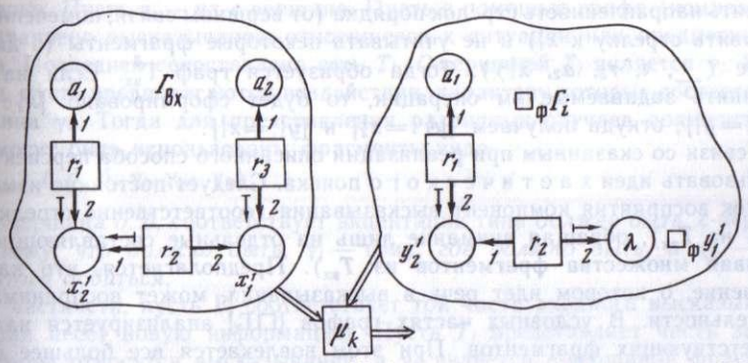


Рис. 5.19. Случай, когда информация ($\Gamma_{вх}$) воспринимается в последовательности, не согласованной со знаниями ($\square\Gamma_i$)

этапа: первый — выделение графов из $\{\square\Gamma_i\}$ за счет $\Gamma_{вх}$, что напоминает эффект «наведения» тока в электрических цепях, и второй — применение графов, что напоминает эффект обратного воздействия — наведенный ток сам начинает воздействовать на цепь, через которую вначале был пропущен ток.

Указанный процесс может быть четко разграничен. Вначале могут быть выполнены все операции, задаваемые $\Gamma_{вх}$, после чего может осуществляться выбор из $\{\square\Gamma_i\}$ подмножества графов, допустимых к применению. В соответствии с ранее сказанным выбираются те графы, каждая из $н$-вершин которых входит хотя бы в одно из множеств найденных значений $н$-вершин $\Gamma_{вх}$ (не обязательно во все множества). При этом в ряде случаев не требуется и выполнения над $T_{вх}$ операций, задаваемых выделенными графами. Если выделен граф $\square\Gamma_i$ и каждая его $н$-вершина y_j^1 была элементом множества значений $н$-вершины x_ξ^1 из $\Gamma_{вх}$, т. е. $[x_\xi^1 := \{..., y_j^1, ...\}]$, то формируется $[y_j^1 := x_\xi^1]$. $З$-сеть как бы разворачивается. Отсюда сразу получаются значения $н$-вершин из $\square\Gamma_i$. Здесь используется свойство симметричности операции сопоставления двух сетей-клише. Часто не играет роли, накладывать одно клише на другое или наоборот.

Следует отметить, что описанный способ организации обладает своим недостатком. Последовательность, в которой осуществляется «восприятие» высказывания (и соответственно направление обработки, задаваемое $\Gamma_{вх}$), может быть не согласована со способом анализа условий, представленных в обязательных графах $\{\square\Gamma_i\}$. Может оказаться, что вначале речь будет идти о соотношениях, которые не представлены в условных частях ни одного из этих графов. В то же время по мере изложения материала могут упоминаться и соотношения, представленные в условных частях, т. е. в принципе в $\{\square\Gamma_i\}$ имеются применимые графы. Но они не будут выделены.

Простой пример, иллюстрирующий описанный случай, изображен на рис. 5.19. В процессе выполнения операций, задаваемых $\Gamma_{вх}$, на $\square\Gamma_i$ (где временно не учитываются стрелки порядка) будут получены пустые множества значений x_1^1 и x_2^1. Уже при сопоставлении первого фрагмента $\langle \_\_, t, r_3, a_2, x_1^1 \rangle$ не будет найдено аналогов в $\square\Gamma_i$. В то же время граф $\square\Gamma_i$ применим к $T_{вх}$. И такая применимость может быть выявлена, если на $\Gamma_{вх}$

изменить направленность стрелок порядка (от вершины связи, помеченной r_2, направить стрелку к x_1^1) и не учитывать некоторые фрагменты (в данном случае $\langle \_\_,\ t,\ r_3,\ a_2,\ x_1^1 \rangle$). Тогда образуется граф $\Gamma^*_{\text{вх}}$. Если на $\square\Gamma_i$ выполнить задаваемые им операции, то будет сформировано $[x_2^1 := y_2^1]$ и $[x_1^1 := y_1^1]$, откуда получаем $[y_2^1 := x_2^1]$ и $[y_1^1 := x_1^1]$.

В связи со сказанным при реализации описанного способа перспективно использовать идеи х а о т и ч е с к о г о поиска. Следует постоянно изменять порядок восприятия компонент высказывания (соответственно стрелки порядка на $T_{\text{вх}}$), обращая внимание лишь на отдельные составляющие (не учитывая множества фрагментов из $T_{\text{вх}}$). Предполагается, что каждое отношение, о котором идет речь в высказывании, может восприниматься в отдельности. В условных частях графов $\{\square\Gamma_i\}$ анализируется наличие соответствующих фрагментов. При этом вовлекается все большее число фрагментов из $T_{\text{вх}}$, как это было описано выше. Анализируется их наличие в $\{\square\Gamma_i\}$. Такой процесс продолжается, пока число применимых графов не окажется в допустимых рамках, к примеру, останется всего один граф.

Общий случай долженствования. Ранее рассматривались два основных случая долженствования: первый — что *должно быть в знаниях* и второй — что *должно быть во входной информации*. И там, и здесь имеет место оттенок обязательности выражаемых соотношений. Возможны и другие случаи долженствования. Остановимся на некоторых из них.

В соответствии со схемой рис. 3.1 возникает случай не обязательности, а о б я з а н н о с т и. Имеются в виду обязанности так изменить объекты предметной области, чтоб имели место заданные соотношения. Если в первых двух случаях изменения осуществляются на уровне системных знаний (СП), то в последнем случае требуется формирование внешних воздействий. Подобные действия подразумеваются в высказываниях типа: *Отряд, прежде чем отправиться в тайгу, должен иметь необходимое количество продуктов.* Требуются соответствующие действия, чтобы запасти эти продукты. Аналогично: *Маркиз должен иметь сына-наследника.* Сравните с высказываниями: *Отряд, который вышел в тайгу, взял необходимое количество продуктов* или *не мог не взять*. Здесь предполагаются изменения на уровне знаний.

Еще один случай, когда выражаются обязанности, связанные с внутренними акциями — умозрительным оперированием, например указывается на необходимость доказательства теоремы, решение задачи: *Нужно доказать теорему о. ..., Должно выполняться* (быть истинным) *соотношение* Γ_1. Имеется в виду не простое изменение знаний с целью учесть указанное соотношение, а построение цепочки доказательства, вывод истинности соотношения $\mathcal{T}_1$. Имеют место совершенно другие акции, которые должны привести к выполнению соответствующих проверок. Такие акции зачастую имеют вид умозаключений, которые сочетаются с другими действиями, например с записью символов (промежуточных результатов) на бумаге. Имеют место достаточно сложные действия.

Все отмеченные случаи различаются материалом, на котором выполняются проверки, и характером действий, инициируемых в случае их невыполнения. Для представления имеющихся в них оттенков обязательности, долженствования могут быть использованы единые фрагменты. Пусть Γ — обязательный граф, задающий операции поиска и проверки. Как уже говорилось, он имеет вид композиции $\Gamma_\zeta \circ \Gamma_i$, где Γ_i — фрагменты, задающие

проверки. Пусть γ_i — их c-вершина. Пусть с помощью графа (композиции) представлено высказывание, относящееся к ситуации или предметной области. Последней сопоставлена сеть T_j. C-вершиной T_j является γ_j. И, наконец, пусть предполагаются воздействия, характеру которых соответствует вершина γ_k. Тогда для представления различных случаев долженствования могут быть использованы фрагменты вида

$$\langle \_\_,\ t,\ д.\ б,\ \gamma_i,\ \gamma_j,\ \gamma_k \rangle, \qquad (5.18)$$

где o-вершина *д. б.* соответствует акцентации типа *должен быть*, c-вершина γ_i — тому, *что должно быть*, γ_j — тому, *где должно быть, и* γ_k — тому, *как этого добиться.*

В частности, пусть Γ_i соответствует той части входного высказывания, которая несет новую информацию. Пусть T_j представляет часть знаний, к которой относится высказывание, а γ_k является c-вершиной фрагмента, указывающего на необходимость изменения знаний. Тогда имеет место случай (см. § 5.1). Соответствующие вершины γ_i и Γ_i метились знаками $\square_\phi$, где индекс «ф» означал *необходимость формирования, изменения.* Этот знак можно считать сокращенной записью фрагмента (5.18) Тогда сама композиция записывается в виде $\Gamma_\zeta \circ \square_\phi \Gamma_i$, где перед Γ_ζ (в соответствии с тем, как ранее было условлено) подразумевается тот же знак $\square_\phi$.

Другой случай, когда с помощью $\Gamma_\zeta \circ \Gamma_i$ представляются знания, а с помощью T_j — высказывание. Этот случай рассматривался в данном параграфе. Тогда композиция, к которой добавлен фрагмент (5.18), записывается в виде $\Gamma_\zeta \circ \square_\phi \Gamma_i$, где перед Γ_ζ уже подразумевается знак $\Diamond$.

Еще один случай имеет место, когда с помощью $\Gamma_\zeta \circ \Gamma_i$ представлена теорема, где Γ_i соответствует тому, что должно выполняться. Тогда γ_k — есть вершина, указывающая на необходимость доказательства. В этом случае для записи оттенков долженствования будем использовать знак $\square_д$, где индекс «д» означает — *нужно доказать.*

Наконец, последний случай связан с внешним воздействием, с обязанностями добиться выполнимости соотношений $\Gamma_\zeta \circ \Gamma_i$. Такие обязанности представляются с помощью фрагмента (5.18), у которого вершина γ_k указывает на внешние воздействия. Будем записывать композицию, к которой добавлен такой фрагмент, через $\Gamma_\zeta \circ \square_F \Gamma_i$, где индекс «F» означает *необходимость внешнего воздействия.*

§ 5.3. СРЕДСТВА НАПРАВЛЕННОГО ПРЕОБРАЗОВАНИЯ

Ранее рассматривался случай, когда изменялись отдельные фрагменты, специальным образом отмеченные. В более сложных случаях могут изменяться множества фрагментов: одни представляют исчезающие или замещаемые части, другие — возникающие, т. е. те, на что замещаются. Более того, во многих случаях допускаются изменения в двух направлениях. Например, правила эквивалентного преобразования могут применяться в обе стороны. Аналогично временные зависимости могут служить как для прогнозирования того, что будет в дальнейшем, так и для восстановления предыдущих ситуаций. Сравнительно часто в процессе прогнозирования или восстановления нужно помнить всю цепочку происходящих изменений. Должно представляться отношение типа *вначале—затем*. Ничего изымать не надо. Хотя если важна лишь текущая ситуация, то разумней использовать действие замещения.

Итак, может быть множество способов использования одной и той же зависимости, для представления которой разумно использовать одну конструкцию. Во введенных ранее средствах не учитывалось подобное обстоятельство. В связи с этим требуется их развитие, в частности языка активных графов и обязательных знаний, чему посвящены два последующих параграфа. В данном параграфе для этого будет введено новое понятие — граф-продукция, обоснована его необходимость и рассмотрены принципы представления сравнительно простых зависимостей — между отдельными объектами, их наборами, а также между соотношениями (без учета контекста).

Об унификации методик. В § 2.1 для представления изменений, преобразований был введен аппарат семантических продукций. Были проиллюстрированы его широкие возможности в плане представления зависимостей, закономерностей, определений и др. Говорилось о том, что прослеживание происходящих изменений обеспечивается путем применения продукции. В частности, продукция, представляющая условную зависимость, применяется к сети, представляющей исходную ситуацию. В результате формируется сеть, представляющая последующую (результирующую) ситуацию. Считалось, что такие действия выполняются специальным механизмом п р е о б р а з о в а н и я. Спрашивается, следует ли рассматривать этот механизм как нечто принципиально отличное от других механизмов, в частности механизма конкретизации, т. е. поиска ответа на запросы? Как задавать действия применения, в рамках какой парадигмы?

Эти вопросы непосредственно связаны с проблемой унификации различных видов внутрисистемной обработки, что весьма актуально с точки зрения создания алгоритмов и построения специализированных устройств. Как указывалось во введении, здесь очень важен сам подход. Нужно ли стараться выявлять множества базовых механизмов, заранее считая, что они могут иметь различную природу? Следует учитывать, что функции таких механизмов весьма сложны — обеспечить использование системных знаний для различных задач, которых можно насчитать десятки. Поэтому, казалось бы, поиск множеств механизмов (реализуемых с помощью собственных алгоритмов) оправдан. Или же проводить исследования в направлении выявления единой основы, придерживаясь точки зрения, что все интеллектуальные функции связаны между собой, имеют единую природу? Нами был выбран второй подход как более перспективный.

В самом деле, преобразования, прослеживания разного рода зависимостей требуют анализа (выявления соответствующих условий) и синтеза (формирования новых фрагментов). Предполагаются определенного сорта обязательные знания. Такое прослеживание зачастую сводится к формулированию «внутренних» вопросов и ответу на них. Например, если описывается какая-либо ситуация, то естественно спросить себя: *А что будет потом*?, *А что было ранее*? Прослеживание зависимостей сводится к ответу на такие вопросы. Если речь идет о каком-то новом отношении или действии, то естественно спросить: *А что оно означает*? Задачи выявления семантической компоненты, преобразования представлений, реализации интегрированных стратегий связаны с умением отвечать на такие запросы.

Словом, если кто-нибудь умеет отвечать на вопросы указанного вида, то он сможет и прослеживать зависимости. Спрашивается, нужно ли отделять эти задачи, которые были названы задачами конкретизации и преобразования? Выясняется, они настолько переплетены, что отделить их практически невозможно. И в том, и в другом случае требуется сопоставление фрагментов с уточнением неопределенных компонент. Ответ на многие вопросы предполагает использование зависимостей. С помощью последних может обеспечиваться «насильственное» уточнение неопределенных компонент. Многочисленные примеры подтверждают сказанное (в частности, см. § 5.2). Следовательно, необходимы единые методики решения задач, которые в одних случаях могут быть использованы для ответа на запросы, в других — для прослеживания изменений, в третьих — для преобразования представлений и т. д.

Поиск единых методик важен не только с точки зрения унификации, построения реализующих устройств. Определенную ценность представляют и теоретические аспекты, связанные с разработкой **единых формализмов** для логико-дискретных систем. В § 2.1 указывалось, что аппарат сетевых продукций шире существующих формальных грамматик. Аналогично аппарат семантических сетей и графов допускает большие возможности, чем традиционные логические средства (см. § 7.2). Существует ли формализм, в который бы вкладывалось и то, и другое? Оказывается, таким формализмом может быть язык семантических графов. Его аналогом являются некоторые теоретико-множественные конструкции (см. § 3.4). Однако операционная часть языка графов богаче, т. е. допускаются операции перебора, проверок и др. Это вопрос будет рассмотрен в § 6.1.

Понятие граф-продукции. Напомним, что механизм преобразования работает над сетевыми продукциями, представляющими семантический эквивалент определенного сорта языковых конструкций, служащих для описания зависимостей, преобразований и др. (см. § 2.1). Такие конструкции имеют вид типовых форм ЕЯ. Разумно считать, что, как и в случае запросов, подобные формы задают направление и способ обработки, т. е. имеют свою операционную семантику, указывающую, как проследить зависимость, реализовать описанное преобразование, воспользоваться определением. Она включает в себя операционную семантику вопросов, сообщений, для представления которых был введен язык семантических графов. Так, чтобы воспользоваться каким-либо определением, нужно найти то, что определяется, уточнить компоненты определяющей части в соответствии с описываемой ситуацией. Требуются и операции формирования, в которых могут принимать участие группы связанных фрагментов.

Для представления таких форм с их операционной семантикой требуются специальные средства, включающие в себя как язык обязательных знаний, так и сетевых продукций. В качестве этих средств будут использоваться так называемые г р а ф - п р о д у к ц и и. Под ними будем понимать продукции, у которых левая и правая части — семантические графы. Граф-продукции задают операции (в том числе формирования), выполнение которых над исходной сетью и приводит к необходимым преобразованиям. Таким способом реализуются и действия применения продукции, и действия пополнения информации.

Граф-продукции будут служить как средство согласованного представления разного рода пояснений, зависимостей с учетом акцентации. Указывается способ уточнения неопределенных компонент, способ выделения анализируемой (изымаемой) и добавляемой частей, а также способ преобразования. Такие способы, как уже не раз говорилось, определяются после-

довательностью изложения материала, языковыми конструкциями, с помощью которых участники диалога стараются сделать излагаемый материал как можно более понятным, что по возможности должно учитываться системой.

Конечно, далеко не всегда способ обработки должен полностью определяться имеющейся акцентацией. Многие зависимости в той или иной степени индифферентны к направлению поиска и уточнения. Следует допускать возможность, когда выбор (и доопределение) направления поручается самой системе, как ей это удобней применительно к конкретному случаю. Для представления такой возможности будем использовать граф-продукции, у которых левую и правую части составляют неполные графы (см. § 3.4), т. е. не везде проставлены стрелки порядка $\mapsto$. Крайний случай — когда таких стрелок нет вовсе. Тогда граф-продукция вырождается в сетевую продукцию. Но чтобы применить последнюю, нужно вначале каким-то образом проставить такие стрелки.

Сетевые продукции. Прежде чем перейти к описанию граф-продукций, вспомним, как осуществляется работа с сетевыми продукциями. В общем виде сетевая продукция записывается следующим образом:

$$T_1(z_1) \underset{r_1}{\rightarrow} T_2(z_2), \qquad (5.19)$$

где z_1 и z_2 — наборы $н$-вершин, соответствующих неопределенным компонентам. Такие наборы могут содержать одни и те же элементы. Пусть с помощью (5.19) представлена некоторая закономерность, определяющая характер изменения объектов класса M_1. Пусть $\mathcal{T}$ один такой объект, представленный с помощью сети T. Чтобы проследить изменения, необходимо настроить закономерность на реальный объект: уточнить неопределенные компоненты, выделить исчезающую и возникающую части. Такая настройка сводится к применению продукции (5.19). Последняя совмещается с сетью T, в процессе чего находятся значения $н$-вершин из z_1 и z_2 (не обязательно всех). Как и в случае обязательных знаний, продукция играет роль формы, клише, которая заполняется материалом, взятым из T. Но способ заполнения никак не оговаривается. Считается только, что обязательно заполнение части $T_1(z_1)$ с ее «слотами» — z_1. Если такое заполнение возможно, то дополняется и оставшаяся часть клише $T_2(z_2)$. Получается изделие, составленное из двух частей. Последующие действия зависят от режима работы. В режиме замещения материал (детали), находящийся в первой части (левой), изымается из T (можно считать, что такое изъятие происходит естественным образом при заполнении клише), а находящийся во второй части (правой) — добавляется. Само клише очищается и годно для последующего применения.

Возможен и режим простого дополнения, а также режим, связанный с представлением цепочки происходящих изменений. В последнем случае клише заполняется материалом из T, сколь это возможно. При этом обязательно заполнение части $T_1(z_1)$. Если последнее условие выполнено, то заполняется оставшаяся часть. Получается изделие, которое и добавляется к T. Заметим, что в изделии будут «детали», взятые из T. К ним как бы привязывается все изделие. Получается цельная конструкция. А само клише остается. Оно очищается и делается готовым для следующего применения.

В § 2.1 говорилось о нескольких разновидностях продукций типа (5.19). В частности, $T_1(z_1)$ может соответствовать исчезающей части, а $T_2(z_2)$ — возникающей. Тогда предполагается уточнение неопределенных компонент лишь по исчезающей части, которая должна быть в исходной ситуации $\mathcal{T}$. При этом в наборе z_2 могут быть $н$-вершины, которых нет в z_1. Они соответствуют новым компонентам, которые появляются. Конечно, данные компоненты также нужно постараться совместить — отождествить с тем, что было известно ранее, если это возможно.

В описанной процедуре остается неясным, как же заполнять клише Никак не оговаривается способ уточнения неопределенных компонент (нахождения значений $н$-вершин), а также способы выделения изымаемой и добавляемой частей. Для указания таких способов будем использовать семантические графы, на основе которых строятся граф-продукции. Последние могут формироваться как за счет входной информации, так и на базе сетевых продукций расстановкой стрелок $\mapsto$, что определяет действия применения продукции. Рассмотрим вначале граф-продукции наиболее простого вида.

Эквивалентные представления об объектах. Чтобы указать на какой-либо объект, необходимо знать его уникальное имя или же описать его свойства и отношения, по которым он может быть однозначно выделен. При этом может существовать множество способов описания, однозначно определяющих объект. Один способ может вводиться через другой. Для этого служат разного рода пояснения типа *это значит*, *это один и тот же объект*. Для представления подобной информации будем использовать граф-продукции следующего вида:

$$\Gamma_1(y_1^1) \circ \langle \beta_1,\ t,\ Df,\ y_1^1,\ y_2^1 \rangle \circ \Gamma_2(y_2^1), \qquad (5.20)$$

где $\Gamma_1(y_1^1)$ — граф, представляющий один из способов описания объекта Y_1 ($н$-вершина y_1^1 является его фокусом), а Γ_2 (y_2^1) — другой. $О$-вершина Df входит в фрагмент, который представляет эквивалентность этих способов. Она соответствует отношению типа *это одно и то же*.

Конструкция (5.20) получается на базе продукции вида (2.1) формированием стрелок порядка. Граф-продукция (5.20) может быть применена к любой сети, представляющей какую-либо ситуацию $\mathcal{T}$ или высказывание о ней, что приведет к пополнению информации о $\mathcal{T}$. Процедура применения сводится к выполнению над T операций, задаваемых $\Gamma_1(y_1^1)$ и $\Gamma_2(y_2^1)$. В зависимости от результатов осуществляются те или иные действия формирования. К сети T приформировываются недостающие фрагменты. Такие действия задаются фрагментом $\langle \beta_1, t, Df, y_1^1, y_2^1 \rangle$, т. е. определяются операционной семантикой отношения типа Df.

В дальнейшем будем рассматривать случаи, когда в графах Γ_1 и Γ_2 имеются только $н$-вершины типа y_i^1, т. е. соответствующие единичным объектам Y_i. Такие графы задают операции перебора с формированием альтернатив (см. § 4.3). В результате их выполнения над T находятся значения $н$-вершин $\{y_i^1\}$. Фактически уточняются абстрактные объекты $\{Y_i\}$ в соответствии с конкретной ситуацией $\mathcal{T}$. Роль уточняющих играют объекты из $\mathcal{T}$. При этом допускается уточнение не всех объектов $\{Y_i\}$, а лишь подмножества $\{Y_j\}$, о чем будет говориться ниже. Соответственно могут быть найдены значения не всех $н$-вершин $\{y_i^1\}$ из (5.20), а лишь $\{y_j^1\}$.

Более того, могут возникать варианты уточнения, например, когда в $\mathcal{T}$ имеется несколько объектов со свойствами $\mathcal{T}_1$. Какой из них взять? Соответственно $н$-вершина y_1^1 может принимать альтернативные значения. Применение продукции (5.20) связывается с выбором одной альтернативы, которая определяет последующие изменения.

Будем различать несколько случаев применения граф-продукции (5.20) и соответственно действий формирования.

Первый случай, когда в выбранной альтернативе значение $н$-вершины y_1^1 оказалось непустым (пусть $[y_1^1 := d_\xi]$), а y_2^1 — пустым. Фактически, одна часть клише (Γ_1) оказалась заполненной, а другая (Γ_2) — нет или частично заполненной. Тогда осуществляется ее дозаполнение. Прежде всего $н$-вершина y_2^1 из Γ_2 (y_2^1) заменяется на d_ξ. Ведь y_1^1 и y_2^1 представляют один и тот же объект. Другие $н$-вершины (y_j^1) из Γ_2 заменяются на их найденные значения. Таким способом на базе Γ_2 формируются новые фрагменты, которые будут привязаны к тому, что имелось в T. Если в $\Gamma_2(y_1^1)$ оказались $н$-вершины y_i с пустыми значениями, то они заменяются на «резервные» вершины, т. е. ранее не входившие в T. Фактически, формируются представления о новых объектах, о которых ранее ничего не было известно. Получившаяся сеть $T_2(d_\xi)$ добавляется к T. При этом исключается дублирование. Фрагменты из $T_2(d_\xi)$, которые ранее имелись в T, не добавляются.

Второй случай применения и формирования, когда в выбранной альтернативе значение $н$-вершины y_1^1 оказалось пустым, а y_2^1 — непустым. Тогда осуществляются те же действия, которые были описаны ранее. Только $\Gamma_1(y_1^1)$ и $\Gamma_2(y_2^1)$ меняются местами. Фактически, имеет место обратное применение — в другую сторону, которое ничем не отличается от обычного применения.

Третий случай возникает, когда значения y_1^1 и y_2^1 (в выбранной альтернативе) оказались непустыми и различными. Пусть $y_1^1 := d_\xi$ и $y_2^1 := d_t$. Фактически, один и тот же объект из $\mathcal{T}$ описан двумя эквивалентными способами, но речь идет как бы о двух объектах. Тогда необходимо их отождествление. У $н$-вершин y_1^1 и y_2^1 формируется одно и то же значение (d_ξ). Сеть $T_2(d_\xi)$ переносится в T и замещает $T_2(d_t)$. Вершины d_ξ и d_t из T сливаются с их окрестностями. В результате образуется одна вершина d_ξ. В другом варианте может быть сформирован фрагмент типа $\langle \_\_, t, =, d_\xi, d_t \rangle$, представляющий равенство (см. рис. 5.5).

Активные функции толковых словарей. В качестве примера обратимся к пояснениям, характерным для толковых словарей. Возьмем одно из них: *Клюква — красная, кислая ягода, которая растет на болоте.* В § 5.2 рассматривалось, как представлять такие пояснения с помощью пассивных структур (знаний). Роль последних играли сети (в частности, см. рис. 5.13), с помощью которых представлялись связи между различными понятиями, в том числе *клюква, ягода, болото.*

В то же время любое пояснение может рассматриваться как имеющее оттенок обязательности, носящее активный характер. Например, переиначим предыдущее высказывание. *Если речь идет о какой-либо ягоде, растущей на болоте, то это должна быть клюква.* Для представления подобной информации уже требуются другого сорта конструкции. Будем использовать для этих целей граф-продукции, составляющие активные знания. Одна из них изображена на рис. 5.20, где y_1^1 и y_2^1 сопоставлены (уточняемым) объектам типа *клюква.* Обратите внимание, что здесь уже представляется связь

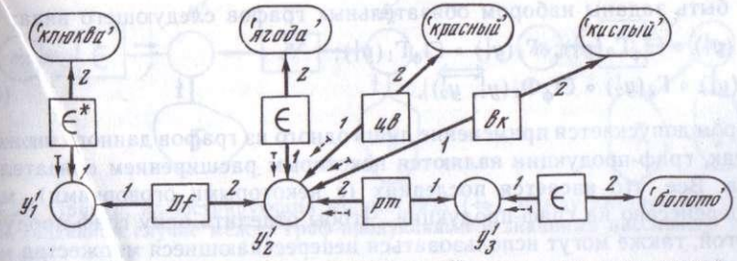


Рис. 5.20. Граф-продукция, представляющая *Клюква — красная кислая ягода, которая растет на болоте. О*-вершина *рт* соответствует отношению *расти на*, а *вк* — *иметь вкус*

между объектами. Далее имеется в виду *некое болото*, которому сопоставлена $н$-вершина y_3^1. Речь идет об объектах, которые могут быть уточнены для каждой конкретной ситуации, а последняя дополнена, что осуществляется путем применения граф-продукции.

Пусть, к примеру, имеется высказывание $\mathcal{T}$, в котором говорится о *некоем болоте* и упоминается *клюква* (D_ξ),— *Петя ходил по болоту и собирал клюкву.* Тогда применением граф-продукции рис. 5.20 к T будет получено $[y_1^1 := d_\xi]$, $[y_3^1 := a_1]$. Сеть T будет дополнена фрагментами, сформированными на базе правой части граф-продукции. В T уже будет представлено, что *Петя собирал красную кислую ягоду, которая растет именно на болоте* A_1. Словом, высказывание дополняется и уточняется.

Пусть в высказывании $\mathcal{T}$ говорится, что *Петя собирал красную кислую ягоду, которая растет на болоте.* Тогда делается возможным применение граф-продукции рис. 5.20 в другую сторону. Отсюда будет получено — *Петя собирал клюкву.* Наконец, пусть высказывание имеет вид *Петя собирал клюкву, ягода растет на болоте и вкусная.* Тогда имеет место третий случай применения граф-продукции. Возникает необходимость отождествления объектов *клюква* и *ягода*, о которых идет речь.

Интересен еще один случай применения граф-продукции типа (5.20). В высказывании $\mathcal{T}$ может много умалчиваться. В конце концов, человек может просто забыть отметить некоторые факты. Например, пусть говорится: *Петя был на болоте и набрал много красной ягоды.* Нам ясно, что это *клюква.* Однако, граф-продукция рис. 5.20 не будет применима, так как в T не представлен вкус — *кислый.* Требуются действия применения, когда не учитываются отдельные фрагменты. В этом случае перспективно использование принципов, рассмотренных в § 5.2 на примере обязательных знаний. Должно допускаться частичное выполнение операций, задаваемых графами Γ_1 и Γ_2 из (5.20). Тогда будут обеспечиваться преобразования при наличии неполной информации.

Отметим, что граф-продукции введенного вида — это определенного сорта о б я з а т е л ь н ы е знания, но с несколько более широкими возможностями. Одна граф-продукция заменяет множество обязательных графов. Запишем граф-продукцию вида (5.20) следующим образом:

$$\Gamma_1(y_1^1) \circ \Phi_1(y_1^1, y_2^1) \circ \Gamma_2(y_2^1). \qquad (5.21)$$

Нетрудно видеть, что описанные выше действия применения граф-продукции

могут быть заданы набором обязательных графов следующего вида:

$$\{\Gamma_1(y_i^1) \circ \square_\phi \Gamma_2(y_i^1);\ \Gamma_2(y_i^1) \circ \square_\phi \Gamma_1(y_i^1);$$
$$\Gamma_1(y_i^1) \circ \Gamma_2(y_2^1) \circ \square_\phi \Phi_1(y_i^1, y_2^1)\}, \qquad (5.22)$$

в котором допускается применение лишь одного из графов данного множества.

Итак, граф-продукции являются некоторым расширением обязательных графов. Все, что касается последних (с некоторыми оговорками), может быть перенесено на граф-продукции. Чтобы отделить одну граф-продукцию от другой, также могут использоваться непересекающиеся множества $н$-вершин, образующих связанные окрестности. Граф-продукции могут организоваться в наборы точно таким же образом, как обязательные графы (см. § 5.2). Они могут выполнять и одинаковые функции. Граф-продукции также могут играть роль демонов, которые следят за входной информацией и пробуют свои силы с целью ее изменения. Только такая проба может осуществляться по-разному, в различных направлениях. Возможность вариантов проб и воздействий задается фрагментом $\Phi_1(y_i^1, y_2^1)$ и определяется операционной семантикой соответствующего отношения.

Следует также отметить, что при применении наборов граф-продукций весьма плодотворной является рассмотренная в § 5.2 идея активизации вершин и фрагментов за счет входной информации (T). Последняя инициирует процесс выполнения операций, задаваемых граф-продукциями. Вызывает как бы пробуждение демонов, которые могут воздействовать в нескольких направлениях. Кстати, последний фактор несколько затрудняет организацию граф-продукций в комплексные структуры, т. е. когда объединяются их левые или правые части. Возникают неоднозначности. Не ясно, в каких случаях и что нужно добавлять.

О соотношении между активными и пассивными структурами (знаниями). При наличии двух сортов знаний возникает естественный вопрос, где следует использовать одни, а где — другие. Казалось бы, значительно лучше все представлять в виде пассивных знаний, как изображено на рис. 5.13. Упрощается процедура поиска по окрестностям. Делается возможным использование обобщенной информации. Обеспечивается пополнение входных высказываний $\mathcal{T}_{вх}$. Напомним, что такое пополнение может осуществляться в процессе идентификации, выбора фрагментов и уточнения значений $н$-вершин из $T_{вх}$ (см. § 4.2 и 5.2). Пусть, к примеру, в $T_{вх}$ имеется вершина d_s, которая входит в $\langle \_\_, t, \in, d_s, m_1 \rangle$, т. е. представляется, что *объект D_s относится к классу M_1*. Тогда в процессе поиска вершина d_s будет идентифицирована с вершиной-классом m_1, находящейся в сети-знаниях T. Из последней может быть выбрана вся окрестность вершины m_1 и перенесена на d_s. Фактически, свойства класса переносятся на конкретные объекты. В результате будет пополнена сеть $T_{вх}$.

Граф-продукции могут служить для аналогичных целей, т. е. для переноса окрестностей, пополнения информации. Но такое пополнение осуществляется направленным и как бы «насильственным» способом, что дает свои преимущества. Условия применения граф-продукций могут иметь достаточно сложный вид и выходить за рамки наличия тех или иных объектов определенных классов. Здесь нет каких-либо ограничений. В левых частях граф-продукций могут быть представлены любые факторы, связанные с наличием указанных свойств и отношений. Из одних свойств могут следовать другие.

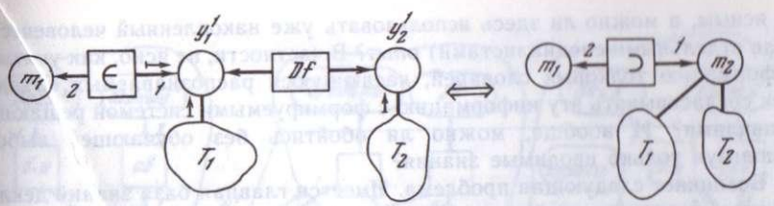


Рис. 5.21. Пример, иллюстрирующий взаимосвязь (⇔) между различными сортами знаний (в данном случае между граф-продукциями и знаниями пассивного типа — о классах)

Принадлежность к классам здесь может играть лишь второстепенную роль.

Для представления подобных зависимостей и предназначены граф-продукции. Если пытаться все представлять с помощью пассивных знаний, то потребуется слишком много вершин-классов. Фактически, по каждому условию придется выделять подмножества объектов и формировать по ним свой класс. О выполнимости условия можно будет судить по соотнесенности к тому или иному классу. Но это чрезвычайно не удобно. Возникнут вершины-классы, не согласованные с понятием ЕЯ и имеющие чисто внутрисистемный характер. Потребуются специальные процедуры, анализирующие входную информацию и соотносящие описываемые объекты к только системе известным классам. Как выясняется, для реализации подобных процедур все равно необходимы действия анализа и синтеза, т. е. средства типа обязательных графов.

В общем-то на базе одних знаний всегда могут быть сформированы другие, т. е. на базе сетей могут формироваться граф-продукции, и наоборот. Простой пример приведен на рис. 5.21. В левой части рисунка изображен пример простейшей граф-продукции. В ней $н$-вершина y_i^1 соответствует *некоему объекту класса M_1 со свойствами $\mathcal{T}_1$*. Представляется, что *если таковой имеется, то он же обладает свойствами $\mathcal{T}_2$*. Пользуясь свойствами $\mathcal{T}_1$ и $\mathcal{T}_2$, можно выделить все множество объектов. На базе их может быть составлен класс с указанными свойствами, что представляется уже с помощью пассивных знаний — сетей. Пример такой сети изображен в правой части рис. 5.21, где новому классу сопоставлена вершина m_2. Возможна и обратная процедура, обеспечивающая формирование граф-продукций на базе представлений о классах. Нетрудно видеть, как это можно сделать.

С процедурами формирования одних структур знаний на базе других связана очень важная проблема — **организации знаний в соответствии с решаемой задачей.** Для этой цели могут быть использованы специальные наборы метаправил, обеспечивающие требуемое формирование. Имеются в виду метаправила, у которых левая и правая части — конструкции СЯ (см. § 4.1). В данном случае левой частью является пассивная структура, а правой — обязательный граф, который может иметь достаточно произвольный вид. Такие метаправила могут служить для формирования направленных процедур анализа по декларативной компоненте (по пассивным знаниям). Заметим, что допускаются возможности, которые выходят за рамки классической задачи распознавания образов.

Напомним, что классическая **задача распознавания** сводится к построению решающих правил на базе обучающей выборки. Однако остается

не ясным, а можно ли здесь использовать уже накопленный человечеством (или отдельными специалистами) опыт? В частности, не ясно, как учитывать информацию толковых словарей, касающуюся распознаваемых объектов? Как согласовывать эту информацию с формируемыми системой решающими правилами? И вообще, можно ли обойтись без обучающей выборки, используя только вводимые знания?

Возникает следующая проблема. Имеется главная база знаний декларативного типа, т. е. набор пассивных структур (сетей). И в соответствии с задачей, порученной системе, она сама должна сконструировать средства и способ решения. Желательно, чтобы такие средства сами имели вид определенного сорта системных знаний. Что же это за знания? (см. § 7.5).

Как выясняется, чем более ограниченна задача, тем более эффективны направленные на ее решение средства. При расширении задачи возникает необходимость в средствах все менее направленного характера, т. е. декларативного вида. Например, для распознавания одного или множества объектов могут использоваться тестовые алгоритмы, обеспечивающие направление выявления свойств. Однако расширим задачи. Пусть необходимо также распознавание по объектам их компонент и пополнение свойств. Тогда для каждого случая потребуется разработка своих тестовых алгоритмов, количество которых будет постоянно расти. Отсюда мы приходим к идее хранения информации в единой форме — в виде соотношений, на базе которых в соответствии с характером очередной задачи должно допускаться автоматическое формирование направленных процедур решения.

Указанная проблема сводится к разработке метаправил, обеспечивающих на базе сетей формирование обязательных графов и граф-продукций. Последние с их операционной семантикой как раз и могут играть роль направленных процедур, т. е. решающих правил (уровня системных знаний), см. [14]. В принципе должна допускаться и обратная процедура, т. е. последовательное формирование конструкций все более декларативного вида (по обязательным графам — граф-продукций — сетевых продукций — пассивных знаний), что обеспечивается применением метаправил в обратном порядке.

Зависимости между соотношениями. Как и для объектов, может существовать множество эквивалентных способов описания соотношений, которые могут быть связаны между собой, определяться одно через другое, например: *Теща — мать жены.* Из одних соотношений могут следовать другие. Имеются в виду также причинно-следственные, временные зависимости. Для их представления в § 2.1 были введены сетевые продукции $T_1 \xrightarrow[r_1]{} T_2$. Будем сводить действие применения такой продукции к формированию на ее фрагментах стрелок порядка $\mapsto$, т. е. указывать способ уточнения неопределенных компонент. При этом стрелки $\mapsto$ должны быть расставлены таким образом, чтобы исключить всякую неоднозначность. Они должны быть направлены ко всем *н*-вершинам, а также *с*-вершинам, соответствующим изымаемой (условной, проверяемой) и добавляемой компонентам.

При выполнении перечисленных условий будет получена конструкция вида

$$\Gamma_1 \circ \langle \beta_3, t, r_1, \beta_1, \beta_2 \rangle \circ \Gamma_2, \qquad (5.23)$$

где β_1 и β_2 — есть *с*-вершины сетей T_1 и T_2. Будем называть такую конструк-

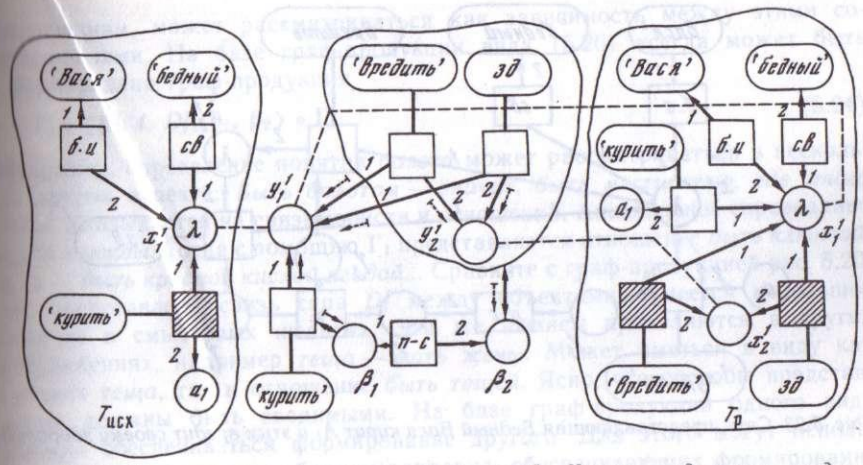


Рис. 5.22. Пример граф-продукции, означающей *Курить — здоровью вредить. О*-вершина *п-с* соответствует *причинно-следственной зависимости*, а *здр — иметь здоровье*

цию г р а ф - п р о д у к ц и е й и записывать в более простом виде: $\Gamma_1 \circ \Phi_1 \circ \Gamma_2$, где фрагмент Φ_1 соответствует зависимости типа R_1. Граф-продукция (5.23) задает операции применения, выполнение которых над некоей исходной сетью ($T_{исх}$) и приведет к требуемым преобразованиям. Они достаточно подробно были описаны в § 2.1. Следует только отметить два важных момента. Во-первых, действия применения реализуются стандартным образом — с помощью введенных ранее операций (см. гл. 4), во-вторых, допускается множество способов уточнения. На базе сетевой продукции может быть сформировано множество граф-продукций. Конечно, выбирается и используется только одна из них.

Каждая граф-продукция $\Gamma_1 \circ \Phi_1 \circ \Gamma_2$ может работать в различных режимах, т. е. допускается несколько вариантов ее применения. Каждый такой вариант сводится к действиям, задаваемым обязательными графами. Например, граф $\Gamma_1 \circ \square_\phi T_2$ задает действия применения продукции в одну сторону с дополнением — формированием новых фрагментов. Другой граф, $\Gamma_2 \circ \square_\phi T_1$, задает действие применения в другую сторону. Граф $\Gamma_1 \circ \square_\phi (\Phi_1 \circ T_2)$ задает применение с представлением зависимости типа R_1 и т. д.

Будем считать, что возможность множества вариантов применения определяется операционной семантикой отношения R_1, т. е. задается фрагментом Φ_1, представляющим данное отношение. В зависимости от задачи граф-продукция настраивается на тот или иной режим работы. Например, если требуется ответ на запрос, то граф-продукция принимает вид одного из обязательных графов $\Gamma_1 \circ \square_\phi T_2$ или $\Gamma_2 \circ \square_\phi T_1$ в зависимости от содержательной части запроса. Если требуется преобразование представлений, то допускаются другие способы варьирования.

В качестве примера обратимся к рис. 5.22. В его средней части изображена граф-продукция, представляющая Y_1 *курит* (не важно, что), *и это является причиной, что* Y_1 *вредит своему здоровью.* Самому *субъекту* сопоставлена *н*-вершина y_1^1, а *его здоровью* — y_2^1. Стрелки $\mapsto$ задают опе-

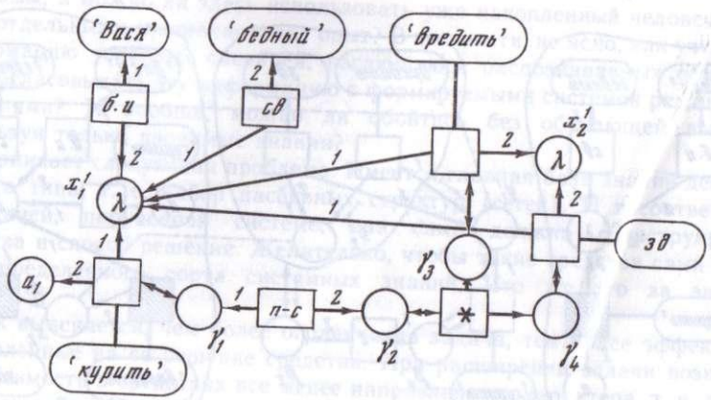


Рис. 5.23. Сеть, представляющая *Бедный Вася курит* A_1 *и этим вредит своему здоровью*

рации согласованного уточнения, см. § 4.3. Пусть имеется сеть $T_{\text{исх}}$, с помощью которой представлено *Существует бедный Вася, который курит* A_1 (*некие сигареты*) (см. рис. 5.22). Самому *Васе* сопоставлена вершина x_1^1, а действие *курить* рассматривается с двумя семантическими падежами *кто курит* (1) и *что курит* (2). Пусть граф-продукция работает в режиме дополнения фрагментов. Тогда выполнением над $T_{\text{исх}}$ задаваемых ею операций будет получено $y_1^1 := x_1^1$, что изображено пунктирной линией. Далее на базе y_2^1 будет сформирована резервная вершина x_2^1, сопоставленная здоровью этого Васи. У вершины x_2^1 будет сформирована окрестность — связанные с нею заштрихованные фрагменты, см. правую часть рис. 5.22. В результате будет получена сеть T_p, представляющая *Бедный Вася курит и вредит своему здоровью*.

Аналогичным способом будет осуществляться поиск ответа на запрос типа *Вредит ли бедный Вася, который курит, своему здоровью*? Только не будут формироваться новые фрагменты (они уже есть в сети-запросе). Но вершине, помеченной знаком ?, насильственным способом будет присвоено значение λ — *Да, вредит*. Граф-продукция будет работать в режиме одного из обязательных графов.

Рассмотрим, каким образом граф-продукция рис. 5.22 может применяться в обратную сторону. Например, пусть имеется сеть T, представляющая *описание бедного Васи, который вредит своему здоровью*. Обратное применение к T граф-продукции сводится к выполнению операций, задаваемых стрелками, которые на рис. 5.22 обозначены пунктиром. Здесь также используется метод согласованного уточнения. В результате будет получена сеть, дополнительно представляющая, что *этот Вася курит*. Таким способом может быть указана причина, восстановлена истекшая ситуация и т. д.

Наконец, рассмотрим еще один случай. Пусть граф-продукция рис. 5.22 применяется к $T_{\text{исх}}$ в режиме представления зависимости. Тогда уже будет получена другая сеть, которая изображена на рис. 5.23. *С*-вершина γ_1 соответствует действию X_1 *курит* A_1, а γ_2 — X_1 *вредит своему здоровью* X_2. Далее представлено, что *из одного следует другое* (*п-с*).

Заметим, что зависимость между объектами, выделяемыми по их со-

отношениям, может рассматриваться как зависимость между этими соотношениями. На базе граф-продукции вида (5.20) всегда может быть сформирована граф-продукция

$$\Gamma_1 \circ \langle \beta_3,\ t,\ Df,\ \beta_1,\ \beta_2 \rangle \circ \Gamma_2. \qquad (5.24)$$

Например, определение понятия *болота* может рассматриваться в несколько другом аспекте: *Быть болотом — значит, быть местностью, где много воды*. Указывается на связь свойств и отношений. Аналогичное справедливо и для *клюквы*. Тогда с помощью Γ_1 представляется отношение *быть клюквой*, а Γ_2 — *быть красной кислой ягодой*... Сравните с граф-продукцией рис. 5.20, где представлена связь типа *Df* между объектами. Имеется небольшое отличие в смысловых нюансах. Эти же нюансы проявляются в других определениях, например *теща — мать жены*. Может иметься в виду как субъект *теща*, так и отношение *быть тещей*. Ясно, что способы представления должны быть сводимыми. На базе граф-продукций одного вида должно обеспечиваться формирование другого. Для этого могут использоваться специальные наборы метаправил, обеспечивающих формирование на базе одних сортов знаний других.

§ 5.4. МОДЕЛИ ДИНАМИЧЕСКИХ СИСТЕМ

В данном параграфе будет продолжено развитие языка обязательных графов и знаний. Будут рассматриваться граф-продукции, представляющие достаточно сложные зависимости, в том числе протекающие в контексте, имеющие корреляционный характер и др. Будут иллюстрироваться возможности таких продукций в плане представления описаний динамических объектов, а также дискретных динамических систем простейшего вида. Фактически, речь будет идти о моделях таких систем, существующих на уровне знаний и согласованных с естественными представлениями. Такая согласованность непосредственно связана с возможностью автоматического формирования моделей на базе входных описаний. Последние могут указывать на определенного сорта стратегии, иметь вид предписаний.

Прослеживание временных зависимостей. В общем случае следует допускать, когда уточняемые компоненты причины остаются в следствии, когда компоненты определяемой части переносятся в определяющую, наконец, когда изменения осуществляются в рамках некоторого контекста или окружения, остающегося неизменным. Для представления подобных зависимостей в § 2.1 были введены сетевые продукции вида $T_1 \circ T_3 \underset{r_1}{\rightarrow} T_2 \circ T_3$, где T_3 соответствует контексту. (Напомним, что в изображении продукции фрагменты из T_3 присутствуют по одному разу.) Сформируем на фрагментах указанной продукции стрелки порядка, направив их *н*-вершинам (y_j) и *с*-вершинам (β_1 и β_2) сетей T_1 и T_2. Тогда будет получена граф-продукция следующего вида:

$$\Gamma_i(y_j,\ \beta_1,\ \beta_2) \circ \langle \beta_3,\ t,\ r_1,\ \beta_1,\ \beta_2 \rangle, \qquad (5.25)$$

где Γ_i — граф или набор графов (с фокусами y_j, β_1 и β_2), построенных на базе $(T_1,\ T_2) \circ T_3$. Подобного сорта конструкции будут играть важную роль для представления временных зависимостей. Тогда *с*-вершина β_1 будет соответствовать исчезнувшей, а β_2 — появившейся части. Остановимся на них более подробно.

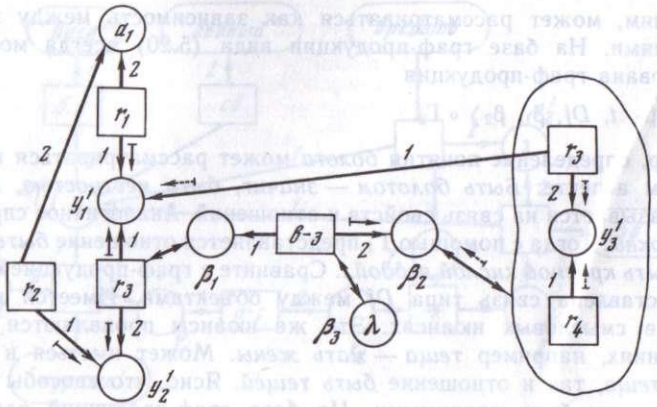


Рис. 5.24. Пример представления сложной зависимости, связанной с разрушением одних связей и появлением других

Начнем с примера. На рис. 5.24 изображена граф-продукция, представляющая временну́ю зависимость следующего вида: *Если имеются два объекта* (Y_1 *и* Y_2), *которые находятся в отношении* R_3 *и связаны с объектом* A_1 *соответственно отношениями* R_1 *и* R_2, *то происходят следующие изменения. Связь типа* R_3 *между ними разрушается и первый объект* (Y_1) *связывается аналогичным отношением* (R_3) *с новым объектом* (Y_3), *обладающим свойством* R_4. Следует заметить, что изменения никак не затрагивают отношений объекта A_1, которые как бы играют роль контекста. Изменились связи совсем у других объектов.

Описанная зависимость может относиться к любой ситуации, для которой выполняются условия. Соответствующие действия реализуются путем выполнения операций, задаваемых граф-продукцией (см. рис. 5.24), над сетью $T_{\text{исх}}$, представляющей какую-либо ситуацию — «исходную» или «текущую». При этом стрелки $\mapsto$ определяют порядок уточнения неопределенных компонент зависимости, способ их отождествления с объектами или частями ситуации. Фрагмент же $\langle \beta_3,\ t,\ в\text{-}з,\ \beta_1,\ \beta_2 \rangle$ определяет вид преобразования.

Проиллюстрируем характер преобразований, задаваемых граф-продукцией рис. 5.24 в **режиме представления зависимости.** Пусть имеется сеть, см. рис. 5.25, представляющая некую исходную ситуацию. Обозначим ее через $T_{\text{исх}}$. Применение к ней граф-продукции приведет к формированию «результирующей» сети, обозначенной через $T_{\text{р}}$. Для простоты рассмотрен случай, когда ранее имевшие место отношения представлены с помощью символа $\tilde{\lambda}$ — *не существует*. *С*-вершинам соответствующих фрагментов присвоены значения $[\gamma := \tilde{\lambda}]$. Текущие же отношения представлены с помощью фрагментов, у которых *с*-вершины имеют значения $[\gamma := \lambda]$. В процессе преобразования учитываются только последние фрагменты, входящие в $T_{\text{исх}}$. К ним относятся и фрагменты, у которых *с*-вершины не изображены (при $\gamma :=$ __ считается $[\gamma := \lambda]$). Хотя возможны и другие способы представления, см. ниже.

При применении продукции рис. 5.24 к сети $T_{\text{исх}}$ прежде всего находятся значения *н*-вершин. С учетом ранее введенных оговорок будет получено

$(y_1^1,\ y_2^1,\ \beta_1) := (a_2,\ a_3,\ \gamma_1)$. Имеет место только один вариант, который определяет последующие преобразования, т. е. формируемые на базе $T_{\text{исх}}$ вершины и фрагменты, что приводит к образованию $T_{\text{р}}$, см. рис. 5.24. В рассматриваемом случае будет сформирована «резервная» вершина x_i^1, соответствующая новому (появившемуся) объекту. Будут также сформированы фрагменты с *с*-вершинами $[\gamma_{j+1} := \lambda]$, $[\gamma_{j+2} := \lambda]$, $[\gamma_{j+3} := \lambda]$. Эти фрагменты представляют возникшие связи объекта A_2. Далее будет сформирован фрагмент, представляющий отношение *вначале—затем* (*в-з*) между бывшей и появившейся ситуациями.

Следует помнить, что прежде чем поместить какой-либо объект в знания, необходимо убедиться, что о нем ранее ничего не было известно. Сказанное относится и к объекту X_i. Нужно проверить, имеются ли знакомые объекты с аналогичными связями, что сводится к нахождению значений x_i^1 из T_p на основе знаний. И если множество таких значений оказалось пустым,

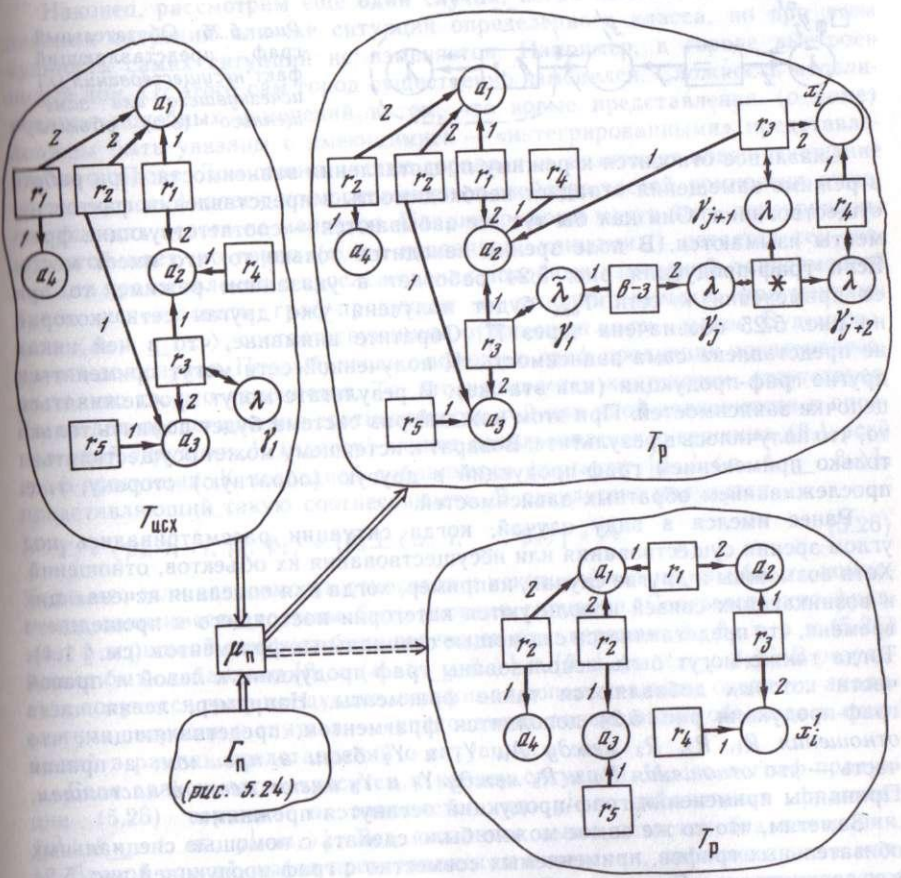


Рис. 5.25. Пример преобразований, вызываемых применением граф-продукции рис. 5.24. Сеть $T_{\text{исх}}$ соответствует исходной, а $T_{\text{р}}$ — результирующей ситуации

то формируется новая *о*-вершина, которая сопоставляется новому объекту и которая с ее фрагментами пополняет знания.

В описанных преобразованиях никак не был представлен тот факт, что отношение R_3 перестало существовать. Необходимо каким-то образом сформировать $[\gamma_1 := \tilde{\lambda}]$ в T_p. В режиме представления зависимости граф-продукции только добавляют фрагменты, не изменяя их. Для такого изменения могут быть использованы **специального вида обязательные графы**. Пример одного такого графа изображен на рис. 5.26. Он представляет, что *если имеется объект* Y_1, *после которого что-то произошло* (с замещением), *то* Y_1 *уже не существует*. Путем применения данного графа к сети T_p будет сформировано $[\gamma_1 := \tilde{\lambda}]$. Подобные графы должны применяться совместно с граф-продукциями. С помощью последних пополняются знания о том, что было вначале, а что затем. Графы же на основе этого как бы «строят заключение» о существовании или несуществовании тех или иных объектов и отношений в текущий момент.

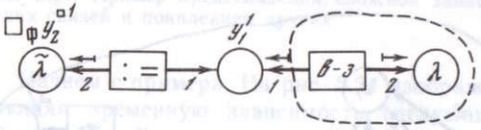


Рис. 5.26. Обязательный граф, представляющий факт *несуществования* ($\tilde{\lambda}$) *исчезнувшего или замещенного* (*в-з*) *объекта*

Сказанное относится к режиму представления зависимости. При работе в режиме замещения отпадает необходимость в представлении фактов несуществования. Они как бы тут же забываются — соответствующие фрагменты изымаются. В поле зрения находится только то, что имеет место. Если граф-продукция рис. 5.24 работает в указанном режиме, то при ее применении к сети $T_{\text{исх}}$ будет получена уже другая сеть, которая на рис. 5.25 обозначена через T_p^*. Обратите внимание, что в ней никак не представлена сама зависимость. К полученной сети могут применяться другие граф-продукции (или эта же). В результате могут прослеживаться цепочки зависимостей. При этом каждый раз система будет помнить только то, что получилось в результате. Возврат к истекшему может осуществляться только применением граф-продукций в другую (обратную) сторону, т. е. прослеживанием обратных зависимостей.

Ранее имелся в виду случай, когда ситуации рассматривались под углом зрения существования или несуществования их объектов, отношений. Хотя возможны и другие случаи, например, когда для описания исчезающих и возникающих связей используются категории настоящего и прошедшего времени, что представляется с помощью специальных фрагментов (см. § 1.4). Тогда также могут быть использованы граф-продукции, к левой и правой части которых добавляются такие фрагменты. Например, левая часть граф-продукции рис. 5.24 дополнится фрагментом, представляющим, что *отношения* R_1, R_2, R_3 *между* A_1, Y_1 *и* Y_2 *были в прошлом*, а правая часть — что *отношения* R_2 *и* R_3 *между* Y_1 *и* Y_3 *имеют место в настоящем*. Принципы применения граф-продукций останутся прежними.

Заметим, что то же самое можно было сделать с помощью специальных обязательных графов, применяемых совместно с граф-продукцией рис. 5.24 в ее прежнем виде. Речь идет об обязательных графах, которые по отношению *вначале—затем* «судят» о том, что было в прошлом, а что в настоящем. Например, граф, формирующий представления о прошедшем, имеет такой

же вид, как и граф рис. 5.26. Только вместо фрагмента с *о*-вершиной: $=$ будет $\langle \_\_, t, вр, y_1^1, y_2^1 \rangle \circ [\square_\phi y_1^{\square 1} := n]$. Представляется, что *объект или ситуация* (которая изменилась на другую) *имели место в прошлом* (n). Применение такого графа к сети, представляющей последовательность событий, обеспечивает формирование фрагментов, играющих роль меток прошедшего времени.

Приведенный пример иллюстрирует широкие возможности, которые обеспечиваются обязательными графами и граф-продукциями. При этом одна и та же зависимость, выраженная различными способами, с помощью различных категорий, представляется с помощью различных конструкций. Но выполнение задаваемых ими операций будет приводить к одним и тем же или сводимым результатам. В некотором смысле будет иметь место инвариантность относительно формы выражения и способа представления. Кстати, это очень важное свойство, характерное для человеческого интеллекта.

Наконец, рассмотрим еще **один случай**, когда изменения касаются отдельных ситуаций или же ситуаций определенного класса, но при этом сущность самих ситуаций не изменяется. Например, в городе выстроен новый дом. От этого сам город существенно изменился. Сложность моделирования подобных изменений в том, что новые представления (о доме) должны быть увязаны с имеющимися — «интегрированными» представлениями о городе. Дом станет частью города, т. е. возникает новое отношение *часть—целое*. Аналогично могут меняться свойства частей, компонент, от чего сущность целого не меняется. Например, дома могут быть перекрашены в другой цвет, но город (с его названием, улицами...) остается тем же.

Для представления и прослеживания подобных изменений, зависимостей будем использовать граф-продукции с дополнительными фрагментами типа $\langle a_i, t, \_\_, \ldots \rangle$, соответствующими отношению *часть—целое*. Будем различать два случая. Первый, когда с помощью граф-продукции представляется только то, что изменяется. То, что называется контекстом, отсутствует. Тогда для представления соотнесенности той или иной зависимости к определенной ситуации A_i (классу) может использоваться *с*-вершина (β_3) всей граф-продукции. К последней добавляется фрагмент $[\beta_3 \perp \langle a_i, t, \_\_, \beta_3 \rangle]$, представляющий такую соотнесенность. В результате получится

$$\Gamma_1 \circ \langle \beta_3, t, r_1, \beta_1, \beta_2 \rangle \circ [\beta_3 \perp \langle a_i, t, \_\_, \beta_3 \rangle] \circ \Gamma_2. \qquad (5.26)$$

Ясно, что если зависимость относится к некоторой ситуации (A_i), то компоненты зависимости должны быть частями ситуации. Соответственно фрагмент с вершиной a_i может быть перенесен на *с*-вершины β_1, β_2, т. е. к (5.26) может быть добавлено $[\beta_1 \perp \langle a_i, t, \_\_, \beta_1 \rangle]$ и $[\beta_2 \perp \langle a_i, t, \_\_, \beta_2 \rangle]$. В результате получится граф-продукция, задающая дополнительные операции анализа типа ситуации и синтеза — «упакования» появившихся объектов и отношений в рамках представлений о ситуации.

Если зависимость относится к классу ситуаций M_k, то в граф-продукции (5.26) вершина a_i заменяется на y_i^1 и добавляется фрагмент $[y_i^1 \perp \langle \_\_, t, \in, y_i^1, m_k \rangle]$. Все сказанное выше остается без изменения.

Второй случай, когда представляются изменения, происходящие в контексте, т. е. имеется неизменная часть. В то же время динамическое отношение типа R_1 связывает лишь изменяющиеся части. Вершина β_3 уже не может быть использована для представления соотнесенности всей

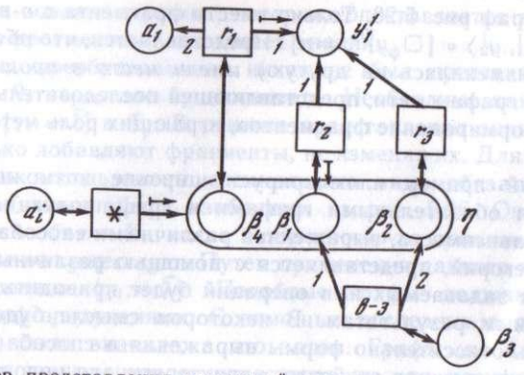


Рис. 5.27. Пример представления изменений, происходящих в рамках ситуации A_i

зависимости к той или иной ситуации. Требуется указывать, что и контекст относится к ситуации A_i. Для этого могут быть использованы фрагменты, представляющие отношение *часть-целое*. На рис. 5.27 приведен простой пример. Представляется *изменение свойств* (*с* R_2 *на* R_3) *у объекта* Y_1. При этом его отношение R_1 с объектом A_1 остается неизменным. Дополнительно указывается, что *все свойства и отношения* (*т. е.* R_1, R_2 *и* R_3) *являются частью ситуации* A_i. Для представления отношения *часть—целое* использован фрагмент $\langle a_2, t, \_\_, \eta \rangle$, отмеченный звездочкой «*». Стрелки $\mapsto$ задают один из вариантов применения. При выполнении соответствующих операций над сетью $T_{\text{исх}}$, представляющей исходную ситуацию, в последней будет сформирован не только фрагмент, соответствующий возникшему свойству R_3, но также и фрагмент, соответствующий факту возникновения новой части.

О представлении динамических систем. Ранее рассматривались изменения, относящиеся к отдельным ситуациям. В то же время многие изменения протекают в рамках фиксированных структур. Например, в электрических схемах изменяются потенциалы, сила тока. Но сама схема остается без изменения. Подобного сорта устройства получили название динамических систем. В них четко различается структура и способ функционирования. Могут иметься входы, воздействие на которые изменяет поведение системы. Многие реальные объекты рассматриваются и описываются как динамические системы. Конструктивные же объекты, как правило, являются таковыми. Отсюда и значительный интерес к данной проблематике.

Как же должны быть устроены внутрисистемные представления о таких объектах, модели на уровне знаний? Это чрезвычайно важный вопрос, решение которого позволит непосредственно подойти к проблеме построения **баз знаний о динамических системах.** Основные функции подобных баз — оказывать помощь человеку в модельных экспериментах. В перспективе таким человеком должен быть так называемый необученный пользователь — исследователь объектов или конструктор, который в основном выражает результат своей работы на ЕЯ. База должна быть устроена таким образом, чтобы обеспечить автоматический ввод и отображение описаний ЕЯ (конечно, выражаемых достаточно точно, по возможности, с использованием однозначно интерпретируемых терминов). Каждое описание должно лечь на свою полочку, что должно приводить к автоматической компоновке цельной модели. В последней должна учитываться и операционная часть, необходимая для «прокручивания» модели, прослеживания происходящих изменений. Все это должна делать сама система. Здесь не предполагается работ, связанных с программированием. Человеку не нужно перекладывать свои представления на какой-либо алгоритмический язык. Конструирование осуществляется с помощью ЕЯ.

Конечно, для реализации описанного проекта еще требуются серьезные исследования. Ниже мы значительно ограничим задачу. Будем рассматривать лишь принципы организации внутрисистемных представлений об динамических системах. Вопросы, связанные с переводом, компоновкой, проверкой правильности и многие другие, останутся в стороне. Более того, речь будет идти лишь о дискретных динамических системах определенного класса — абстрактных (математических) машинах и автоматах. Их типовыми примерами являются конечные автоматы, алгоритмы Маркова, машины Тьюринга, Поста. Их иногда называют системами на бумаге.

Чем же они хороши? Все, что касается их структуры и принципов функционирования, может существовать лишь в представлениях человека, выражаемых с помощью ЕЯ. Здесь может привлекаться специальная символика, разного рода математические языки. Например, автомат может описываться как множество... [8]. Подобного сорта математические описания будут рассматриваться в следующей главе. Сейчас речь будет идти о более «естественных» описаниях — неформализованных, выражаемых с помощью привычных слов. Важно, что такие описания обладают необходимой точностью и достаточно просты в смысле восприятия. Весь автомат можно «держать в голове», т. е. облегчается задача их внутрисистемного представления. Оказывается, что для этих целей могут быть использованы введенные ранее средства. Для представления структуры и состояния компонент могут служить семантические сети, а принципов функционирования — обязательные графы и продукции. Прослеживание динамики (на уровне системных знаний) осуществляется путем их применения, т. е. выполнения задаваемых ими операций над сетями.

Для иллюстрации сказанного возьмем наиболее простую динамическую систему — **конечный автомат.** Напомним, что он может находится в одном из множества состояний $\{B_i\}$. На его вход поступают сигналы $\{A_j\}$, которые вызывают «срабатывание» автомата. В результате изменяется его состояние и формируется выходной сигнал (C_k). Такое изменение и формирование определяется диаграммой переходов. Она состоит из вершин B_i, соответствующих состояниям, и дуг связывающих эти вершины. Дуги метятся символами A_j. И если, например, автомат находился в состоянии B_i и на вход его был подан сигнал A_j, то он переходит в состояние, которое легко найти, перейдя от вершины B_i ао дуге A_j к другой вершине. В дальнейшем для простоты будем иметь в виду автоматы, которые не выдают выходных сигналов, а только переходят из одного состояния в другое. От этого общность рассмотрения существенно не изменится.

На рис. 5.28 изображены внутрисистемные представления (модель) подобного автомата. С помощью сети $T_{\text{дп}}$ представлена диаграмма переходов, которая сама показана левее. Сигналы B_j рассматриваются как отношения, связывающие состояния A_i (хотя возможны и другие способы представления). Сеть $T_{\text{а}}$ соответствует состоянию автомата. С помощью

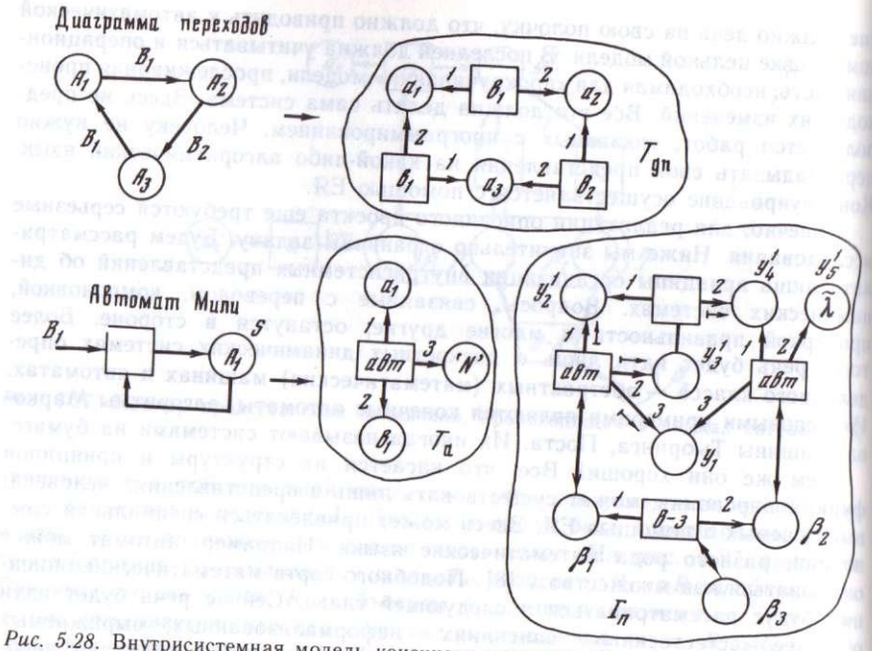


Рис. 5.28. Внутрисистемная модель конечного автомата. Сеть $T_{дп}$ соответствует диаграмме переходов, T_a — входу (W_1) и состоянию (A_1) автомата N, а граф Γ_1 представляет правило перехода из одного состояния в другое

фрагмента $\langle \_\_, t, авт, a_1, b_1, 'N' \rangle$ представлено, что *автомат с номером N находится в состоянии* A_1 *и на вход его подан сигнал* B_1 (*о*-вершина *авт* соответствует тернарному отношению). Граф-продукция $\Gamma_п$ представляет описанное выше правило перехода: *если автомат* Y_1 *находился в состоянии* Y_2 *и на вход был подан сигнал* Y_3, *то он перейдет в состояние, связанное с* Y_2 отношением Y_3, *а сам сигнал пропадет* ($y_5^1 := \tilde{\lambda}$).

Нетрудно видеть, что $\Gamma_п$ имеет вид корреляционной продукции. В ней имеется фрагмент $[y_4^1 \perp \langle \_\_, t, y_3^1, y_2^1, y_4^1 \rangle]$, соответствующий переходу в новое состояние, что определяется зависимостями типа Y_3 (диаграммой переходов). Но сами зависимости никак не меняются. Изменяется лишь состояние, что осуществляется путем применения граф-продукции $\Gamma_п$ к композиции $T_a \circ T_{дп}$, т. е. выполнением задаваемых ей операций. Если $\Gamma_п$ работает в режиме замещения, то будет получено $(y_1^1, y_2^1, y_3^1) := (a_1, b_1, 'N')$, откуда $y_4^1 := a_2$, что соответствует переходу по диаграмме состояний. Далее, в T_a прежний фрагмент будет замещен на $\langle \_\_, t, авт, a_2, x_j^1, 'N' \rangle \circ [x_j^1 := \tilde{\lambda}]$, т. е. *автомат N находится в состоянии* A_2 *и ждет нового сигнала.* Такой сигнал приведет к изменению значения x_j^1, в результате чего сделается возможным повторное применение продукции рис. 5.28 (значение y_4^1 будет непустым) и т. д. Если $\Gamma_п$ работает в режиме представления зависимости, то в T_a будет запоминаться вся предыстория, т. е. через какие состояния прошел автомат и из-за каких воздействий.

Выше была рассмотрена наиболее простая модель, которую нетрудно

соответствующим образом усложнить. В общем случае вводятся понятия автоматов с преобразователями, групп автоматов, работающих совместно, различаются последовательностные машины и т. д. К автоматам добавляются ленты, считывающие головки, разные устройства. Например, машины Тьюринга могут сдвинуть ленту на одну клетку влево или вправо, воспринимая записанный символ. Подобного сорта динамические системы также могут быть представлены в системных знаниях. При этом каждое описание займет свою «полочку». В частности, описание функций и происходящих изменений будет представлено в виде обязательных знаний — графов, граф-продукций. Выполнением задаваемых ими операций обеспечивается «функционирование» модели. Еще раз отметим, что для этого не потребуется специального программирования, разработки алгоритма. Достаточно рассказать, как работает машина (автомат).

Известно, что существует множество способов описания одной и той же машины или автомата. Соответственно в представлениях будут различные модели (сети, графы). Но все они будут приводить к одним и тем же результатам. В следующей главе мы вернемся к сказанному. В ней будут затронуты разного рода абстрактные построения, формализованные конструкции. Они могут вводиться по-разному, т. е. возможны их различные представления.

Описание изменений. Описание работы какой-либо динамической системы зачастую сводится к указанию акций, которые приводят к тем или иным изменениям. Более того, может быть определена последовательность выполнения акций, т. е. задана стратегия. Здесь возникают задачи следующего сорта. Во-первых, как связать описания акций с теми изменениями, к которым они приводят, и, во-вторых, как обеспечить прослеживание изменений с учетом указанных стратегий. Остановимся вначале на первой из них.

Будем различать два типа описаний. Первый, когда описывается, что исчезло и что возникло, например: *Триггер перешел из состояния «о» в состояние «1», На шине возник импульс J* и т. д. Такие описания непосредственно отображаются на введенные ранее средства (продукции), с помощью которых можно проследить, что же имеет место в текущий момент.

Другой тип описаний, когда изменения выражаются с помощью более емких категорий, например: *Автомат перешел в другое состояние, Регистр очистился, При подаче импульса J схема F сработала.* Здесь описываются действия, процессы, приводящие к изменениям. Последние должны каким-то образом связываться с самими описаниями. В § 2.1 было введено понятие составной продукции (см. (2.8)). Расширим его. Для указания способа применения зададим операционную часть, для чего будем использовать введенные ранее стрелки порядка. Продукция примет следующий вид:

$$\Gamma_1 \xrightarrow[Df]{} \Gamma_2, \qquad (5.27)$$

где Γ_1 — граф, представляющий ту или иную описательную форму (ее семантическую структуру — СС), а Γ_2 — обязательный граф или граф-продукция, представляющая имеющие место изменения. Будем называть конструкцию (5.27) составной граф-продукцией.

На рис. 5.29 приведен простой пример. Справа изображена граф-продукция, представляющая действие *перехода* (*автомата*) в *другое состояние* (*пдс*). Самому действию сопоставлен фрагмент $\langle \beta_1, t, пдс, y_1^1 \rangle$.

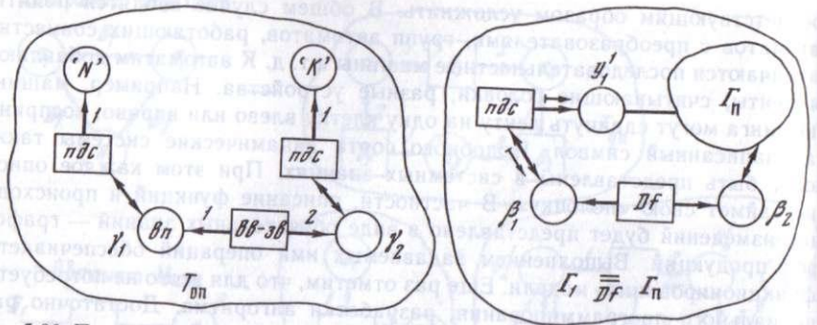


Рис. 5.29. Пример обработки входных описаний ($T_{оп}$), имеющих вид указаний на действия

Такое действие связывается (Df) с внутрисистемными акциями, задаваемыми графом $\Gamma_п$, см. рис. 5.28. Пусть на вход системы поступило описание следующего типа: *автомат N перешел в другое состояние*, что представляется в виде фрагмента $\langle \gamma_1,\ t,\ пдс,\ 'N' \rangle$, входящего в $T_{оп}$. Процесс прослеживания происходящих изменений состоит из двух этапов. Первый этап связан с применением упомянутой граф-продукции к $T_{оп}$. В результате совмещения ее левой части будет получено $y_1^1 := 'N'$. Это значение будет передано в $\Gamma_п$. Последняя как бы настроится на работу с представлениями, касающимися автомата с номером N. На втором этапе осуществляется применение $\Gamma_п$ к $T_а$. В результате будет сформирована сеть, представляющая новое состояние автомата с номером N.

Сравнительно часто с помощью одной и той же высказывательной формы, относящейся к различным объектам, выражаются различные изменения. Например, возьмем предложение *Триггер перешел в другое состояние*. Предполагается, что *если он находился в состоянии «О», то будет находиться в состоянии «1», а если в состоянии «1», то — «О»*. Выражаются совсем другие изменения, чем *автомат перешел в другое (новое) состояние*. В связи с этим в левой части (Γ_1) конструкции (5.27) должны быть более дифференцированные представления. В данном случае должна быть представлена принадлежность объектов к тому или иному классу (триггеров, автоматов). В общем случае могут представляться отличительные свойства и др. Вообще любые две высказывательные формы, выражающие различные изменения, должны представляться с помощью отличающих друг от друга графов Γ_1. Система сама должна «чувствовать» различия и вводить более дифференцированные представления, обеспечивая (в процессе обучения) справедливость указанного правила.

Выполнение последовательности действий. Рассмотрим случай, когда на вход системы поступило описание множества действий с указанием их последовательности, стратегии. Спрашивается, как обеспечить выполнение таких действий в соответствии с указанной стратегией? Казалось бы, проблема «не стоит выеденного яйца». В самом деле, если под рукой имеется ЭВМ, то она сама все сделает, как надо. Однако в данной работе речь идет о системах совершенно другого плана — способных накапливать знания, обучаться, в том числе умению правильно выполнять

действия, о которых говорится в описаниях. Предполагается, что это не дано априорно.

Имеется множество способов выполнения стратегий на уровне системных знаний. Вначале рассмотрим один из них — когда стратегии реализуются на основе тех же принципов, что и в задачах поиска и конкретизации (см. § 4.1). Описание представляется в виде сети. Каждому действию F_i с аргументами $D_1, ..., D_k$ и результатом X_i сопоставляется фрагмент $\langle \_\_,\ t,\ f_i,\ d_1, ...,\ d_k,\ x_i^1 \rangle$. Далее, на сети формируются стрелки порядка. Получается граф. Только такие стрелки будут служить уже для указания не способа уточнения неопределенных компонент, а порядка выполнения действий, вычисления результатов. Соответственно, с фрагментами типа $[x_i^1 \perp \langle \_\_,\ t,\ f_i,\ d_1, ...,\ d_k,\ x_i^1 \rangle]$ связываются не операции группового ассоциирования, а сами действия $F_i(D_1, ...,\ D_k)$. Подобная связь обеспечивается с помощью специальных наборов метаправил, которые в работе [24] назывались правилами интерпретации. Это некоторая разновидность акт-продукций с правой частью — оператором и левой — формулой метаязыка (см. § 4.1). С помощью таких метаправил обеспечивается непосредственное отображение конструкции графов на реализующие устройства F_i. Последние включаются в работу в определенном порядке, что определяется стрелками $\mapsto$.

На рис. 5.30 приведен простой пример. С помощью графа представлено уравнение вида

$$F_2(F_1(A_1)) = F_3(F_1(A_1)) = X_2, \qquad (5.28)$$

где F_1, F_2 и F_3 — есть функции. Стрелки $\mapsto$ указывают порядок вычисления их значений. Собственно такой порядок определяется скобочной символикой, от которой зависит расстановка стрелок $\mapsto$. Если убрать их, то просто будет представлена система равенств

$$F_1(A_1) = X_1,\ F_2(X_1) = X_2,\ F_3(X_1) = X_2.$$

Расстановкой таких стрелок уточняется способ вычисления, что можно связать с **задачей автоматического формирования алгоритма.**

Естественно, возможности вычисления определяются наличием реализующих устройств, от которых во многом зависит, куда можно направлять стрелки порядка, а куда нет. Правила расстановки таких стрелок определяются и типом реализуемых функций (действий) F_i. Если для всех F_i имеются обратные функции (и обеспечивается реализация и тех, и других), то допускается достаточно свободная расстановка стрелок. Как правило, не допускаются лишь случаи, когда от одной вершины связи (квадратика) исходит сразу несколько стрелок. Предполагается, что на основе функции однозначно может быть вычислена лишь одна переменная.

Проиллюстрируем сказанное на примере систем арифметических равенств. Пусть имеется фрагмент $\langle \_\_,\ t, +,\ x_1^1,\ x_2^1,\ x_3^1 \rangle$, представляющий $X_3 = X_1 + X_2$. Направим стрелку порядка к x_3^1, получим $[x_3^1 \perp \langle \_\_,\ t, +, x_1^1,\ x_2^1,\ x_3^1 \rangle]$, что соответствует сложению. Если же направить стрелку к x_1^1, то будет получено $[x_1^1 \perp \langle \_\_,\ t, +, x_1^1,\ x_2^1,\ x_3^1 \rangle]$, что соответствует $X_1 = X_3 - X_2$. Аналогично, знак «+» можно заменить на знак умножения. Тогда вместо минуса будет стоять знак деления.

Следует отметить два важных момента. Во-первых, изменению направленности стрелок порядка будут соответствовать определенного сорта пре-

образования арифметических выражений, в частности, соответствующие переносу слагаемых из одной части равенства в другую, умножению или делению обеих частей на одну и ту же величину. Например, рассмотрим граф рис. 5.31, представляющий один из способов вычисления X_4. Изменим стрелки $\mapsto$ таким образом, чтобы они были направлены от квадратика со знаком умножения к x_5^1, а со знаком плюс — к x_3^1. Тогда уже будет представлено $X_4/X_3-X_2=X_1$. Именно на подобной возможности изменения направления вычисления основаны модели Нариньяни и Тыугу [36, 46]. Очень близко к ним примыкают задачи планирования действия робота. Только вместо арифметических функций будут другого сорта действия — перехода из одной комнаты в другую, перемещения предметов и т. д.

И, второе, в принципе от одного и того же квадратика может исходить несколько стрелок $\mapsto$. Например, можно постараться по равенству $X_1+X_2=3$ найти пары значений (X_1, X_2). Но таких пар в случае арифметических выражений может оказаться множество. Если X_1 и X_2 — целые положительные числа, то множество будет состоять лишь из двух элементов. Если же X_1 и X_2 — вещественные числа, то будет бесконечное количество пар. Именно поэтому не удается путем расстановки стрелок $\mapsto$ решать более сложные системы равенств. Просто невозможно расставить стрелки таким образом, чтоб получилось дерево с корнем — *н*-вершиной (соответствующей одной из переменных) и ветвями, заканчивающимися *о*-вершинами (соответствующими заданным значениям, параметрам). Требуется преобразование систем равенств. Для этого могут быть использованы сетевые продукции (см. § 6.3). Задача автоматического построения алгоритма сводится к формированию стрелок порядка и применению наборов сетевых продукций.

Реализация стратегий. Следует отметить, что путем использования стрелок порядка и введения соответствующей операционной семантики могут быть реализованы лишь достаточно простые стратегии, в частности, выражаемые в виде суперпозиции функций. В более общем виде требуются условные переходы, циклы. Одни функции могут определяться через другие — базисные и др. Для учета подобных действий можно было бы

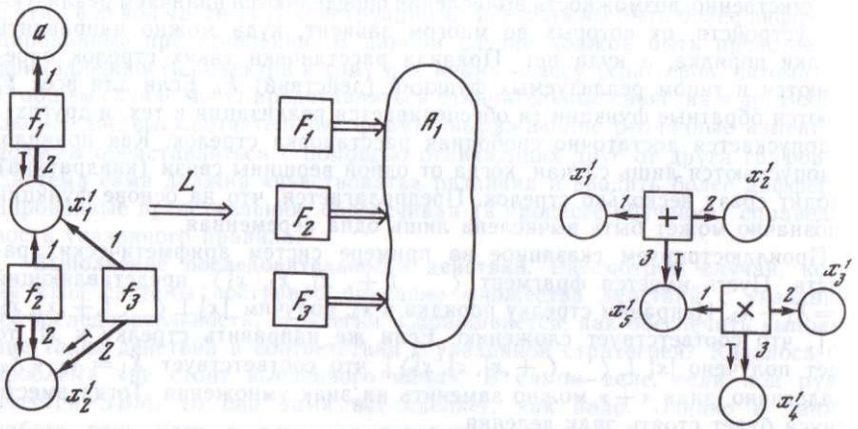


Рис. 5.30. Граф, представляющий случай вычисления значений F_2 и F_3 по F_1

Рис. 5.31. Граф, представляющий $(X_1+X_2)\times X_3=X_4$

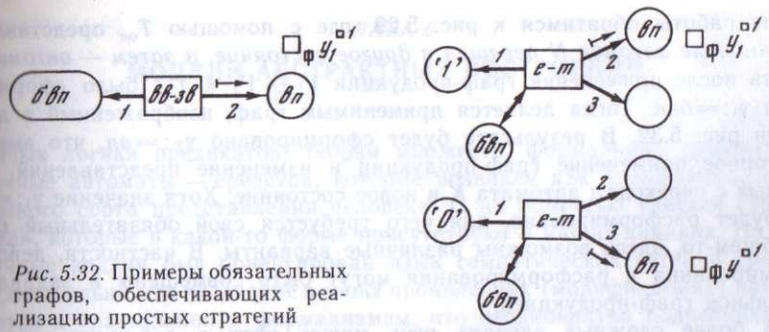


Рис. 5.32. Примеры обязательных графов, обеспечивающих реализацию простых стратегий

ввести специальные стрелки порядка. Но тогда мы потеряем в плане согласованности представлений, возможности автоматического отображения информации.

Для представления сложных стратегий лучше использовать язык семантических сетей, на которых с помощью обычных стрелок $\mapsto$ указывать объекты действия, направление вычисления (см. § 2.4).

При использовании сетей и графов (T_c), представляющих стратегии, можно предложить два способа организации вычислений (действий). Один из них основан на непосредственном отображении L^* конструкций T_c уровня СП на некую реализующую среду, управляющую физическими устройствами F (см. рис. 3.1). Другой способ предполагает использование **специальных знаний** (обязательных графов), осуществляющих слежение за тем, какое действие выполняется, какое уже выполнено, и указывающих на то, какое действие нужно выполнять. Подобное слежение обеспечивается путем применения графов к сети T_c, представляющей стратегию. Рассмотрим, как это делается.

В сети T_c *с*-вершине, соответствующей текущему действию, присваивается значение *вп — выполнять*. По окончании действия это значение сменяется на *бвп — только что было выполнено*. Если действие реализуется на уровне внутрисистемных представлений, то формирование значения *бвп* осуществляется в процессе применения составных продукций. Если же действие реализуется с помощью физических устройств (на них отображаются фрагменты СЯ), то изменение значений осуществляется самими устройствами.

Важно, что за состоянием сети T_c, значениями ее вершин следят специальные обязательные графы, которые применяются к T_c. И как только в T_c появилось значение типа *бвп*, то по фрагментам, соответствующим отношениям *вначале выполнить—затем выполнить* (*вв-зв*), *если, то выполнять* (*е-т*), выявляется *с*-вершина следующего действия, которой присваивается значение *вп*. А значение *бвп* у предыдущей *с*-вершины снимается. В результате как бы внимание переносится на следующее действие, а предыдущее — временно выносится за пределы «сферы видения». Хотя возможен повторный возврат к нему.

Примеры обязательных графов, выполняющих указанные функции, изображены на рис. 5.32. С помощью них обеспечиваются действия, соответствующие условным и безусловным переходам. Для иллюстрации прин-

ципов работы обратимся к рис. 5.29, где с помощью $T_{оп}$ представлено, что *вначале автомат N перешел в другое состояние, а затем — автомат K*. Пусть после применения граф-продукции $\Gamma_1 \xrightarrow[Df]{} \Gamma_2$ к $T_{оп}$ было сформировано $\gamma_1 := бвп$. Тогда делается применимым граф, изображенный в левой части рис. 5.32. В результате будет сформировано $\gamma_2 := вп$, что вызовет повторное применение граф-продукции и изменение представлений, связанных с переходом автомата *K* в новое состояние. Хотя значение $\gamma_1 := бвп$ не будет расформировано, для чего требуется свой обязательный граф. В общем-то, здесь возможны различные варианты. В частности, действие формирования и расформирования могут быть совмещены и задаваться отдельной граф-продукцией.

В более сложных случаях речь может идти о действиях, которые вначале нужно свести к базисным, что сделает возможным их выполнение. Например, в описании алгоритма может быть обращение к процедурам (блокам), заранее определенным. Здесь можно предложить несколько способов организации вычислительного процесса (на уровне знаний). Один из них заключается в предварительном преобразовании сети к виду, представляющему лишь базисные отношения. Затем уже осуществляется выполнение действий (операторов). Другой способ основан на использовании обязательных знаний, которые следят за выполнением действий. И если имело место обращение к процедуре (блоку), то по отношению *часть-целое* выделяется ее первый оператор. Соответствующей *с*-вершине присваивается значение *вп*. После выполнения последнего оператора *с*-вершине самой процедуры присваивается значение *бвп* и т. д.

Преимущество описанного подхода — гибкость, возможность настройки системы на тот или другой способ вычисления, что может осуществляться путем передачи ей соответствующих знаний. В частности, система может быть настроена на работу в режиме компиляции (чему соответствует первый способ) или интерпретации (второй). Может обеспечиваться расширение категорий, служащих для описаний стратегий. Ограничений в этом плане практически никаких нет. В перспективе система может быть настроена (за счет знаний) на способ вычисления или управления действиями, характерный для того или иного языка программирования (Алгол-60, Лисп, Рефал и др.). Соответствующие стратегии могут быть скомпонованы на едином уровне СС, что очень важно при построении многоязыковых систем программирования.

ГЛАВА 6

МОДЕЛИ АБСТРАКТНЫХ ПОСТРОЕНИЙ

Язык логики предикатов, теории множеств, формальные грамматики, конечные автоматы — средства, которые возникли как отражающие определенного сорта представления человека. С их помощью удается строить модели, которые в какой-то форме присутствуют в наших знаниях. На аналогичные задачи был ориентирован язык семантических сетей и графов. Не удивительно, что последние в своих проявлениях (модификациях) приближаются к традиционным формализмам, что неоднократно подчеркивалось ранее, проводились соответствующие аналогии.

В данной главе будет несколько смещен акцент исследований. Их объектом будут сами формализмы (математические объекты, средства их описания), которые будут рассматриваться как образующие собственную предметную область. Она также должна отображаться во внутрисистемные представления. При этом будет учитываться тот факт, что типовые формализмы существуют в рамках математических текстов, с помощью которых они вводятся (строятся), описываются, изучаются. Все это должно представляться в знаниях (базе), а не только традиционные компоненты — синтаксис формального языка, его семантика, способы оперирования над конструкциями.

Вообще, что такое математическая или знаковая модель? Это набор символов и действий над ними, осуществляемые человеком. Последний может записать эти символы на бумаге, рассказать, что с ними нужно делать, что в результате получится. Система по меньшей мере должна уметь отображать все это на свои представления и воспроизводить указанные действия. Имеется в виду система, которую можно было бы настраивать на определенную модель, вводя математические описания. Можно назвать такую систему базой математических формализмов, знаковых моделей. Спрашивается, как она должна быть устроена? В данной главе будет сделана попытка в какой-то степени прояснить данный вопрос. При этом будут рассматриваться лишь некоторые математические построения, в основном касающиеся оперирования над символьными конструкциями, их преобразования, выявления семантической компоненты. Пока что не будет затрагиваться одна из важнейших задач, можно сказать, цель многих построений — процедура обоснования, доказательства. Эту задачу мы рассмотрим в следующей главе.

Итак, наше внимание будет акцентировано на одной из наиболее интересных предметных областей, связанных с математическими построениями. Такая область будет играть роль «пробного камня» для иллюстрации необходимости предлагаемых средств, возможностей введенных методик. Математические объекты и построения в большей степени подходят для этой роли, чем какие-либо другие области в силу их достаточной точности и однозначности.

§ 6.1. О МЕТАМАТЕМАТИЧЕСКИХ ИССЛЕДОВАНИЯХ НА БАЗЕ СЕТЕЙ

В данном параграфе будет рассматриваться, каким образом предложенные ранее средства могут быть использованы в качестве инструмента исследования деятельности математика. Будет обосновываться необходимость таких средств для проведения исследований более широкого плана, чем выполняемых в рамках традиционных подходов.

Об инструментах исследования. Изучение математических построений традиционно относится к области математической логики. С начала становления логика рассматривалась как *наука, пользуясь которой с карандашом в руках можно было бы вычислить правильность любого рассуждения или высказывания* [18]. Для этой цели был создан аппарат математической логики, развит аксиоматический подход. Считается, что в сочетании с аппаратом теории множеств этот подход может быть использован для полной формализации любых математических рассуждений. Однако в действительности, как правило, пользуются неполными формализациями, т. е. *когда опыт математика подсказывает, что перевод на формальный язык был бы лишь упражнением в терпении* (см. [5, с. 24]). Все, что касается процедуры формализации, относится к области так называемой метаматематики, которая считается разделом логики. Метаматематика — это дисциплина, которая, *абстрагируясь от всякого значения, рассматривает формализованные тексты как композиции неких заранее данных объектов, для которых важен лишь порядок их расположения*, см. [5, с. 27].

В настоящее время многие исследования логического плана приняли методологический характер. Здесь важное место занимают проблемы полноты, непротиворечивости, неразрешимости и т. д. Для исследования все новых математических объектов развиваются инструменты исследования. Например, для изучения алгоритмических и ряда других процедур создан аппарат динамических логик. За последние годы большой интерес проявляется к теории допустимых множеств, теории моделей.

Важно отметить, что логические концепции и инструменты постоянно совершенствуются, но такое совершенствование в основном определяется теоретическими исследованиями. А можно ли такие инструменты использовать на практике, например, для решения поставленных в данной работе задач (см. § 1.1)? Ведь понятие систем с СП можно считать некоторой модификацией (расширением) парадигмы теории моделей (см. § 3.1). Более того, задачи данной работы очень близки к тому, ради чего создавался аппарат математической логики. Так, хотелось бы, пользуясь карандашом и бумагой (по строгим правилам), «вычислять» ответ на запросы, строить варианты гипотез и т. д. Более того, хотелось бы, чтобы такого сорта деятельность можно переложить на «плечи» какой-либо машины, т. е. автоматизировать процедуру. Выясняется, что это далеко не просто. Опыт построения даже сравнительно простых диалоговых систем говорит о неприспособленности логических средств. В чем же основная причина? Существующие инструментарии допускают в принципе реализацию любых построений. Но вся «поддержка этих инструментариев» идет через интеллектуальные возможности человека. Предполагается, что все построения будет делать человек, который возьмет текст, переиначит его в нужную форму, запишет на формальном языке и по указанным правилам будет оперировать с его конструкциями. Лишь последняя фаза поддается автоматизации.

Возникают естественные сомнения, а все ли будет учтено в процессе записи на формальном языке? Не исказится ли реальная картина. Ведь сама процедура записи не формализована, а в процессе формализации учтены лишь «существенные» (с точки зрения человека) факторы. Многое останется за кадром. Давно известно, что человеческая деятельность крайне многообразна. Взять хотя бы математика. Он не только доказывает, т. е. выводит из одних истинных высказываний другие (что формализуется средствами логики). Им выдвигаются разного рода гипотезы, осуществляются необходимые конструктивные построения, в конце концов создается свой язык описания. Эта сфера средствами логики практически не исследуется. Хотя в методологическом плане и здесь все должно быть в порядке. Где гарантия, что построения не «зациклятся» сами на себя, или что нет аномалий лингвистического характера? А как известно самим логикам, такие аномалии возможны. Вспомните, например, парадокс Берри (*наименьшее натуральное число, которое нельзя назвать меньше, чем с помощью тридцати трех слогов*). А в чем значение самих значений? Подобные примеры можно было бы продолжить.

Какие средства, инструменты нужны для воспроизведения полной картины? Прежде всего такие средства должны быть адекватны самой задаче. Они должны допускать согласованное представление (в явном виде, на единой основе) смысловых эквивалентов математических текстов, с помощью которых вводятся, исследуются формализмы. Каждое новое предложение должно лечь на свою «полочку», достраивая общую картину (модель). Предполагается наличие некоторой абстрактной машины, которую будем называть метаматематической. Будем рассматривать ее в рамках выбранной организации, т. е. как частный случай систем с СП. Роль упоминавшихся средств будет играть СЯ.

О понятии «метаматематическая машина». Любой математический текст, претендующий на строгость, насыщен специальной символикой, служащей для обозначения абстрактных или формальных объектов. Как правило, такая символика вводится в процессе изложения, рассмотрения все более сложных абстрактных объектов. Характерным является текст, взятый из [5, с. 33]. *Обозначим через* $\tau_y(A)$ *знакосочетание, получаемое следующим образом: мы пишем знакосочетание* τA, *соединяем связью каждый экземпляр буквы* x *в* A *со знаком* τ, *написанным слева от* A, *и заменяем букву* x, *каждый ее экземпляр, символом* $\square$. Фактически вводится собственный язык обозначения абстрактных объектов, их описания. Такой язык может рассматриваться и вне интерпретации. Например, нас может не интересовать, что значит $\tau_y(A)$, какие реальные предметы обозначает это знакосочетание. Важно, как его получить. При этом способ получения или построения описывается в естественном языке (ЕЯ), который обязательно должен быть интерпретированным. Чтобы понять какой-либо текст, необходимо, кроме всего прочего, еще и уметь делать то, что в нем выражается, т. е. соединять, заменять, выделять знаки, стоящие слева, справа и т. д. Такие действия могут выполняться над реальными объектами (символами, записанными на бумаге) или прослеживаться чисто умозрительно. И в том, и в другом случае важно, чтобы был получен результат построений, который во многом определяет, о чем же идет речь.

Могут ли математические описания быть полностью формализованы? Считается, что да и что для этой цели может быть использован язык логики

предикатов. Однако, как бы ни уточнялись описания ЕЯ, всегда остается неформализованная часть, ориентированная на здравый смысл математика [52]. Что же это за здравый смысл? В предыдущем примере предполагалось умение понимать описания, делать то, что в них записано. Предполагалось также умение пользоваться определениями для создания, расширения языка описания. Однако и этого, безусловно, мало. Математические тексты насыщены и материалом другого сорта, предполагающим свой «здравый смысл», свои умения, например умения оперировать с посылками, следствиями, придерживаться хода определенного сорта рассуждений, умения на базе различных предпосылок строить цепочки доказательств и т. д. Все это подразумевается. Многие такие умения оказываются связанными между собой (родственными). Собственно, с выявлением базисного набора таких умений связана очень важная проблема — как должна быть устроена машина (система), которая могла бы читать математические тексты, формировать соответствующие представления и использовать их для ответа на запросы, для доказательства, а также для последующего понимания других текстов, проверки их правильности, корректности. В рамках такой машины должны каким-то образом укладываться новые формализмы, знаковые системы, используемые для построения математических моделей.

Ясно, что речь идет не о математической машине в традиционном понимании этого термина. Под математической машиной, как правило, понимают созданную на бумаге *знаковую систему*. Имеется в виду некое описание, которое человек может однозначно понять, построив в своих представлениях соответствующую модель. В нее входят и акции оперирования, т. е. представления о том, как работает машина. Естественно, такие акции могут быть запрограммированы или реализованы, например, на ЭВМ. Ниже речь будет идти о машине, способной «переварить» описание любой знаковой системы (в том числе вводимую символику, указанные преобразования), построить в своих представлениях соответствующую модель и уметь пользоваться ею. Именно такую машину будем называть *метаматематической*. Заметим, что она тоже может иметь вид знаковой системы, но несколько другого сорта.

На каких же принципах должна быть основана метаматематическая машина? Ясно, что она должна иметь внутренние представления или знания, в которые укладывались бы математические тексты и, по возможности, без потерь. Должны быть базовые механизмы, работающие над знаниями и обеспечивающие ввод математических описаний, их понимание, а также накопление и использование содержащейся в них информации. Как нетрудно видеть, обратившись к предыдущем примеру, для этого требуется умение выделять объекты по указанным свойствам, отношениям, проверять их наличие или отсутствие и многое другое. Все это должно осуществляться на уровне внутрисистемных представлений или знаний. Далее, требуется умение проследить или смоделировать те построения, изменения, преобразования, о которых идет речь. И, наконец, требуется умение пользоваться новыми терминами. Должно допускаться постоянное расширение языковых возможностей за счет описаний типа *назовем..., будем говорить* ... и т. д. Как выясняется, это очень непростая проблема. В математических текстах, как уже говорилось, многое умалчивается, подразумевается. В процессе же отображения во внутрисистемное представление все должно выражаться в явном виде. Следует учитывать, что часто имеют место неоднозначности,

которые должна устранять сама система. Например, возьмем предложение *обозначим* $A+B$ *через* ω. Что обозначается — само выражение или результат сложения? Вообще, различение знаков и их значений — весьма тонкая вещь. Понимание математических текстов требует соответствующих знаний, которыми должна обладать метаматематическая машина.

Проблема формализации описаний. Математические построения весьма любопытны по следующей причине. Фактически с помощью одного языка (ЕЯ) вводится другой — формальный. Исследование такой процедуры важно, прежде всего, с точки зрения изучения способов создания языка, реализации этих способов в рамках систем с СП. Более того, часто говорится о точности математических описаний, о возможности полной формализации. Но многое из того, что связано с описаниями (и относится к их «интеллектуальной поддержке»), остается за пределами формализации, т. е. понимается чисто умозрительно. Фактически неисследованной остается та часть, в рамках которой рождается новый формализм. Что это за часть? Может ли она быть формализована и каким-то образом вложена в одну из существующих знаковых систем или машин? Могут ли для этих целей быть использованы типовые инструментарии, позволяющие задавать синтаксис, операционную или какую-либо другую семантику, позволяющие вводить новые возможности? В частности, достаточно ли для этой цели бэкусовских нормальных форм, средств ввода операторов алгоритмических языков, разного рода описателей, допустимых в языке программирования? Ясно, что нет. Иначе язык программирования можно было бы расширить до ЕЯ, хотя бы ее части — математических описаний. Но это не удается сделать. Тогда какими должны быть средства, обеспечивающие такое расширение? Какое место должны занимать существующие формализмы, в частности грамматики — линейные, графовые и др.? Ведь такие формализмы сами существуют благодаря их описаниям, понимаемым человеком. Последний может проследить все то, что описывается, построить «систему на бумаге», пользоваться ей. В частности, некоторые такие системы могут прикладываться для работы с самими описаниями, текстами. Тогда спрашивается, какие должны быть формализмы, чтобы можно было их использовать для работы с текстами их же записи? Заметим, что в ЕЯ это можно делать.

Конечно, в настоящий момент дать ответ на все эти вопросы невозможно. Однако некоторые исследования в данной проблематике имеют смысл. В дальнейшем представляется перспективным проводить их под углом зрения построения метаматематических машин, в частности, в рамках парадигмы — систем с СП. И еще раз отметим, что речь будет идти не об инструментариях, с помощью которых разработчикам удается строить разного рода конструктивные объекты, а о внутрисистемных средствах, в рамках которых допускается существование таких инструментариев.

При исследованиях в указанной проблематике следует учитывать, что математические тексты насыщены разнообразными описаниями, каждое из которых имеет свое назначение. Например: *Обозначим ... через* $(B|X)A$. *Читать «*B *замещает* X *в* A*»*, см. [5, с. 33]. *Формативная конструкция теории* $\mathcal{T}$ *есть последовательность знакосочетаний* ... [5, с. 35], *Заменим каждое вхождение* τ в A *на* B. Тогда (примеры взяты наугад) каждое такое описание (часть текста) должно укладываться на свою полочку, заранее приготовленную. Для организации некоторых таких полочек (т. е. формализации некоторых видов описаний) в какой-то степени могут быть исполь-

зованы существующие средства и подходы. Однако если учитывать, что все такие полочки связаны между собой, одна подготавливает другую и т. д., то становится ясной необходимость общей основы, организации. Отсюда мы приходим к идеологии систем с СП. Остановимся на принципах их использования для представления достаточно простых математических объектов и описаний.

Схема обработки. Речь будет идти об общей организации метаматематической машины — на базе введенных в предыдущих главах концепций. В § 2.3 и 2.4 уже поднимались вопросы, связанные с представлением некоторых типов абстрактных объектов — слов, чисел, таблиц, выражений алгоритмических языков. При этом мы различали уровни поверхностных структур (*ПС*) и семантических структур (*СС*). Напомним, что любое выражение Ф какого-либо языка (формального, ЕЯ) может рассматриваться как набор значков, расположенных в определенном порядке, что представляется с помощью сети, называемой *ПС* выражения. Будем обозначать такую сеть через *ПС* [Ф]. Сеть, представляющая значения или содержание выражения Ф, была названа его семантической структурой, что будем обозначать через *СС* [Ф]. Например, выражение $\{X \mid R_1(X) \wedge R_2(X)\}$ может рассматриваться как набор значков, т. е. как знакосочетание. Тогда оно представляется на уровне *ПС*, т. е. с помощью сети $ПС[\{X \mid R_1(X) \wedge R_2(X)\}]$. Принципы представления таких сетей будут описаны в следующем параграфе. В то же время нас может интересовать, что означает данное выражение. Тогда уже будут иметь место представления на уровне СС, где выражению будет сопоставлен семантический граф. Его фокус будет соответствовать множеству X, а область — условиям, ограничениям $R_1(X) \wedge R_2(X)$. Принципы подобного представления рассматривались ранее (см. § 1.4 и 3.4). В общем случае здесь могут быть использованы как семантические графы, так и сети. В соответствии с ранее введенными обозначениями будем записывать их по типу $CC[\{X \mid R_1(X) \wedge R_2(X)\}]$ или обычным способом — в рамках СЯ.

Далее, каждое выражение может иметь свою операционную семантику. В частности, с указанным выше выражением могут быть связаны действия по выделению множества X. Для такого выделения будем использовать операторы ОЯ, введенные в гл. 4. Более того, уровни *ПС* и *СС* связаны между собой. Предполагается возможность отображения с одного уровня на другой. Итак, имеет место достаточно сложная организация, в рамках которой реализуются процессы ввода новых терминов, расширения языка записи моделей. Рассмотрим такую организацию более подробно.

На рис. 6.1 изображена схема обработки описаний, касающихся символьных выражений и выполняемых над ними действий. В рамках данной схемы укладываются и некоторые виды определений, о которых будет говориться ниже. Пока что речь будет идти лишь об конкретных описаниях, касающихся отдельных объектов или множеств. В схеме рис. 6.1 имеется предметная область (ПО). В нее входят как абстрактные объекты (знакосочетания), обозначенные через Ω, так и их описания *Оп*Ω. Различают два типа описаний — описание конструкции Ω и описание действий над Ω. И те, и другие имеют вид текстов на ЕЯ, в которых встречаются знакосочетания Ω.

Объекты ПО (знакосочетания, описания) через специальное устройство восприятия V отображаются во внутрисистемные знания — на уровень *ПС*. На этом уровне любые описания на ЕЯ рассматриваются как обычные знакосочетания, т. е. они представляются точно таким же образом, как и объек-

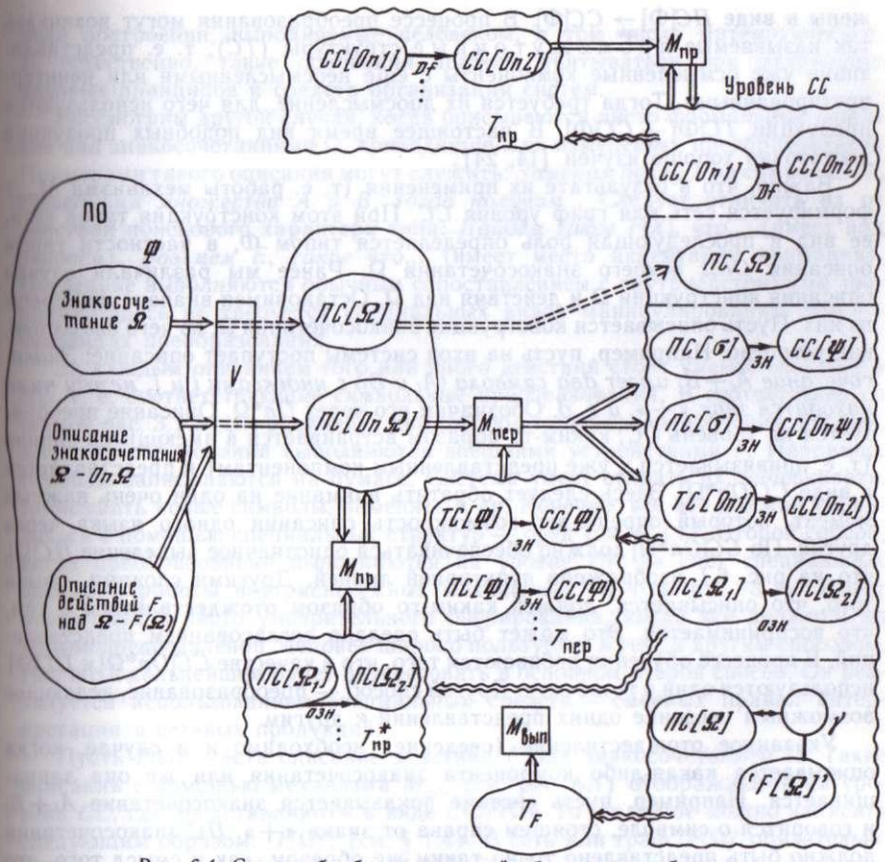


Рис. 6.1. Схема обработки описаний абстрактных объектов

ты Ω. Обозначим такие знакосочетания через Ф. Будем считать, что любое знакосочетание Ф, в том числе с новыми, ранее не встречавшимися символами, может быть отображено на уровень *ПС*, где как бы запечатляется все то, что система воспринимает. Представляются базовые (различаемые) символы и пространственные отношения между ними. Для этого и используются сети $ПС[Ф]$, где в общем случае на месте Ф может стоять Ω (конкретное знакосочетание) вместе с *Оп*Ω (конкретным описанием), т. е. $ПС[Ф] = ПС[Ω, Оп Ω]$.

Далее выявляется семантическая компонента, т. е. что означает то или иное знакосочетание. Осуществляется преобразование знакосочетаний Ф с уровня *ПС* на уровень *СС*. Для этого используется механизм $M_{пер}$ (см. рис. 6.1), управляемый с помощью специального сорта знаний $T_{пер}$. Последние определяют, какие знакосочетания являются интерпретированными, т. е. могут быть преобразованы на уровень *СС*. Роль знаний $T_{пер}$ будут играть наборы сетевых продукций (см. § 2.1). На рис. 6.1 они изобра-

жены в виде $ПС[Ф] \xrightarrow[зн]{} СС[Ф]$. В процессе преобразования могут возникать так называемые промежуточные структуры ($ТС$), т. е. представляющие уже осмысленные компоненты с еще неосмысленными или неинтерпретированными. Тогда требуется их доосмысление, для чего используются продукции $ТС[Ф] \xrightarrow[зн]{} СС[Ф]$. В настоящее время вид подобных продукций достаточно хорошо изучен [14, 24].

Важно, что в результате их применения (т. е. работы механизма $М_{пер}$) формируется сеть или граф уровня $СС$. При этом конструкция такой сети, ее вид и проследующая роль определяется типом Ф, в частности типом описания $Оп\Omega$ некоего знакосочетания Ω. Ранее мы различали случаи описания конструкции Ω и действия над Ω. Остановимся вначале на первом из них. Пусть описывается конструкция знакосочетания Ω, из чего оно состоит, как устроено. Например, пусть на вход системы поступает описание: *Знакосочетание A_i+B_j имеет два символа (A_i и B_j) с индексами i и j, между ними находится знак «+», и т. д.* Обозначим его через $Оп^*\Omega$. Описание преобразуется на уровень $СС$, каким-то образом встраивается в имеющиеся знания (т. е. привязывается к уже представленным компонентам) и представляется в виде $СС[Оп^*\Omega]$. Здесь следует обратить внимание на один очень важный момент, который определяет возможность описания одного языка через другой. По $СС[Оп^*\Omega]$ должно обеспечиваться однозначное выделение $ПС[\Omega]$, что на рис. 6.1 изображено пунктирной линией. Другими словами, смысл того, что описывается, должен каким-то образом **отождествляться** с тем, что воспринимается. Это может быть сделано согласованием представлений. В процессе обучения добиваются того, что в качестве $СС[Оп^*\Omega]$ и $ПС[\Omega]$ используются одни и те же сети. Другой способ — преобразование, делающее возможным сведение одних представлений к другим.

Указанное отождествление (сведение) необходимо и в случае, когда описывается какая-либо компонента знакосочетания или же она запрашивается. Например, пусть системе показывается знакосочетание A_i+B_j и говорится о символе, стоящем справа от знака «+». $ПС$ знакосочетания должно быть представлено точно таким же образом, как и смысл того, что говорится, т. е. $СС$ описания. Иначе говоря, выделение вершины, соответствующей символу B_j, оказывается невозможным. Система не сможет понять, о чем же идет речь.

В более сложных случаях системе может показываться какой-либо составной объект, например иероглиф, и говориться о его частях. Здесь также должна обеспечиваться однотипность, согласованность представлений, необходимая с точки зрения поиска, отождествления.

Отмеченная однотипность может восприниматься как нечто естественное — в наших представлениях как-то все очень легко согласуется. Однако за этим естественным кроется многое. Прежде всего, умение отождествлять вырабатывается в процессе обучения языку. Вначале отождествляются слова и предметы, затем описания с более сложными конструкциями, ситуациями. Аналогичными возможностями в перспективе должна обладать и система. А если их нет, то разработчик должен сам каким-то образом предусматривать однотипность упоминавшихся представлений.

Вообще, всегда следует помнить о том, что при рассмотрении систем с $СП$ достаточно часто приходится обращать особое внимание на то, что с точки зрения здравомыслящего человека является само собой разумеющимся. Но именно здесь следует искать основания, делающие возможными разного

рода построения, выполняемые человеком, в том числе математические. И, естественно, такие основания должны учитываться при разработке базовых принципов и средств организации систем.

Рассмотрим другой случай, когда описываются чисто формальные действия над знакосочетаниями Ω, приводящие к их изменению, преобразованию. Примерами такого описания могут служить: *Заменим букву X* на символ □ ..., *Объединим множества A и B, тогда получим...* Следует отличать их от действий поискового характера типа: *Найдем такое F(X), что...* (имеет вид запроса), *Возьмем E, такое что...* (имеет место акцентация внимания). Последние выполняются обычным сопоставлением с внутрисистемными знаниями. Здесь не требуется специальных видов манипулирования, обеспечивающих преобразования.

За каждым описанием того или иного действия стоит умение выполнять его, т. е. соответствующие символьные преобразования. В соответствии со схемой рис. 3.1 будем различать два способа таких преобразований, первый, когда преобразования выполняются внешними устройствами F. Например, символы записываются на бумаге. Система умеет стирать их, зачеркивать, записывать новые символы, заменять и др. Конечно, все это должно управляться с помощью специальных структур — сред (см. § 3.1). Второй способ, когда преобразования выполняются на уровне $СП$ за счет специальных знаний и работы внутрисистемных механизмов $М$ (см. рис. 3.1). Имеет место случай чисто умозрительного оперирования, когда все делается на уровне представлений. Человек широко пользуется и тем, и другим способом. Нас же в дальнейшем будет интересовать в основном второй способ. Он реализуется использованием специальных средств — сменных правил интерпретации и сетевых продукций.

Пусть $F(\Omega)$ — есть описание действия F над знакосочетанием Ω. Такие описания с помощью механизма $М_{пер}$ (см. рис. 6.1) отображаются на уровень $СС$, где представляются в виде $СС[F(\Omega)]$. То же самое можно записать следующим образом: '$F(\Omega)$', (см. § 1.4). В сеть или граф '$F(\Omega)$' обязательно будет входить $с$-вершина γ сети $ПС[\Omega]$. С ее помощью представляется аргумент, т. е. объект действия. Выполнение действия осуществляется с помощью механизма $М_{вып}$, управляемого знаниями о действиях $Т_F$. Имеются в виду сетевые продукции, сопоставляющие представлениям о действиях обязательные графы или знания. Последние задают соответствующие операции преобразования на уровне внутрисистемных представлений (см. § 6.3). В $Т_F$ также могут быть введены специальные метаправила, с помощью которых структурам СЯ (представлениям о действиях) могут непосредственно сопоставляться операторы обработки (см. § 4.1). Далее, в $Т_F$ могут находиться специальные графы, обеспечивающие метку действий в соответствии с указанным порядком их выполнения.

Определения, обозначения. Математические тексты насыщены разного рода описаниями, с помощью которых вводятся новые значки, символы, понятия, фиксируется тот или иной способ говорения и др. В таких описаниях затрагиваются и чисто формальные, и содержательные вещи. Все это весьма причудливым образом переплетается в рамках единого текста описания. В процессе представления каждое такое описание должно лечь на свою «полочку», заранее заготовленную. Для этого введем некоторый вариант классификации определений, ориентированный на обработку в рамках схемы рис. 6.1.

Будем различать определения (описания) четырех типов. Первый тип относится к вводу новых символьных обозначений. К таким определениям относятся следующие: *Обозначим через σ множество отрезков S множества E, для которых существует ... или Пусть σ — есть множество таких отрезков S множества E, что f и f′ совпадают на S*. Возможны и другие формы, например: *(B|X)A — читать «B замещает X в A»*.

Обозначим подобного сорта определения через *Оп* Ω. В них можно выделить две части — вводимое обозначение σ и что оно обозначает (*Оп* Ψ). Первая часть рассматривается как набор значков и представляется на уровне ПС, т. е. в виде *ПС*[σ]. Вторая часть должна быть понятной. Предполагается выявление семантической компоненты *Оп* Ψ с отображением на уровень *СС*, где будет сформировано *СС*[*Оп* Ψ]. Все определение представляется в виде пары (сетевой продукции)

$$ПС[\sigma] \underset{зн}{\rightarrow} СС[Оп\ \Psi], \qquad (6.1)$$

где о-вершина *зн* соответствует отношению *означать*, т. е. представляется связь между уровнями *ПС* и *СС*.

Продукция (6.1) формируется на уровне *СС*, но в дальнейшем может быть перенесена в лингвистические знания $T_{пер}$, что на рис. 6.1 изображено волнистой линией. Применением продукций к *ПС* входных описаний обеспечивается выявление в них соответствующих обозначений с формированием структуры уровня *СС*, т. е. представлением значения или смысла. В результате будет обеспечиваться некоторое расширение языковых возможностей системы.

Остановимся на **определениях второго и третьего типа**. Сравнительно часто новые понятия, термины вводятся в контексте. Например: *Пусть G — график, X — предмет. Срезом графика G по X называется множество* $G\langle\{X\}\rangle$, см. [5, с. 85]. Здесь имеется определяемая часть (словосочетание *срез графика G по X*) и определяющая — все остальное. Обозначим первую часть через *Оп*1, а вторую — через *Оп*2. Естественно, *Оп*2 должна быть «понятной», т. е. отображаемой на уровень *СС*, где *Оп*2 представляется в виде *СС*[*Оп*2]. В то же время *Оп*1 может быть как понятной, так и частично понятной или совсем непонятной. Соответственно *Оп*1 может быть отображено на уровень *СС* и представлено в виде *СС*[*Оп*1] или не может. В последнем случае оно представляется на уровне *ТС* (промежуточных структур) или *ПС*, т. е. в виде *ТС*[*Оп*1] или *ПС*[*Оп*1].

Например, если про срезы графиков или каких-либо других объектов уже говорилось и система знает, что это такое, то и новое словосочетание будет отображаемой на *СС* (имеющимися средствами). Тогда *Оп*1 представляется в виде *СС*[*Оп*1]. Если же в словосочетания входят новые термины (например, слово *срез* встречается впервые) или известные термины используются в новых сочетаниях (ранее не было добавки *по X*), то определяемая часть не может быть отображена на уровень *СС* и представляется в виде *ТС*[*Оп*1]. У некоторых терминов казался понятным смысл, т. е. была выявлена семантическая компонента, у других — нет. Все это представляется в виде сети *ТС*[*Оп*1]. И уж крайний случай, когда вообще ничего в *Оп*1 непонятно. Тогда *Оп*1 рассматривается как набор значков-символов и представляется в виде *ПС*[*Оп*1].

Соответственно будем относить определения с совершенно непонятной определяемой частью к первому типу, с частично понятной — ко второму

и с полностью понятной — к третьему. Первый тип был рассмотрен выше. Определения второго типа представляются в виде

$$ТС[Оп1] \underset{зн^*}{\rightarrow} СС[Оп2], \qquad (6.2)$$

где *зн** соответствует указанию на связь между двумя уровнями. Продукция типа (6.2) может быть перенесена в знания $T_{пер}$ и использоваться для «досмысления» текстов.

Определения третьего типа представляются в виде

$$СС[Оп1] \underset{Df}{\rightarrow} СС[Оп2]. \qquad (6.3)$$

О-вершина *Df* соответствует отношению типа *означать то же самое*. Связываются структуры одного и того же уровня *СС*. Продукция (6.3) может быть перенесена в специальные знания $T_{пер}$, что на рис. 6.1 изображено волнистой линией, и использоваться для преобразования представлений. Имеются в виду преобразования над осмысленными текстами с целью их сведения к базисным представлениям, расшифровки действий и др.

Пусть описывается, например, что значит *применять правило подстановки к исходному слову* или *склеивать два графа*. Эта (описываемая) часть (*G*1) является понятной, т. е. представляется в виде *СС*[*G*1], или что то же самое, '*G*1'. В описательной части дается некоторая стратегия, что представляется в виде '*F*1' (см. § 2.4). Все определение будет представлено в виде $'G1' \underset{Df}{\rightarrow} 'F1'$. И если в каком-либо входном тексте речь идет об указанном действии, то последнее всегда может быть расшифровано, что осуществляется применением продукции к его *СС*. В результате действие может быть сведено к базовым операциям, реализуемым системными механизмами ($M_{вып}$).

И, наконец, **четвертый тип** определений служит для обозначения одних значков, символов (Ω_1) через другие (Ω_2). Такие обозначения носят чисто формальный характер и необходимы, чтобы облегчить восприятие сложных символьных выражений. При этом не обращается внимание на то, что означают значки. Например, в определении *Обозначим выражение* $A\uparrow B|X$ *через* $F(X)$ может подразумеваться чисто «механическая» замена. Такие определения могут содержать разного рода пояснения, говорящие об особенностях символьных конструкций. Они преобразуются на уровень *СС* и представляются в виде

$$ПС[\Omega_1] \underset{озн}{\rightarrow} ПС[\Omega_2], \qquad (6.4)$$

где *о*-вершина *озн* соответствует отношению *обозначать*. Представляется связь между конструкциями уровня *ПС*. Продукции вида (6.4) объединяются и образуют специальные знания $T^*_{пр}$, используемые для преобразования представлений на уровне *ПС*. Такие действия соответствуют чисто формальному манипулированию, без вникания в содержание, смысл.

Отметим, что **формальные системы**, как считается, должны быть основаны именно на упомянутом манипулировании, которому соответствуют действия применения набора продукций $T^*_{пр}$ к *ПС* знакосочетаний. Однако трудно себе представить, как работать с математическими текстами без выявления семантической компоненты. Конечно, возможен еще один уровень рассмотрения, когда все, что касается схемы рис. 6.1, представляется только на уровне *ПС*. В частности, сети уровня *СС* представляются как наборы вершин, связанных ребрами. В другом варианте может представляться *ПС*

формульной записи сетей и графов. Словом, все, что описано в данной работе, в принципе может рассматриваться как формальная система и быть представлено лишь на уровне ПС. Однако тогда на этом уровне появляются подуровни, соответствующие символическим записям различного сорта знаний, возникнет необходимость в связях и т. д. Вся схема рис. 6.1, все равно каким-то образом будет представлена, но как комбинация символов.

Схема рис. 6.1 предполагает согласованную работу всех ее звеньев. На одном и том же уровне должны иметь место о д н о т и п н ы е или сводимые представления. Иначе никакая обработка невозможна. Система должна допускать постоянную «подстройку» за счет своих знаний. Здесь очень важен вопрос организации набора продукций $T_{\text{пер}}$ (т. е. знаний о внешнем языке) и принципов их применения (т. е. работа $M_{\text{пер}}$). Система должна уметь различать случаи появления новых терминов, а также известных терминов, но в новых сочетаниях. От этого зависит, может ли описание быть преобразовано на уровень СС. Далее, механизм $M_{\text{пер}}$ должен быть индифферентен относительно знаний $T_{\text{пер}}$ — он должен обеспечивать работу с любыми структурами $T_{\text{пер}}$, которые отображают соответствующие математические конструкции, описания. При этом следует учитывать многоэтапность преобразования с уровня ПС на уровень СС, т. е. через уровень ТС. Процесс преобразования связан с реализацией морфологического, синтаксического, а зачастую и семантического анализа. Новые знания должны умело встраиваться в уже имеющиеся, управляющие этапами преобразования. Помимо этого должны выявляться противоречивые, избыточные случаи. Система должна обладать возможностью обобщения (экстраполяции). Иначе любая новая комбинация символов будет ставить ее в тупик. Словом, возникают те же задачи, что и при работе с обычными текстами ЕЯ.

§ 6.2. ПРЕДСТАВЛЕНИЕ СИМВОЛЬНЫХ ВЫРАЖЕНИЙ

В чем же сущность задачи? Нужны ли какие-то специальные представления, когда общепринятая запись символьных выражений гораздо проще, нагляднее? Когда речь идет о реализации, то ясно, что нужно кодировать символы, перекладывать их на машинный язык. А здесь зачем? Так рассуждает человек — конструктор, разработчик. Отсюда — как бы он делал, что ему для этого нужно, что удобнее. Ниже будет рассматриваться несколько другой вопрос: а как лучше машине, т. е. удобней с точки зрения организации системной обработки.

Следует учитывать тот факт, что человек от природы обладает прекрасной способностью **видения**. Стоит ему показать какой-либо сложнейший символ, и ему сразу станет ясно, что к чему. Подсознательно будут выделены компоненты, по которым он поймет, что же означает данный символ. У системы (машины) такого видения нет. И в то же время для работы с символами, для выявления, что за ними стоит, ей нужно очень четко «чувствовать» различия. Стоит передвинуть какой-либо значок (индекс), и символ может означать уже совсем другое. Более того, символьные выражения каким-то образом должны сочетаться с их описаниями, с помощью которых они вводятся, указываются допустимые соотношения, дается их значение. Поэтому просто закодировать символы — еще полдела. Нужно закодировать так, чтобы допускалось отображение указанных описаний на системные коды. Отсюда мы и приходим к необходимости согласованных представлений,

в которых все должно выражаться в явном виде, к необходимости специального уровня для сочетания символьных конструкций и их описаний (см. § 6.1).

В данном параграфе будут затронуты лишь отдельные вопросы, связанные с представлением символьных выражений. Последние будут рассматриваться как чисто формальные объекты, т. е. как наборы значков, находящихся в определенных пространственных отношениях. Речь будет идти о представлениях на уровне ПС, т. е. без учета интерпретации. Подобные представления уже рассматривались в § 2.4 применительно к числам, словам, таблицам. Ниже будет продолжено такое рассмотрение. При этом будет учитываться тот факт, что любой формальный объект вводится в контексте интерпретируемых (понимаемых) выражений, в которых указывается, из чего он состоит, к каким классам относятся его компоненты и др. Им также должно находиться место в системных представлениях.

Представление знакосочетаний. Рассмотрим наиболее типовые выражения. Они составляются из набора базовых символов. Пусть ими будут следующие: буквы A, B, C, ... алфавита, скобки (открывающиеся, закрывающиеся), знаки операций (арифметических, логических и др.) и пробелы. Сопоставим таким символам свои вершины-классы, как показано в верхней части рис. 6.2. Всем символам сопоставлена вершина *'кл. символов'*. При этом с помощью фрагмента $\langle$—, *t*, *эн*, *'кл. символов'*, $\mathfrak{A}'\rangle$ представлено, что *класс символов обозначен через* $\mathfrak{A}$. Аналогичным образом представлено, что *класс букв обозначен через* $\mathfrak{B}$, *скобки — через* $\mathfrak{C}$, *а операции — через* $\mathfrak{R}$. Далее представлено, что A, B, ... $\in \mathfrak{M}$; (,) $\in \mathfrak{C}$; +, ×,... $\in \mathfrak{R}$, т. е. *соотнесенность базовых символов к классам*, а также $\mathfrak{M} \subset \mathfrak{A}$, $\mathfrak{C} \subset \mathfrak{A}$, $\mathfrak{R} \subset \mathfrak{A}$, — $\in \mathfrak{A}$, т. е. *принадлежность букв, скобок, операций, пробелов к классу символов*.

Отметим, что указанные соотношения как раз и составляют ту содержательную часть, которая необходима для проверки правильности или допустимости синтаксической конструкции, для последующей интерпретации знакосочетания (для этого нужно знать, какой символ является знаком операции, какой обозначает число и др.) И, наконец, содержательная часть позволяет говорить о знакосочетаниях — указывать на их компоненты, запрашивать их. Классы, как правило, отображаются на понятия, используемые в вопросах и сообщениях. Конечно, чтоб поиск был возможен, необходимы представления о самих знакосочетаниях. Рассмотрим, как они устроены.

В нижней части рис. 6.2 представлена ПС выражения следующего вида:

$$(A+B)\times B. \tag{6.5}$$

Такие выражения обычно называют т е р м а м и. Буквы в них различаются не только по своему внешнему виду, но и по позициям. На рис. 6.2 им сопоставлены *н*-вершины x_1^1, x_2^1, ... Одна и та же буква может находиться на нескольких позициях, что проиллюстрировано на примере буквы B. При этом факт, что буква B находится на предпоследней позиции (последняя занята пробелом) представляется следующим образом: $\langle$__, *t*, *нх*, *'B'*, $x_8^1\rangle$, где *о*-вершина *нх* соответствует отношению *находиться на*, а x_8^1 — соответствующей *позиции*. Тот факт, что *упомянутая позиция является предпоследней*, представлен лишь косвенно — на это указывает фрагмент $\langle$__, *t*, *л-п*, x_8^1, $x_9^1\rangle$, где *н*-вершина x_9^1 больше не входит в другие фрагменты такого же типа (*л-п*).

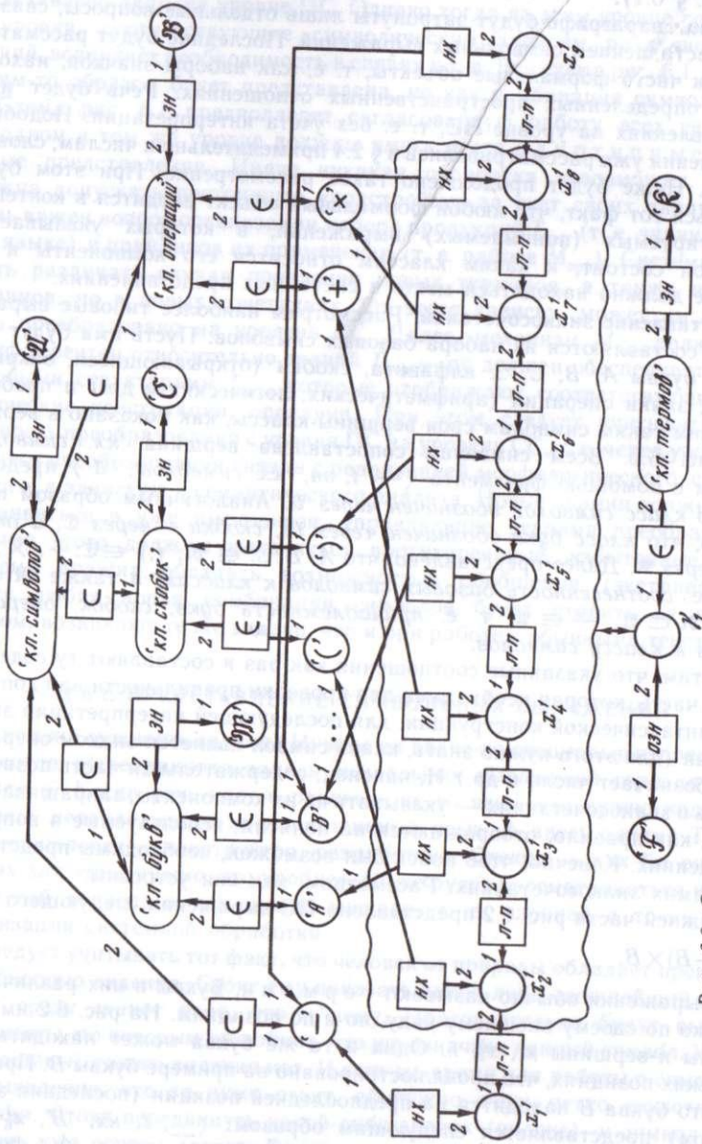


Рис. 6.2. Сеть, представляющая поверхностную структуру (ПС) выражения вида $(A+B)\times B$. *С*-вершина γ_1 соответствует всему выражению (терму)

В дальнейшем будем обозначать сети, представляющие месторасположение символов и их пространственные отношения, через $ПС[\Omega]$, где Ω — есть само выражение. Например, часть сети рис. 6.2, обведенная контурной линией и представляющая отношение между символами в (6.5), обозначается через $ПС[(A+B)\times B]$.

Символьное выражение или любая его часть может рассматриваться как единое целое. Тогда ему сопоставляется своя собственная вершина. В частности, на рис. 6.2 выражению (6.5) сопоставлена вершина γ_1. Она является *с*-вершиной сети, обведенной контурной линией. Напомним, что такие *с*-вершины могут формироваться самой системой — замыканием (см. § 1.4). Это осуществляется, если речь идет о всем выражении (или его части), для которого вводится обозначение, указывается его отношение с другими выражениями (частями) и т. д. Для представления подобной информации используются *с*-вершины.

Например, на рис. 6.2 с помощью *с*-вершины γ_1, соответствующей всему выражению, представлено, что последнее *относится к классу термов, обозначенному через* $\mathfrak{R}$. *Само же выражение обозначено через* $\mathfrak{F}$. Соответствующие фрагменты составляют окрестность *с*-вершины γ_1. При этом обратите внимание, что для представления обозначений использованы различные *о*-вершины *зн* и *озн*. Дело в том, что соответствующие фрагменты связывают различные уровни (см. рис. 6.1).

Пока что речь шла об обозначениях наиболее простого типа. В более сложных случаях требуется использование не отдельных фрагментов, а специального вида сетевых продукций (см. § 2.1 и 5.3). Например, в предложении $(B|X)A$ — *читать*: «*B замещает X в A*» знак | указывает на действие замещения, а B и X — есть компоненты достаточно произвольного выражения A, см. [5 с. 33]. Подобного сорта оговорки учитываются с помощью левой и правой части продукции (см. § 6.3).

Нетрудно видеть, что сеть рис. 6.2 как бы «сшита» из различных кусков — фрагментов, имеющих различное назначение. Одни необходимы сугубо для интерпретации, другие — для поиска и т. д. При этом следует помнить, что данная сеть иллюстрирует лишь определенный способ представления. И таких способов может быть множество. В частности, с помощью специальных фрагментов могут быть указаны номера позиций. Тогда сделается возможным ответ на запросы типа: *Какой символ стоит на второй позиции*? или *На какой позиции стоит символ B*?.

Представление индексных выражений. На рис. 6.2 был проиллюстрирован способ представления букв, не содержащих индексации. Рассмотрим, как же могут быть представлены индексы. Один из способов — по их пространственному расположению относительно основного символа (буквы). Другой способ базируется на использовании специальной *о*-вершины *б. инд*, соответствующей отношению *быть индексом*. Предполагается, что такое отношение выявляется в процессе восприятия — V, см. рис. 6.1. Тогда, например, символ A_i будет представлен в виде фрагмента $\langle \gamma_k, t, б.\ инд, 'I', 'A' \rangle$, где $'I'$ соответствует самому индексу, а γ_k — всему символу A_i. Для представления *верхних индексов* может быть использована другая *о*-вершина — *б. в. инд*. Если у символа несколько индексов, то последние будут представлены с помощью нескольких фрагментов.

Вообще-то, здесь возможны различные способы представления. Символ со сложной индексацией может рассматриваться как комплексный объект,

а сами индексы — как компоненты некоего комплексного отношения. Тогда для представления может быть использован лишь один фрагмент. В конце концов, могут представляться лишь пространственные отношения. Важно отметить, что при любом способе представления самому символу (вместе с его индексами) всегда может быть сопоставлена отдельная вершина типа γ_k, называемая *с*-вершиной (см. § 1.3). Она используется для представления факта нахождения символа на той или иной позиции. При этом один и тот же символ может находиться в нескольких местах. В самом выражении такие символы могут располагаться один сверху другого и т. д. Принципы представления существенно не изменятся. Также для различения будут важны позиции.

Рассмотрим пример. На рис. 6.3 изображена сеть, представляющая более сложное выражение:

$$\frac{A_i+C}{C}. \qquad (6.6)$$

Вершина x_1^1 соответствует *символу* A_i, а x_2^1 — *позиции, на которой он находится*. Вершины x_4^1 и x_6^1 соответствуют *позиции, на которой находится символ С*. *О*-вершина *в-н* соответствует отношению *находиться сверху—вниз*, *о*-вершина '_' — *горизонтальной линии*, а *н*-вершина x_5^1 — *ее позиции*. Если бы в выражении было несколько горизонтальных линий, то имелось бы и несколько *н*-вершин такого типа. Наличие одной *н*-вершины x_5^1 говорит от том, что имеется только одна горизонтальная линия.

Нетрудно обобщить рассмотренный пример на другие случаи. Можно показать, что с помощью сетей введенного типа обеспечивается представление любого выражения — с любым количеством всевозможных знаков,

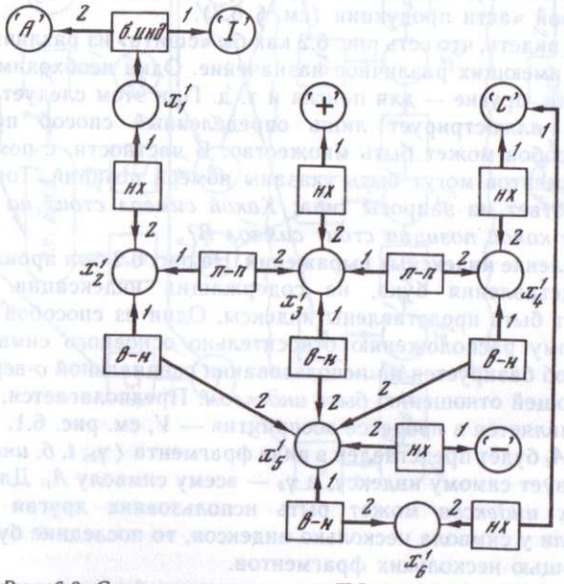


Рис. 6.3. Сеть, представляющая ПС выражения (6.6)

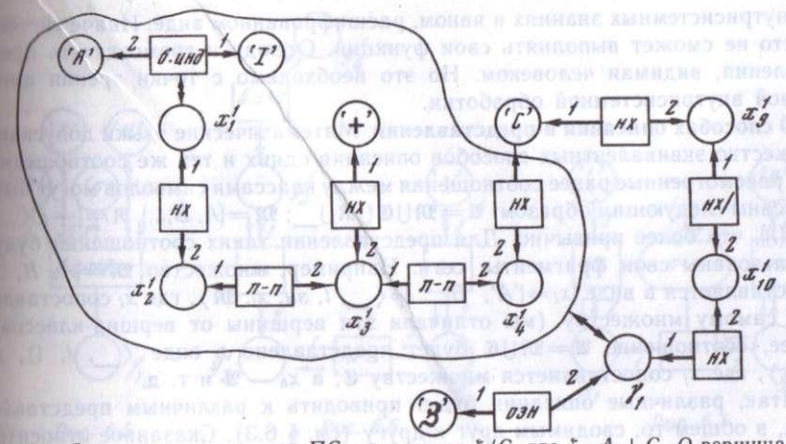


Рис. 6.4. Сеть, представляющая ПС выражения ξ/C, где $\xi \underset{Df}{=} A_i+C$. *О*-вершина *нх/* соответствует комплексному отношению *находиться в позициях через горизонтальную черту*

в том числе относящихся к отдельным символам, как индексы, или к целым выражениям, как, например, знак корня. При этом всегда одни и те же символы, встречающиеся в различных частях выражения, должны каким-то образом различаться. Для этого мы использовали вершины, соответствующие их позициям. Конечно, как правило, один и тот же символ означает нечто одно. Соответственно на уровне СС ему будет сопоставлена одна вершина. Но это не всегда так. Значение может зависеть и от контекста, как в иероглифах. Аналогично один и тот же знак, используемый как индекс, может означать совсем другое, чем в роли основного символа. Тогда на уровне СС ему должны сопоставляться различные вершины.

Следует обратить внимание еще на один существенный момент. Бросается в глаза видимая громоздкость представления. Казалось бы, линейная запись куда проще. Почему бы ее не использовать во внутренних представлениях? Но тогда все описания, вопросы, касающиеся символьных выражений, должны каким-то образом сводиться к такой записи, где многое предполагается, подразумевается. Как выясняется, это сделать чрезвычайно трудно. Записать в линейном виде многие вопросы, связанные с поиском на структурах, не удается. Отсюда необходимость во внутренних представлениях структурного характера.

Более того, человек обладает способностью воспринимать всю информацию, которая имеется в выражении. У машины же такого видения нет. В частности, человек видит, что слева или справа от того или иного символа ничего больше не стоит. Это не говорится, а подразумевается. Именно поэтому при описании какой-либо математической конструкции достаточно записать само символьное выражение и обратить внимание лишь на отдельные стороны. Все остальное человек сам увидит. А за этим «увидит» кроется очень многое. Например, постарайтесь точно описать какое-либо выражение (все, что в нем есть), не приводя его. Получится весьма длинный текст. И все, что в нем выражено, должно быть каким-то образом представлено

во внутрисистемных знаниях в явном, расшифрованном виде. Иначе система просто не сможет выполнять свои функции. Отсюда и громоздкость представления, видимая человеком. Но это необходимо с точки зрения автономной внутрисистемной обработки.

О способах описания и представления. Математические языки допускают множество эквивалентных способов описания одних и тех же соотношений. Так, рассмотренные ранее соотношения между классами символов могут быть записаны следующим образом: $\mathfrak{A}=\mathfrak{M}\cup\mathfrak{C}\cup\mathfrak{R}\cup\_$; $\mathfrak{M}=\{A, B,...\}$, $\mathfrak{R}=\{+,\times,...\}$, $\mathfrak{C}=\{(,)\}$, что более привычно. Для представления таких соотношений будут использованы свои фрагменты, сети. Например, множество $\mathfrak{M}=\{A, B, ...\}$ представляется в виде $[x_i=\{'A', 'B', ...\}]\circ\langle\_, t, зн, x_i, \mathfrak{M}\rangle$, где x_i сопоставляется самому множеству (мы отличали эти вершины от вершин-классов). Далее, соотношение $\mathfrak{A}=\mathfrak{M}\cup\mathfrak{C}$ будет представлено в виде $\langle\_, t, \cup, x_i, x_j, x_k\rangle$, где x_j сопоставляется множеству $\mathfrak{C}$, а x_k — $\mathfrak{A}$ и т. д.

Итак, различные описания будут приводить к различным представлениям, в общем-то, сводимым друг к другу (см. § 6.3). Сказанное относится и к конструкциям формальных выражений, которые также могут описываться по-разному. Например, можно (вместо варианта расположения символов по позициям) взять другой вариант, сводимый к первому. Можно считать, что имеются классы букв, а в выражениях используются их экземпляры. Словом, имеются как бы коробки, в каждой из которых содержатся фишки с записью одной и той же буквы. А выражения составляются из таких фишек, которые располагаются слева—направо, сверху—вниз и т. д. Тогда на рис. 6.2 и 6.3 вместо *о*-вершины *них* будет другая — $\in$, соответствующая отношению *принадлежности к классу*. Сами же представления существенно не изменятся.

Характерным для математических текстов является постоянное укрупнение объектов, понятий, категорий, введение комплексных отношений. Как уже говорилось, удобным оказывается рассмотрение некоторых выражений и их частей как самостоятельных единиц с введением необходимых обозначений. Соответственно возникает потребность в комплексных отношениях, с помощью которых связываются отдельные выражения, удается простым способом описать достаточно сложные связи. Все это (с нужной точностью) может быть представлено в языке семантических сетей (СЯ). Последние очень чувствительны к способу выражения информации, используемым категориям, что делает возможным ее представление без искажений и потерь.

Рассмотрим пример. Переиначим выражение (6.6). Обозначим в нем A_i+C через $\mathfrak{Z}$. Тогда само выражение примет более простой вид $\mathfrak{Z}|C$. Будем считать, что символы $\mathfrak{Z}$ и C связаны между собой отношением *находиться в позициях через горизонтальную черту*, т. е. сама черта не рассматривается как отдельный символ. Тогда требуется уже другое представление, что изображено на рис. 6.4. Упомянутому отношению сопоставлена *о*-вершина *нх/*. Части выражения, обозначенной через $\mathfrak{Z}$, сопоставлена *с*-вершина γ_1, с помощью которой представлены ее отношения с символом C.

Остановимся еще на одном примере. В математических текстах весьма емким и полезным является понятие **«слово в заданном алфавите»**. Оно может рассматриваться как специального вида отношение, связывающее буквы. Тогда слову сопоставляется следующий фрагмент:

$$\langle\gamma_1, t, 'б.\ словом', x_1^1, x_{2,...}^1\rangle,$$

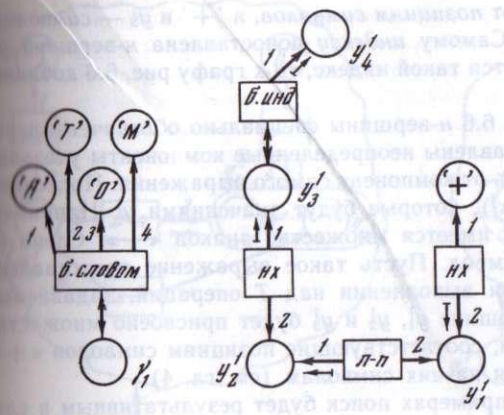


Рис. 6.5 Рис. 6.6. Рис. 6.7.

Рис. 6.5. Фрагмент, представляющий слово АТОМ

Рис. 6.6. Граф, соответствующий *индексу (б. инд) символа, стоящего слева от знака* +

Рис. 6.7. Граф, сопоставленный *ближайшей букве В, находящейся справа от открывающейся скобки*

где x_i^1 (для $i=1, 2, ...$) соответствует символу, стоящему на i-й позиции, γ_1 — всему слову, а *б. словом* — отношению *быть словом, состоящим из линейной последовательности символов*. При этом местность данного отношения не фиксируется. Далее предполагается, что на позициях в слове могут стоять только символы (буквы, скобки, знаки операций), но не пробелы, см. рис. 6.5.

Пользуясь подобного сорта фрагментами, можно в значительной степени упростить сети, служащие для представления простых символьных выражений. Так, выражение $(A+B)\times B$ может быть представлено в виде

$$\langle\gamma_1, t, 'б.\ словом', '(', 'A', '+', 'B', ')', '\times', 'B'\rangle, \qquad (6.7)$$

т. е. с помощью лишь одного фрагмента. Вершины типа γ_1 могут служить для представления последовательности выражений (слов алфавита). Однако возможности результативного использования подобных представлений оказываются ограниченными. Например, в тексте, приведенном в самом начале параграфа, говорилось о знаке, написанном (находящемся) слева от A. Пусть сказанное относится к выражению, представленному с помощью фрагмента (6.7). Оказывается, что выделить в нем вершину, соответствующую указанному знаку, очень не просто. Нужно каким-то образом связать отношение *находиться слева* с позициями фрагмента. Для этого требуется преобразование представлений. Все равно, на базе фрагмента (6.7) вначале должна быть получена сеть рис. 6.2, а затем уже сделается возможным поиск (преобразовать вопрос в форму типа (6.7) нельзя).

Выделение компонент. Как уже говорилось, для представления указаний на разного рода объекты — компоненты символьных выражений — используются семантические графы. Последние задают операции нахождения или выделения соответствующих вершин. На рис. 6.6 изображен граф, соответствующий *индексу символа, стоящего непосредственно слева от знака «+»*.

Вершины y_1^1 и y_2^1 соответствуют *позициям символов*, а '+' и y_3^1 — *символам, стоящим на этих позициях*. Самому *индексу* сопоставлена *н*-вершина y_4^1. В частности, если запрашивается такой индекс, то к графу рис. 6.6 добавляется *з*-сеть $[y_4^1 := ?]$.

Отметим, что в графе рис. 6.6 *н*-вершины специально обозначены через y_i^1, с помощью которых представлены неопределенные компоненты указания (вопроса); их следует отличать от компонент самого выражения. Последним сопоставляются *н*-вершины (x_i^1), которые будут значениями y_i^1. Например, пусть в исходном выражении имеется множество знаков «+». Слева от каждого знака стоит свой символ. Пусть такое выражение представлено с помощью сети T. Тогда при выполнении над T операций, задаваемых графом рис. 6.6, каждой *н*-вершине y_1^1, y_2^1 и y_3^1 будет присвоено множество значений. Ими будут вершины, соответствующие позициям символов «+», левым соседям и находящимся на них символам (см. гл. 4).

В рассматриваемых выше примерах поиск будет результативным в случае, когда структура графа, сопоставленного тому или иному объекту, в какой-то степени повторяет структуру сети, в которой представлена информация о подобного сорта объектах, например сети-знания. Однако это не всегда так. Сравнительно часто могут иметься в виду объекты или компоненты, связанные цепочками отношений. При этом число звеньев цепочки может быть не указано. В сети-знаниях допускаются различные варианты. Для представления подобной информации в гл. 4 было введено понятие р е к у р с и в н о г о графа. Пример такого графа изображен на рис. 6.7. Граф сопоставлен *ближайшей букве B, находящейся справа от открывающейся скобки*. Вершина y_2^1 сопоставлена позиции такой буквы, а *н*-вершина $\hat{y}_3$ указывает на возможность цепочки отношений типа *находиться слева—направо*. Предполагается, что между скобкой и буквой B могут быть другие символы.

Остановимся на более **сложных примерах**. На рис. 6.8 изображен граф, сопоставленный *букве B, находящейся в выражении $\mathcal{F}$ рядом с закрывающейся скобкой*. Проиллюстрировано два способа изображения — один с учетом введенных ранее обозначений (см. слева), другой детализированный. При этом отношение *быть рядом* рассматривается как *находиться непосредственно слева или справа*, что представляется с помощью *з*-сети $y_2^1 := \{y_3^1, y_4^1\}$ (хотя то же самое можно было бы представить с помощью фрагмента, соответствующего дизъюнкции, см. § 1.3). Самому выражению $\mathcal{F}$ сопоставлена *с*-вершина γ_1. Нетрудно видеть, что если выполнить операции, задаваемые графом рис. 6.8, над сетью рис. 6.2, то будет получено $[y_2^1 := x_5^1]$, где x_5^1 соответствует *одной из позиций, занимаемой буквой B (или одному их ее экземпляров)*.

На рис. 6.9 приведен еще один пример. На нем изображен граф, сопоставленный *первой позиции слова* Ф. Его фокус y_1^1 сопоставлен *той позиции слова* Ф, *слева от которой ничего нет*, на что указывает символ $\hat{\lambda}$. Если слово представлено в контексте пробелов, как это имело место на рис. 6.2, то граф будет соответствовать *позиции левого пробела*. Если пробелы не представлены — то *позиции первого символа*. В дальнейшем такие графы будем изображать, пользуясь мнемоническими обозначениями, как это показано в правой части рис. 6.9.

Средства активного воздействия. Рассмотрим случай, когда описания несут новую информацию. Конечно, понятие новизны многоаспектно. Любая

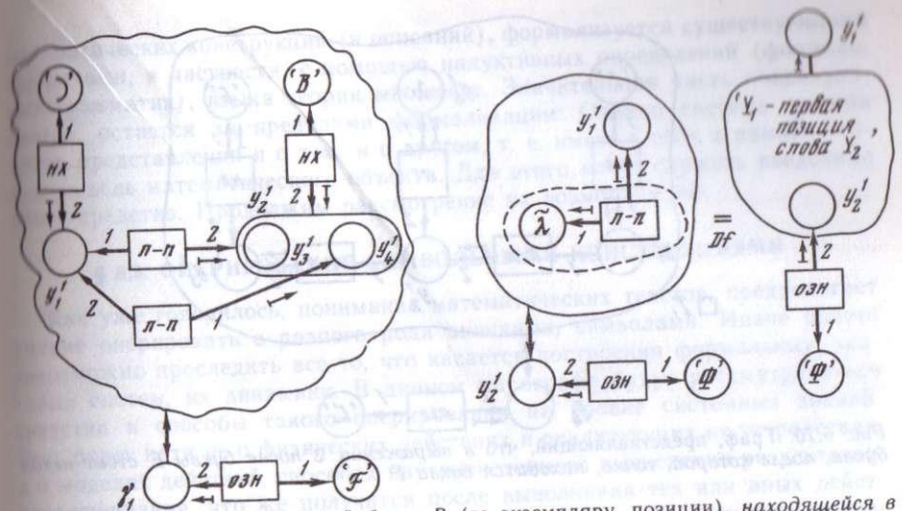


Рис. 6.8. Граф, сопоставленный *букве B* (ее экземпляру, позиции), *находящейся в выражении $\mathcal{F}$ рядом с закрывающейся скобкой*

Рис. 6.9. Граф, сопоставленный *первой позиции слова* Ф. В правом изображении использованы мнемонические обозначения

часть математического текста в какой-то степени является новой. Как устроен такой текст? Как правило, вначале даются исходные посылки, вводятся обозначения. Далее следуют утверждения. Например: *Пусть F — функция двух переменных, а D — область ее определения. Обозначим (или тогда ...)* ... И новые обозначения, и посылки, и утверждения должны каким-то образом встраиваться во внутрисистемные представления, вызывая соответствующие изменения, дополнения. При этом каждое утверждение, высказывание, определение должно лечь на свою полочку (см. § 6.1). Новые обозначения, основанные на введенных ранее, должны пополнять лингвистические знания. Новые утверждения должны находить свое место среди представлений об истинных высказываниях, теоремах и т. д. Система сама должна постоянно следить, не было ли этого ранее, куда нужно поместить то, о чем говорится, как связать с тем, что уже было сказано ранее. В § 5.2 были введены специальные графы, называемые активными. Они ориентированы на представление сообщений с учетом акцентации на новую информацию. Такие графы задают не только операции поиска и проверки, но и формирования новых фрагментов.

Например, пусть системе сообщается, что *объединение последовательности счетных множеств счетно*. Такое утверждение отображается на уровень *СС* и представляется в виде графа, задающего операции поиска на сети-знаниях T о теоремах. При эттом некоторых фрагментов графа (с вершинами, отмеченными знаком $\square_\phi$) в T может и не быть. Тогда предполагается действие достройки. Пусть в знаниях T есть сеть, представляющая понятие *объединения последовательности счетных множеств*. Тогда к этой сети просто подсоединяется фрагмент, представляющий свойство *быть счетным*. Такие действия обеспечиваются самим обязательным графом, задающим соответствующие операции.

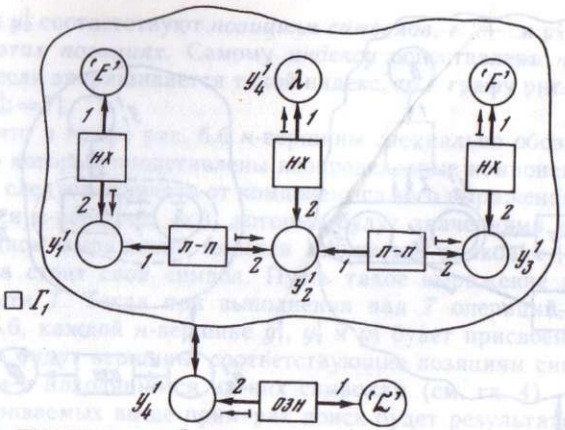


Рис. 6.10. Граф, представляющий, что *в выражении* Ω *после буквы E стоит некая буква, после которой, точно, находится буква F*

Продолжим рассмотрение роли активных графов. Остановимся на случае представления описаний, в которых речь идет о неких новых компонентах символьного выражения. На рис. 6.10 изображен пример. Представлено следующее: *Имеется выражение* Ω; *в нем после буквы E стоит некая буква, после которой находится буква F*. Граф задает операции поиска значений *н*-вершин y_4^1, y_1^1 и y_2^1. Если о букве, стоящей после *E*, системе известно, то y_4^1 и y_2^1 получат непустые значения. В противном случае значения будут пустыми. Тогда формируются резервные вершины x_i^1 и x_j^1 и фрагменты $\langle \_\_, t, нх, x_i^1, x_j^1 \rangle \circ \langle \_\_, t, л\text{-}п, x_k^1, x_i^1 \rangle$, где x_k^1 — значение y_1^1. То же самое осуществляется, если системе ничего неизвестно об упомянутой букве *F*. Тогда формируются фрагменты со своими вершинами. Только на месте x_j^1 будет стоять '*F*'. Эти фрагменты пополняют представления о символьном выражении.

Напомним, что любой активный граф может быть перенесен во внутрисистемные знания. Тогда он начинает играть роль некоего п о с т у л а т а, с помощью которого можно пополнять и корректировать представления о входных сообщениях. Соответствующая информация будет иметь условный оттенок. Например, если граф рис. 6.10 перенести в знания, то будет уже представлено: *Если речь идет о выражении* Ω *и в нем упоминается буква E, то после нее должна стоять некая буква, после которой находится F*. Подобные графы были названы обязательными. С их помощью представляется некий костяк, в рамках которого должна находиться вводимая информация. Если что-то выходит за рамки представлений, то будут предприниматься действия изменения, корректировки. В математических текстах роль такого костяка играют не только аксиомы, постулаты, но и многочисленные допущения — что является правильной записью, а что нет, какого сорта обозначения недопустимы и многое другое. Для их представления могут быть использованы обязательные графы и граф-продукции. Они образуют так называемые обязательные знания, служащие для представления допустимых сочетаний компонент, возможных свойств и соотношений и т. д.

Следует отметить, что далеко не все, что связано с допустимостью

математических конструкций (и описаний), формализуется существующими средствами, в частности, с помощью индуктивных определений (формальных грамматик), языка теории множеств. Значительная часть подразумевается, остается за пределами формализации. Однако система должна иметь представления и о том, и о другом, т. е. иметь у себя в памяти полную модель математического объекта. Для этого могут служить введенные ранее средства. Продолжим рассмотрение их возможностей.

§ 6.3. ОПЕРИРОВАНИЕ СИМВОЛЬНЫМИ КОНСТРУКЦИЯМИ

Как уже говорилось, понимание математических текстов, предполагает умение оперировать с разного рода значками, символами. Иначе просто невозможно проследить все то, что касается построения формальных, знаковых систем, их динамики. В данном параграфе будут рассматриваться средства и способы такого оперирования на уровне системных знаний. Речь будет идти не о физических действиях и реализующих их устройствах, а о моделях действий, способах внутрисистемного вычисления результатов, прослеживания, что же получится после выполнения тех или иных действий. Для этого могут быть использованы типовые конструкции сетей и графов, что будет рассматриваться ниже. Будут приводиться наиболее типовые примеры, которые позволяют составить общую картину.

Оперирование математическими объектами предполагает их постоянные преобразования. В соответствии со схемой рис. 6.1 будем различать несколько их видов. Первый — это преобразование неинтерпретированных выражений, которое сводится к замене одних знакосочетаний на другие. Подобные преобразования отображаются и реализуются на уровне поверхностных структур (ПС). В более сложном случае могут предполагаться разного рода конструктивные изменения, например составление из одних значков других. Способов составления может быть множество. Все такие действия должны «прослеживаться» системой, что осуществляется на уровне ПС.

Другой вид преобразований относится к значениям, смыслам. Одно и то же можно выразить множеством различных способов с использованием различных категорий, отношений. При внутрисистемной обработке возникает необходимость сведения одних конструкций к другим, что предполагает соответствующие преобразования на уровне семантических структур (СС). Сюда же относятся некоторые виды эквивалентных преобразований (связанные с изменением базиса отношений), а также преобразования, имеющие целью расшифровку действий, прослеживание вызываемых ими изменений.

И, наконец, еще один вид преобразований связан с интерпретацией, когда выявляется, что же означает то или иное выражение, что будет рассматриваться в следующем параграфе.

Важно отметить, что для реализации указанных преобразований могут быть использованы одни и те же средства: обязательные графы, семантические продукции. Для иллюстрации их возможностей остановимся вначале на втором виде преобразований.

Преобразование представлений. Неоднозначности. На рис. 6.11 изображена сетевая продукция, представляющая, что *если сверху от горизонтальной черты* (Y_3) *находится объект* Y_1, *а снизу* — Y_2, *то это значит* ($\equiv$),

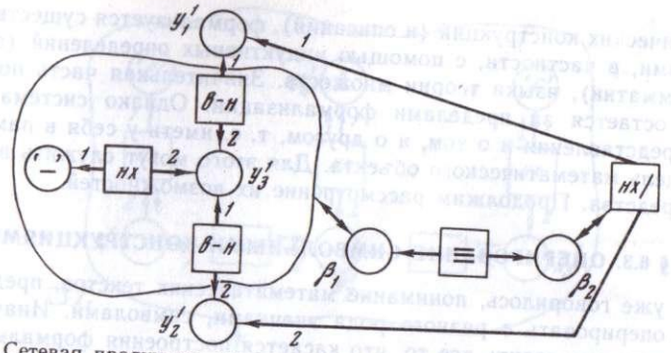


Рис. 6.11. Сетевая продукция, представляющая пояснение, что значит, *находиться через горизонтальную черту* (*нх/*)

что объекты Y_1 *и* Y_2 *находятся через горизонтальную черту* (*нх/*). Фактически связываются два разных представления. Данная продукция может быть применена к сети рис. 6.3, представляющей выражение (6.6). При многократном применении получится сеть, представляющая, что *и символ* A_i, *и знак «+», и буква С находятся через горизонтальную черту от другой буквы С, т. е. другого экземпляра.*

Обратим внимание на одну существенную деталь. Если в сети, к которой применяется продукция, представлена лишь одна горизонтальная черта, то никаких неоднозначностей не возникает. При многократном применении в продукции рис. 6.11 *н*-вершина y_3^1 будет принимать одно и то же значение. Возможно и обратное ее применение, т. е. уже к полученной сети, но в другую сторону — когда левая и правая части продукции меняются местами. Тогда на базе y_3^1 также будет формироваться одна и та же «резервная» вершина (так как известно, что всего одна линия). В результате такого применения будет получена прежняя сеть.

Если же допускается множество горизонтальных линий, то возникают неоднозначности. Сеть, полученная в результате прямого применения продукции, может интерпретироваться по-разному. Соответственно при обратном применении может быть не получена та же сеть. В самом деле (см. пример на рис. 6.11), после выполнения преобразований над сетью (см. рис. 6.2) уже будет неясно, находятся ли символы A_i, $+$ и C через одну и ту же горизонтальную черту от C или через разные. Допускается как выражение (6.6), так и

$$\frac{A_i}{C} \quad + \quad \underline{C}\,. \tag{6.8}$$

Словом, при прямом применении продукции сети, соответствующие и тому, и другому выражению, будут преобразованы в одну и ту же результирующую сеть. Если же к последней снова применять продукцию рис. 6.11 (но уже в другую сторону), то может быть получена любая упомянутая сеть. Все определяется способом присвоения значений *н*-вершине y_3^1. Если ей присваивать одно и то же значение, то получится одна сеть, а если различные, то другая. Отсюда — источник альтернатив.

Конечно, приведенный пример может показаться не совсем удачным.

Выражение (6.8) с виду не привычно. Любой скажет, что это неправильная запись. Ее можно всегда скорректировать. Но пусть в (6.8) снизу (в знаменателе) имеется несколько различных символов. Тогда возникают указанные ранее неоднозначности, которые уже записываются в виде правильных выражений. Итак, **с укрупнением категорий становится возможной потеря информации**, возникновение альтернатив. Их количество будет возрастать с увеличением количества неопределенных компонент в определяющей части какого-либо понятия или отношения. Таким компонентам сопоставляются *н*-вершины левой части соответствующей продукции.

В общем случае с помощью сетевых продукций (на уровне СС) могут быть представлены более сложные виды связей, например, когда какое-либо понятие в контексте указанных отношений выражает то же самое, что и другое понятие в контексте его отношений. Применением соответствующих продукций обеспечиваются преобразования групп отношений, относящихся к различным базисам. Здесь также при применении даже одной продукции в обе стороны можно не получить исходной сети. Возникают альтернативы, варианты. И их количество будет возрастать с увеличением числа *н*-вершин, входящих только в левую или только в правую часть продукции. Указанный эффект будет усиливаться с возрастанием числа используемых продукций.

Разумно допустить, что отмеченный эффект является одним из источников тех неоднозначностей, неопределенностей, которые постоянно возникают в диалоге. Различные люди, владеющие одним и тем же языком, пользуются различными категориями, понятиями. В процессе общения приходится постоянно подлаживаться, т. е. сводить категории собеседника к привычной для себя форме. Здесь и возникают неодозначности, альтернативы. Сказанное в неменьшей степени относится и к математическим языкам, которые также допускают многообразие способов выражения.

Указанные неоднозначности приобретают принципиальный характер в области, занимающейся разработкой систем перевода и лингвистических процессоров. В каждом языке имеется большое количество специфических понятий, для которых нет эквивалентов в другом языке. Отсюда необходимость введения их в контексте с привлечением других понятий, категорий. Но последние в каждом конкретном случае должны уточняться. Здесь и возникают отмеченные альтернативы. Нужно уметь каким-то образом конкретизировать те новые объекты, которые возникают в процессе сведения категорий. Для этого требуются специальные семантические фильтры (обязательные знания), организация внешней активности с целью уточнения. Только таким способом можно бороться с неоднозначностями.

При использовании лишь синтаксических методов перевода подобных проблем, естественно, не возникает. Но с развитием методов, ориентированных на семантическую компоненту, придется учитывать отмеченный эффект. В общем-то он имеет место и в том случае, когда перевод осуществляют люди. Видимо, указанные неоднозначности являются одним из факторов, определяющих почему при переводе какой-либо фразы вначале на один язык, затем на другой, на третий, как правило, получается нечто совсем другое.

Следует отметить, что для учета семантической компоненты весьма перспективно использование наборов сетевых продукций, на базе которых могут строиться достаточно мощные лингвистические процессоры. Экспери-

ментальные исследования показывают [14], что таким способом могут осуществляться преобразования на уровень СС достаточно сложных предложений ЕЯ. На этом уровне за счет системных знаний могут устраняться многие неоднозначности. Более того, такой уровень может играть роль языка-посредника. Отсюда — один из подходов к построению системы перевода. Последняя рассматривается, как состоящая из двух лингвистических процессоров. Первый осуществляет преобразование предложений одного языка на уровень СС, а второй — обратное преобразование, но уже на другой язык. В рамках подобного подхода допускаются широкие возможности исследований, в том числе природы неоднозначностей, что может осуществляться на достаточно точной основе.

Эквивалентные преобразования. Рассмотрим преобразования, осуществляемые на уровне СС. В качестве примера возьмем а р и ф м е т и ч е с к и е выражения. Будем различать несколько видов преобразований. Один из них был рассмотрен в § 5.4 и сводился к изменению направленности стрелок порядка. Напомним, что такое изменение соответствует некоторым видам синтаксических преобразований, в частности, связанных с перестановкой слагаемых, расстановкой скобок и др. Стрелки порядка определяют последовательность вычисления и соответственно отражают приоритеты операций и скобочную символику.

Рассмотрим эквивалентные преобразования, связанные с изменением операций (базиса отношений). Будем различать преобразование функций (вычисляемое значение остается тем же) и выражений (типа равенства) или конструкций.

Для представления преобразований первого вида будем использовать граф-продукции (5.20). На рис. 6.12 изображен пример. Представлено равенство типа $(Y_1+Y_2)\times Y_3=Y_1\times Y_3+Y_2\times Y_3$. Имеется в виду равенство результатов, вычисленных значений. Им сопоставлены *н*-вершины y_5^1 и y_6^1 со своими окрестностями Γ_1 и Γ_2. Другим переменным (аргументам) сопоставлены *н*-вершины y_1^1, y_2^1 и y_3^1. Стрелки $\mapsto$ расставлены в соответствии со скобочной символикой. Продукция может применяться к любой сети, представляющей то или иное арифметическое выражение. И если в этой сети имеется вершина x_i^1 с окрестностью, аналогичной Γ_1, то к x_i^1 добавляется другая окрестность, сформированная на базе Γ_2. При этом добавляемая окрестность может замещать прежнюю. Более того, продукция может применяться в любую сторону.

Эквивалентные преобразования второго вида относятся к конструкциям типа «равенство». Для них используются сетевые продукции вида $T_1\equiv T_2$, где с помощью T_1 и T_2 представлены сами конструкции, а $\equiv$ соответствует *тождеству*. Если необходимо учитывать способ вычисления, то на месте T_1 и T_2 будут стоять графы. Получится граф-продукция вида (5.25).

Пример такой граф-продукции изображен на рис. 6.13. В ее левой части (Γ_1) представлено равенство $(Y_1+Y_2)\times Y_3=Y_5$, а в правой (Γ_2)—$Y_1\times Y_2=Y_5-Y_2\times Y_3$. С помощью стрелок порядка указано направление вычисления. При этом в случае вычисления значений промежуточной переменной возникает необходимость в фрагменте $\langle \_\_, t, =, y_6^1, y_8^1\rangle$, представляющем равенство. Он имеется в Γ_2. В общем-то, стрелки порядка могут быть изъяты, а *н*-вершины y_6^1 и y_8^1 — объединены в одну. Тогда получится обычная сетевая продукция (см. § 2.1). На базе нее могут формироваться различные граф-продукции типа рис. 6.12. Из эквивалентности конструк-

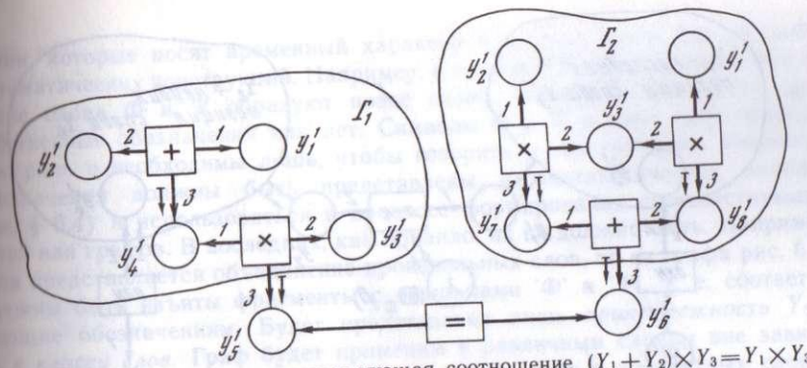


Рис. 6.12. Граф-продукция, представляющая соотношение $(Y_1+Y_2)\times Y_3=Y_1\times Y_3+Y_2\times Y_3$

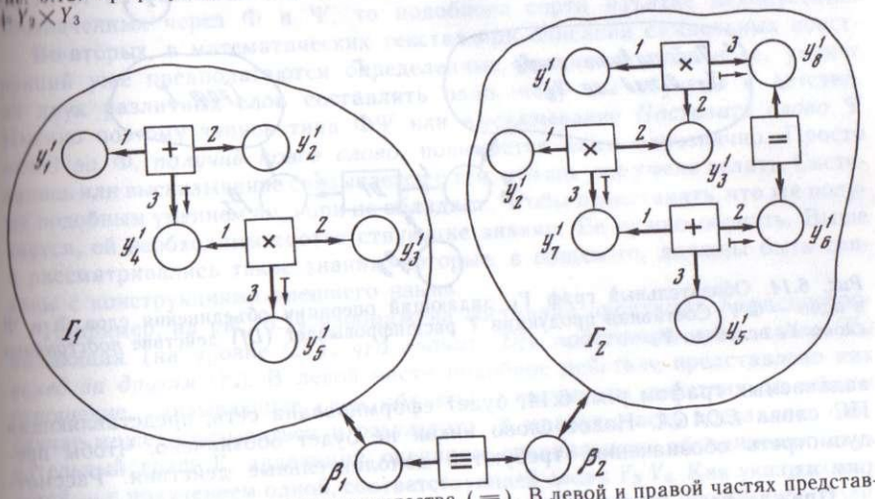


Рис. 6.13. Пример представления тождества ($\equiv$). В левой и правой частях представлены равенства

ций всегда следует эквивалентность однотипных объектов. Хотя обратное не всегда справедливо.

Действия композиции. Перейдем к рассмотрению преобразований, осуществляемых на уровне ПС. Начнем с простого примера. На рис. 6.14 изображен обязательный граф Г, который представляет указание об *объединении двух слов, обозначенных через* Ф *и* Ψ, *в одно* — ΦΨ. Операции, задаваемые таким графом, выполняются над сетью, представляющей ПС различных слов. В результате находятся сети, соответствующие словам Ф и Ψ. К ним достраивается фрагмент $\langle \_\_, t, л\text{-}п, x_i^1, x_j^1\rangle$, представляющий, что *после последней позиции* (X_i) *слова* Ф *следует* (*л-п*) *первая позиция слова* Ψ. Образовавшаяся сеть и будет соответствовать слову ΦΨ.

Например, пусть Ф и Ψ — обозначения слов *ЕС* и *АСА*, представленных с помощью сетей T_1 и T_2. Тогда выполнением операций над $T_1\circ T_2$,

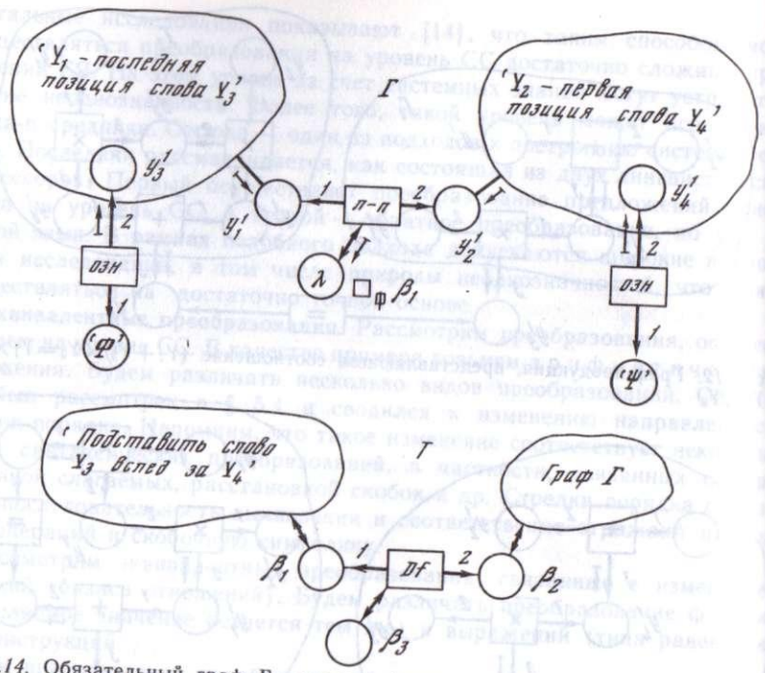


Рис. 6.14. Обязательный граф Γ_1, задающий операции объединения слов Ф и Ψ в одно — ФΨ. Составная продукция T расшифровывает (Df) действие *подстановки слова* Y_4 *вслед за* Y_3

задаваемых графом рис. 6.14, будет сформирована сеть, представляющая ПС слова *ECACA*. Новое слово никак не будет обозначено. Чтобы предусмотреть обозначения, требуются дополнительные действия. Рассмотрим их.

Прежде всего в граф Г рис. 6.14 необходимо ввести фрагменты, задающие операции п е р е з а м ы к а н и я. Новое слово должно рассматриваться как единое целое. Ему должна быть сопоставлена своя c-вершина и представлено, что *частями этого слова являются* Y_3, Y_4, *расположенные слева—направо* (B_1). Подобные действия задаются фрагментами $\langle \beta_2, t, \_, y^1_3, y^1_4, \beta_1 \rangle \circ [\square_\phi \beta_2 := \lambda]$. На базе β_2 формируется c-вершина. В частности, выполнением действий (операций) над ПС *ECACA* будет сформирована c-вершина этой сети. Пусть это γ_i (она ранее не должна входить в сеть). Далее система должна воспроизвести (сконструировать) новое обозначение, отличное от тех, которые были известны ей ранее. Пусть такому обозначению (*озн*) сопоставлена вершина 'D'. На основе указанных вершин должен быть сформирован фрагмент $\langle _, t, $ *озн*, 'D', $\gamma_i \rangle$. Последние операции задаются следующим образом: $\langle \_, t,$ *озн*, $y^1_6, \beta_2 \rangle \circ [\square y^1_6 := k]$, где $з$-сеть указывает на необходимость конструирования (k) обозначения. Такие фрагменты должны быть введены в граф рис. 6.14.

Следует обратить внимание на следующие **моменты**. Во-первых, в математических текстах широкое распространение находят абстрактные обозначения, которые носят временный характер и служат лишь для описания математических конструкций. Например, в записи ФΨ совершенно не важно, какие слова Ф и Ψ образуют новое слово. Имеют ли эти слова свои собственные обозначения или нет. Символы Ф и Ψ играют вспомогательную роль и необходимы лишь, чтобы говорить о конструкциях. Подобные обозначения должны быть представлены в лингвистических знаниях (см. § 6.4) и использоваться в процессе формирования соответствующих сетей или графов. В последних, как правило, их не должно быть. Например, если представляется объединение произвольных слов, то из графа рис. 6.14 должны быть изъяты фрагменты с вершинами 'Ф' и 'Ψ', т. е. соответствующие обозначениям. Будет представлена лишь *принадлежность* Y_3 *и* Y_4 *к классу слов*. Граф будет применим к различным словам вне зависимости от их обозначений. Конечно, если речь идет о конкретных словах, обозначенных через Ф и Ψ, то подобного сорта изъятие недопустимо.

Во-вторых, в математических текстах при описании символьных конструкций уже предполагаются определенные умения, в том числе, умение из двух различных слов составлять одно, чему мы обучены с детства. Именно поэтому запись типа ФΨ или высказывание *Поставить слово* Ψ *вслед за* Ф, *получив новое слово*, понимается нами однозначно. Просто запись или высказывание связывается с тем, что мы уже умели делать. Система подобным умением априори не обладает. Чтобы представить, что же получается, ей необходимы соответствующие знания. Ее нужно обучать. Выше и рассматривались такие знания, которые, в общем-то, должны быть связаны с конструкциями внешнего языка.

Например, на рис. 6.14 изображена составная продукция T, расшифровывающая (на уровне СС), *что значит* (Df), *подставить одно слово* (Y_3) *вслед за другим* (Y_4). В левой части подобное действие представлено как отношение, связывающее свои объекты — аргументы действия (в общем случае может связываться и результат). А правую часть составляет обязательный граф Г, задающий операции специального объединения двух сетей — с получением одной, соответствующей слову $Y_3\,Y_4$. Как указывалось ранее, обозначения типа Ф и Ψ здесь не должны играть какой-либо роли. В правой части продукции должны отсутствовать соответствующие фрагменты.

Продукция T рис. 6.14 может быть применена к любой сети, представляющей описание каких-либо действий. И если упоминается действие *подстановки* (указанного типа), то продукция делается применимой. Формируется граф. Задаваемые им операции выполняются уже над другой сетью — представляющей ПС слов. В результате на уровне СС будут «прослеживаться» изменения, построения, о которых идет речь в описаниях.

В-третьих, нам кажется просто то, что мы хорошо умеем делать. А что за этим стоит, мало кого интересует, разве что специалистов в области «искусственный интеллект». Как-то само собой соответствующие структуры переносятся с уровня сознания на рефлекторный уровень — действие доводится до автоматизма. И нет труда в его выполнении. В рамках систем с СП подобный перенос — пока еще нерешенная проблема (см. § 3.1). Все действия должны поддерживаться структурами знаний. И как выясняется, то, что просто для нас (мы легко это можем делать), не обязательно будет простым с точки зрения внутрисистемной обработки. Для поддержания, казалось бы, «простых» преобразований, действий зачастую могут

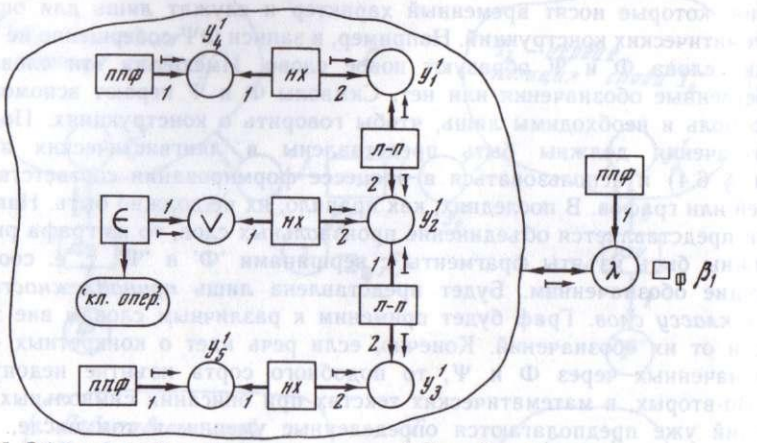


Рис. 6.15. Обязательный граф, представляющий *Если* Y_4 *и* Y_5 *— правильно построенные формулы (ППФ), а* $\vee$ *— знак операции, то* $Y_4 \vee Y_5$ *— ППФ*

требоваться достаточно сложные структуры знаний. Например, если детализировать изображение рис. 6.14, то полученный граф будет не столь простым. Хотя его функции элементарны (для нас). Это обстоятельство в дальнейшем не следует упускать из виду.

И, наконец, в-четвертых, в математических текстах слишком многое умалчивается, подразумевается. Именно поэтому их чтение требует специальной подготовки. При рассмотрении внутрисистемных представлений все приходится восстанавливать, выражать в явном виде. Отсюда — большое количество пояснений, оговорок. Некоторые из них могут восприниматься как «масло масляное». Более того, возникает видимость «громоздкости» используемых конструкций. Казалось бы, куда лучше для целей представления использовать значительно более емкие языки математической записи. Но еще раз отметим, что это рассуждения человека, судящего по себе — как ему просто. При разработке же автономных систем разработчик должен думать, как проще системе (см. § 1.2 и 6.2).

Индуктивные определения. В математических текстах часто используются определения, с помощью которых вводятся конструктивные объекты, задается тот или иной способ их анализа и синтеза. Будем различать определения двух типов. Первые относятся к формальным конструкциям, например: *Если* Ф *и* Ψ *— слова, то* ФΨ *— слово* или *Если* $\mathfrak{A}$ *и* $\mathfrak{B}$ *— термы, а* δ *— знак операции, то* $\mathfrak{A}\,\delta\,\mathfrak{B}$ *— терм*. Предполагается умение пользоваться такими определениями для построения математических объектов, анализа правильности их конструкции, а также соотнесенности конструкций к тем или иным классам. При этом не обязательно знать, что означают сами объекты.

Определения второго типа относятся к содержанию, смыслу. В них значительную роль играет семантическая компонента, например: *Если* A *и* B *— целые величины, то* $A+B$ *— целая величина*. Предполагается, что A и B — есть обозначения величин или переменных с целыми значениями. Под $A + B$ понимается не набор символов, а результат выполнения операции,

который является целой величиной. Подобные определения могут использоваться для «вычисления» типов данных, вида результата.

Для представления определений обоих типов могут быть использованы введенные ранее средства — обязательные графы, продукции. Они образуют знания, которые необходимы для «поддержания» упоминавшихся ранее умений — анализировать, конструировать. При этом определения первого типа представляются на уровне ПС (т. е. рассматриваются как чисто формальные преобразования), а второго типа — на уровне СС. Остановимся на определениях **первого** типа.

На рис. 6.15 изображен обязательный граф, представляющий следующее: *Если* $\mathfrak{A}$ *и* $\mathfrak{B}$ *— правильно построенные формулы* (ППФ), *а* $\vee$ *— знак логической операции, то* $\mathfrak{A}\,V\mathfrak{B}$ *— есть* ППФ. В таком определении подразумевается, что символы $\mathfrak{A}$ и $\mathfrak{B}$ являются обозначениями достаточно произвольных формул (ППФ), из которых составляется единая конструкция $\mathfrak{A} \vee \mathfrak{B}$. Эти символы, как уже говорилось, играют вспомогательную роль и служат лишь для построения самой конструкции. Все это учтено в графе рис. 6.15, где *н*-вершины y_4^1 и y_5^1 соответствуют исходным ППФ, а y_1^1 и y_3^1 — их позициям. Сами обозначения не представлены. *Быть ППФ* рассматривается как свойство, которому сопоставляется фрагмент вида $\langle \_, t, ппф, y_i^1 \rangle$. Вершина β_1 соответствует всей конструкции. Она является *с*-вершиной сети, представляющей последовательность расположения частей данной конструкции, т. е. ее ПС.

Граф рис. 6.15 может применяться к ПС любого выражения. Пусть выполняются соответствующие условия, т. е. в выражении имеется часть

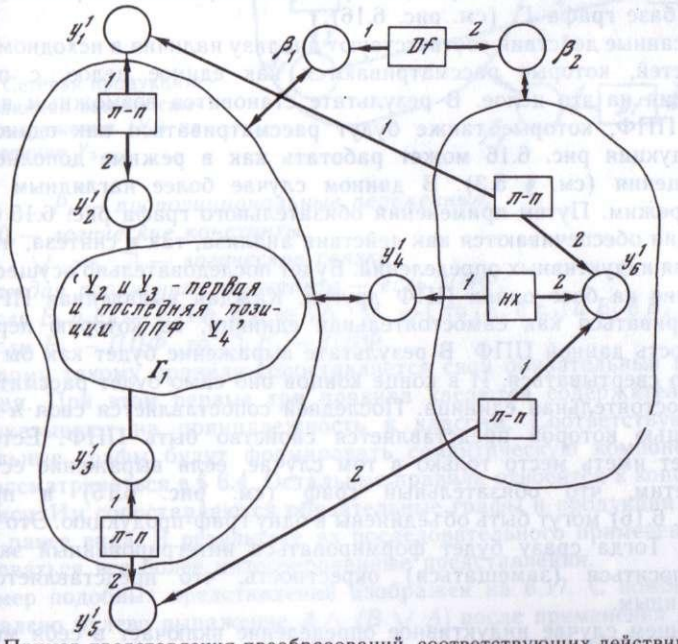


Рис. 6.16. Пример представления преобразований, соответствующих действию *подстановки на место слова* (*ППФ*) *его интегрированного эквивалента* Y_4

вида $\mathfrak{A} \vee \mathfrak{B}$. Тогда этой части сопоставляется своя (резервная) вершина (γ_j) и в ПС выражения формируется $\langle \_\_, t, ппф, \gamma_j \rangle$, что соответствует образованию интегрированных представлений. Словом, упомянутая часть рассматривается как единое целое, относящееся к классу ППФ. Подобные действия задаются фрагментами $[\square_\phi \beta_1 := \lambda]^\circ \langle \_\_, t, ппф, \beta_1 \rangle$.

Граф рис. 6.15 может быть обобщен заменой *о*-вершины $\vee$ на *н*-вершину y_6^1. Тогда будет иметься в виду *любая операция*. Путем многократного применения такого графа к ПС какого-либо выражения обеспечивается последовательная интеграция с формированием *с*-вершин, сопоставляемых выявляемым ППФ. Имеются в виду лишь ППФ, непосредственно входящие в выражения. Индукции пока не будет, т. е. система не сможет на базе одних выявленных ППФ выявлять другие. Дело в том, что хотя ППФ и будет рассматриваться как единое целое, но будет оставаться неясным, что же находится слева и справа от этого целого. Требуются дополнительные действия, заключающиеся в переносе окрестности вновь выявленной ППФ на указанное целое, т. е. на интегрированный эквивалент. Конечно, сказанное представляется нам как само собой разумеющееся. Однако в системе подобного сорта перенос должен специальным образом предусматриваться.

Перенос окрестности (на уровне внутрисистемных представлений) обеспечивается сетевой продукцией рис. 6.16. Продукция применяется к сети, представляющей ПС какого-либо выражения. И если в сети имеется *с*-вершина (γ_j), соответствующая какой-либо ППФ, то продукция будет применимой. Значением y_4^1 будет указанная выше вершина — γ_j. На базе y_6^1 будет сформирована вершина, соответствующая *позиции интегрированного эквивалента* (ППФ). На эту вершину и будет перенесен «образец», полученный на базе графа Γ_1 (см. рис. 6.16).

Описанные действия соответствуют анализу наличия в исходном выражении частей, которые рассматриваются как единое целое, с переносом окружения на это целое. В результате становится возможным выявление других ППФ, которые также будут рассматриваться как единое целое.

Продукция рис. 6.16 может работать как в режиме дополнения, так и замещения (см. § 5.3). В данном случае более наглядным является второй режим. Путем применения обязательного графа рис. 6.15 и данной продукции обеспечиваются как действия анализа, так и синтеза, что характерно для индуктивных определений. Будет последовательно осуществляться выявление на базе одних ППФ других. Каждая выявленная ППФ будет рассматриваться как самостоятельная единица, на которую переносится окрестность данной ППФ. В результате выражение будет как бы последовательно свертываться. И в конце концов оно само будет рассматриваться как самостоятельная единица. Последней сопоставляется своя *н*-вершина, с помощью которой представляется свойство быть ППФ. Естественно, это будет иметь место только в том случае, если выражение есть ППФ.

Отметим, что обязательный граф (см. рис. 6.15) и продукция (см. рис. 6.16) могут быть объединены в одну граф-продукцию. Это нетрудно сделать. Тогда сразу будет формироваться интегрированный эквивалент и переноситься (замещаться) окрестность, что представляется более естественным.

В общем случае индуктивное определение включает в себя множество пунктов — правил. Например, выражения языка б у л е в о й алгебры (БА) определяются следующим образом:

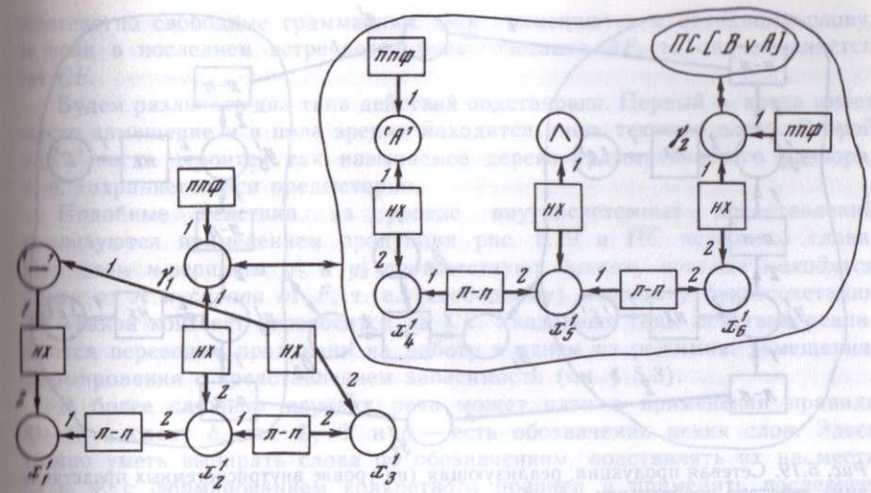


Рис. 6.17. Сеть, представляющая ПС выражения $A \wedge (B \vee A)$, после выявления интегрированных эквивалентов и переноса на них окрестностей. *С*-вершина γ_1 соответствует всему выражению, а γ_2 — его части $B \wedge A$

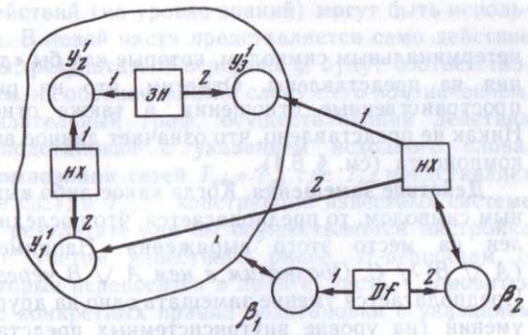


Рис. 6.18. Сетевая продукция, обеспечивающая замену символа или выражения (Y_2) на его обозначение Y_3

1) $P_1, ..., P_k$ — *пропозициональные переменные;*
2) 1,0 — *логические константы;*
3) $\wedge$, $\vee$, $\Rightarrow$, $\neg$ — *логические связки;*
4) *каждая переменная, константа — есть ППФ;*
5) *если* E_1 *и* E_2 *— ППФ, а* $\delta \in \{\wedge, \vee, \Rightarrow\}$, *то* $(E_1 \delta E_2)$ *и* $E_1 \delta E_2$ *— ППФ;*
6) *если* E_1 *— ППФ, то* $\neg E_1$ *— ППФ.*

Каждому такому правилу сопоставляется свой обязательный граф или продукция. При этом первые три правила касаются содержательных вещей — указывают на принадлежность к классам. Соответствующие им обязательные графы будут формировать семантическую компоненту, что будет рассматриваться в § 6.4. Остальные правила относятся к конструкции, синтаксису. Им сопоставляются обязательные графы и продукции рассмотренного ранее вида. В результате их последовательного применения будут формироваться все более интегрированные представления.

Пример подобных представлений изображен на 6.17. С помощью сети представлено булево выражение $A \wedge (B \vee A)$ после применения к его ПС графов и продукций введенного типа. Сеть соответствует дереву грамматического разбора, где *с*-вершины γ_1 и γ_2 соответствуют так называемым

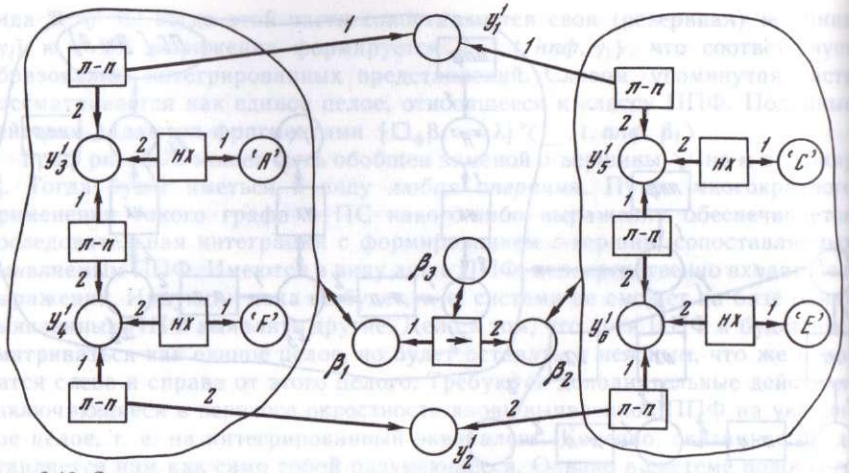


Рис. 6.19. Сетевая продукция, реализующая (на уровне внутрисистемных представлений) действие подстановки $AF \rightarrow CE$

нетерминальным символам, которые как бы «держатся в уме». Их обозначения не представлены. Отметим, что на рис. 6.17 представлены лишь пространственные отношения, а также отношения типа *часть — целое*. Никак не представлено, что означает данное выражение, т. е. семантическая компонента (см. § 6.4).

Действие замещения. Когда какое-либо выражение обозначается отдельным символом, то предполагается, что последний всегда может быть поставлен на место этого выражения. Например: *Возьмем знакосочетание* $(A \vee B) \vee C$. *Обозначим в нем* $A \vee B$ *через* D. *Тогда получится* $D \vee C$. Предполагается умение замещать одно на другое. Для реализации подобных умений (на уровне внутрисистемных представлений) будем использовать продукции. На рис. 6.18 изображена сетевая продукция, представляющая, что *если на некоторой позиции* Y_1 *находится символ или выражение* Y_2, *обозначенное через* Y_3, *то символ может быть замещен на его обозначение*. Соответствующие действия реализуются путем применения продукции к сети T, представляющей исходное выражение.

Пусть продукция рис. 6.18 работает в режиме замещения. Тогда если в T, например, представлена компонента, обозначенная через D, то продукция будет применимой. В результате будет сформирован фрагмент $\langle \underline{\cdot}, t, нх, 'D', x_i^1 \rangle$, где x_i^1 соответствует позиции упомянутой компоненты. Символ D займет место компоненты. Слева и справа от D окажутся символы, которые ранее окружали компоненту. Все это будет представлено в T. Если же продукция работает в режиме дополнения, то окружение у компоненты не изменится. Но ее окрестность будет перенесена на D. Здесь также предполагалось, что обозначение кем-то введено, уже имеется. Если это не так, то его нужно сгенерировать или сконструировать, см. выше.

Перейдем к случаю подстановки слов. Рассмотрим, когда слова указаны в явном виде. На рис. 6.19 изображена сетевая продукция, представляющая правило подстановки вида $AF \rightarrow CE$. Напомним, что такие правила образуют

контекстно свободные грамматики. Они применяются к исходному слову, и если в последнем встречается буквосочетание AF, то оно заменяется на CE.

Будем различать два типа действий подстановки. Первый — когда имеет место замещение и в поле зрения находится лишь текущее слово. Второй тип — когда строится так называемое дерево грамматического разбора, т. е. сохраняется вся предыстория.

Подобные действия на уровне внутрисистемных представлений реализуются применением продукции рис. 6.19 и ПС исходного слова. При этом $н$-вершины y_1^1 и y_2^1 соответствуют *буквам, которые находятся слева от A и справа от F*, т. е. (свободному) контексту буквосочетания AF. Такой контекст переносится на CE. Указанные типы действий реализуются переводом продукции на работу в одном из режимов: замещения, формирования с представлением зависимости (см. § 5.3).

В более сложных случаях речь может идти о применении правила $\Phi \rightarrow \Psi$ к слову δ, где Φ, Ψ и δ — есть обозначения неких слов. Здесь нужно уметь выбирать слова по обозначениям, подставлять их на места Φ и Ψ с формированием конкретного правила и применять последнее. Для реализации подобных действий (на уровне знаний) могут быть использованы составные продукции. В левой части представляется само действие подстановки — его аргументы, результат. H-вершины y_i^1 будут соответствовать правилу, его компонентам, их обозначениям и слову с его обозначением. В правой части будет обязательный граф, осуществляющий действия конструирования правила подстановки с указанием исходного слова. Продукция применяется к композиции сетей $T_{оп} \circ T_{сл}$ где $T_{оп}$ представляет входное описание (на уровне СС), а $T_{сл}$ — конструкции известных системе слов с их обозначениями. В результате как бы осуществляется настройка на то, о чем идет речь с учетом известного ранее. H-вершинам y_i^1 присваиваются значения, которые переносятся в правую часть. Становится возможным конструирование конкретных правил подстановки с указанием конкретного слова, к которому оно должно применяться.

Наконец, речь может идти о некоторых стратегиях, связанных с чисто формальными преобразованиями. Например, машина Тьюринга или конечный автомат могут быть описаны на чисто формальном уровне, т. е. сведены к некоторым последовательностям действий оперирования над символьными конструкциями. Для представления таких действий могут быть использованы сети (см. § 2.4), а для учета всего, что с ними связано,— составные продукции вместе с обязательными графами (см. § 5.4). Пользуясь ими, система сможет воспринимать описания, насущенные разного рода символикой, что будет приводить к изменению ее представлений, к соответствующим преобразованиям на уровне знаний.

§ 6.4 ПРИНЦИПЫ ВЫЯВЛЕНИЯ СЕМАНТИЧЕСКОЙ КОМПОНЕНТЫ

С помощью математических текстов не только описываются абстрактные объекты, обосновываются их свойства, даются конструктивные построения ... По мере изложения материала создается сам язык описания, вводятся разного рода обозначения, указывается на тот или иной способ говорения. Для этого используются предложения типа: *Назовем ..., Обозначим ..., Будем говорить ...* . Их примеры приводились в § 6.1. Говорилось, что в рамках

математических машин (систем с СП) подобного сорта предложения должны укладываться на свои полочки, заранее приготовленные. Они должны служить для последующего осмысления материала. Спрашивается, что это за полочки, как они должны быть устроены?

С этим вопросом связана очень важная проблема — построения л и н г в и с т и ч е с к и х процессоров, т. е. устройств, воспринимающих тексты описания, выявляющих семантическую компоненту и формирующих соответствующую структуру. Для решения данной проблемы необходимо прежде всего изучение математических описаний. Каким образом создается точный язык? Как вводимые формы изложения или говорения должны соотноситься с содержанием, семантической компонентой? Важным преимуществом строгих математических описаний является не только достаточная точность языка, однозначность передаваемого содержания (о чем уже говорилось выше), но и ограниченность приемов, с помощью которых создается такой язык. В частности, не используется принцип обучения по примерам. Имеют место ограничения, несколько сужающие проблему построения лингвистических процессоров. Но сама проблема остается.

В настоящее время указанная проблематика решается в области, связанной с созданием **языков программирования** (ЯП). Роль семантической компоненты играет машинный язык (или какой-либо другой язык, отображаемый на машинный), а под лингвистическим процессором понимается транслятор с ЯП на этот язык. Для записи лингвистических знаний (их называют семантикой языка) используются специальные формализмы, например Бэкусовские нормальные формы. Допускается постоянное расширение ЯП, для чего вводятся макроопределения. С помощью таких средств удается приближать ЯП к типовым математическим формализмам, в частности, разрабатывать ЯП, ориентированные на запись теоретико-множественных соотношений, предикатов и т. д.

Тогда спрашивается, чего же не хватает у современных трансляторов, почему их нельзя использовать для выявления семантического эквивалента предложений ЕЯ? Данный вопрос непосредственно связан с проблемой построения перспективного класса систем («универсальных» трансляторов), которые можно было бы настраивать на тот или иной язык, например Паскаль, ЛИСП и т. д. Предполагается настройка за счет декларативной компоненты — лингвистических знаний, согласованных с формами, с помощью которых в ЕЯ вводятся новые понятия. Такие системы должны строиться на хорошей структурной основе, допускающей (с любой точностью) представление семантических эквивалентов не только формализмов (конструкция ЯП), но и описаний, что они значат, как осуществлять вычисления и т. д. Эта основа должна позволить системе отвлечься от линейной формы записи и посмотреть, что за ней стоит Оказывается, что далеко не везде следует учитывать порядок слов, в ряде случаев большую роль играют денотаты и др. Конечно, развитие такой основы предполагает и разработку новых классов реализующих устройств, базирующихся на ассоциативных принципах, см. [14].

В данном параграфе будет показано, каким образом указанные проблемы могут решаться с помощью введенных ранее средств — семантических сетей, графов, продукций. Сами лингвистические процессоры или трансляторы будут рассматриваться в рамках парадигмы — систем с СП, т. е. как осуществляющие отображение сетей (графов) с уровня п о в е р х н о с т-

н ы х структур (ПС) на уровень с е м а н т и ч е с к и х структур (СС) (см. § 6.1 и 6.2). Такое отображение будет управляться с помощью наборов сетевых продукций, которые и будут играть роль лингвистических знаний. Ниже мы рассмотрим принципы представления в таких знаниях типовых математических определений, форм, с помощью которых указывается, что означает то или иное выражение, т. е. дается интерпретация.

Принципы выявления семантической компоненты. Следует заметить, что формальные языки разумно рассматривать как часть ЕЯ, но со «словами» специального сорта и с крайне ограниченным синтаксисом. Для их автоматической интерпретации перспективно использование общих принципов выявления смысла сообщений. Тогда появляется возможность их расширения, приближения к ЕЯ. Остановимся на этих принципах.

Когда смотришь на предложение, записанное на неизвестном языке, то оно воспринимается всего лишь как набор каких-то значков. Их представление и образует уровень ПС. Если язык знаком, то делаются постоянные попытки осмысления компонент предложения, которым сопоставляются их семантические эквиваленты. Здесь важную роль играет умение видеть за словами, что же они значат. Семантические эквиваленты как бы ставятся на места слов. В результате становится возможным осмысление все новых компонент предложения — слов, словосочетаний, отдельных языковых форм. При этом в процессе осмысления используются не только грамматические и синтаксические категории. Важную роль играют и уже выявленные семантические эквиваленты. Это давно известный факт. Таким способом может быть понято все предложение. Полученный продукт представляется в виде семантической структуры (СС). В процессе же осмысления будет иметь место промежуточный продукт, т. е. конструкции, в которых часть компонент осмыслена, а часть нет. Их представление приводит к образованию уровня так называемых п р о м е ж у т о ч н ы х структур (ТС).

Как же мыслится реализация данного процесса? Рассмотрим **основную идею** на примере. Пусть сообщается *Брат Ивана является мужем сестры Петра.* Анализ первых двух слов приводит к формированию семантического эквивалента, т. е. элемента, сопоставленного некоему субъекту — *брату Ивана.* У данного элемента формируется окрестность, т. е. представляется отношение *быть братом,* а сам элемент замещает свое словосочетание *брат Ивана.* То же самое касается последних двух слов. Далее анализируются оставшиеся слова *является мужем.* Анализ осуществляется в контексте уже сформированных элементов. Учитывается семантическая компонента, т. е. что элементы сопоставлены людям. Тогда эти элементы связываются отношением *быть мужем — женой.* В результате и будет сформирован семантический эквивалент всего предложения (см. приложение 1).

В общем случае слова какого-либо словосочетания, имеющего самостоятельный смысл, могут находиться в различных частях предложения, т. е. быть «разбросаны». Могут иметь место неоднозначности, эллиптические конструкции и многое другое. Процесс анализа существенно не изменится. Принципы останутся теми же.

В дальнейшем процедура выявления семантической компоненты будет рассматриваться под углом зрения преобразования с уровня ПС на уровень СС. Для этой цели будут использоваться наборы сетевых продукций, в том числе корреляционных (см. § 2.2). Они являются мощным средством преобразования. В частности, в работе [14] показано, что всего 12 корреляцион-

ных продукций достаточно для анализа простых распространенных предложений русского языка, т. е. их количество невелико. При этом продукции могут быть сделаны индифферентными к месту расположения слов и даже к их грамматическим категориям. Может учитываться лишь семантическая компонента. Делается возможным осмысление предложений типа *Моя твоя непонимай*, *Один штука* и т. д. Допускается крайне свободное выражение мысли.

Следует отметить, что такого сорта возможности представляют большой интерес с точки зрения **развития ЯП**. В настоящее время в ЯП слишком большую роль играет синтаксическая часть. Любая даже незначительная погрешность (не так набит пробел, не там поставлена точка, запятая, пропущен символ в идентификаторе ...) классифицируется как ошибка. Трансляция делается невозможной. В перспективе же языки должны быть устойчивы к определенного сорта ошибкам, соответственно необходимы трансляторы, допускающие (в некоторых пределах) варьирование форм записи, умеющие восстанавливать пропуски и т. д. Они в значительной степени должны опираться на семантическую компоненту. Важно, что выражает запись.

Здесь перспективно использование аппарата сетевых продукций и других средств СЯ. Сам процесс трансляции организуется следующим образом. Вначале формируется ПС алгоритмического выражения (см. § 2.4), затем с помощью продукций осуществляется преобразование на уровень СС, на котором задается операционная семантика. Становятся возможными разного рода вычисления (см. § 5.4). При этом операционная часть может быть записана в языке машинных команд, т. е. в частном случае (с помощью правил интерпретации) может быть сформирована машинная программа. При такой организации делается возможным постоянное расширение языковых форм (пополнением набора продукций). Может быть обеспечена проверка допустимости выражения, в том числе семантической правильности, что осуществляется за счет обязательных знаний. Система может быть настроена на выявление определенного сорта ошибок, на реализацию той или иной процедуры вычисления.

Рассмотрим вначале средства отображения на уровень СС на примере наиболее простой задачи — выявления семантических эквивалентов алгебраических выражений. Затем покажем, как решаются более сложные задачи.

Интерпретация алгебраических выражений. Любое выражение какого-либо формального языка может рассматриваться как последовательность символов, что представляется на уровне ПС. В то же время за каждым символом может что-то стоять, к примеру, он может обозначать константу, переменную, знак арифметической (логической) операции. С помощью конструкции выражения указывается, как они связаны между собой. Например, результат одной операции может быть аргументом другой. Все это представляется на уровне СС (см. § 1.4 и 2.4). Интерпретация сводится к отображению с уровня ПС на уровень СС, в процессе чего выявляются семантические эквиваленты — формируются соответствующие вершины, фрагменты. Последние как бы замещают те части выражения, на базе которых они были сформированы.

Например, пусть на вход системы поступило выражение

$$(A+B)\times B. \qquad (6.9)$$

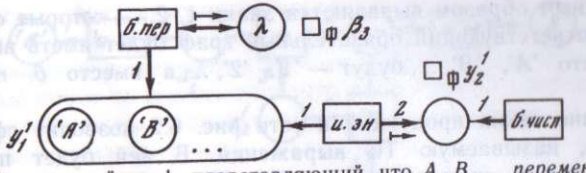


Рис. 6.20. Обязательный граф, представляющий, что *A*, *B*, ... *переменные, имеющие числовое значение*

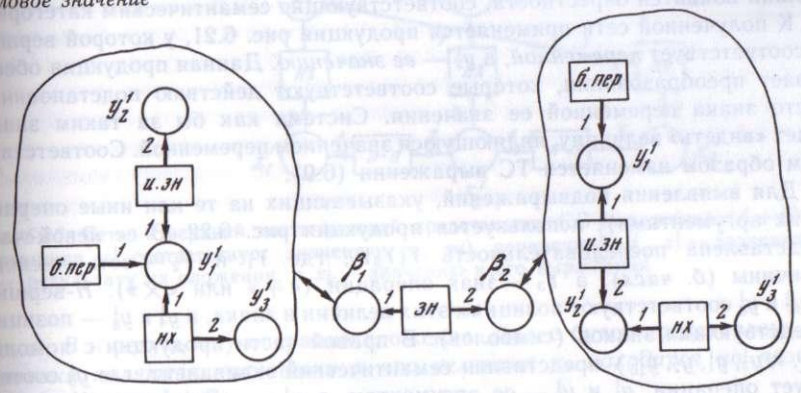


Рис. 6.21. Сетевая продукция, представляющая, что *если переменная* Y_1 *находится на каком-либо месте* (Y_3) *в выражении, то имеется в виду ее значение* (Y_2)

Вначале формируется ПС выражения (см. рис. 6.2), где символам *A*, *B*, +, ... сопоставляются собственные вершины '*A*', '*B*', ... Далее выявляется, что означают эти символы. Представляется принадлежность к классам. Затем учитывается тот факт, что в операции принимают участие не сами переменные, а их значения, т. е. некие величины. Соответственно переменные как бы замещаются на эти величины. После этого анализируется, что же означает $A+B$. Пусть выяснилось, что это некоторая величина, являющаяся результатом сложения. Тогда ей сопоставляется своя вершина. Сама же величина ($\mathfrak{A}$) как бы замещает выражение $A+B$, что осуществляется соответствующими преобразованиями на уровне внутрисистемных представлений. Затем анализируется, что означает $\mathfrak{A}\times B$. Результату умножения также сопоставляется своя вершина, окрестность которой составит СС всего выражения (6.9).

Подобные действия реализуются с помощью обязательных графов и продукций, изображенных на рис. 6.20 — 6.22. Рассмотрим вначале первый из них. Граф рис. 6.20 задает операции выявления символов *A*, *B*, ..., которые объявляются переменными (*б. пер.*) с числовыми значениями. Такие операции выполняются над сетью рис. 6.2. И если в ней имеется какая-либо вершина типа '*A*' (или '*B*', или ...), то формируется ее окрестность $\langle _, t, $ *б. пер*, '*A*' $\rangle \circ \langle \_, t,$ *и. зн*, '*A*', $x_i^1\rangle \circ \langle \_, t,$ *б. числ*, $x_i^1\rangle$, где *б. пер* — *быть переменной*, *и. зн* — *иметь значение* (следует отличать ее от *о*-вершины *зн*, представляющей связь уровней ПС и СС), *б. числ* — *быть числом*, а резервная *н*-вершина x_i^1 сопоставляется *значению переменной A*.

Аналогичным образом выявляются знаки 1, 2, ..., которые объявляются числами. Соответствующий обязательный граф будет иметь вид рис. 6.20. Только вместо *'A'*, *'B'*, ..., будут — '1', '2', ..., а вместо *б. пер* — стоять *б. числ.*

Применение таких продукций к сети рис. 6.2 позволит сформировать другую сеть, называемую ТС выражения. В ней будет представлена последовательность символов, как в ПС. Но дополнительно у некоторых вершин появятся окрестности, соответствующие семантическим категориям.

К полученной сети применяется продукция рис. 6.21, у которой вершина y_1^1 соответствует *переменной*, а y_2^1 — *ее значению*. Данная продукция обеспечивает преобразования, которые соответствуют действию подстановки на место знака переменной ее значения. Система как бы за таким знаком будет «видеть» величину, являющуюся значением переменной. Соответствующим образом изменяется ТС выражения (6.9).

Для выявления подвыражений, указывающих на те или иные операции (с их аргументами), используется продукция рис. 6.22. В ее левой части представлена последовательность $Y_1Y_3Y_2$, где Y_1 и Y_2 — есть числовые величины (*б. числ*), а Y_3 — знак операции («+» или «×»). *Н*-вершины y_5^1, y_7^1 и y_6^1 соответствуют позициям этих величин и знака, а y_4^1 и y_8^1 — позициям соседствующих знаков (символов). В правой части продукции с помощью $\langle \_, t, y_3^1, y_1^1, y_2^1, y_{10}^1 \rangle$ представлен семантический эквивалент, где y_3^1 соответствует операции, y_1^1 и y_2^1 — ее аргументам, а y_{10}^1 — результату. *Н*-вершина y_9^1 соответствует позиции результата.

Формирование семантического эквивалента (фрагмента уровня СС) обеспечивается применением продукции рис. 6.22 и ТС выражения (6.9). В результате на базе $A+B$ будет сформировано

$$\langle \_, t, u.\,зн, 'A', x_i^1 \rangle \circ \langle \_, t, u.\,зн, 'B', x_j^1 \rangle \circ \langle \_, t, +, x_i^1, x_j^1, x_k^1 \rangle,$$

где x_i^1, x_j^1 и x_k^1 — резервные вершины, соответствующие значениям перемен-

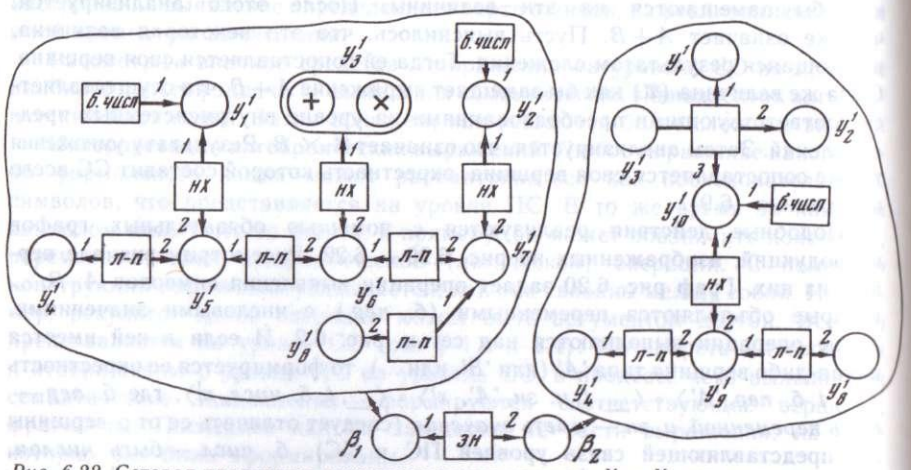


Рис. 6.22. Сетевая продукция, представляющая, что *если* Y_1 и Y_2 *есть числовые величины, а* Y_3 *— знак операции сложения или умножения, то* $Y_1Y_3Y_2$ *есть числовая величина, являющаяся результатом сложения*

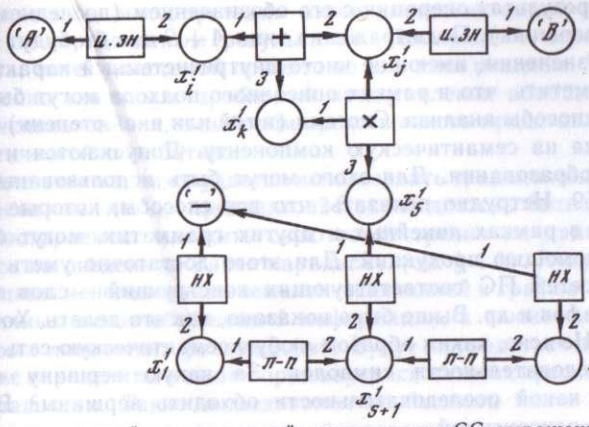


Рис. 6.23. Сеть, в верхней части которой представлена СС выражения $(A+B)\times B$. *Н*-вершина x_i^1 соответствует значению (*и. зн*) переменной A, x_j^1 — значению B, x_k^1 — результату их сложения, а x_5^1 — значению всего выражения

ных A, B и результату операции сложения. Далее будет представлено, что *значения являются числовыми и что величина* x_k *находится на позиции подвыражения* $A+B$.

Конечно, продукция рис. 6.22 должна работать совместно с еще одной продукцией, выявляющей подвыражение вида $(A+B)$, т. е. со скобками. В ее левой части будет представлена последовательность $(Y_1Y_3Y_2)$, а в правой — то же, что и на рис. 6.22. Причем, такая продукция должна иметь более высокий приоритет, т. е. применяться первой — скобки должны раскрываться в первую очередь. Путем применения указанных продукций к ТС выражения будет получена сеть, изображенная на рис. 6.23, где x_i^1 — резервная *н*-вершина, соответствующая *значению всего выражения* (*результату умножения*), а x_{s+1}^1 — *позиции этого результата*. Пока еще останутся фрагменты, представляющие пространственные отношения типа *слева и справа от величины* X_s *ничего нет — стоят пробелы*. Последние могут быть удалены специальной продукцией. Тогда остается «чистая» СС выражения (6.9).

Выше рассматривался лишь пример, иллюстрирующий определенный способ анализа арифметических выражений. Аналогичным образом могут анализироваться и выражения других языков. Еще в середине 70-х годов автором были разработаны средства, обеспечивающие преобразование на уровень СС любого выражения языков булевой алгебры, теоретико-множественных соотношений, исчисления предикатов [25].

В общем случае может быть множество способов анализа. Система может быть настроена на любой из них за счет своих знаний. В частности, тот факт, что в операциях принимает участие не сама переменная, а ее значение, можно учитывать в процессе выявления подвыражений. Соответственно отпадет необходимость в продукции рис. 6.21, а левая часть продукции рис. 6.22 несколько видоизменится. В ней будут представлены позиции переменных с указанием значений. В правой же части будет фрагмент с вершинами, соответствующими этим значениям. Будет также

представлен результат операции с его обозначением (последнему сопоставляется своя вершина). Подвыражения типа $A+B$ как бы будут замещаться на такие обозначения, имеющие чисто внутрисистемный характер.

Важно отметить, что в рамках описанного подхода могут быть реализованы любые способы анализа. Система (в той или иной степени) может быть ориентирована на семантическую компоненту. Допускаются и чисто формальные преобразования. Для этого могут быть использованы продукции вида рис. 6.19. Нетрудно показать, что все способы, которые могут быть реализованы в рамках линейных и других грамматик, могут быть реализованы и с помощью продукций. Для этого достаточно уметь записывать с помощью сетей ПС соответствующих конструкций — слов в заданном алфавите, графов и др. Выше было показано, как это делать. Хотя обратное не очевидно. Не ясно, каким образом любую семантическую сеть записывать в виде последовательности символов. За какую вершину «цепляться» вначале и в какой последовательности обходить вершины? Ведь запись должна быть однозначной.

ГЛАВА 7

ОБОБЩЕННЫЕ ПРЕДСТАВЛЕНИЯ

В связи с развитием вычислительной техники одной из важнейших задач, стоящих на повестке дня, является необходимость передать машине тот громадный опыт, который накоплен специалистами в их областях знаний. Такой опыт достаточно часто выражается в виде сравнительно точных предложений ЕЯ, т. е. однозначно интерпретируемых. Здесь следует учитывать, что наиболее ценный опыт выражается с помощью «обобщающих» предложений, т. е. носящих характер гипотез, закономерностей, описывающих структурные особенности классов объектов, их общие черты и отличия. Подобные предложения могут служить для распознавания, классификации, проверки семантической правильности ответа на запросы, порождения новой информации и пр. Спрашивается, как вводить такую информацию в машину (систему) и обеспечить ее использование?

Далеко не лучший способ — применение для этих целей промежуточного звена, т. е. посредников — формализаторов, интерпретаторов, которым специалист передает свои знания. Эти посредники осуществляют формализацию или алгоритмизацию передаваемой им информации и обеспечивают интерпретацию системных результатов. Получается что-то похожее на игру в испорченный телефон. Специалист, не зная формализмов, может лишь догадываться, правильно ли его поняли, не перепутали ли чего-либо, то ли делает машина, что ему нужно. По мере разработки какой-то задачи, требующей для решения вычислительной техники, возникают новые, более точные и совершенные идеи. И так может продолжаться достаточно долго. А изучить сложнейшие формализмы специалисту не под силу.

Конечно, наиболее приемлемо — обеспечить специалисту непосредственный доступ к системной информации. Отсюда возникают задачи автоматического ввода «обобщающих» предложений ЕЯ, их уточнения в режиме

активного диалога и интерпретации актов внутрисистемной деятельности, приводящей к определенным результатам. По крайней мере специалист должен знать, почему системой принято то или иное решение. Такие задачи непосредственно связаны с проблемой представления обобщенной информации и ее использования, с выбором средств такого представления и методик автоматической обработки.

Почему для этих задач не удается использовать традиционные формализмы, в частности язык логики предикатов? Какими должны быть формализмы? Ниже мы постараемся изложить свою точку зрения на данные вопросы, ориентируясь на введенную ранее парадигму (см. § 1.1). Будут рассматриваться принципы использования семантических сетей и графов для представления различных видов обобщенной информации, а также способы реализации типовых логических рассуждений. Будут проводиться исследования, позволяющие определить, какого сорта внутрисистемные представления приводят к парадоксам. Такие исследования необходимы как с точки зрения обеспечения «чистоты» системных акций, правильности ее знаний, так и для обоснования случаев нерезультативных реакций и противоречивых реплик.

§ 7.1. ПРЕДПОСЫЛКИ К РАЗВИТИЮ ЛОГИЧЕСКИХ ЯЗЫКОВ

В настоящее время имеются две основные тенденции в развитии логических средств. Первая из них связана с задачами чисто теоретического характера. Языки рассматриваются как некоторое подспорье (инструментарий) человеку, пытающемуся решать какие-либо принципиальные вопросы и осуществляющего необходимые для этого построения. Здесь следует упомянуть языки динамических логик, допустимых множеств, алгоритмических алгебр и др. Вторая тенденция связана с изучением естественных представлений, категорий, используемых человеком в рассуждениях. Подобные исследования привели к созданию расплывчатых, модальных и других логик. Однако и в том, и в другом случае языки создавались как инструментарий человеку с тем, чтобы последний мог построить нужную ему модель и, пользуясь указанными правилами, осуществить необходимые преобразования и т. д. Некоторые из этих акций поддаются автоматизации, но далеко не все. Фактически языки разрабатываются при допущении, что вся «поддержка инструментария» будет идти через интеллектуальные возможности человека. Отсюда насыщенность языков специальной символикой, высокая синтаксическая загруженность. Пользоваться ими может лишь специалист.

Ниже будет рассматриваться несколько другое направление развития, связанное с задачами автоматического ввода и обработки естественно-языковых форм. Как правило, логики относят такое направление к «реализационным» аспектам. С этим трудно согласиться, так как речь идет об одном из подходов в направлении создания логических автоматов, которые можно было бы «настроить» на работу с той или иной логикой, используемой человеком в рассуждениях.

Особенности формализма логики предикатов. Возьмем простое предложение *Все люди смертны.* Или *Человек — смертен. Если кто-то является человеком, то он смертен. Только те, кто смертны, могут быть людьми. Никто из людей не является бессмертным. Если ты человек, то ты не можешь быть бессмертным.* Возможны и другие способы переиначивания, варианты

выражения сути. Каким же образом представлять такие предложения в знаниях системы и осуществлять обработку?

Наиболее распространенный из существующих подходов заключается в создании формализмов, в которых все эти предложения по возможности представляются в виде одной и той же конструкции. В худшем случае — в виде нескольких эквивалентных конструкций. С точки зрения логического вывода, видимо, такой подход является наиболее правильным. Если основной задачей является порождение одних истинных высказываний (теорем) из других (аксиом), то ясно, что с уменьшением способов записи одного и того же уменьшится число логических аксиом (правил эквивалентного преобразования). Именно с учетом данного критерия создавался язык логики предикатов.

Однако такой подход оказывается неприемлемым, если расширить круг задач, включив задачи ввода естественно-языковых описаний и проверки правильности содержащейся в них информации. Сразу возникают трудности преобразования на уровень системных знаний и естественной обработки. Так, все рассмотренные в примере предложения преобразуются в предикатное выражение вида $\forall X\,(P_1(X) \supset P_2(X))$, которое дословно интерпретируется *Для всех объектов (субъектов X) справедливо, если объект относится к классу людей* (P_1), *то он обладает свойством смертности* (P_2). Возникает вопрос, как же осуществлять преобразование предложений в нужную форму с построением предикатного выражения? Здесь невозможно использование методов последовательного выявления семантических эквивалентов с автоматическим формированием внутреннего представления. Что сопоставлять словам типа *только*, *может быть*, *если-то*, *никто* и др. Или же анализировать только исходные формы с такими словами, сопоставляя им соответствующие предикатные выражения? Ясно, что количество таких форм будет сильно возрастать с усложнением предложений. Кто им будет сопоставлять предикатное выражение, если последовательный анализ затруднен? Или как-то на поверхностном уровне, не вдаваясь в суть, обеспечивать их переиначивание, преобразование в интерпретирующее предложение? Как показывает опыт, пока что такого сорта деятельность по плечу только человеку достаточно высокой квалификации. Построение автоматизированных процедур здесь крайне затруднено. И основных причин две. Во-первых, сравнительно высокая синтаксическая загруженность языка логики, его неоднородность. И, во-вторых, несогласованность средств языка с информацией на ЕЯ. В этом легко убедиться, обратившись к предыдущему примеру, т. е. сравнив интерпретирующее предложение с исходным.

Об операционных оттенках. Зачем же человеку понадобилось наличие множества естественно-языковых форм, которые с точки зрения логики предикатов выражают одно и то же? Прежде всего, для задания операционных оттенков, которые никак не выражаются в языке логики. Например, возьмем предложение *Никто не является бессмертным*. Естественно со словом *никто* связать оператор проверки пустоты найденных объектов. Если данное предложение — гипотеза, которая проверяется на материале конкретных знаний, то она будет справедливой в случае выполнения проверки. Аналогично естественно связать со словом *только* оператор теоретико-множественного включения. Тогда форма вида *Только* X_1 *могут быть* X_2 будет задавать операции нахождения X_1 и X_2 с проверкой $X_2 \subset X_1$. Форма *Если ... то ...* выражает условную зависимость, т. е. ее естественно связать

с операциями проверки условия и выполнения соответствующих действий. Подобные примеры можно было бы продолжить.

В особо показательной форме операционный оттенок проявляется в случае сложных предложений, имеющих составные части со словами *какие-либо*, *все*, *только*, *если* и др. Выясняется, что **от формы или конструкции предложения во многом зависит степень его читабельности**.

Возьмем, например, высказывание *Имеется город, на каждом заводе которого есть цеха, выпускающие только детали со знаком качества.* Или *Среди цехов любого завода некоего города имеются такие, в которых каждая выпускаемая деталь имеет знак качества.* Нетрудно видеть, что первое предложение заметно отличается от второго. Прежде всего, в первом акцент сделан на города, а во втором — на цеха. Подобное явление в лингвистике получило название **темы** (на чем сделан акцент) и **ремы** (все остальное). Акцентация как бы указывает, что нужно проверять, на что обращать внимание. В первом предложении как бы напрашивается, что нужно постараться найти соответствующие города, а потом проверить, что они есть, т. е. что их множество непусто. Во втором — что множество цехов непусто. Операционные оттенки в этих предложениях, как нетрудно видеть, различные. Более того, степень читабельности второго предложения ниже, чем первого. Выясняется, что форма во многом определяет степень такой читабельности. Если читателю приведенный пример показался не столь убедительным, то предлагаем выразить приведенное высказывание на языке логики предикатов и прочитать его, т. е. дословно проинтерпретировать. Тогда понять высказывание будет значительно труднее.

Описанное явление можно объяснить, учитывая операционный характер самих языковых конструкций или форм. Естественно допустить, что трудночитабельными в большинстве случаев являются предложения таких конструкций, с которыми связаны сложные операции по обработке, в частности операции перебора, многошаговые алгоритмы. Подобные предложения часто хочется представить в более понятном виде (переиначить), после чего проверять их правильность, искать ответ на вопрос и др. Хотя такое переиначивание связано со своими трудностями.

Следует заметить, что для некоторых конструкций может быть просто неясным, как и в какой последовательности искать неопределенные компоненты. Тогда переиначивание просто необходимо. При этом по возможности конструкция должна быть приведена к виду с наиболее ясным операционным оттенком. Подобный фактор должен учитываться в процессе организации внутрисистемной обработки.

Неоднозначности в формализации, проблема уточнения. Обратимся к примерам, используемым в типовых работах по математической логике. Например, в [51] рассматривается предложение *Ни один пациент не любит знахаря*. Оно переиначивается следующим образом: *Для каждого пациента и каждого знахаря справедливо, что пациент не любит знахаря*, что записывается в нотации языка логики. Видно, что при переиначивании имеющие место неоднозначности каким-то образом устраняются, а само предложение уточняется. В частности, выше понималось, что речь идет о некоем обобщенном представителе класса знахарей, в связи с чем и был использован квантор всеобщности (*каждый знахарь*), хотя возможен и другой вариант, т. е. на самом деле речь могла идти о конкретном человеке — знахаре.

Вообще, при восприятии подобного сорта предложений хочется сразу переспросить, что имеется в виду. Причем встречные вопросы наиболее легко воспринимаются, если они согласуются с формой предложений, сообщений. При существенном изменении такой формы восприятие делается чрезвычайно трудным. Например, в предыдущем случае естественно переспросить *Какого знахаря Вы имеете в виду?* Сравните с другой формой *Правильно ли я Вас понял — для каждого пациента существует знахарь, которого он не любит?*

Предыдущий пример иллюстрирует некоторые побочные эффекты, возникающие в силу несогласованности языка логики предикатов с информацией на ЕЯ. В процессе формализации возможны искажения, связанные с переиначиванием. Информация может уточняться совсем не таким способом, как это имелось в виду. Все это характерно для большинства примеров. Возьмем из той же работы [51] другие предложения — *Некоторые пациенты любят своих докторов* (где неясно, относится ли слово *своих* к конкретным пациентам или ко всем); *Ни один пациент не любит знахаря* (неясно, имеется ли в виду конкретный знахарь или вообще класс). Они записываются в языке предикатов в одном из вариантов, видимо, удобном с точки зрения этого языка, но далеко не самом лучшем [51 с. 48]. При формализации же сложных предложений искажения еще больше усиливаются.

Сказанное справедливо и для многих других языков, основанных на логике предикатов. Здесь следует упомянуть языки типа реляционного исчисления на доменах, реляционного исчисления на кортежах, используемых для записи запросов в реляционных базах данных [47].

Аналогичными недостатками обладают и семантические сети, в которых для представления обобщенной информации вводятся конструкции, повторяющие структуру предикатных выражений. Например, в сетях Хендрикса [33], Ригера [62], Шуберта [56] вводятся сетевые пространства, соответствующие областям действия кванторов. Имеются и фрагменты-кванторы, связанные с областями почти таким же образом, как в предикатных выражениях. При этом в качестве наиболее веского аргумента выдвигается следующий: безразлично, как представлять, можно так, а можно иначе. С этим нельзя согласиться. Представления должны быть **согласованными** с естественно-языковыми формами. Только тогда будет возможен последовательный анализ таких форм с выявлением семантической компоненты и выполнением необходимых внутрисистемных акций, т. е. требования к выбору средств и способов представления обобщенной информации должны быть достаточно жесткими.

Способы представления, кванторные сети. С точки зрения обеспечения однородных и согласованных представлений привлекателен подход, при котором кванторы рассматриваются как специальные виды отношений типа *быть каждым* ($\forall$), *быть какими-либо* ($\exists$), *быть каким-либо* ($\exists_1$) и др. Отношения представляются обычным способом. Но они носят чисто внутрисистемный (служебный) характер. Их нет в реальных ПО. Считается, что каждое такое отношение как бы связывает две компоненты. Первая соответствует множеству, которое пробегает свободная переменная, а вторая — представителю этого множества с его свойствами и отношениями. Представитель фактически соответствует связанной переменной. Например, считается, что в предложении *Каждый сотрудник лаборатории DC — передовик* отношение *быть каждым* связывает множество сотрудников лаборатории *DC*

с его представителем, для которого указано свойство *быть передовиком*.

Указанный подход привлекателен прежде всего возможностью использования для представления обобщенной информации обычных сетей и графов. Последние дополняются специальным набором вершин, соответствующих упомянутым отношениям ($\forall, \exists, \exists_1, \ldots$). Будем называть элементарные фрагменты (ЭФ) с такими вершинами кванторными, а сеть или граф с кванторными ЭФ — кванторной (ым). Кванторные ЭФ как бы разбивают сеть на две независимые части: первая соответствует ограничениям на свободную переменную, а вторая — области действия квантора. Каждая такая часть может быть легко выделена по вершинам ЭФ, соответствующим множеству и его представителю. При таком подходе не требуется специальных средств указания типа переменной (свободная, связанная), области действия квантора и ограничений, которые накладываются на свободную переменную. Соответствующие вершины и фрагменты могут быть легко найдены по кванторной ЭФ — по его вершинам и связанным с ними фрагментам.

Описываемый подход привлекателен и с точки зрения обеспечения представлений, согласованных с формами ЕЯ. Напомним, что согласованность предполагает выполнение следующих требований (см. § 1.1). Любое естественное дополнение или уточнение предложения, при котором оно остается осмысленным, должно вызывать лишь достройку внутреннего представления без существенного изменения общей конструкции. При этом смысловые эквиваленты слов или словосочетаний, самостоятельные по своему характеру, должны оставаться без изменения.

Например, в рассмотренном ранее предложении допускаются достаточно произвольные уточнения типа *Имеются в виду все сотрудники, кроме Иванова и Сидорова, Сотрудники, проработавшие не менее трех лет, Не каждый, а некоторые* и т. д. Желательно, чтобы все множество таких уточнений сводилось лишь к достройке внутреннего представления. При этом то, что касается лаборатории *DC*, свойства *передовик* и всего множества сотрудников, не должно изменяться. Их представления должны оставаться теми же.

При использовании кванторных сетей и графов обеспечивается выполнение требований согласованности. Более того, могут быть учтены операционные оттенки информации, что важно не только в плане проверки правильности сообщений, выполнимости гипотез, но и с точки зрения поиска ответа на запросы «обобщенного» характера. Сети и графы могут служить для представления различных смысловых оттенков. Для этого будем использовать введенные ранее средства: знак «?», стрелки порядка $\mapsto$, области графов (соответствующие оборотам типа *тот, который*) и др. Например, вопросы *Все ли люди смертны?, Смертен ли каждый человек?* (где акцент сделан на слове *смертен*) будут представляться по-разному. Знаком «?» будут отмечены различные *н*-вершины.

Для учета операционных оттенков языковых форм со словами *каждый, какие-либо, только, некоторые* и др. будем использовать специальные наборы операторов, которые по мере необходимости будем вводить в ОЯ (см. § 4.1). Принципы учета останутся прежними, т. е. формы представляются с помощью графов, задающих операции ОЯ. При этом различные формы представляются с помощью различных графов (у которых стрелки $\mapsto$

направлены по-разному). Например, предложения *Каждый человек смертен* и *Только смертные могут быть людьми* представляются с помощью различных графов. Первый из них будет задавать операции перебора людей с последовательным выделением представителей и проверкой для каждого из них свойства *быть смертным*, а второй — операции нахождения множества людей и множества субъектов, обладающих свойством *смертности*; далее осуществляется проверка включения ($\subset$) первого множества во второе. Словом, задаваемые операции будут зависеть от того, куда направлены стрелки $\mapsto$, и конечно же, от конструкции самого графа.

Последующие разделы будут посвящены рассмотрению нового формализма представления обобщенной информации, допускающего непосредственное отображение форм ЕЯ. При этом следует учитывать, что с помощью ЕЯ может быть «прочитано» любое выражение языка логики. Отсюда мы приходим к требованию вложимости, т. е. возможности представления любого выражения языка функционального исчисления первого и второго порядков средствами предлагаемого формализма. Ниже будет показано, что семантические сети и графы удовлетворяют данному требованию.

§ 7.2. ПРЕДСТАВЛЕНИЕ ДЕКЛАРАТИВНОЙ КОМПОНЕНТЫ

В данном параграфе будут рассматриваться вопросы представления обобщенной информации, пока что без учета операционного оттенка. Будет развиваться язык так называемых кванторных сетей, т. е. содержащих фрагменты с o-вершинами из $\{\forall, \exists, \exists_1\}$. Напомним, что o-вершина $\forall$ соответствует отношению *быть каждым*, $\exists$ — *быть какими-либо* и $\exists_1$ — *быть каким-либо* (*одним*). Мы покажем, что с помощью таких сетей обеспечивается согласованное представление различной информации обобщенного характера, в том числе формализуемой средствами логики предикатов. Еще раз отметим, что речь будет идти о представлении чисто декларативной компоненты, т. е. без учета направления обработки. Предполагается, что задачи выбора наилучшего направления решаются самой системой, которая соответствующим образом сформирует на сети стрелки порядка. В результате получится граф, задающий эффективную процедуру поиска и обработки. Такие графы будут рассматриваться в следующем параграфе.

Простые кванторные сети. Как уже говорилось, слова типа *только*, *некоторые*, *каждый* и др. служат для указания на специального вида связи, имеющие место в представлениях. Такие связи играют важную роль в процессе обработки на базе обобщенных сведений. Например, в словосочетании *некоторый сотрудник* слово *некоторый* указывает на связь множества сотрудников с его представителем. Такая связь, к примеру, может быть использована для выбора представителя. Слово *некоторые* указывает на связь множества с подмножеством. Слова типа *любой*, *всякий*, *каждый* указывают на связь множества с множеством или множества с представителями, которые требуется перебрать. Например, словосочетание *любой сотрудник* может означать *все множество сотрудников* или же *какого бы сотрудника мы не взяли, для него будет справедливо...*

Для представления подобного вида связей будем использовать следующие фрагменты (ЭФ):

$$\langle \gamma_1, p_1, \sigma, x_1, x_2 \rangle, \text{ где } \sigma \in \{\forall, \exists, \exists_1\}, \tag{7.1}$$

которые будем называть к в а н т о р н ы м и. O-вершина δ определяет вид связи, $н$-вершина x_1 соответствует *множеству*, а x_2 — *отдельному представителю* (при $\sigma = \exists_1$), *подмножеству* (при $\sigma = \exists$) или *всему множеству* (при $\sigma = \forall$). С помощью вершины p_1 представляется логическая составляющая. Если заменить p_1 на t (или добавить $[p_1 := t]$), то будет представлено, что *указанная связь имеет место*, т. е. *факт истинности*, а если заменить p_1 на f (или добавить $[p_1 := f]$) — то *факт ложности*.

В § 7.1 сети с кванторными фрагментами были названы к в а н т о р н ы м и. Рассмотрим несколько типовых примеров. В кванторной сети

$$T_1(x_1) \circ \langle \gamma_1, t, \exists, x_1 \,|\, \textit{сотрудники} \,|\, x_2 \rangle \circ T_2(x_2) \tag{7.2}$$

$н$-вершина x_1 соответствует *множеству сотрудников*, а x_2 — *некоторым из них*. С помощью T_1 представлены ограничения на множество сотрудников, т. е. какими они обладают свойствами. С помощью T_2 представлены *свойства и отношения, справедливые* (t) *для некоторых сотрудников*.

На рис. 7.1 изображена кванторная сеть, представляющая *Все объекты класса* M_1 *обладают свойством* R_1. Такая сеть записывается в виде

$$\langle \_, t, \in, x_1, m_1 \rangle \circ \langle \gamma_1, t, \forall, x_1, x_2 \rangle \circ \langle \_, t, r_1, x_2 \rangle. \tag{7.3}$$

Если o-вершина m_1 соответствует *классу людей*, а r_1 — свойству *быть смертным*, то сеть рис. 7.1 будет представлять *Все люди смертны*. Если m_i соответствует *критянам*, а r_1 — *быть лжецом*, то будет представлено *Все критяне лжецы* и т. д. Если на рис. 7.1 заменить o-вершину $\forall$ на $\exists_1$, то будет представлено *Некоторый объект класса* M_1 *обладает свойством* R_1, а если заменить на $\exists$ — то *некоторые*. Если в кванторном фрагменте сети рис. 7.1 заменить o-вершину t на f, то будет представлено *Неверно, что объекты класса* M_1 *обладают свойством* R_1. Если же заменить t на $н$-вершину p_1, то будет представлено *Неизвестно, обладают ли объекты класса* M_1 *свойством* R_1.

Сети вида (7.2) и (7.3) могут быть **обобщены** до следующей конструкции:

$$T_i(x_1) \circ \langle \gamma_1, t, \sigma, x_1, x_2 \rangle \circ T_j(x_2), \tag{7.4}$$

где имеется только один кванторный фрагмент. При этом $T_i(x_1)$ — связанная сеть, представляющая *свойства и отношения объектов* X_1, по которым может быть выделено все их множество. Сеть $T_j(x_2)$ также должна быть связанной. Она представляет: при $\sigma = \forall$ — *общие* (*дополнительные*) *свойства или отношения указанного множества объектов*, при $\sigma = \exists$ — *свойства или отношения некоторого подмножества*, а при $\sigma = \exists_1$ — *отдельного представителя*.

Заметим, что на конструкцию вида (7.4) накладывается только одно ограничение: сети T_1 и T_j не должны иметь общих $н$-вершин. Иначе делается невозможным однозначное выделение множества X_1. На последнее будут оказывать влияние его представители X_2.

Роль вершины γ. Вершина γ_1 в кванторном фрагменте на рис. 7.1 (и в (7.3)) соответствует всему высказыванию, т. е. играет роль c-вершины (см. § 1.4). Такие вершины могут служить для представления различных связей, например для указания автора высказывания, для представления зависимостей между отдельными высказываниями — причинно-следственного, временного характера и др. Так предложение A_1 *сообщил, что все люди смертны* на уровне СС представляется, как показано на

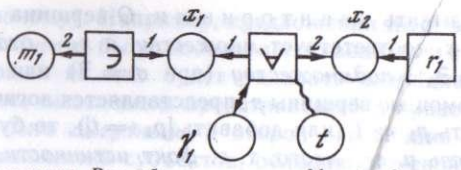


Рис. 7.1. Сеть, означающая *Все объекты класса* M_1 *обладают свойством* R_1

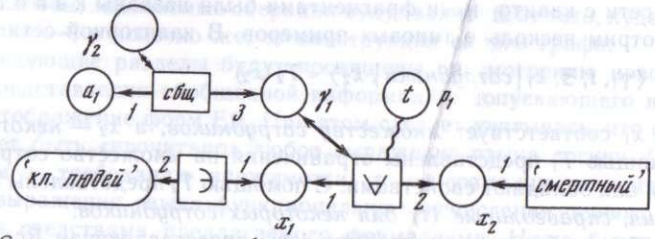


Рис. 7.2. Сеть, означающая, что A_1 *сообщил (сбщ), что все люди смертны*

рис. 7.2. С помощью $x_1|$*люди*$|$ представлена *принадлежность* A_1 *к классу людей*, а действие *сообщить* (*сбщ*) рассматривается как имеющее три семантических падежа — *кто сообщил*, *кому* и *что сообщил* (хотя их на самом деле больше). *С*-вершина γ_2 соответствует всему предложению, а γ_1 — той информации, которую сообщил A_1.

С-вершина γ_2 сама может служить для представления более сложных взаимоотношений. Например, добавим к сети (7.5) фрагмент $\langle \_, t, в\text{-}з, \gamma_3, \gamma_2 \rangle$, где γ_3 соответствует некоторому событию $\mathcal{T}_3$. Тогда будет представлено, что *после того, как имело место событие* $\mathcal{T}_3$, *субъект* A_1 *сообщил, что все люди смертны*.

Следует отметить, что если речь идет не о самих высказываниях, а о фактах их истинности или ложности, то для представления связей будут использоваться не *с*-вершины (типа γ), а *р*-вершины кванторных фрагментов. Например, предложение A_1 *утверждает факт истинности того, что все люди смертны* представляется с помощью сети

$$\langle \_, t, гов, a_1, p_1 \rangle \circ \langle \_, p_1, \forall, x_1\ |люди|, x_2 \rangle \circ \langle \_, t, 'смертный', x_2 \rangle, \quad (7.5)$$

где действие *утверждать* или *говорить* (*гов*) рассматривается как имеющее два семантических падежа: *кто утверждает* и *какой факт утверждает*. Конечно, такие представления сводимы. Отличие от того, что рассматривалось ранее, лишь в незначительных смысловых нюансах.

Представление вопросов. Перейдем к рассмотрению принципов представления требований и вопросов обобщенного характера. Для этого используются *з*-сети вида $[p_i := ?]$ или $[x_i := ?]$, которые добавляются к кванторным сетям (см. § 2.2). Напомним, что в изображениях сетей знак ? ставится внутри вершин p_i, x_i. Оказывается, что в зависимости от места этого знака будут представляться различные смысловые оттенки вопроса. Например, на рис. 7.3 изображена сеть, представляющая *Все ли люди смертны?*. Изменим расположение знака ?. Поставим его внутри *н*-вершины p_2, а внутри p_1 поставим t. Тогда уже будет представлено *Смертны ли все люди?*. Акцент сделан на другом слове — *смертный*. На рис. 7.4 приведен

еще один пример. Представлено *Какими свойствами обладают некоторые люди?* Наконец, обратимся снова к рис. 7.3. Подставим знак ? на место *кл. людей*, а t — внутри *н*-вершины p_1, сделав $[p_1 := t]$. Тогда будет представлено *Какие имеются классы с элементами, каждый из которых обладает свойством смертности?*

В принципе знак ? может быть поставлен на место (или внутри) любой *о*-вершины (*н*-вершины). Хотя в ряде случаев это может приводить к интерпретирующим выражениям несколько **искусственного** характера. Например, если на рис. 7.3 поставить знак ? внутри *н*-вершины x_1, предварительно сделав $[p_1 := t]$, то будет представлено следующее: *Какие множества, элементы которого относятся к классу людей, включены в другое множество, элементы которого обладают свойством смертности?* В естественном общении человек не пользуется вопросами подобного типа. Такая же искусственность интерпретирующих выражений будет иметь место, если поставить знак ? внутри x_2. Если же на рис. 7.3 заменить *о*-вершину ∀ на ?, поставив ? внутри p_1, то интерпретирующее выражение будет иметь вид *Как связаны между собой множество людей и множество тех, кто смертен?* Подобные примеры можно было бы продолжить, включив требования нахождения согласованных значений. Итак, знаки ? могут расставляться в сети достаточно произвольным способом. В результате сеть будет представлять вопрос, имеющий свой оттенок.

Следует обратить внимание, что одна и та же сеть может интерпретироваться несколькими способами. Например, сеть рис. 7.3 может быть проинтерпретирована: *Для каждого ли человека справедливо, что он обладает свойством смертности?*, а также *Только ли смертные могут быть людьми?*. Эти предложения имеют различные операционные оттенки, которые мы будем связывать с понятием графа. Уточнение варианта интерпретации предполагает формирование стрелок порядка $\mapsto$.

Представление предикатных выражений. Ниже будут рассматриваться примеры использования кванторных сетей для представления выражений многосортной логики предикатов [18, 21]. Сети фактически будут повторять структуру предикатных выражений с тем только отличием, что специальным образом не будут выделяться области действия кванторов. Для этого

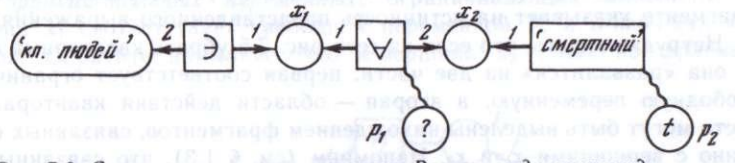


Рис. 7.3. Сеть, представляющая вопрос *Все ли люди смертны?*

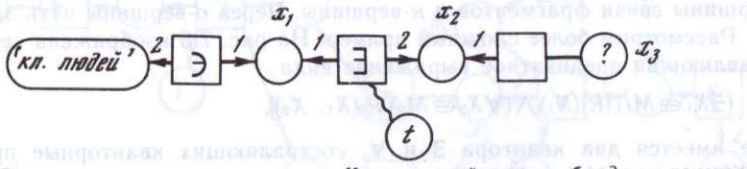


Рис. 7.4. Пример представления вопроса *Какими свойствами обладают некоторые люди?*

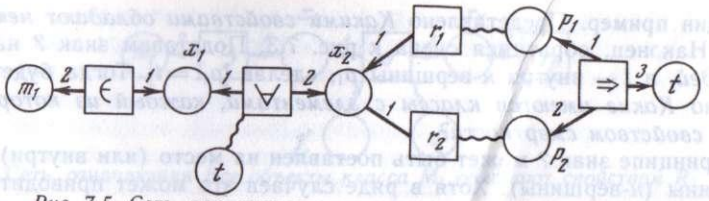


Рис. 7.5. Сеть, представляющая предикатное выражение (7.6)

будет использовано понятие с в я з а н н о й сети (см. § 1.3). Далее сети будут дополнены фрагментами, представляющими факт вхождения одних кванторов в область действия других, или, как будем говорить, факт «подчинения» кванторов. С помощью таких сетей обеспечивается представление любого выражения языка логики предикатов [25].

Сразу следует оговориться. Возможности сетей, соответствующих предикатным выражениям, оказываются ограниченными — в смысле согласованного представления естественно-языковой информации обобщенного характера. Понятия области действия кванторов, их вложимости и др. никак не согласуются со многими традиционными естественно-языковыми формами. Для представления таких форм лучше использовать кванторные графы, которые будут введены в § 7.3. Последние будут служить для обеспечения направленной обработки (т. е. алгоритмическими методами), необходимой с точки зрения проверки правильности сообщений, выполнимости гипотез, ответа на запросы и во многих других случаях.

Для представления предикатных выражений, содержащих один квантор, достаточно введенных ранее средств. На рис. 7.5 изображена к в а н т о р н а я сеть, представляющая предикатное выражение вида

$$(\forall X_1 \in M_1)\,[R_1(X_1) \Rightarrow R_2(X_1)]. \qquad (7.6)$$

H-вершина x_1 соответствует свободной переменной, ее значения представляют множество, которое пробегает свободная переменная. H-вершина x_2 соответствует связанной переменной, а $н$-вершины p_1 и p_2 — логическим значениям предикатов $R_1(X_1)$ и $R_2(X_1)$. С помощью фрагмента $\langle \_\_,\ t,\ \Rightarrow,\ p_1,\ p_2,\ t\rangle$ представлена связь типа импликации ($\Rightarrow$). O-вершина t в кванторном фрагменте указывает на истинность представленного выражения.

Нетрудно видеть, что если из сети рис. 7.5 убрать кванторный фрагмент, то она «развалится» на две части: первая соответствует ограничениям на свободную переменную, а вторая — области действия квантора. Обе эти части могут быть выделены нахождением фрагментов, связанных соответственно с вершинами x_1 и x_2. Напомним (см. § 1.3), что связанными с вершиной x_i фрагментами считаются такие, к которым от x_i имеется путь через вершины связи фрагментов и $н$-вершины. Через $о$-вершины путь запрещен.

Рассмотрим более сложный пример. На рис. 7.6 изображена сеть, представляющая предикатное выражение вида

$$(\exists X_1 \in M_1)\,[R_1(X_1) \wedge (\forall X_2 \in M_2)\,R_2(X_1,\ X_2)], \qquad (7.7)$$

где имеется два квантора $\exists$ и $\forall$, составляющих кванторные приставки $(\exists X_1 \in M_1)$ и $(\forall X_2 \in M_2)$. Этим кванторам с их приставками и областями

действия сопоставляются $с$-вершины γ_1 и γ_2. С помощью фрагмента $\langle \_\_,\ t,\ \leftarrow,\ \gamma_1,\ \gamma_2\rangle$ представляется, что *второй квантор входит в область действия первого*. Будем говорить, что второй квантор ($\forall$) п о д ч и н я е т с я ($\leftarrow$) первому ($\exists$). Данный фрагмент необходим для указания порядка выделения связанных частей, представляющих области действия кванторов. При таком представлении не требуется специальных средств выделения областей.

Остановимся более подробно на **принципах выделения областей**. Выбирается фрагмент, соответствующий с т а р ш е м у квантору, т. е. такому, который никому не подчиняется. На рис. 7.6 таким фрагментом будет $\langle \gamma_1,\ t,\ \exists,\ x_1,\ x_2\rangle$. Временно запрещается прохождение через его $н$-вершины x_1, x_2 и γ_1. Фрагмент как бы изымается. Тогда фрагменты, связанные с вершиной x_2, образуют область действия квантора $\exists$, а фрагменты, связанные с x_1, представляют ограничение на множество X_1. Далее выбирается фрагмент, соответствующий следующему по старшинству квантору. На рис. 7.6 таким фрагментом является $\langle \gamma_2,\ t,\ \forall,\ x_3,\ x_4\rangle$. Также временно запрещается прохождение через его $н$-вершины x_3, x_4 и γ_2. При этом предыдущие запреты остаются в силе. Тогда фрагменты, связанные с x_4, и образуют область действия квантора $\forall$, а фрагменты, связанные с x_3, представляют ограничения на множество X_3, т. е. $X_3 = M_2$.

Следует отметить, что описанные действия нахождения связанных фрагментов (сетей) реализуются стандартной процедурой выделения окрестностей, а для выбора кванторов в порядке их старшинства используются фрагменты типа $\langle \_\_,\ t,\ \leftarrow,\ \gamma_1,\ \gamma_2\rangle$. Никаких специальных процедур при этом не требуется. В этом одно из преимуществ описанного способа представления.

Введенные кванторные сети могут быть использованы для представления выражений языка **логики второго порядка**. Например, выражение вида $(\exists F \in \mathscr{F})\ F\ (A_1)$, где F — функциональная переменная, представляется следующим образом:

$$\langle \_\_,\ t,\ \in,\ x_1,\ '\mathscr{F}'\rangle \circ \langle \gamma_1, t,\ \exists,\ x_1,\ x_2\rangle \circ \langle \_\_,\ t,\ x_2,\ a_1\rangle, \qquad (7.8)$$

т. е. используются обычные конструкции. O-вершина '$\mathscr{F}$' соответствует *класссу функциональных переменных*, ограничивающих множество X_1. H-вершина x_1 соответствует свободной переменной F, а $н$-вершина x_2 — связанной. Обратите внимание, что $н$-вершина x_2 стоит во фрагменте

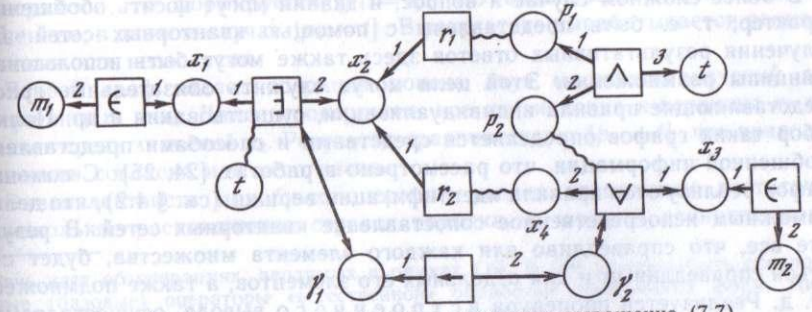


Рис. 7.6. Сеть, представляющая предикатное выражение (7.7)

$\langle \_\_, t, x_2, a_1 \rangle$ на третьем месте, т. е. она представляет *неизвестное отношение* или *функциональную зависимость*.

Выше рассматривались лишь отдельные примеры. Обобщив их, нетрудно показать, что любому предикатному выражению первого и второго порядков можно сопоставить сеть. Более того, подобные действия могут осуществляться автоматически, т. е. с помощью обязательных графов и продукций (см. § 6.4). Предикатные выражения будут рассматриваться как образующие внешний язык, отображаемый во внутрисистемные представления.

Ответ на запросы. Введенные сети могут играть роль пассивных знаний, используемых для ответа на запросы и проверки правильности сообщений. Будем различать несколько случаев. Наиболее простой из них, когда структура сети-запроса повторяет структуру одной из кванторных сетей, входящих в знания. Например, сеть-запрос имеет вид рис. 7.3, а сеть-знания — рис. 7.1, где m_1 соответствует *классу людей*, а r_1 — свойству *быть смертным*. Тогда ответ может быть найден с помощью типовой процедуры конкретизации (см. гл. 4). При этом вершины и фрагменты, представляющие кванторы ($\forall$, $\exists$), сопоставляются обычным образом.

Другой случай, когда вопрос носит конкретный характер, а в знаниях имеется нужная (обобщенная) информация, представленная с помощью кванторных сетей. Например, пусть спрашивается, *Смертен ли Сократ*?, что представляется

$$\langle \_\_, p_i, \text{'смертный'}, x_1^1 | \text{кл. людей} | \rangle \circ [p_i := ?], \tag{7.9}$$

где x_1^1 сопоставлена *Сократу* (для простоты его имя не представлено). Пусть в знаниях имеется сеть рис. 7.1, где M_1 — *класс людей*. Тогда в процессе поиска и конкретизации будет получено $x_1^1 := x_1$. На этом поиск закончится. Для выполнения последующих шагов необходимо, чтобы x_1^1 была каким-то образом идентифицирована с x_2, т. е. ей было присвоено значение $x_1^1 := x_2$. Тогда сделается возможным использование фрагмента, представляющего свойство R_2 (*смертный*). Для этого могут служить обязательные графы, обеспечивающие размножение вершин (см. § 5.3):

$$[\square_\phi y_1 := y_2] \circ [y_2 \perp \langle \_\_, t, \forall, y_1, y_2 \rangle]. \tag{7.10}$$

Применением графа (7.10) к композиции, составленной из сети-запроса (в котором $[x_1^1 := x_1]$) и сети-знаний, будет получено $[x_1^1 := \{x_1, x_2\}]$. Тогда становится возможным выполнение последующей операции группового ассоциирования, что приведет к $[p_i := t]$. Ответом будет *Да, смертен*.

В более сложном случае и вопрос, и знания могут носить обобщенный характер, т. е. быть представлены с помощью кванторных сетей. Для получения результативных ответов здесь также могут быть использованы принципы размножения. Этой цели могут служить обязательные графы, представляющие правила индивидуализации, существования и др. Полный набор таких графов определяется средствами и способами представления обобщенной информации, что рассмотрено в работах [24, 25]. С помощью графов реализуются правила идентификации вершин (см. § 4.2), что делает возможным непосредственное сопоставление кванторных сетей. В результате все, что справедливо для каждого элемента множества, будет считаться справедливым и для отдельных его элементов, а также подмножеств и т. д. Реализуется процедура встроенного вывода, осуществляемого в процессе сопоставления образцов — сетей.

§. 7.3. НАПРАВЛЕННАЯ ОБРАБОТКА ОБОБЩЕННОЙ ИНФОРМАЦИИ

Ниже будут рассматриваться принципы учета операционных оттенков, имеющих место при естественных формах выражения обобщенной информации. Речь будет идти о направленных (алгоритмических) способах ее обработки. Для этого будут вводиться конструкции СЯ, называемые кванторными графами. Напомним, что под такими графами понимаются кванторные сети, на которых сформированы стрелки порядка. Будут рассмотрены примеры их использования для представления вопросов и высказываний обобщенного характера, а также принципы реализации так называемой техники естественных рассуждений.

Простые конструкции. Рассмотрим два высказывания *Каждый критянин — лжец* и *Только лжецы могут быть критянами*. С точки зрения математической логики эти предложения выражают одно и то же, т. е. записываются в виде

$$(\forall X_1 \in M_1) R_1(X_1) \text{ или } \forall X_1 [(X_1 \in M_1) \Rightarrow R_1(X_1)],$$

где M_1 — *множество критян*, а R_1 — *быть лжецом*. И все же в этих высказываниях есть отличия, заключающееся в акцентации, в неявно задаваемом способе проверки их правильности. Первое высказывание, если оно поступило впервые, может восприниматься как гипотеза. Тогда оно звучит так: *Возьми любого известного тебе критянина и посмотри, ведь он лжец*. Подразумеваются проверки, так ли это. Если высказывание выражается достаточно уверенно, т. е. как некоторого сорта «обязательная» информация, то оно будет звучать так: *Помни, что каждый известный тебе критянин лжец*. Как бы навязываются представления о новых свойствах, предполагаются соответствующие изменения. Второе же высказывание имеет другой акцент. Говорится, что множество критян включено ($\subset$) в множество лжецов. И здесь тоже может быть оттенок обязательности. Только он не связан с указанием какого-либо свойства.

Для представления подобных оттенков будем использовать стрелки порядка и знаки $\square_\phi$. Рассмотрим вначале случай, когда речь идет о входном высказывании (гипотезе), требующей проверки. На рис. 7.7 и 7.8 изображены соответствующие графы. В первом стрелки $\mapsto$ сходятся к вершине $[p_2 := t]$, а во втором — к $[p_1 := t]$. Такие графы задают различные операции [1], но с эквивалентными результатами. Граф рис. 7.7 задает операции поиска множества значений x_1, их перебора с проверкой, что для каждого из них имеет место $[p_2 := t]$. И если во всех случаях проверки удовлетворяются, то формируется $[p_1^* := t]$, что сравнивается с имеющимся значением, в данном случае $[p_1 := t]$. В результате вырабатывается реакция согласия или несогласия.

Второй граф задает операции поиска двух множеств значений для n-вершины x_1 и x_2. Далее проверяется, все ли значения n-вершины x_1 находятся среди значений x_2. Результат сравнивается с $[p_2 := t]$, что приводит к реакции согласия или несогласия.

Эквивалентность подобных операций делает возможным достаточно произвольное расположение стрелок порядка, которые в принципе могут

[1] Речь идет об операциях, вводимых в рамках ОЯ. В ОЯ должны быть специальные (базовые) операторы «естественной» обработки обобщенных форм. Ниже такие операторы будут рассматриваться на примерах.

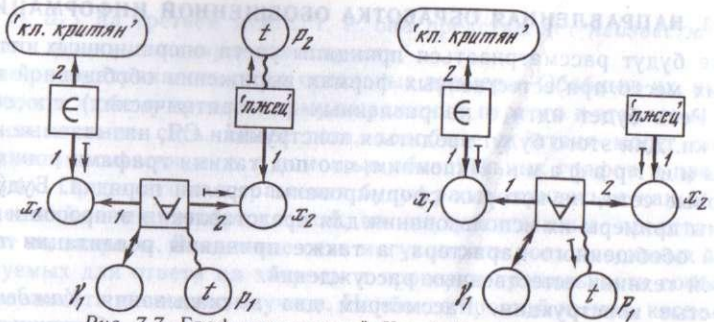


Рис. 7.7. Граф, означающий *Каждый критянин лжец*

Рис. 7.8. Граф, означающий *Только лжецы являются критянами*

сходиться к любой вершине кванторной сети. И с каждым таким расположением могут быть связаны свои операции. В § 4.1 говорилось, что для этой цели служат специальные метаправила. Тогда для выбора наилучшего направления поиска достаточно соответствующим образом расположить стрелки порядка, что может осуществляться самой системой (алгоритмы формирования рассмотрены в работах [26, 27]).

Пусть упомянутые высказывания имеют оттенок обязательности, т. е. это всем известные истины. Тогда для их представления также могут использоваться графы рис. 7.7 и 7.8. Только вершины, к которым сходятся стрелки порядка, будут помечены знаком $\square_ф$. У первого графа будет $[\square_ф p_2 := t]$. Он задает рассмотренные ранее проверки. Последние выполняются над сетью $T_{исх}$, представляющей исходную или анализируемую информацию. И если имеет место случай невыполнимости проверок, т. е. в $T_{исх}$ представлен некий критянин без указанного свойства, то осуществляется формирование фрагмента, соответствующего данному свойству.

Граф рис. 7.8, у которого $[\square_ф p_1 := t]$, представляет более индифферентное высказывание с его оттенком обязательности. В явном виде не указывается, какого свойства может нехватать и что нужно формировать в $T_{исх}$. Предполагается, что система каким-то образом сама должна добиться выполнения проверок. Это может быть сделано, в частности, изменением направленности стрелок порядка с формированием других (эквивалентных) обязательных графов. Операции, задаваемые ими, выполняются над $T_{исх}$, что вызовет ее пополнение. Таким способом можно добиться выполнения проверок, включения ($\subset$) множеств.

Граф рис. 7.7 может быть расширен до конструкции следующего вида:

$$[x_1 \perp \Gamma_1(x_1)] \circ [x_2 \perp \langle \gamma_1,\ p_1,\ \forall,\ x_1,\ x_2 \rangle] \circ \Gamma_2(x_2). \qquad (7.11)$$

Представляется, что *каждый объект, имеющий свойства* $\mathcal{T}_1$ (*или связанный указанными отношениями*), *обладает и свойствами* $\mathcal{T}_2$ (*или связан отношениями* $\mathcal{T}_2$). Граф $\Gamma_1(x_1)$ задает операции поиска множеств значений x_1. Такие операции соответствуют определенному способу выявления множества (объектов), которое пробегает свободная переменная. Далее осуществляется действие перебора, что задается фрагментом с *о*-вершиной $\forall$. Элементы множества значений последовательно ставятся в $\Gamma_2(x_2)$ на место x_2

(или присваиваются как значения x_2). Далее выполняются операции, задаваемые графом $\Gamma_2(x_2)$. Соответственно анализируется справедливость свойств $\mathcal{T}_2$ у перебираемых объектов (элементов). В результате вырабатывается реакция согласия или несогласия, о которой говорилось выше.

Поставим (7.11) перед $\Gamma_2(x_2)$ знак $\square_ф$. Получится обязательный граф, представляющий дополнительно *должен обладать свойствами* $\mathcal{T}_2$. Соответственно задается операция ф о р м и р о в а н и я (см. § 5.3). Если для какого-либо из перебираемых элементов (d_i) не выполняются операции, задаваемые $\Gamma_2(x_2)$, то формируются фрагменты $T_2(d_i)$, которые добавляются к $T_{исх}$.

Заметим, что аналогичные операции задаются и обязательным графом $\Gamma_1(y_1^!) \circ \square_ф \Gamma_2(y_1^!)$, где $y_1^!$ — фокус Γ_1. Конструкция (7.11) со знаком $\square_ф$ перед Γ_2 может быть преобразована к более простому виду, не содержащему кванторного фрагмента. При этом будет представлено *объект, обладающий свойствами* $\mathcal{T}_1$, *должен обладать и свойствами* $\mathcal{T}_2$. Обратите внимание на смысловую близость интерпретирующих предложений. Источник подобной близости в эквивалентности операционных оттенков.

Рассмотрим некоторые варианты конструкции (7.11). Заменим в ней *о*-вершину $\forall$ на $\exists$. Тогда будет представлено уже другое высказывание: *некоторые объекты, имеющие свойства* $\mathcal{T}_1$, *обладают и свойствами* $\mathcal{T}_2$, а если заменить $\forall$ на $\exists_1$, то *Некоторый объект* ... Тогда будут задаваться рассмотренные ранее операции поиска значений x_2 и перебора. Но уже будет требоваться, чтобы проверки, задаваемые $\Gamma_2(x_2)$, выполнялись хотя бы для одного элемента. В зависимости от этого будет вырабатываться реакция согласия или несогласия.

Если к конструкции (с *о*-вершиной $\exists$) добавить знак $\square_ф$, поставив его впереди $\Gamma_2(x_2)$, то будет представлен оттенок обязательности. Получится обязательный граф, задающий операции проверки и формирования. Если проверки, задаваемые Γ_2, не удовлетворяются ни для какого элемента из множества значений x_2, то осуществляются следующие действия. Берется «резервная» *н*-вершина $x_i^!$ (не входившая ранее в $T_{исх}$) и формируется сеть $T_2(x_i^!)$, которая добавляется к $T_{исх}$. Подобные действия соответствуют предположению о наличии неизвестного ранее объекта со свойствами $\mathcal{T}_1$ и $\mathcal{T}_2$. В дальнейшем могут делаться попытки уточнения, что это за объект, что сводится к нахождению значений $x_i^!$.

Перейдем к рассмотрению конструкций, которые получаются на базе графа рис. 7.8. Он может быть расширен следующим образом:

$$[x_1 \perp \Gamma_1(x_1)] \circ [p_1 := t] \circ [p_1 \perp \langle \_\_,\ p_1,\ \forall,\ x_1,\ x_2 \rangle] \circ [x_2 \perp \Gamma_2(x_2)]. \qquad (7.12)$$

Представляется, что *только объекты со свойствами* $\mathcal{T}_2$ (*или связанные отношениями* $\mathcal{T}_2$ *с другими объектами*) *могут обладать свойствами или отношениями* $\mathcal{T}_1$. Соответственно графы Γ_1 и Γ_2 задают операции поиска множеств значений *н*-вершин x_1 и x_2, а оставшиеся фрагменты — операцию теоретико-множественного включения. В зависимости от ее результата вырабатывается реакция согласия или несогласия. Если в конструкции перед p_1 поставить знак $\square_ф$, сделав $[\square_ф p_1 := t]$, то будет представлен оттенок обязательности.

Заменим в (7.12) *о*-вершину $\forall$ на $\exists$. Тогда уже будет представлен *факт существования объектов со свойствами* $\mathcal{T}_1$ и $\mathcal{T}_2$. Фрагмент с *о*-вершиной $\exists$ будет задавать операции проверки наличия непустого пересечения

найденных множеств. Заметим, что такая же проверка задается более простой конструкцией $[x_1 \perp \Gamma_1(x_1) \circ \Gamma_2(x_1)]$, т. е. имеет место сводимость.

Естественные способы представления. На базе введенных ранее графов могут составляться более сложные конструкции. Части $\Gamma_1(x_1)$ и $\Gamma_2(x_2)$ из (7.11) и (7.12) сами могут иметь вид кванторных или составных графов. Конструкции такого типа могут служить для согласованного представления обобщенной информации, выражаемой с помощью типовых форм ЕЯ. Графы задают операции, с помощью которых учитываются операционные оттенки форм и обеспечиваются различные виды обработки. Рассмотрим несколько наглядных примеров.

На рис. 7.9 изображен кванторный граф, представляющий *каждый любит кого-либо*. *О*-вершина *лб* соответствует отношению *любить* (с двумя семантическими падежами). *Н*-вершина x_2 сопоставлена *каждому*, а x_3^1 — *какому-либо человеку*. С помощью $[x_3^1 := \lambda]$ представляется *факт существования такого человека*. Граф задает операции, которые соответствуют нахождению множества X_1 (известных системе) людей и их перебору. Далее для каждого X_2 из этих людей находится множество X_3 тех, кого он любит, и проверяется его непустота, т. е. что имеются любимые люди. Такие операции могут быть выполнены над любой сетью T, представляющей конкретную информацию. В результате может быть проверено, сколь правильно данное высказывание, сколь оно согласуется с этой информацией.

Поставим на рис. 7.9 перед *н*-вершиной x_3^1 знак $\square_\phi$. Тогда дополнительно будут задаваться операции формирования. Если у кого-либо из множества X_1 не было найдено любимого человека, то в T формируются представления о нем. Хотя, что это за человек, не уточняется. Ему сопоставляется своя резервная *н*-вершина.

На рис. 7.10 изображен другой граф, который представляет (дословно) *Имеется человек, который любим каждым*. Такому человеку сопоставлена вершина x_3 |*люди*|, что в сокращенном виде представляет *принадлежность к классу людей*. Граф задает операции, которые выполняются над T. Они соответствуют перебору известных системе людей с отсевом вариантов (см. § 4.3). Для каждого человека проверяется, что те, кого он любит, включают в себя все множество людей. Если проверка не удовлетворяется, то вариант считается неприемлемым. Берется другой человек ... и так до первого случая, когда проверка будет выполнена. Если такой случай най-

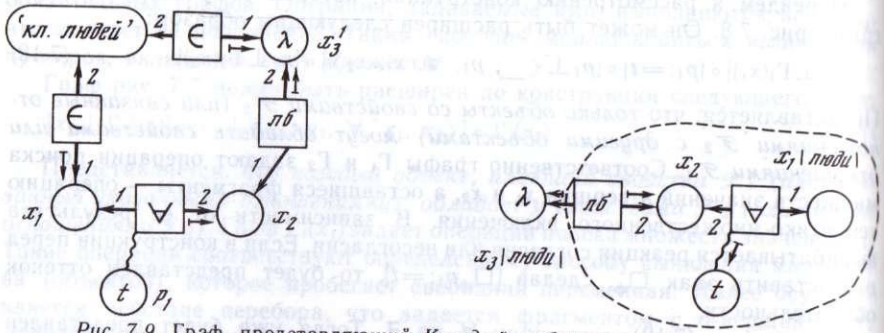


Рис. 7.9. Граф, представляющий *Каждый любит кого-либо*

Рис. 7.10. Составной граф, представляющий *Кто-либо любим каждым*

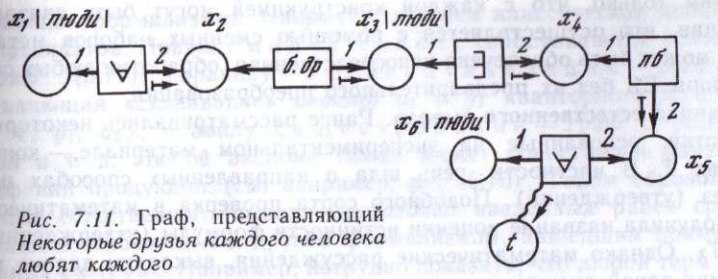


Рис. 7.11. Граф, представляющий *Некоторые друзья каждого человека любят каждого*

ден, то это говорит о правильности высказывания, т. е. что оно согласуется с представленной в T информацией.

Поставим на рис. 7.10 перед *н*-вершиной x_3^1 знак $\square_\phi$. Тогда будет представлено, что *должен быть такой человек, который любим каждым*. Дополнительно задаются операции формирования представлений о таком человеке (если ранее о нем ничего не было известно).

На рис. 7.11 изображен более сложный граф, представляющий *Некоторые друзья каждого человека любят каждого*. *Н*-вершина x_3 сопоставляется *друзьями каждого человека*, x_4 — *некоторым из них*, а x_5 — *тем, кого последние любят*. Граф задает операции двойного перебора, т. е. цикла в цикле. Перебираются люди X_1 с нахождением для каждого из них множества друзей X_3. Далее перебираются эти друзья с попыткой нахождения такого, чтобы удовлетворялось $X_5 \supset X_6$, где X_5 — множество тех, кого он любит, а X_6 — множество всех людей. Если такой друг найден, то берется следующий человек из X_1. Для каждого из них должен иметься указанный друг.

Приведенные примеры иллюстрируют принципы использования кванторных графов для **согласованного представления** форм ЕЯ. Такие графы могут строиться автоматически в процессе восприятия форм. Каждой категории последних (имеющей самостоятельный смысл) сопоставляется свой элемент графа. Подобной согласованности не удается добиться при использовании существующих логических средств. Требуется переиначивание, которое во многих случаях будет приводить к тому, что предложение станет менее понятным. Например, постарайтесь записать в языке логики предикатов то, что представлено на рис. 7.11. Прочтите полученное выражение и сравните его с исходным. Вы сами убедитесь в справедливости сказанного.

Выше рассматривались наиболее простые конструкции. Но уже с их помощью может быть обеспечено согласованное представление достаточно большого количества типовых форм ЕЯ. С помощью знаков $\square_\phi$ удается учитывать оттенки долженствования, изменением стрелок порядка — акцентацию и т. д. Полученный граф можно постараться преобразовать к более простому виду, в частности, изменив направленность стрелок порядка, изъяв некоторые кванторные фрагменты, о чем говорилось выше. Таким способом информация может быть сделана более ясной, понятной. Возможности здесь высокие.

Конечно, в общем случае введенных конструкций недостаточно. Они должны быть дополнены другими, например представляющими формы типа *Для каждого существует*, *Существует такой, что каждый* и др. Достаточно полный набор таких конструкций (графов) рассмотрен в работе [14]. Чтобы не загромождать изложение, мы не будем останавливаться на них.

Отметим только, что с каждой конструкцией могут быть связаны свои операции, что осуществляется с помощью сменных наборов метаправил. Тогда может быть обеспечена непосредственная обработка любых обобщенных форм ЕЯ без их предварительного преобразования.

Техника естественного вывода. Ранее рассматривались некоторые виды обработки, основанные на экспериментальном материале,— конкретных знаниях *Т*. В частности, речь шла о направленных способах проверки гипотез (утверждений). Подобного сорта проверка в математической логике получила название «оценки истинности формулы (утверждения) в модели *Т*». Однако математические рассуждения, выкладки далеко выходят за рамки такой оценки.

Возьмем для примера задачи геометрии. Дана фигура. Указаны ее особенности. Требуется доказать некоторые свойства или соотношения. Процесс доказательства состоит из множества этапов, имеющих целью достройку фигуры (проведение новых линий...) и порождение на базе одних свойств других. Для этого используются определенного сорта знания. Как выясняется, и те, и другие знания могут быть представлены в виде обязательных графов или продукций. Процесс доказательства реализуется применением их к сети, представляющей исходную фигуру. В результате сеть будет достраиваться ... пока в ней не появятся фрагменты, соответствующие тому, что требуется доказать.

Сказанное подтверждается экспериментальной проверкой. Были взяты типовые примеры из области **планиметрии**, см. [37]. Оказалось, что введенных ранее семантических сетей достаточно для представления любой фигуры (с ее свойствами) с любой степенью точности. Обязательных графов и продукций достаточно для представления теорем, а также различных приемов, связанных с построением и выявлением новых фигур — компонент исходной. Выполнение задаваемых ими операций реализует процедуру доказательства. Точно таким же способом реализуются и более естественные рассуждения, принципы реализации которых рассматривались в гл. 5.

Следует заметить, что в процессе доказательств, в частности, в области планиметрии, человек широко использует возможности своего восприятия, видения. Многие шаги определяются наблюдаемыми метрическими отношениями. Доказывается равенство лишь тех углов или линий, которые на чертеже выглядят равными. Утверждения, не совместимые с метрическими отношениями, заведомо отбрасываются. Система лишена подобного видения. Ей необходимы специальные средства («эвристики») целенаправленного порождения. Человек обладает способностью воспринимать (держать в голове) только то, что может пригодиться в текущий момент. Соответственно при внутрисистемной обработке возникает задача выбора и минимизации представлений. Дело в том, что в фигурах имеется множество различных компонент (точки, отрезки, линии, углы). И если представлять все отношения между ними, то возникают весьма сложные семантические сети, которые, по возможности, должны быть упрощены самой системой.

В общем случае истинные высказывания (аксиомы, теоремы), а также типовые правила вывода представляются в виде обязательных графов и продукций. Сама **процедура доказательства** (порождения на основе аксиом — теорем) реализуется применением таких графов и продукций друг к другу. Ранее уже рассматривались некоторые примеры такого сорта (см. § 5.3). Указанным способом могут быть реализованы процедуры, задаваемые в рам-

ках различных формализмов. Например, возьмем классический аппарат — логику предикатов. Правилу модус поненс сопоставляется обязательный граф (5.17), правилу индивидуализации — продукция, осуществляющая «склеивание» вершин y_1 и y_2 кванторного фрагмента $\langle \_,\ t,\ \forall,\ y_1,\ y_2 \rangle$, правилу существования — граф с резервной вершиной и т. д. Любой аксиоме также может сопоставлен обязательный граф или продукция (см. например, рис. 5.16). Таким образом, для реализации естественного вывода достаточно введенных ранее средств.

Сказанное относится и к другим формализмам, нашедшим применение в различных системах. Например, нетрудно показать, что любой терм языка ПРОЛОГ может быть представлен с помощью сети, а любая редукция («секвенция») — с помощью продукции. Отсюда, все, что делается в языке ПРОЛОГ, может быть реализовано и в рамках предлагаемого подхода.

Наконец, остановимся на **методе резолюций** и процедуре доказательства Эрбрана [51]. Они основаны на специальной бескванторной записи (скулемовской стандартной форме) и интерпретации (т. е. предполагается порождение примеров). Подобные формализмы отражают некоторые виды рассуждений, используемых человеком. Нетрудно показать, что для реализации таких рассуждений также достаточно введенных средств. При этом можно провести четкие аналогии, в частности, между процедурой поиска рассогласований, используемой в методе резолюций, и сопоставлением фрагментов (с использованием правил идентификации). Аналогом процедуры порождения примеров на базе эрбрановского универсума является перенос сетей и графов, представляющих высказывания, в системные знания (см. § 5.2). В процессе такого переноса осуществляется формирование вершин, соответствующих новым (неизвестным ранее) объектам. Более того, многие эвристические приемы, нашедшие применение в методе резолюций, могут быть в той или иной форме перенесены в аппарат сетей и графов. В частности, правилу чистых литер можно сопоставить действие изъятия из знаний обязательных графов с о-вершинами, отсутствующими в других графах, и т. д.

Сказанное иллюстрирует, во-первых, близость подходов и, во-вторых, достаточную универсальность предлагаемого подхода, где удается найти место тому, что характерно для типовых формализмов. Система может быть настроена на работу в соответствующем режиме. При этом могут реализовываться различные методы, в том числе метод обратных рассуждений (см. § 5.2). В процессе доказательств могут широко использоваться нечеткие понятия (см. § 38 в работе [14]). Хотя задача обеспечения целенаправленного вывода остается открытой. Видимо, здесь также перспективно использование обязательных и кванторных графов, представляющих внутрисистемные акции на более высоких уровнях (см. § 3.1).

§ 7.4. АНАТОМИЯ ПАРАДОКСОВ

Вызывают улыбки миниатюры типа:

— Почему Вы ушли от своего мужа?

— Потому что я ему твердила, какой у меня мягкий характер, а он не соглашался.

Парадоксальные и противоречивые суждения присущи человеческому мышлению и, в общем-то, в обычном общении не вызывают каких-либо непри-

ятностей. По-настоящему они дают себя знать, лишь когда ставится задача создания стройной теории, т. е. в области математики. Тогда возникают вопросы **Почему методы рассуждений, казавшиеся столь убедительными, приводят к парадоксам?, Как изменить теорию, чтоб в ней не возникали парадоксы? [18].**

Для изучения данных вопросов представляется перспективным использование введенного в § 6.1 понятия математической машины. Как было показано ранее, математические построения и рассуждения могут рассматриваться как определенного вида процессы, происходящие в рамках такой машины (см. рис. 6.1). Нетрудно видеть, что в ней допускаются различные «контуры» обратных связей. В частности, за счет знаний пополняются и корректируются сами же знания. Знания могут применяться сами к себе, что допускается на любом из уровней [2]. Контуры обратных связей могут возникать как на отдельных уровнях, так и между уровнями за счет явления переноса.

Далее, в привычном нам языке выражается не только информация о ПО, но и способ пользования этой информацией (что отражено на схемах рис. 3.1 и 4.1). И то, и другое отображается на знания. Получается, что знания в какой-то спепени определяют способ использования самих же знаний. Имеют место свои контуры обратных связей более сложного вида.

Как известно в системотехнике, при наличии обратных связей возможны разного рода **неустойчивые** явления — колебательные и др. Процесс может быть затухающим или, наоборот, усиливающимся. Аналогичные явления характерны и для схемы рис. 6.1. Естественно допустить, что важному понятию чистоты математических рассуждений может быть поставлен в соответствие процесс, когда происходит лишь накопление знаний на всех уровнях. При этом недопустимы случаи, когда справедливо и то, и совершенно другое, т. е. противоречия, парадоксы. Как выясняется, их можно связать с колебательными, неустойчивыми явлениями, которые в принципе характерны для схемы рис. 6.1.

Как же добиться протекания процессов в нужном русле? Просто разорвать обратные связи — значит, сделать невозможными никакие процессы, в том числе связанные с накоплением лингвистических знаний, с порождением на базе одних утверждений других и т. д. Следовательно, необходимы разумные ограничения, чтобы исключить возможность отрицательных обратных связей. И в общем-то при создании математических теорий, как правило, накладываются такие ограничения. Например, вводятся специальные средства, делающие невозможными некоторые виды суждений, как в теории типов Рассела. Объявляются недопустимыми утверждения (аксиомы), делающие ту или иную теорию противоречивой и др. Однако при этом затрагиваются лишь некоторые уровни, контролируются отдельные контуры обратных связей. А их множество. Все они должны в какой-то степени контролироваться. В частности, сказанное относится к контуру управления, см. рис. 3.2. Управляющие сети (T_y) должны правильным образом «взаимодействовать» с сетями (T_c), составляющими уровень обработки. Кстати, именно этот контур является источником так называемых неразрешимых проблем. Еще один важный контур вызван наличием метауровня и метаправил (см. рис. 4.1). Последние образуют специального сорта знания,

[2] Интересные обратные связи (типа «описательной рекурсии») возникают при наличии c-вершины или p-вершины какой-либо сети внутри ее самой.

определяющие внутрисистемные умения. Здесь также возможны отрицательные обратные связи. Из сказанного следует важность изучения процессов, приводящих к нежелательным явлениям. Ниже будет начато такое изучение на примерах типовых парадоксов.

Парадокс Рассела. Он связан с множеством всех множеств, которые не являются элементами самих себя. Рассмотрим, какие внутрисистемные акции приводят к этому парадоксу. На рис. 7.12 изображена продукция $\Gamma_1 \xrightarrow[Df]{} \Gamma_2$, представляющая на уровне СС, что *если* Y_1 *не является (f) элементом самого себя, то он соотносится к множеству* S. Справедливым считается и обратное. Имеет место разновидность обязательных знаний, с помощью которых могут пополняться и уточняться другие знания, что осуществляется путем применения продукции. При этом допускается применение в обе стороны, на что указывает отношение типа Df.

Пусть на выход системы поступил вопрос *Является ли S элементом самого себя?*, что представляется в виде сети, обозначенной на рис. 7.12 через $T_{вх}$. Имеется два допущения. Первому из них *Да, является* соответствует $[p_1 := t]$. Тогда делается возможным применение продукции в обратную сторону, что приведет к $[y_1^1 := s]$ и формированию $[p_1 := f]$. А раз так, то уже невозможно $[p_1 := t]$. (Подобный вывод осуществляется применением соответствующего обязательного графа, он не изображен на рисунке.)

Второму допущению *Нет, не является* соответствует $[p_1 := f]$. Тогда продукция будет применимой напрямую, что приведет к формированию в $T_{вх}$ $з$-сети $[p_1 := t]$, исключающей первую. Возникает колебательный процесс. Заметим, что если на рис. 7.12 присвоить значение $[y_1^1 := s]$, что соответствует действию индивидуализации, то продукция «разделится» на две противоречивые части, т. е. исключающие одна другую. Имеет место коллизия.

Аналогичный характер имеет **парадокс парикмахера.** *Некий человек S бреет тех и только тех, которые не бреются сами.* Спрашивается, *бреет ли он сам себя?* Утверждение переиначивается *S бреет* Y_1, *если* Y_1 *не бреется сам, и наоборот,* что представляется в виде продукции рис. 7.12. Только вместо $\in$ будет o-вершина *'брить'*, соответствующая действию с двумя семантическими падежами — *кто бреет* и *кого бреет*.

Парадокс лжеца. На рис. 7.13 изображен граф Γ_1, представляющий *Критянин утверждает, что все критяне говорят (гов) ложь* (т. е. *только то, чего нет*). Последней части высказывания сопоставлен граф $\square_\phi \Gamma_2$ с p-вершиной p_1. С помощью $[p_1 := t]$ представлено, что X_1 *говорит истину*, а с помощью $[\square_\phi p_2 := f]$ — что X_3 (*любой критянин*) *говорит ложь*.

Пусть система «поверила» критянину и поместила $\square_\phi \Gamma_2$ в свои знания. С помощью них может быть скорректирована любая входная информация, в том числе и сам граф Γ_1. Граф $\square_\phi \Gamma_2$ оказывается применимым к Γ_1, что приведет к насильственному формированию $[p_1 := f]$. А раз так, то $\square_\phi \Gamma_2$ будет уже представлять ложное высказывание. При этом $\square_\phi \Gamma_2$ может быть преобразован в другую конструкцию, означающую *Существует такой критянин, который говорит правду.* На этом процесс закончится.

Итак, имеет место самоприменимость, приводящая к замене значения p_1 с t на f, что соответствует «модификации» исходного высказывания,

делающей невозможным его последующее использование. За счет обратной связи как бы осуществляется «самосброс», т. е. при попытке установления определенного состояния система сама переходит в другое — устойчивое состояние.

Дилемма Крокодила. *Крокодил украл ребенка, но обещал вернуть его отцу, если тот угадает, вернет ли ему крокодил ребенка.* Неразрешимая дилемма возникает перед крокодилом, если отец скажет ему, что *он не вернет ребенка*. На рис. 7.14 изображен граф Г, представляющий «знания» крокодила. С помощью T_1 (p_1) $\circ$ $[p_1 : = f]$ представлено *утверждение (утв) отца (A_1), что крокодил не вернет ребенка*, а с помощью T_2 (p_2) — действие крокодила: $[p_2 : = t]$ означает, что *крокодил возвращает ребенка*, а $[p_2 : = f]$ — что *нет*. Фрагмент $\langle \_\_, p_2, =, p_1, p_2 \rangle$ соответствует *угадыванию*, т. е. если p_1 и p_2 имеют одно и то же значение, то $[p_3 : = t]$, иначе — $[p_3 : = f]$. Еще один фрагмент $(e\text{-}т)$ представляет *условный переход*. Граф Γ_3 (p_3) $\circ$ $[\square_ф\, p_3 : = t]$ соответствует *обещанию вернуть ребенка* (такой граф допускается к применению при $[p_3 : = t]$), а граф Γ_4 (p_4) $\circ$ $\circ$ $[\square_ф\, p_4 : = f]$ — не вернуть. Он допускается к применению при $[p_3 : = f]$.

Сеть T_2 и графы Γ_1, Γ_2 представляют одно и то же действие *вернуть*. Они содержат одни и те же фрагменты, только с различными *н*-вершинами

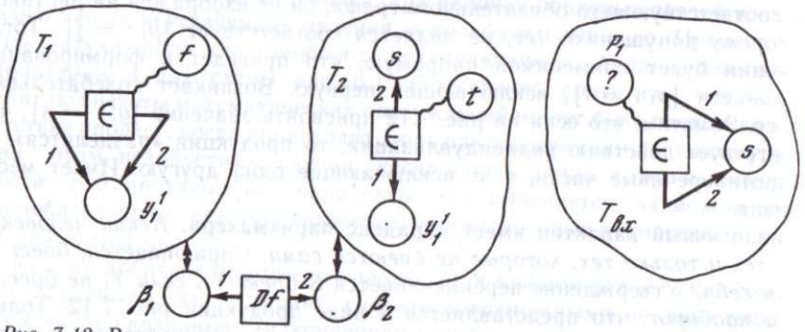


Рис. 7.12. Внутрисистемные представления, связанные с парадоксом Рассела

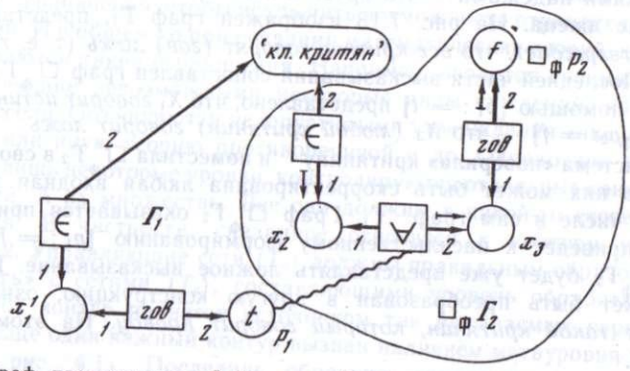


Рис. 7.13. Граф, представляющий *Критянин* (X_1) *утверждает* (*гов*), *что все критяне говорят ложь* (*f*)

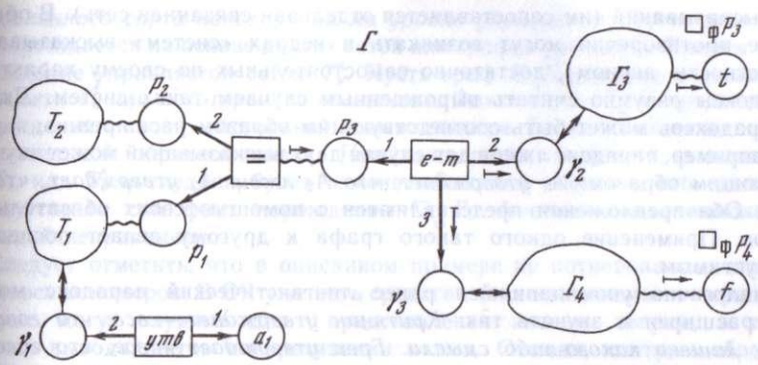


Рис. 7.14. Внутрисистемные представления, приводящие к дилемме крокодила

(добавляются и *з*-сети). Отсюда оба графа будут применимыми к T_2. При этом применение первого графа приводит к $[p_2 : = t]$. В результате уже будет допущен к применению граф Γ_4, что приведет к $[p_2 : = f]$ и вызовет применение Γ_3. Возникает к о л е б а т е л ь н ы й процесс, вызванный отрицательной обратной связью. Он и находит свое выражение в виде упоминавшейся дилеммы.

Каждый из рассмотренных парадоксов имеет множество различных вариантов, модификаций. Их суть одна: допустимые действия над знаниями (за счет обратных связей) приводят к недопустимым явлениям, т. е. в данном случае противоречиям. При этом ранее речь шла об обратных связях, возникающих на одном из уровней. Имеется в виду уровень СС. Однако контуры обратных связей могут возникать и за счет нескольких уровней. Рассмотрим пример.

Лингвистические парадоксы. Наиболее представительный из них — парадокс Берри, который уже упоминался в § 6.1. Но самым простым является следующий: *Никакое предложение, в котором встречается слово «значение», не имеет никакого значения.* Если соответствующий граф поместить в обязательные знания, с помощью которых анализируется необходимость осмысления предложений, то он исключит и возможность осмысления данного предложения, т. е. сделает невозможным формирование самого себя. В данном случае в обратной связи принимает участие несколько уровней — ПС, ТС и $Т_{пер}$. Другой вариант данного парадокса: *Никакое предложение, в котором говорится о значении чего-либо, не имеет значения.* Еще один парадокс: *Во всех предложениях, где Петр упоминается не более двух раз и речь идет об Иване, под последним имеется в виду Петр.* Это утверждение может быть применено к самому себе. Тогда под *Иваном* будет пониматься уже *Петр*, но последний будет упоминаться более двух раз. Следовательно, такого сорта понимание недопустимо.

Подобных парадоксов множество. И в общем-то они сравнительно легко могут порождаться на базе схемы рис. 6.1. Любая обратная связь, в том числе вызванная явлением переноса (~~~), может служить источником парадоксов. Несколько в стороне стоит лишь парадокс Ришара, имеющий свою природу.

Об оценке допустимости. Парадоксы — есть иллюстрация противоречий, которые возникают в недрах отдельных высказываний или же группы связан-

ных высказываний (им сопоставляется отдельная связанная сеть). В общем случае противоречия могут возникать в недрах «систем» высказываний (в частности, аксиом), достаточно самостоятельных по своему характеру. Парадоксы разумно считать вырожденным случаем таких систем. Любой из парадоксов может быть соответствующим образом расширен.

Например, парадокс лжеца для случая двух высказываний может звучать следующим образом: *A_1 утверждает, что A_2 лжец. A_2 утверждает, что A_1 лжец.* Оба предложения представляются с помощью своих обязательных графов. Применение одного такого графа к другому делает последний недопустимым.

Аналогично упоминавшийся ранее лингвистический парадокс может быть расширен и звучать так: *Критянин утверждает — все, что говорят греки, лишено какого-либо смысла. Грек утверждает — все, что говорят критяне, бессмысленно.* Здесь одно предложение, отображенное на лингвистические знания, исключает возможность отображения (осмысления) другого.

Следует учитывать, что понятие противоречивости не охватывает всех видов недопустимых явлений. Например, при организации базы знаний недопустимыми могут быть объявлены случаи наличия альтернатив, представлений о неуточненных объектах и др.

Сказанное позволяет подойти к задаче оценки допустимости процессов и соответственно «чистоты» реализуемых акций. Такая задача для уровня СС (см. рис. 6.1) формулируется следующим образом. Имеется множество обязательных графов $\{\square_{\phi} \Gamma_i\}$, представляющих истинные утверждения (знания о ПО), и множество $\{\Gamma_j\}$, представляющих допустимые явления. Знания считаются допустимыми, если при любых применениях одних графов из $\{\square_{\phi} \Gamma_i\}$ к другим будут выполнимы проверки, задаваемые $\{\Gamma_j\}$.

В другом варианте задача может рассматриваться относительно «универсальных» знаний T, которые в теории моделей названы интерпретацией. Предполагается применение $\{\square_{\phi} \Gamma_i\}$ к T в различных вариантах. И если не удается породить конструкции, не удовлетворяющей проверкам $\{\Gamma_j\}$, то знания $\{\square_{\phi} \Gamma_i\}$ объявляются допустимыми.

Задача более общего вида затрагивает все звенья схемы рис. 6.1. При этом понятие допустимости должно определяться вне данной схемы (на метауровне) и вовлекать в себя все ее уровни. В частности, недопустимыми могут быть объявлены случаи использования знаний для представления акций работы над этими же знаниями. Чем же чревато такое использование? Остановимся на этом вопросе несколько подробнее.

О неразрешимых проблемах. Они возникают в рамках теорий, которые допускают формализацию самой процедуры вывода и порождения новых теорем (для этой цели вводятся специальные гёделевские номера). Оказывается, что некоторые виды утверждений, касающиеся вывода и порождения, нельзя ни доказать, ни опровергнуть. В общем-то, именно этот факт положен в основу знаменитых теорем Гёделя о неполноте и неразрешимости теорий [3, 4].

Причина описанного явления в так называемой самоприменимости. Здесь сравнительно легко просматривается один из контуров обратных связей. Для иллюстрации обратимся к схеме рис. 3.2, где имеется уровень обработки (T_c) и уровень управления, т. е. управляющие сети (T_y). Пусть на уровне T_c имеется граф $\square_{\phi} \Gamma_i$, который говорит *о невозможности*

определенного сорта конструкций на уровне управления. Пусть этот граф сам порождается на основе других сетей или графов уровня T_c. Причем такое порождение управляется сетью T_y^*. Пусть граф $\square_{\phi}\Gamma_i$ говорит *о невозможности сети T_y^* или некоторых ее компонент.* Фактически $\square_{\phi} \Gamma_i$ представляет утверждения о своей собственной невыводимости. Тогда и возникает контур отрицательной обратной связи. При выполнении операций, задаваемых $\square_{\phi} \Gamma_i$, над T_y^* последняя будет изменяться таким образом, что сделается недопустимым сам факт порождения $\square_{\phi} \Gamma_i$. А стало быть, выполнение операций неправомерно.

Следует отметить, что в описанном примере не потребовалось никаких геделевских номеров. В СЯ уже есть средства (так называемые *с*-вершины), которые могут служить для представления аксиом, теорем, а также цепочек вывода, актов конкретизации и т. д. Отсюда, контуры обратных связей просматриваются в более явном виде.

Выше были затронуты лишь некоторые аспекты. Без внимания остались процессы, связанные с канторовским диагональным методом, проблемой алгоритмической неполноты и многое другое. Представляется перспективным их комплексное рассмотрение в рамках метаматематической машины, введение разумных ограничений с целью направить процессы в нужное русло.

Чрезвычайно интересной является проблема **«устойчивости» при рекурсивных обращениях**. Можно ли рассуждать о каких-либо построениях (абстрактных, логических), находясь в рамках процессов, основанных на этих построениях? Останутся ли правильными подобные рассуждения при изменении базиса (элементов, приемов), на котором держатся построения? Ведь при этом будет меняться и русло, в котором протекают сами рассуждения. С этими вопросами непосредственно связана проблема обоснованности некоторых исследований логического плана. Может оказаться, что в принципе возможны различные базисы или логики (в широком понимании). Более того, основанные на них интеллектуальные автоматы никогда не смогут понять друг друга.

§ 7.5. ОБУЧЕНИЕ ПО ПРИМЕРАМ. СТАТИСТИЧЕСКИЕ ЗАКОНОМЕРНОСТИ

Ниже будут рассматриваться индуктивные механизмы на семантических сетях и связанные с ними задачи обучения. Такие механизмы необходимы в тех случаях, когда система не может справиться с заданием пользователя имеющимися у нее средствами. Тогда делается попытка формирования новых знаний — обобщенного характера, что осуществляется на основе конкретного материала, опытных данных. Формируются знания об особенностях классов объектов (эмпирических закономерностях).

К настоящему моменту разработан ряд интересных методик автоматического формирования гипотез, поиска закономерностей, например *ДСМ*-метод, см. также [12]. В основном они базируются на нотации языка логики предикатов, в котором записываются «протокольные» предложения и сами гипотезы, закономерности.

В данном параграфе будут развиваться методики, работающие на плоских структурах (т. е. не привязанные к линейным формам записи, что дает определенные преимущества в плане унификации). Для представления эмпирического материала будут использоваться семантические сети, а для представления гипотез, закономерностей — графы с их операционной

семантикой (см. § 4.1). Будут рассматриваться различные закономерности, в том числе статистические. Для их представления будет введено новое понятие — поверхностного графа. Будет показано, что формирование закономерностей сводится к построению графов по упомянутым сетям, а использование закономерностей — к применению графов (выполнению операций, задаваемых графами, над сетью, представляющую очередное задание).

Задачи обучения. Ниже в основном речь будет идти о системной ориентации на задания, выражаемые в виде требований нахождения неизвестных объектов, компонент, свойств по фиксированной схеме отношений (задача конкретизации, недоопределенность по значению). Будут затрагиваться также простейшие задания, связанные с требованиями дополнения исходной информации (недоопределенность по структуре, см. [36]). К указанным требованиям сводится сравнительно широкий круг традиционных задач: ответ на вопросы, распознавание образов (уточняется принадлежность объекта к классу), диагностика, принятие решений (описание текущей ситуации дополняется указанием на способ действия) и др.

Для результативного выполнения заданий требуется обучение системы. Будем различать несколько типов обучения, определяемых формой передачи знаний и характером заданий. Прежде всего отметим, что наиболее распространенный в настоящее время способ передачи знаний — через алгоритмические предписания. Но такой способ не следует связывать с обучением. Он требует постоянного участия разработчика, которому приходится выражать все знания в алгоритмической форме, постоянно достраивать, переделывать алгоритмы. Трудозатраты по разработке и сопровождению систем здесь громадные.

Более совершенный способ передачи знаний основан на участии учителя — специалиста в определенной ПО, эксперта. Он передает свой опыт, выраженный в виде обобщающих предложений ЕЯ. Системы, воспринимающие подобную информацию, рассматривались выше. Поступающие на их вход предложения представляются в виде обобщенных знаний, роль которых играли кванторные сети, семантические графы, в том числе, обязательные. Они служили для выполнения заданий. Это уже определенного типа обучение, позволяющее учителю передавать системе знания в удобных для него формах. Трудозатраты по сопровождению подобных систем значительно уменьшаются.

Совершенно другой тип **обучения — по примерам**. Человек передает системе свои конкретные знания, эмпирический материал, указывает на характер заданий. Система сама должна найти зависимости, закономерности, позволяющие ей успешно справляться с заданиями. Предполагается формирование новых знаний. Естественно, они не должны противоречить имеющимся знаниям, быть как можно более простыми и обеспечивать достаточно определенные решения. Более того, система должна быть уверена в своих знаниях. Соответственно они должны быть проверены на достаточно большом количестве примеров.

С точки зрения трудозатрат такая форма передачи знаний является наиболее благоприятной и привычной для человека. При этом следует учитывать, что эмпирический материал — это не обязательно набор объектов с указанием их свойств или значений признаков. Объекты могут быть связаны между собой, иметь части и т. д. Такой материал представляется в конкретных знаниях. В них удается найти большое количество разнообраз-

ных примеров, выявить множество различных зависимостей, закономерностей, которые представляются в обобщенных знаниях. Конечно, их может оказаться чересчур большое количество. Многие из них просто никогда не будут использоваться. Поэтому рационально строить их, исходя из характера задания.

Будем различать здесь два случая. Первый — когда новые знания формируются под текущее задание и имеют целью успешно справиться с ним. Например, такой случай имеет место, когда система не может ответить на запрос. Недостаточно ее знаний. Причем запросы могут быть достаточно произвольными, в том числе непредусмотренными. И второй случай, когда система ориентируется на задания определенного типа, например, требуется лишь распознавание объектов. Система специально готовится к выполнению таких заданий. Формируются необходимые знания, которые должны служить как можно в большем числе случаев.

Перечисленные типы обучения могут в различных вариациях встречаться в привычных нам речевых актах. Образуются промежуточные формы. Например, обобщенные описания могут даваться с той или иной степенью полноты, но подкрепляться примерами. Системы, допускающие такие акты, должны вначале формировать соответствующие знания, а затем на основе примеров проверять их, дополнять.

В ряде случаев обучение может происходить на **нескольких уровнях**. Например, при создании формальных теорий требуется как умение каким-то образом порождать новые теоремы (претенденты на доказательство), так и умение доказывать. Для порождения зачастую используются конкретные знания, имеющийся здесь опыт. Это один уровень обучения. Но чтобы доказывать, нужно иметь знания, определяющие приемы, последовательности используемых правил вывода. Такие знания могут складываться на основе своего опыта. Это уже другой уровень.

И, наконец, ранее рассматривалось обучение, осуществляемое за счет накопления знаний. Однако параллельно с ними должны совершенствоваться и механизмы работы над знаниями, т. е. внутрисистемные «умения». Например, с вводом новых категорий и конструкций должны формироваться средства, отражающие особенности их операционной семантики (см. § 3.1). Должны постоянно совершенствоваться стратегии поиска. Из сказанного следует необходимость специальной организации. С этой целью была введена схема (см. рис. 3.1).

В дальнейшем будем рассматривать лишь обучение, связанное с накоплением, формированием знаний. Почему же здесь оказываются неудовлетворительными существующие подходы? Остановимся на этом вопросе.

Распознавание образов, экспертные системы. В области распознавания образов также ставится задача обучения. Ее суть — в построении на базе обучающей выборки (набора описаний известных объектов) решающих правил. Последние должны обеспечить соотнесение к классам все новых объектов (по их описаниям). Однако в этой области в основном ориентируются на сравнительно простые описания, которыми являются наборы признаков с их значениями. Такие описания могут рассматриваться как точки в координатном пространстве, что делает возможным использование математических средств. На них основаны методы разделяющих граней, потенциальных функций и др. Здесь не требуется, чтобы решающие правила были согласованы с формами ЕЯ, чтобы система могла вводить и использовать привычные нам описания, которые в своем большинстве

носят структурный характер. Предполагается участие специально подготовленных людей, которые формализуют очередное задание, сведут новое описание к типовой форме, интерпретируют системные результаты и обоснуют полученное решение. Все это требует громадных трудозатрат по сопровождению системы.

Ничего существенного не дают и методы, базирующиеся на формальных грамматиках (известные как синтаксические) и логике предикатов (структурные). Основные причины — в сильной ограниченности допустимых конструкций, недостаточной гибкости процедур анализа, в трудностях обобщения на уровне линейных форм. Сказанное относится и к различным видам продукций, в том числе к секвенциям языка PROLOG продукциям с операторами в правой части. Более того, каждый из упомянутых формализмов ориентируется лишь на определенные задания, формы ЕЯ. Чуть последние выходят за рамки допустимого — и без участия человека уже не обойтись.

Опыт показывает, что **чем в меньшей степени учитываются факторы однородного, согласованного представления форм ЕЯ (ориентации на автоматический ввод и обработку), тем большие трудозатраты требуются по доработке и сопровождению систем**. Отсюда становится понятным, почему так и не нашли себя многие, казалось бы, перспективные системы — информационные, логические, общения на ЕЯ. Для них требуется специальный формализм. Конечно, он должен обладать возможностями и предикатов, и продукций, и грамматик, и других средств, источниками которых явились наши естественные представления. И в то же время он должен удовлетворять своим требованиям (см. § 1.1).

Отсутствие адекватного формализма привело к распространенной в настоящем времени тенденции существенного ограничения задачи. Эта тенденция привела к направлению, получившему название экспертных систем. Заведомо ограничивается область, допустимые задания, фиксируются формы сопровождения системы. Например, одна из наиболее известных систем MYCIN ориентирована на диагностику заболеваний, DENDRAL — на задачи масс-спектрометрии и т. д.

Указанное направление в значительной степени стимулирует разработку все новых формализмов, постановку и решение новых задач, в том числе обучения. Например, в системе EURISKO реализовано обучение по аналогии и т. д. Но в то же время при наличии существенных ограничений снижаются требования к используемым формализмам, становится возможным большое разнообразие средств. Это хорошо видно на примере экспертных систем, где находят свое применение различные языки логического программирования, фреймовые конструкции, предикаты и многое другое. Казалось бы, знания могут быть организованы по-разному. Но это не так. Если снижать ограничения, расширять круг задач и заданий, то требования к формализму будут все возрастать, что делает его вид все более определенным. Думается, что здесь есть **единая научная истина,** к которой мы, собственно, и стремимся в настоящей работе.

О сочетании концептуального и вероятностного подходов (в рамках единого формализма представления знаний). Одна из разновидностей задач обучения связана с формированием и использованием статистических закономерностей. Между утверждениями типа *Товар A_1 поставляется (неким поставщиком, не важно каким)* и *Не поставляется* существует множество промежуточных утверждений *Поставляется с вероятностью Р.*

Они также должны представляться в системных знаниях. Это уже нашло свое отражение в ряде экспертных систем, например MYCIN, ADVICE и др.

Дело в том, что в ряде случаев просто не удается найти детерминированные закономерности, например, говорящие об особенностях какого-либо класса объектов. Такие особенности могут быть и у других объектов. Возникает необходимость в вероятностных оценках, знаниях и статистических закономерностях (их частными случаями являются рассмотренные ранее обязательные знания, для которых все вероятности есть 1 или 0). Естественно, система сама должна уметь формировать и использовать такие знания. Для этого необходимо привлечение методов математической статистики и типовых формул пересчета вероятностей, которые должны каким-то образом сочетаться со стандартными процедурами работы над знаниями — поиска, проверки, дополнения. Спрашивается, на какой основе может быть обеспечено такое сочетание? Постараемся ответить на этот вопрос.

Прежде всего следует отметить близость концептуального и вероятностного подходов. Можно провести четкие аналогии между используемыми понятиями. Например, предметной области (ПО) соответствует понятие генеральной совокупности. Восприятие объектов ПО, их изучение — это испытания. Знания о ПО — это выборочная совокупность (если иметь ввиду открытые ПО и считать, что в них воспринимаются кое-какие объекты, взятые «наугад»). Объекты, ситуации (представленные в знаниях), а также значения признаков, факты наличия или отсутствия отношений, различные акции — это исходы или события. Некоторые из них рассматриваются как неделимые — элементарные. Другие же носят составной характер. Это более сложные события. Фактически с помощью различных слов называются одни и те же вещи.

Далее следует отметить родство задач. Формирование статистических закономерностей по выборочной совокупности — это частный случай задачи обучения по примерам. Требования нахождения вероятностей — это определенного сорта запросы. Соответственно решение вероятностных задач сводится к ответу на запросы, дополнению информации.

И в то же время между указанными подходами есть существенные отличия. Главное из них в том, что аппарат теории вероятности предполагает интеллектуальную поддержку со стороны человека, который поймет условие задачи, представит ее в нужном виде, вычислит требуемую вероятность. И как следствие — многое умалчивается. Все, что касается содержательных вещей (к каким событиям относятся вероятности, как связаны события, испытания и др.), дается через ЕЯ. Нам же важно, чтобы процесс ввода условий задач с их решением осуществлялся автоматически. Предполагается выделение семантического эквивалента, согласованное представление текстовых сведений и вероятностных категорий. Все должно быть представлено в явном виде и быть ориентировано на автоматическую обработку. Отсюда вытекают жесткие требования к используемому формализму — его свойство однородности, согласованности при высоких возможностях представления структурной информации (см. § 1.1). Здесь не годятся линейные формы записи (из-за их высокой синтаксической загруженности). Между тем на таких формах основана вся теория вероятности.

Еще одно отличие — в характере моделей. Вероятностные методики ориентируются на выборочные совокупности, которые, как правило, имеют вид пар действие — результат или объект — набор признаков. Этого недостаточно в базах знаний, которые должны служить для представления

и обработки структурной информации. Объекты, действия могут быть связаны между собой достаточно сложным способом. Здесь делается условным и понятие признака, и понятие объекта, и понятие испытания. Любое свойство может рассматриваться как объект, а любая цепочка отношений — как признак.

Несмотря на отличия, следует отметить необходимость сочетания подходов, взаимного проникновения областей. Это основной путь к построению новых классов **систем** и прежде всего **принятия решений.** Ясно, что такие системы должны базироваться на знаниях, которые в перспективе должны обеспечивать решения и в условиях неопределенности. Здесь следует учитывать, что информация, касающаяся исходной ситуации и способов принятия решений, может поступать по многим каналам. Это могут быть тексты ЕЯ, информация с выходов различных устройств (обнаружения сигналов, измерения и др.). Здесь могут быть и структурные компоненты, и вероятностные, и ошибки измерения, и нечеткие компоненты Все они должны каким-то образом суммироваться на общей основе, где должна происходить постоянная работа по устранению разного рода неопределенностей, неточностей, а если такое устранение невозможно, то происходит их оценка или переоценка, Проводимые нами исследования показывают, что такой основой может служить язык семантических сетей и графов (СЯ). В принципе он может быть использован и для работы с дикретными случайными величинами, на чем мы остановимся чуть ниже.

Особо хочется остановиться на возможностях СЯ в плане работы с нечеткими категориями. Для учета их относительного характера предложена специальная методика, основанная на преобразовании представлений (см. [14]). Например, возьмем высказывание *Петр молодой.* Его содержание во многом зависит от возраста автора (A_1) высказывания. В связи с этим рационально его преобразование к виду *Петр моложе сверстников* A_1. *Очень молодой* преобразуется в *моложе большинства сверстников* A_1 и т. д. В данном случае стандартизируются лишь понятия *больше, значительно больше,* ... При этом требуются представления об A_1, его сверстниках. На этой основе могут формироваться функции принадлежности, обеспечивается ответ на различные запросы (с нечеткими категориями), обоснованное принятие решений.

Выявление зависимостей, формирование закономерностей. Любая зависимость предлагает наличие двух компонентов, связанных между собой. Первая компонента указывает, что зависит, а вторая — от чего зависит. Например, погода может зависеть от времени года (в одной местности). Это некая форма, в которой фиксируется связь компонент, но не уточняется, как зависит. Форма может быть заполнена содержанием, т. е. могут быть найдены конкретные примеры, привязанные ко времени, местности и указывающие, какой была погода в тех или иных случаях — ветреной, морозной и т. д. Примеры выбираются на основе конкретных знаний. Они позволяют уточнить характер самой зависимости. На их основе могут быть сформированы новые знания, представляющие утверждения (локальные закономерности) типа *В Хабаровске в первой половине января погода всегда ветреная и морозная.*

Итак, формирование новых знаний предполагает порождение соответствующих форм, их заполнение с последующим анализом содержимого. При этом порождение должно определяться текущим заданием. Например,

если задание имеет вид запроса, то форма должна строиться на базе компонентов запроса. Тогда есть гарантия, что формируемые знания позволят получить ответ. Если задание связано с распознаванием очередного объекта, то формы должны строиться по его описанию. Если система специально готовится к такому распознаванию, то для построения форм берутся описания известных объектов и т. д.

На базе какого-либо запроса или описания может быть получено множество форм. Здесь возникает задача выбора среди них наиболее удачной, т. е. приводящей к формированию знаний, позволяющих как можно более успешно справляться с текущим заданием. Например, если вопрос требует ответа типа *да, нет*, то форма должна приводить к формированию как можно более определенных знаний.

Для выбора форм будем использовать **метод последовательных приближений**. Вначале строится наиболее простая из них. Если форма оказалась неудачной, то анализируются причины, которые вызывают действия детализации (добавления новых компонент, которые берутся из запроса, описания), переориентации (берутся новые компоненты взамен старых) и обобщения (не принимаются во внимание некоторые детали, факторы). Таким образом организуется сходящийся процесс. При этом перечисленные действия затрагивают содержательную часть, т. е. компонентами являются свойства, отношения, о которых идет речь в запросе (описании), а деталями — сами объекты, их типы. Отсюда необходимость реализации процесса на уровне семантических структур (СС), а не слов. Остановимся на принципе такой реализации.

Для представления форм будем использовать семантические графы. Наиболее типовой является конструкция вида

$$[y_2 \perp T_2\,(y_2,\ y_1^{\dagger})] \circ [y_1^{\dagger} \perp T_1\,(y_1^{\dagger})], \qquad (7.13)$$

где представляется зависимость компоненты Y_2 от Y_1. Сеть $T_2\,(y_2,\ y_1^{\dagger})$ представляет отношения между Y_1 и Y_2, а $T_1\,(y_1^{\dagger})$ — ограничения на компоненту Y_1, накладываемые условия.

Например, граф (7.13) может быть использован для представления *зависимости типа товара от поставщика.* При этом сеть $T_1\,(y_1^{\dagger})$ будет представлять Y_1 *является поставщиком*, а также имеющиеся ограничения на множество поставщиков. Сеть $T_2\,(y_2,\ y_1^{\dagger})$ будет представлять Y_1 *поставляет товар типа* Y_2.

Возможны и другие конструкции. Так, граф

$$[y_2 \perp (y_1^{\dagger},\ y_2) \perp T_2\,(y_2,\ y_1^{\dagger})] \qquad (7.14)$$

представляет зависимость, в которой ограничения по компоненту накладываются за счет связи $\mathcal{T}_2$ компонет. Например, может быть представлена *зависимость поставляемого товара от того, кто его поставляет.* Это может быть человек, завод т. д.

Для представления зависимостей между множествами компонент используются эти же конструкции (7.13) и (7.14). Только в них на местах $y_1^{\dagger}$ и y_2 будут стоять наборы n-вершин. Например, качество товара (Y_{11}) может рассматриваться в паре с его количеством (Y_{12}) и зависеть от множества факторов Y_{21}, ..., Y_{2n}. Для представления такой зависимости используется граф (7.13), в котором на месте $y_1^{\dagger}$ будет стоять $(y_1^{\dagger},\ y_{12}^{\dagger})$, а на месте y_2 — n-ка вида $(y_{21},\ ...,\ y_{2n})$. Сети же T_1 и T_2 будут заменены на соответствующие фрагменты.

При последующем изложении будем в основном ориентироваться на конструкцию (7.13), которую легко распространить на другие случаи.

Заполнение форм сводится к нахождению значений *н*-вершин из (7.13). Граф достраивается до полного, см. § 4.1. Задаваемые им операции выполняются над сетью T, представляющей конкретный материал (знания). Напомним, что граф указанного вида будет задавать операции нахождения по T_1 множества значений y_1^1, их перебора с нахождением значений y_2. Пусть оказалось $y_1^1 := \{b_1, ..., b_k\}$. Путем перебора формируются согласованные значения пары

$$(y_1^1, y_2) := \{(b_1, \{a_{i1}\}), (b_2, \{a_{i2}\}), ..., (b_k, \{a_{ik}\})\}, \qquad (7.15)$$

где на местах $\{a_{ij}\}$ для $j = \overline{1,k}$ могут быть и пустые множества $\varnothing$. Данная *з*-сеть означает, например, что *поставщик* B_1 *поставляет товары* $\{A_{i1}\}$, *поставщик* B_2 — $\{A_{i2}\}$ и т. д.

Формирование гипотез, выявление закономерностей сводится к анализу полученных пар, т. е. *з*-сети (7.15). Делается попытка выполнения над множествами $\{a_{ij}\}$ различных операций. Если попытка оказалась удачной, то формируется граф, представляющий г и п о т е з у (претендент на роль знаний). После соответствующих оценок граф может быть перенесен в знания. Тогда он уже будет представлять уверенное высказывание, закономерность. Вид графа и характер закономерности будет определяться операциями. Под них как бы подбирается граф (он будет задавать эти операции). При этом осуществляется акция обобщения — объекты $\{B_1, ..., B_k\}$ перестают различаться. Им сопоставляется *н*-вершина.

В последующей части параграфа будут рассматриваться операции, которые приводят к формированию утверждений (графов, сетей) с оттенком всеобщности, существования, а также возможности или вероятности.

Вначале отметим, что прежде чем перенести граф в системные знания, должны осуществляться **специальные оценки**, в частности на п р о т и в о р е ч и в о с т ь. Далее должна вычисляться с т е п е н ь д о в е р и я к графам, т. е. степень подтверждаемости соответствующих утверждений. Здесь большую роль играет количество элементов $\{b_1, ..., b_k\}$. Образованные ими пары фактически представляют опытный материал, те примеры, которые были подобраны в соответствии с конструкцией (7.13). Естественно, таких элементов должно быть как можно больше. Если граф перенесен в знания, то он может использоваться для выполнения заданий.

Конструкции с оттенками всеобщности (обязательности), существования. К формированию знаний (графов, сетей), представляющих **общие свойства** объектов, приводят операции поиска непустого пересечения $\bigcap \{a_{ij}\}$, где $j = \overline{1,k}$. Пусть множества $\{a_{ij}\}$ содержат общий элемент — *о*-вершину a_t. Тогда формируется фрагмент $[\square_{\phi} y_2 := a_t]$, который может быть добавлен к графу (7.13). Получится обязательный граф, в котором y_1^1 — *н*-вершина (у нее не будет значений). Например, такой граф может представлять, что *поставщик* Y_1 *обязательно поставляет товар типа* A_t. Указанный граф, перенесенный в обязательные знания, может служить для дополнения и уточнения входной информации, т. е. играть роль решающего правила. Когда речь идет о поставщиках, удовлетворяющих условию $\mathcal{T}_1$, то всегда может быть добавлено, что они поставляют товар типа A_t. Это обеспечивается выполнением операций, задаваемых графом, над соответствующей входной сетью (см. § 5.1).

Полученный обязательный граф может быть преобразован в кванторную сеть вида

$$T_1(y_1) \circ \langle \_, t, \forall, y_1, y_3 \rangle \circ T_2(a_t, y_3), \qquad (7.16)$$

представляющую, что *все объекты со свойствами* $\mathcal{T}_1$ *связаны отношениями* $\mathcal{T}_2$ *с* A_t. Например, таким способом может быть представлено, что *все поставщики, обладающие свойствами* $\mathcal{T}_1$, *поставляют товар типа* A_t. Подобная сеть может быть перенесена в пассивные знания. Тогда она будет материалом, на котором осуществляется поиск. Повышается результативность выполнения соответствующих заданий, в частности, имеющих вид запросов о поставляемых товарах. Более того, на базе кванторной сети могут формироваться различные обязательные графы (см. § 7.3), т. е. сети могут быть основой порождения новых знаний, играющих роль решающих правил.

Пусть множества $\{a_{ij}\}$ содержат несколько общих элементов, которые обозначим через $\{a_t\}$. Тогда формируется фрагмент $[\square_{\phi} y_2 := \{a_t\}]$, который добавляется к (7.13). Будет представлена обязательность любого элемента из $\{A_t\}$. Все сказанное ранее остается в силе. При этом может быть сделана попытка найти сеть T_3, представляющую особенности только элементов $\{A_t\}$. Если это удается, то формируются фрагменты $[y_2 \perp T_3(y_2)] \circ [\square_{\phi} y_2 := \lambda]$, которые добавляются к (7.13). Первый из этих фрагментов есть граф, соответствующий множеству $\{A_t\}$. Например, он может представлять *товары, которые изготовлены на заводе* N_1 ($\mathcal{T}_3$).

Следует заметить, что кванторные сети можно формировать несколько **другим способом** — за счет конструкций $[y_1 \perp T_1(y_1)]$ и $[y_3 \perp T_2(a_t, y_3)$. По ним находятся множества значений $[y_1 := \eta]$ и $[y_3 := \gamma]$. *О*-вершина a_t подбирается таким образом, чтобы имело место включение $\eta \subset v$. В результате формируется сеть (7.16), на основе которой может быть построен обязательный граф. Итак, к формированию конструкций с оттенками всеобщности, обязательности, приводят как операции перебора с поиском общих элементов, так и операции проверки теоретикомножественного включения ($\subset$).

Пусть множества $\{a_{ij}\}$ не содержат общих элементов. Тогда анализируется, есть ли среди этих множеств пустые. Если нет, то может быть сформирован фрагмент $[\square_{\phi} y_2 := \lambda]$, который добавляется к графу (7.13). В результате будет представлено, например, что *(любой) поставщик* Y_1 *обязательно поставляет какой-либо товар*. Если, наоборот, все множества оказались пустыми, то может быть сформировано $[\square_{\phi} y_2 := \tilde{\lambda}]$. В результате будет представлено, что *никакой поставщик не поставляет никаких товаров*. Конечно, подобные графы не смогут обеспечить выполнение заданий, требующих конкретных решений. В основном такие графы служат для целей проверки, т. е. в качестве семантических фильтров.

К формированию конструкций, представляющих **свойства** $\mathcal{T}_2$ **некоторых объектов** Y_1, приводят операции выделения из множеств $\{a_{ij}\}$ каких-либо элементов (a_t). В результате строится сеть вида (7.16), в которой вместо $\forall$ стоит $\exists$. Будет представлено, например, что *некоторые поставщики поставляют товар* A_t. К такой же сети приводят операции проверки непустого пересечения (вместо проверки $\subset$).

Все сказанное ранее нетрудно распространить на случай наборов, т. е. когда представляются зависимости между множествами компонент. Следует также допускать композиции конструкций, возможность рассмотре-

ния сети T_2 как конструкции вида (7.13). Таким способом могут формироваться более сложные кванторные сети, например, представляющие *Все поставщики поставляют товары, обладающие знаком качества*, а также *Каждый поставщик поставляет товары* $\{A_i\}$, *некоторые из них обладают знаком качества* и т. д. Здесь может быть много различных смысловых оттенков. Как показано в [14], все они укладываются в рамки введенных нами средств.

Вероятностные графы, статистические закономерности. К их формированию приводят операции подсчета относительного числа пар $(b_j, \{a_{ij}\})$, где $j = \overline{1,k}$, в правой части которых встречается o-вершина a_s из множества $\bigcup \{a_{ij}\}$. Такой подсчет осуществляется для всех элементов указанного множества. Пусть o-вершина a_s встречалась в m парах. Тогда относительная частота будет равна m/k. При $k \to \infty$ такая частота будет стремиться к вероятности (P_s) наличия объекта A_s (или исхода A_s).

Введем необходимые средства представления вероятностных категорий. Речь будет идти лишь о дискретных событиях. Элементарным событиям или исходам будем сопоставлять o-вершины. Более сложным событиям, составленным из элементарных (путем их перечисления), будем сопоставлять вершины типа y с соответствующими значениями. Если же событие определяется условием, то ему сопоставляется граф: его фокус будет соответствовать самому событию, а область — условию.

Например, пусть o-вершины a_1, a_2, a_3 соответствуют *деталям* (т. е. элементарным событиям, связанным с их появлением, наличием). Тогда вершина $y^1_1 := \{a_1, a_2, a_3\}$ будет соответствовать группе событий — *появлению одной детали из указанного множества*. Граф $[\Gamma_1 \,|\textit{детали}| \perp T_1(y_1)]$ будет соответствовать *детали, обладающей свойством* $\mathcal{T}_1$. Это тоже событие. Причем относительно них могут подсчитываться вероятности других событий, например, если последние связываются с фактом появления упомянутых деталей. Заметим, что события, относительно которых подсчитываются вероятности, часто связываются с испытаниями. В дальнейшем будем пользоваться этим термином.

Самим вероятностным оценкам будем сопоставлять o-вершины p_1, p_2, ... (не следует путать их с p-вершинами, см. § 1.4), а также o-вершины '0,1'; '0,2'; ..., соответствующие числам в интервале 0,1.

Для представления *вероятности* $P(Y_2)$ *некоторые события* Y_2 будем использовать фрагмент вида

$$\langle \_\_,\ t,\ \textit{вер},\ y^1_1,\ y_2,\ \rho \rangle, \tag{7.17}$$

где *вер* — быть вероятностью; y_2 соответствует исходам; ρ — вероятностной оценке (это $н$-вершина типа y^1). H-вершина y^1_1 соответствует испытаниям. Относительно них подсчитывается вероятность. Будем называть такую вершину и множество ее значений опорными.

Для представления заданных вероятностей фиксированных событий (из одной группы) A_1, ..., A_s будем использовать сети типа

$$\langle \_\_,\ t,\ \textit{вер},\ y^1_1,\ y_2,\ \rho \rangle \circ [(y_2, \rho)^1 := \{(a_1, \text{'0,1'}),\ ...,\ (a_s, \text{'0,2'})\}]. \tag{7.18}$$

Данная сеть соответствует записи $P(A_1) = 0{,}1$; ...; $P(A_s) = 0{,}2$. На местах '0,1'; ...; '0,2' могут стоять вершины p_1, ..., p_s, соответствующие обозначениям вероятностей.

Будем называть фрагмент (7.17) и сети (7.18) вероятностными. Они будут служить в композиции с графом (7.13) или (7.14) для представления статистических закономерностей (о дискретных величинах, в том числе случайных). Подобные композиции будем называть вероятностными графами. Примером одного из них является

$$[y_2 \perp \langle y^1_1,\ t,\ \textit{бр. мн},\ y_2 \rangle] \circ [\square_{\phi}\ (y_2, \rho)^1 := \{(\text{'}\textit{орел}\text{'}, \text{'0,5'}),\ (\text{'}\textit{решка}\text{'}, \text{'0,5'})\}], \tag{7.19}$$

где представлено, что *при бросании монеты (бр. мн.) имеет место (или должно быть) два исхода: с вероятностью 0,5 будет орел и с такой же вероятностью — решка*. H-вершина y^1_1 соответствует акции бросания, т. е. испытаниям.

Вероятностный граф наиболее типового вида:

$$[y_2 \perp T_2(y_2, y^1_1)] \circ [y^1_1 \perp T_1(y^1_1)] \circ \langle \_\_,\ t,\ \textit{вер},\ y^1_1,\ y_2,\ \rho \rangle \circ$$
$$\circ\ [\square_{\phi}\ (y_2, \rho)^1 := \{(a_1, p_1), ..., (a_s, p_s)\}]. \tag{7.20}$$

Например, его частным случаем является

$$[y_2 \,|\textit{товар}| \perp \langle \_\_,\ t,\ \textit{пост},\ y_2,\ y^1_1 \,|\textit{поставщик}| \rangle] \circ \langle \_\_,\ t,\ \textit{вер},\ y^1_1,\ y_2,\ \rho \rangle \circ$$
$$\circ\ [\square_{\phi}\ (y_2, \rho)^1 := \{(a_1, p_1), ..., (a_s, p_s)\}], \tag{7.21}$$

где представлено, что *поставщик* Y_1 *с вероятностью* P_1 *поставляет (пост) товар* A_1, ..., *с вероятностью* P_s — *товар* A_s. Здесь вероятности оценивались относительно множества поставщиков, т. е. факт наличия одного из них был испытанием.

Нетрудно видеть, что в графах (7.20) и (7.21) все представлено в явном виде: связь $\mathcal{T}_2$ между событиями Y_2 и испытаниями Y_1, ограничения $\mathcal{T}_1$ на Y_1. Вероятности как бы связывают Y_1 с конкретными проявлениями Y_2, т. е. исходами A_1, ..., A_s. Подобное представление необходимо для автоматической обработки. В общепринятой же записи — $P(A_1)$, ..., $P(A_s)$ — указанные связи должны оговариваться особо. Кто-то постоянно должен держать их в голове.

В общем случае будем допускать несколько опорных вершин, входящих во фрагменты (7.17), сети (7.18) и графы (7.20). Таким способом представляются зависимости от множества факторов. Причем некоторые из них могут быть зафиксированы. Тогда будет иметь место так называемая условная вероятность (имеет место случай, когда условия и исходы взяты из разных групп событий). Например, пусть вероятность того, что поставляется товар Y_2, вычисляется относительно пары *Поставщик* Y_1 — *город поставки* Y_3, которая представляется с помощью $\langle \_\_,\ t,\ \textit{пост}^*,\ y_2,\ y^1_1 \,|\textit{поставщик}|,\ y^1_3 \,|\textit{город}| \rangle$. Тогда, сделав $\langle \_\_,\ t,\ \textit{вер},\ y^1_1,\ y^1_3,\ y_2 \rangle \circ$ $\circ\ [y^1_3 := e_k]$, получим граф, представляющий: *Поставщик поставляет в город* E_k *товар, которым с вероятностью* P_1 *является* A_1, *с вероятностью* P_2 — A_2 *и т. д.* Такие вероятности обычно записываются в виде $P(A_1 \mid E_k)$, $P(A_2 \mid E_k)$, ... Графы, представляющие условные вероятности событий, отличаются от других графов тем, что некоторым $н$-вершинам присвоены конкретные значения. Ясно, что вероятностные оценки будут зависеть от таких присвоений.

Статистические закономерности, в которых исходами являются неэлементарные события, представляются с помощью той же конструкции (7.20). Только в ней на местах a_1, ..., a_s будут стоять вершины типа y^1. Они могут быть вершинами с заданными значениями или фокусами каких-либо графов.

Например, если в графе (7.21) поставить на место a_1 вершину y^1_3, которая $y^1_3 := \{a_{s+1}, ..., a_m\}$, то уже будет представлено, что *поставщик с вероятностью* P_1 *поставляет какой-либо товар из перечня* $\{A_{s+1}, ..., A_m\}$. Если на место a_1 поставить вершину y^1_3, которая $[y^1_3 \perp T_3(y^1_3)]$, то будет представлено, что *с вероятностью* P_1 *поставляется товар, обладающий свойством* $\mathcal{T}_3$.

Формирование вероятностных графов. Рассмотрим принципы такого формирования, которое осуществляется на базе конструкций вида (7.13), (7.14). Для этого используется «опытный» материал, т. е. пары $(b_j, \{a_{ij}\})$, где $j = \overline{1,k}$. Подсчитывается относительное число пар, в правой части которых встречается o-вершина a_1. Такой подсчет осуществляется для всех вершин $\{a_1, ..., a_s\} = U\{a_{ij}\}$. В результате находятся вероятности $P_1, ..., P_s$, которым сопоставляется $(a_1, p_1), ..., (a_s, p_s)$. Из них формируется $з$-сеть. Таким способом на базе конструкции (7.13) будет сформирован граф (7.20). При этом элементы множества $\{b_1, ..., b_k\}$, которое определяло подсчет, как бы перестают различаться. Им сопоставляется $н$-вершина y^1_1. Например, становится неважным, какие поставщики поставляют товары. Главное, что они есть.

Отметим, что в зависимости от вида исходной конструкции будут формироваться графы, представляющие те или иные статистические закономерности. Ранее рассматривалось, когда подсчет вероятностей шел относительно поставщиков, а исходами являлись товары. В других случаях подсчет может вестись относительно множества товаров, относительно городов поставки и т. д. Опорное множество может быть достаточно произвольным. Оно определяется лишь условиями $\mathcal{T}_1$, т. е. сетью $T_1(y^1_1)$. Далее, исходами могут быть оценки количества и качества товаров, их какие-либо свойства. Такие исходы определяются отношениями $\mathcal{T}_2$, т. е. сетью $T_2(y_2, y^1_1)$. Соответствующим образом будет меняться характер закономерностей. Например, будут различаться *Вероятность того, что поставщик поставляет товар высокого качества* и *Вероятность того, что товар, поставленный поставщиком, имеет высокое качество.* В первом случае вероятность подсчитывается относительно множества поставщиков, а во втором — относительно поставляемых товаров.

В процессе формирования вероятностных графов будут выявляться разного рода группы событий, что сводится к нахождению значений $н$-вершин на базе конкретных знаний T. При этом, если в T представлены не все объекты ПО (или ПО является открытой), то никогда не будет уверенности, что группа является полной. В дальнейшем это понятие будет рассматриваться относительно текущих знаний T. Сказанное касается и группы исходов $\{A_1, ..., A_s\}$, которая в общем случае может быть дополнена пустым исходом. Например, в множество $\{(a_1, p_1), ..., (a_s, p_s)\}$ из (7.21) может быть введена соответствующая пара $(\tilde{\lambda}, p_0)$. Тогда дополнительно будет представлено, что *с вероятностью* P_0 *поставщики не поставляют никакого товара.*

При построении графов могут осуществляться специальные проверки — на несовместность исходов, независимость событий. В случае их выполнения к графу (7.20) добавляются новые фрагменты. Для проверки несовместности исходов $\{A_1, ..., A_s\}$ используются граф (7.13) с $з$-сетью $[y^1_2 := \{a_1, ..., a_s\}]$, где $\square 1$ — означает *только один.* Если проверка удовлетворилась, то в (7.20) вместо $\square_ф (y_2, \rho)^1$ будет поставлено $\square_ф (y_2, \rho)^{\square 1}$.

Если такое изменение произвести в графе (7.21), то дополнительно будет представлено *(Любой) поставщик поставляет только один из товаров перечня* $\{A_1, ..., A_s\}$.

Проверка на независимость событий сводится к оценке изменения величин $P_1, ..., P_s$, представленных в (7.20), после того, как какой-либо вершине присвоено значение (e_k). Таким способом проверяется $P(A_s) = P(A_s \mid E_k)$, где A_s и E_k — события из разных групп. Можно показать, что и в других случаях для упомянутых проверок достаточно введенных ранее средств («механизмов»).

Использование вероятностных графов. Они будут служить для выполнения текущих заданий. Будем различать два основных случая. Первый — когда задание требует вероятностных оценок, т. е. нужно подсчитать вероятность какого-либо события. И второй случай — когда задание требует сугубо определенного решения, например, требуется указать конкретный объект, ответить *да* или *нет* и т. д. Остановимся на **первом случае.**

Пусть $\{\Gamma_t\}$ — множество вероятностных графов, а $T_{вх}$ — сеть, представляющая входное задание. Требуется дополнение $T_{вх}$ фрагментами, представляющими вероятностные оценки. Пусть среди $\{\Gamma_t\}$ есть граф, повторяющий структуру сети $T_{вх}$. Тогда дополнение обеспечивается за счет операций (в том числе формирования), задаваемых этим графом. Например, пусть в $T_{вх}$ представлены конкретные сведения о неком человеке — поставщике. Тогда путем выполнения над $T_{вх}$ операций, задаваемых графом (7.21), получим $T_{вх}$, в которой дополнительно будет представлено *С вероятностью* P_1 *этот человек поставляет товар* A_1, *с вероятностью* $P_2 - A_2$, и т. д.

Пусть среди $\{\Gamma_t\}$ нет графов, повторяющих структуру сети $T_{вх}$, где представлено условие задачи. Тогда требуется сформировать такой граф. Как указывалось ранее, это может быть сделано за счет опытного (статистического) материала.

Другой способ основан на методиках решения задач теории вероятности. Такие методики реализуются в рамках введенных ранее средств. Например, пусть в $T_{вх}$ представлено *Монета брошена два раза, требуется найти вероятность того, что оба раза оказался орел.* А в $\{\Gamma_t\}$ имеется лишь граф (7.19). Тогда на основе него должен быть сформирован другой вероятностный граф, что должно осуществляться в соответствии с условием задачи.

Для формирования одних вероятностных графов на основе других вводятся специальные композиции, которые соответствуют связям между событиями (операциям над ними). Например, пусть $\Gamma_v(y^1_v)$ и $\Gamma_v(y^1_v) \circ \Gamma_v(y^1_v)$ будет соответствовать произведению этих событий, а $\Gamma_v(y^1_v) \circ \Gamma_w(y^1_w) \circ [y_k := \{y^1_v, y^1_w\}]$ — их сумме. Применительно к ним (по формулам умножения, сложения вероятностей) пересчитываются вероятностные оценки, которые представляются с помощью $з$-сетей. В результате получаются новые графы.

Аналогичным образом (с помощью своих композиций) представляются повторяющиеся события, их цепочки и т. д. Для пересчета вероятностей берутся соответствующие формулы. Важно отметить, что во всех композициях графы объединяются в конструкцию с общими опорными вершинами и вершинами — исходами. Подобные композиции вместе с формулами пересчета вероятностей были названы исчислением вероятностных графов. С точки зрения теории вероятности ничего нового здесь нет —

все делается по типовым методикам. Только идет ориентация на внутрисистемную обработку, где сами композиции подбираются под задание.

На основе вероятностных графов (как и обычных обязательных графов) могут формироваться кванторные сети вида (7.16), которые могут служить для ответа на запросы. Только такие сети будут представлять процентные отношения, удельное число случаев. Соответственно, на месте $\forall$ будут стоять свои вершины, которые вместе с окрестностью (представляющей проценты) будем обозначать через $\exists_{p1}$. Например, на основе графа (7.20) может быть сформирована сеть

$$T_1(y_1) \circ \langle \_\_,\ t,\ \exists_{p1},\ y_1,\ y_3 \rangle \circ T_2(y_2,\ y_3) \circ [y_2 : = a_1], \qquad (7.22)$$

представляющая, что $P_1 \circ 100\%$ *объектов* Y_1, *обладающих свойством* $\mathcal{T}_1$, *связано отношением* $\mathcal{T}_2$ *с* A_1. Ясно, что при $P_1 = 1$ вершина $\exists_{p1}$ вырождается в $\forall$. Сети вида (7.22) могут формироваться на основе любой пары $(a_\xi,\ p_\xi)$ графа (7.20). При этом возможен еще один способ их формирования — за счет конструкций $[y_1 \perp T_1(y_1)]$ и $[y_3 \perp T_2(a_\xi,\ y_3)]$, по которым находятся значения $[y_1 : = \eta]$ и $[y_3 : = \nu]$. Далее подсчитывается количество элементов в $\eta \cap \nu$ и η (обозначим такие количества через $|\eta \cap \nu|$ и $|\eta|\rangle$. Первое делится на второе. В результате будет вычислена величина $P_1 = |\eta \cap \nu| \ / \ |\eta|$.

Остановимся на **втором случае** — когда задание требует сугубо определенного решения. Тогда ответ уже не может носить характер альтернатив (исходов) с вероятностными оценками. Для получения определенных ответов необходимы детерминированные знания, т. е. обязательные графы и сети вида (7.16). Как уже говорилось, для их формирования требуется организация с х о д я щ е г о с я процесса. Вначале (на основе компонент задания) порождается форма. Затем она заполняется и формируется статистическая закономерность. Последняя оценивается по степени определенности, в результате чего принимается решение относительно последующих действий. Например, могут инициироваться действия изменения формы (предыдущая квалифицируется как неудачная) или же вырабатывается ответ.

В таком процессе вероятностные графы вида (7.20) будут играть роль промежуточного звена, т. е. продукта, который оценивается. Такая оценка осуществляется на основе вектора $(P_1, ..., P_3)$. Здесь возможно множество различных случаев. Наиболее благоприятный из них, когда все вероятности $P_1, ..., P_3$ оказались близкими по величине и меньше некоторого н и ж н е г о порога l_1. Имеют место маловероятные события $\{A_1, ..., A_s\}$. Тогда требуется такое изменение формы, при котором близость будет нарушена. Для несовместных событий нужно стремиться к тому, чтобы одна из вероятностей (P_ξ) значительно превышала все остальные и была бы больше в е р х н е г о порога l_2. В лучшем случае должно быть $P_\xi = 1$, а все остальные вероятности равны 0. Если имеют место совместные события (и задание предполагает множество решений), то несколько вероятностей могут стремиться к 1, а остальные — к 0.

В общем случае для выработки решения может быть использован метод максимального правдоподобия (его разновидность). Вводится еще один порог l_0. Относительно него оценивается величина $P_\xi \ / \ P_\tau$, где P_ξ — максимальная вероятность $(P_\xi \geqslant l_2)$, а P_τ — следующая за ней $(P_\tau \leqslant l_1)$. Если $P_\xi \ / \ p_\tau > l_0$, то с некоторой долей уверенности формируются соответствующие знания и выдается ответ A_ξ. Подобный метод может быть расширен на случай, когда допускается множество решений.

Обучение при ответе на запросы. Пусть на вход системы поступил запрос, на который она не может дать ответа. Недостаточно ее знаний. Тогда возникает необходимость в новых знаниях. Запрос служит основой для порождения форм, в рамках которых строятся гипотезы, ищутся закономерности, формируются недостающие знания.

Например, пусть на вход системы поступил запрос *Смертен ли Сократ (человек с именем Сократ)*?, а в знаниях представлен материал типа *Петров смертен, Сидоров смертен и т. д.* Но о Сократе ничего не известно. Тогда естественно попытаться отвлечься от некоторых компонентов запроса, свести его к форме *А вообще, человек смертен или нет*? На основе полученного ответа может быть сформировано новое утверждение (гипотеза) *Человек обязательно смертен* или *Все люди смертны.*

В приведенном примере сформированное утверждение касалось общих свойств класса субъектов. В других случаях могут формироваться знания, представляющие общие свойства или особенности различных множеств объектов. Более того, такие множества могут формироваться по мере необходимости. Важно только, чтобы они включали в себя элемент (объект, значение признака и др.), о котором идет речь в запросе и на который падает подозрение, что недостаток знаний о нем послужил причиной пустого ответа. Собственно упомянутые множества и берутся в качестве первой компоненты зависимости. А вторая компонента — это запрашиваемый элемент.

В данном случае рационально использование метода последовательных приближений, который реализуется следующим образом. Пусть $T_{вх}(x_i^1) \circ [x_i^1 : = ?]$ — сеть, представляющая запрос. Например, может требоваться *указать деталь* (X_i), *которую поставляет поставщик, являющийся компаньоном Петра и проживающий в городе* N_1. Пусть поиск ответа ни к чему не привел. Было получено пустое множество значений. Тогда делается попытка породить на базе $T_{вх}$ конструкцию (граф) вида (7.20), достаточную для формирования недостающих знаний. Остановимся на принципах такого порождения.

В $T_{вх}$ выделяется фрагмент, содержащий x_i^1 и еще какую-либо $н$-вершину x_j (желательно, чтобы это был фрагмент, на который падает подозрение как на источник пустого ответа). H-вершина x_i^1 заменяется на y_2, а x_j — на y_1^1. Сам фрагмент начинает играть роль $T_2(y_2,\ y_1^1)$. Далее выделяется один или несколько фрагментов из $T_{вх}$, связанных с x_j. В них также осуществляются указанные замены (все $н$-вершины перекодируются с x на y). Сами фрагменты начинают играть роль $T_1(y_1^1)$. Получается граф, представляющий очередную форму, например *зависимость деталей* Y_2 *от поставщиков* Y_1 *из города* N_1.

На базе графа находятся значения (7.15) и формируется вероятностный граф, который будет представлять исходы. Например, такой граф может представлять *вероятности* $P_1, ..., P_s$ *того, что поставщик из города* N_1 *поставляет детали* $A_1, ..., A_s$. Далее оценивается вектор $(P_1, ..., P_s)$ вероятностей исходов. В результате инициируются действия детализации, переориентации, обобщения (одно из них). Остановимся на них более подробно.

Пусть в векторе $(P_1, ..., P_s)$ все вероятности оказались невысокими. Например, *мало кто из поставщиков* $\{B_1, ..., B_k\}$ *города* N_1 *поставляет деталь* A_1. *То же самое относится и к деталям* $A_2, A_3, ..., A_s$. Тогда разумно попытаться сузить множество $\{B_1, ..., B_k\}$, выделив из них поставщиков

какой-либо одной детали. Это осуществляется путем детализации — добавления к $T_1(y_1^1)$ новых фрагментов, которые берутся из $T_{вх}$. Например, может быть взят фрагмент, представляющий *иметь компаньона*. В результате ограничивается множество $y_1^1 := \{b_1, ..., b_k\}$, где будут представлены лишь *поставщики, имеющие компаньонов*. Снова подсчитываются вероятности. Таким способом ищутся пути (последовательности вовлекаемых фрагментов), при которых начинает возрастать какая-либо одна вероятность (в случае совместных событий их может быть несколько). Так, если добавление к $T_1(y_1^1)$ одного из фрагментов привело к возрастанию вероятности P_ξ, то делаются попытки вовлечь фрагменты, связанные с n-вершинами предыдущего фрагмента. Другими словами, вовлекаются цепочки фрагментов. Например, она может представлять *иметь компаньоном человека по имени Петр*.

Если выбранный путь не привел к нужным результатам, то осуществляется переориентация. Взамен одних фрагментов берутся другие, которые добавляются к $T_1(y_1^1)$. Каждый раз выявляется, какие изменились вероятности и в какую сторону, что определяет последующие действия. Таким образом осуществляется подслеживание, позволяющее направить процесс в нужное русло.

Действие обобщения заключается в заменах o-вершин на вершины с множеством значений (куда входит данная o-вершина), а также на n-вершины. В результате как бы в меньшей степени учитываются некоторые детали — они заменяются на перечни или не принимаются во внимание вовсе. Например, таким путем может быть получена сеть $T_1(y_1^1)$, соответствующая *поставщикам, которые проживают в каком-либо городе* (сам город не уточняется). Далее допускаются замены вершин-классов на вершины-суперклассы. В результате утверждение *Все люди смертны* может быть расширено до *Все животные смертны*. Действие обобщения необходимо не только, чтобы сделать утверждение более сильным. Во многих случаях оно требуется для расширения множества $\{b_1, ..., b_k\}$, от которого зависит степень доверия к формируемым знаниям.

Ранее рассматривался случай, когда на базе запроса порождались зависимости, в которых Y_1 и Y_2 связаны одним отношением, заранее фиксированным. Соответственно роль $T_2(y_2, y_1^1)$ играл отдельный фрагмент, выделенный из T_k. Однако, если такое выделение не привело к желаемым результатам (не было получено знаний, достаточных для ответа на запрос), то может быть сделана попытка заменить упомянутый фрагмент или добавить к нему новые фрагменты. Имеют место действия переориентации и детализации, но примененные к $T_2(y_2, y_1^1)$. В результате роль $T_2(y_2, y_1^1)$ уже будут играть цепочки фрагментов, выделенных из $T_{вх}$. Будут порождаться графы, представляющие более сложные зависимости, например *зависимость деталей Y_2 от городов, в которых проживают поставщики*. На основе них будут формироваться вероятностные графы уже другого вида. В них опорная вершина будет соответствовать городам, а не поставщикам. Здесь также требуется организация сходящегося процесса, но за счет действий, изменяющих характер связи между компонентами зависимости.

Порождение определений, структурное распознавание. Остановимся вначале на задаче поиска структурных особенностей объектов фиксированного класса. К ней сводится задача поиска минимальных описаний, определяющих понятие того или иного объекта.

Имеется множество объектов, относящихся к классам $\{M_1, ..., M_s\}$. Даны

их структурные описания. Далее указывается один из классов M_k. Требуется найти особенности, которые характеризуют все объекты класса M_k и только их. Ясно, что если описания объектов имеют вид наборов признаков, то задача вырождается и сводится к поиску значимых признаков (или их комбинаций), однозначно определяющих M_k. Однако в случае сложных описаний требуются специальные действия выделения подструктур. Ниже будет рассматриваться методика такого выделения, заключающаяся в последовательных приближениях.

Берется один из объектов — представителей класса M_k. На основе описания объекта выделяются структурные компоненты. Находится все множество объектов, у которых имеются эти компоненты. Данное множество сравнивается с элементами класса M_k. Выявляются рассогласования. В зависимости от их характера вызываются те или иные действия — добавления новых компонент (детализации), их изменения (переориентация) и обобщения. На основе полученных компонент снова находится множество объектов и сравнивается с элементами класса M_k. Подобные действия продолжаются до совпадения. Тогда выделенные (структурные) компоненты объявляются особенностями класса M_k. Они и будут составлять определение понятия M_k.

Остановимся на принципах реализации указанной методики. Пусть T — сеть, представляющая описание различных объектов с указанием их классов. Из T выделяется подсеть $T_v(d_v) \circ \langle \_\_, t, \in, d_v, m_k \rangle$, соответствующая одному из объектов (D_v) класса M_k. Строится граф вида $[y_2 \perp \langle \_\_, t, \in, y_2, m_k \rangle]$ и на основе знаний T находятся значения $[y_2 := \nu]$. Множество ν будет соответствовать известным системе объектам класса M_k.

Далее начинается последовательное вовлечение фрагментов из T_v, связанных с d_v. Вначале (на первом шаге) выделяется один фрагмент, затем (на последующих шагах) — другие, наконец — их цепочки и т. д. Каждый раз на основе выделенных фрагментов (T_1) строится граф $[y_1 \perp T_1(y_1)]$, по которому находятся значения $[y_1 := \eta]$. Множество η будет соответствовать известным системе объектам, обладающим особенностями $\mathcal{T}_1$. Находится пересечение $\eta \cap \nu$ и подсчитываются количества элементов $|\eta \cap \nu|$, $|\eta|$ и $|\nu|$. Вычисляются величины $P_k = |\eta \cap \nu|/|\eta|$ и $G_k = |\eta \cap \nu|/|\nu|$, где $G_k \cdot 100\%$ характеризует процент объектов класса M_k, обладающих особенностями $\mathcal{T}_1$, а $P_k \cdot 100\%$ — сколько (процент) объектов, обладающих особенностями $\mathcal{T}_1$, относится к классу M_k. Формируются соответствующие сети

$$T_1(y_1) \circ \langle \_\_, t, \exists_{P_k}, y_1, y_2 \rangle \circ \langle \_\_, t, \in, y_2, m_k \rangle, \tag{7.23}$$

$$\langle \_\_, t, \in, y_2, m_k \rangle \circ \langle \_\_, t, \exists_{G_k}, y_2, y_1 \rangle \circ T_1(y_1). \tag{7.24}$$

Оцениваются величины P_k, G_k, на основе чего принимаются решения по изменению T_1. Здесь будем различать несколько случаев.

Пусть $P_k < 1$, $G_k = 1$. Это означает, что *все объекты класса M_k обладают особенностями $\mathcal{T}_1$, но такими же особенностями обладают и другие объекты*. Тогда требуется детализация — добавление к $T_1(y_1)$ новых фрагментов, которые берутся из T_v. Учитываются все новые компоненты. При этом ищется такой вариант добавления, который не изменяет G_k, но приводит к увеличению величины P_k.

Пусть $P_k = 1$, $G_k < 1$. Это означает, что *особенностями $\mathcal{T}_1$ обладают только объекты класса M_k, но не все*. Тогда требуется обобщение, которое

сводится к заменам вершин в $T_1(y_1)$. Не принимаются во внимание некоторые детали. При этом ищется вариант замен, приводящий лишь к увеличению G_k.

Пусть $P_k = G_k = 0$. Это означает, что *среди объектов класса M_k нет таких, которые обладали бы особенностями $\mathcal{T}_1$*. Требуется переориентация — вовлечение в $T_1(y_1)$ новых фрагментов взамен имеющихся. Рассматривается другой вариант определения объектов.

И, наконец, пусть $P_k < 1$ и $G_k < 1$. Это значит, что *объекты класса M_k имеют непустое пересечение с объектами, обладающими особенностями $\mathcal{T}_1$*. Здесь могут быть использованы любые из упомянутых ранее действий изменения T_1. Делается попытка найти такое действие, которое значительно увеличивает хотя бы одну из величин P_k или G_k.

Пусть оказалось $P_k = G_k = 1$. Это означает, что *$\mathcal{T}_1$ — есть особенности всех объектов класса M_k и только их*. Тогда в (7.23) на место $\exists_{P_k}$ ставится o-вершина $=$. Полученная сеть будет представлять определение понятия M_k. На основе сети может быть сформирован обязательный граф, представляющий особенности объектов из M_k. Этот граф может служить в качестве решающего правила для выявления и распознавания объектов.

В общем случае описанный процесс может не привести к желаемому результату, т. е. не удается добиться, чтобы $P_k = G_k = 1$. Тогда можно пойти по нескольким путям. Во-первых, могут быть занижены требования. Вместо равенства может использоваться критерий близости к 1 ($P_k \to 1$, $G_k \to 1$). В результате будут строиться графы, представляющие *особенности $\mathcal{T}_1$ большинства объектов из M_k*. Более того, могут быть выделены объекты, которые не удовлетворяют этим особенностям. Они будут играть роль исключений.

Во-вторых, можно искать особенности подклассов, которые покрывают класс M_k. Пусть оказалось $P_k = 1$, а $G_k < 1$. Тогда из M_k могут быть выделены объекты, которые не обладают особенностями $\mathcal{T}_1$. Из них берется представитель и ищутся свои особенности ($\mathcal{T}_2$). Для объектов, которые не обладают ни теми, ни другими особенностями, ищутся третьи особенности ($\mathcal{T}_3$) и т. д. В результате порождается составное определение, которое будет представлено в следующем виде:

$$T_1(y_{11}) \circ T_2(y_{12}) \circ T_3(y_{13}) \circ \ldots \circ [y_1 := \{y_{11}, y_{12}, y_{13}, \ldots\}] \circ$$
$$\circ \langle \_\_, t, =, y_1, y_2 \rangle \circ \langle \_\_, t, \in, y_2, m_k \rangle. \qquad (7.25)$$

Таким способом определяются классы, в которых объекты могут сильно отличаться друг от друга, например, когда имеют место заглавные и рукописные буквы и др.

Введенная методика может быть использована в задачах распознавания объектов. Имеется в виду **распознавание, основанное на выявлении структурных особенностей** классов объектов. Тогда последовательно выделяются классы M_k из $\{M_1, \ldots, M_s\}$, т. е. $k = \overline{1,S}$. Берутся представители и на основе их описаний ищутся особенности. На основе (7.25) строятся графы, которые и играют роль решающих правил.

ПРИЛОЖЕНИЕ 1

ЛИНГВИСТИЧЕСКИЙ ПРОЦЕССОР НА ПОЛУЛИНЕЙНЫХ ПРОДУКЦИЯХ

Общая организация. Напомним, что под лингвистическим процессором понимается устройство, отображающее предложения ЕЯ на базу знаний. Такие процессоры необходимы для любой системы или машины, ориентированной на так называемую категорию необученных или слабообученных пользователей. Имеются в виду различные разработчики, техники, а также геологи, медики, биологи и др., пользующиеся для постановки и решения своих задач (информационного обслуживания, конструирования, распознавания, принятия решений, порождения гипотез с поиском закономерностей и др.) профессиональным ЕЯ, дополненным таблицами, графиками, математической символикой. Типовая схема обработки поступающей от них информации заключается в предварительном выявлении семантического эквивалента, т. е. в отображении предложений ЕЯ (и разного рода формализованных описаний) на уровень таких эквивалентов, который мы называли уровнем СС базы знаний. На нем уже осуществляется обработка — поиск, формирование, корректировка, прослеживание и др. (см. гл. 5).

Упомянутая схема позволяет пользователю самому что называется разрабатывать базу знаний, т. е. вводить в нее соответствующие сведения. При этом разработка должна осуществляться в удобных для человека формах и должна касаться не только статических данных, но и разного рода динамических систем и моделей. Их описания должны отображаться на базу (уровень СС), где уже осуществляется прослеживание происходящих изменений. При этом формальные модели должны каким-то образом сочетаться с концептуальными схемами и моделями, разрабатываемыми через ЕЯ. Вопрос такого сочетания до настоящего времени недостаточно изучен. Здесь нужен единый язык представления (см. гл. 1).

Для указанного отображения требуются специальные лингвистические процессоры — настраиваемые, управляемые с помощью своих знаний (л и н г в и с т и ч е с к и х). Последние должны служить и для обратного отображения — с уровня СС на ЕЯ или формализованные описания. Система должна уметь передавать человеку свои знания, обосновывать решения и др.

Аналогичная обработка перспективна и с точки зрения **построения трансляторов** — с одного языка (Я1) на другой (Я2). Тогда уровень СС будет играть роль так называемого я з ы к а - п о с р е д н и к а. Выражения (предложения) Я1 отображаются на этот уровень. Там уже они дополняются, корректируются, т. е. доводятся до конструкций, удовлетворяющих критериям семантической правильности, допустимости. И затем уже формируются соответствующие выражения Я2. Заметим, что Я1 и Я2 — это могут быть как различные ЕЯ, так и формальные языки, в том числе языки программирования, например Я1 — PL-1, а Я2 — язык Ассемблера. Ясно, что мощные

лингвистические процессоры, ориентированные на ЕЯ, должны легко перерабатывать и гораздо более простые выражения формальных языков. Ведь последние есть продукт формализации некоторых видов естественных представлений.

Итак, процедура трансляции языков програмирования может основываться на использовании лингвистических процессоров, которые в общем случае могут быть ориентированы и на формы ЕЯ. Открываются широкие возможности в плане насыщения языков программирования привычными человеку формами ЕЯ — в качестве операторов и данных. При этом удается снять жесткие ограничения на синтаксис, сделать систему более устойчивой к определенного сорта синтаксическим и семантическим ошибкам. В результате может быть повышена степень «благожелательности» проектируемой системы программирования.

Для реализации обсуждаемой обработки требуются специальные средства — формализмы. Как выясняется, здесь уже недостаточно линейных форм и соответственно линейных грамматик для отображения таких форм. Информация, поступающая от пользователя, в основном носит структурный характер. Поэтому уровень СС должен быть уровнем однородных структур. В качестве последних в гл. 2 и 3 был введен язык семантических сетей и графов. При этом рассматривался случай, когда вначале по выражению $\mathfrak{A}$ входного языка (Я1, ЕЯ) формируется его поверхностная структура — $ПС[\mathfrak{A}]$, т. е. некоторая сеть, представляющая пространственные отношения между символами (см. § 2.3). Тогда вся последующая обработка сводится к оперированию над сетями, управляемому с помощью обязательных графов и сетевых продукций. Последние и образуют лингвистические знания. Средства их представления те же самые, что и средства представления знаний о предметной области.

Описанная организация предпочтительна в том случае, когда имеется специальный процессор быстрого оперирования над семантическими сетями. Но пока еще такого процессора нет. Есть лишь предложения, проекты [36]. В связи с этим приходится ориентироваться на существующую технику — ЭВМ, их входные языки, которые имеют дело с числами, словами, текстами, т. е. линейными формами. Конечно, последние могут быть использованы для записи всевозможных структур, в том числе и $ПС[\mathfrak{A}]$,— множество фрагментов представляется в виде набора кортежей, которые сводятся в единую таблицу (массив) (см. § 2.3). Однако это не всегда удобно, так как требуется большое число фрагментов (кортежей).

Если входной язык достаточно ограничен (удовлетворяет свойству проективности, слова располагаются в определенном порядке и др.), то лингвистические знания могут быть устроены более простым способом. Линейные формы здесь могут быть сохранены. То же самое относится и к выражениям $\mathfrak{A}$ входного языка. Тогда процедура преобразования выражений $\mathfrak{A}$ на уровень СС будет заключаться в последовательном замещении компонентов (слов, словосочетаний) из $\mathfrak{A}$ на их семантические эквиваленты, т. е. на соответствующие структуры. В конце концов должны остаться только структуры уровня СС. Здесь возникает необходимость в специальных формализмах, сочетающих в себе линейные формы и семантические сети. Ниже будут рассматриваться такие формализмы.

Полулинейные конструкции. Пусть имеется базовый алфавит $\{A, Б, В, \ldots\}$, из символов которого составляются исходные слова. Пусть $\mathfrak{A}$ — одно такое

слово. Заменим в нем один из символов на какую-нибудь вершину семантической сети. Получится конструкция, которую будем называть полулинейной. Например, CBx_1AB есть полулинейная конструкция, у которой на третьей позиции стоит $н$-вершина x_1. В общем случае будем допускать замену символов на наборы вершин (n-ки). При этом замещаться могут сразу несколько символов, а стоящие на их местах вершины могут быть определенным образом связаны между собой и с другими вершинами, т. е. они могут входить в свои сети и графы. Будем записывать последние (через символ $\perp$) рядом с вершинами. Например,

$$CB[x_1 \perp T_1(x_1)]DE[x_2 \perp T_2(x_2)]KB$$

есть полулинейная конструкция, у которой на третьей и шестой позициях стоят $н$-вершины x_1 и x_2 сетей $T_1(x_1)$ и $T_2(x_2)$. При этом сами сети стоят как бы в стороне, т. е. не занимают собственной позиции. Они образуют окрестности $н$-вершин x_1 и x_2 (окрестности запоминаются в спектральной памяти).

Будем считать, что в базовом алфавите находятся разного рода синтаксические знаки — пробел, точка, запятая и др. В таком алфавите уже могут записываться предложения, тексты ЕЯ. Их компоненты — слова, словосочетания — также могут замещаться на вершины сети. Получаются полулинейные конструкции, у которых на местах отдельных слов стоят их семантические эквиваленты — вершины. Например, в конструкции

$$x_1^1 | человек | ЯВЛЯЕТСЯ\ БРАТОМ\ [x_2^1 | человек | \perp \langle \_\_, t, би, x_2^1, 'Иван' \rangle] \tag{1}$$

имеются вершины x_1^1 и x_2^1, сопоставленные двум людям. Второй имеет имя *Иван*. Их отношение *являться братом* пока еще записано в линейной форме. Семантический эквивалент данного отношения как бы не выявлен.

Если в (1) заменить словосочетание *ЯВЛЯЕТСЯ БРАТОМ* на $[\gamma \perp \gamma_1, t, б.бр, x_1^1, x_2^1\rangle]$, то получится набор фрагментов, из которых легко может быть составлен семантический граф, а если удалить в последнем стрелки порядка — то сеть. Итак, частными случаями полулинейных конструкций являются введенные ранее семантические сети и графы, а также обычные последовательности символов — слова, предложения, тексты.

Полулинейные продукции. Под ними будем понимать продукции, у которых левая (Δ_1) и правая (Δ_2) части есть полулинейные конструкции. Такие продукции записываются в виде

$$\Delta_1 \rightarrow \Delta_2. \tag{2}$$

В частном случае (если Δ_1 и Δ_2 — слова в заданном алфавите) они вырождаются в обычные правила подстановок формальных грамматик. Если Δ_1 и Δ_2 — семантические сети, то получаются сетевые продукции (см. § 2.1).

Продукция (2) применяется к полулинейной конструкции (обозначим ее через Δ) как правило подстановки, только с некоторыми дополнениями. В Δ ищутся вхождения конструкции Δ_1, т. е. такая последовательность символов и вершин, которая совпадает с Δ_1. При этом понятие совпадает несколько расширяется. Допускается отождествление $н$-вершин из Δ_1 (будем обозначать их через y_i^1) с вершинами из Δ, содержащими аналогичную окрестность. Такие вершины также считаются совпадающими.

Пусть в Δ найдена часть, совпадающая с Δ_1. Тогда осуществляются действия замены. При этом линейные последовательности замещаются

обычным образом (как в случае правил подстановок). При замене вершин осуществляются специальные действия переноса значений, формирования резервных вершин, как в случае применения сетевых продукций (см. § 2.1). В результате на основе Δ_2 подготавливается часть, которая и замещает в Δ часть, совпавшую с Δ_1. Остановимся на этих действиях более подробно.

Напомним, что применение сетевой продукции $T_1 \rightarrow T_2$ к Δ сводится к поиску в Δ части (вершин с их окрестностями), которая удовлетворяет «клише» T_1. Сеть T_1 с ее $н$-вершинами, играющими роль слотов, как бы заполняется материалом, взятым из Δ. В результате каждой $н$-вершине из T_1 будут присвоены значения. Ими будут вершины, которые считаются совпадающими со своей $н$-вершиной, т. е. имеет место отождествление. Далее найденная часть изымается из Δ. Последующие же действия зависят от вида продукции. Пусть и в левой T_1, и в правой T_2 части продукции имеется одна и та же $н$-вершина. Тогда её найденное значение переносится в T_2. И если других $н$-вершин нет, то полученная сеть T_2^* просто добавляется к Δ. При этом за счет переноса значений вставляемая часть заведомо привязывается к тем вершинам, которые остались в Δ. Если в $T_1 \rightarrow T_2$ имеются $н$-вершины, которые входят только в правую часть, то на основе них формируются так называемые р е з е р в н ы е значения, т. е. новые вершины (ранее не встречавшиеся), представляющие новые (появляющиеся) объекты. Полученная сеть T_2^* добавляется к Δ. Еще раз отметим, что описанные действия совмещаются с обычными подстановками линейных форм (если они имеются в Δ_1 и Δ_2). При этом такие формы могут замещаться и на вершины.

Рассмотрим несколько типовых примеров. Продукцция вида

$$БРАТ \rightarrow \text{'быть братом'} | кл.\ р.\ о | \qquad (3)$$

применяется к предложениям или конструкциям Δ, в которых встречается слово *БРАТ*. Оно заменяется на $н$-вершину *'быть братом'*, к которой добавляется фрагмент $\langle \_\_,\ t,\ \in,\ \text{'быть братом'},\ кл.р.о \rangle$, т. е. представляется *принадлежность к классу родственных отношений* (*кл. р. о*). Такой фрагмент в (3) записан в сокращенном виде. Другая продукция

$$ИВАН \rightarrow [y_1^1 | человек | \perp \langle \_\_,\ t,\ би,\ «Иван»,\ y_1^1 \rangle] \qquad (4)$$

применяется к предложению и вызывает замену каждого слова *ИВАН* на резервную $н$-вершину, которая формируется на базе y_1^1. При этом различным словам *ИВАН* будут сопоставлены свои $н$-вершины, которые (на основе знаний) могут быть соответствующим образом конкретизированы — найдены их значения. Фактически как только встречается слово *ИВАН*, то вначале имеется в виду некий человек с именем Иван, а затем может быть уточнено, о каком человеке идет речь. Для этого используются системные знания (см. гл. 4). Конечно, уточнение будет однозначным, если в знаниях представлен лишь один человек с именем Иван. Если их множество, то возникают альтернативы, которые могут быть устранены лишь по мере выявления семантических эквивалентов других слов или же за счет организации активного диалога (см. § 5.3).

Другая продукция

$$y_1^1 | человек | БРАТ y_2^1 | человек | \rightarrow [\beta_1 \perp \langle \beta_1,\ t,\ б.\ бр,\ y_1^1,\ y_2^1 \rangle] \qquad (5)$$

применяется лишь к полулинейным конструкциям Δ, в которых упоминаемым

людям уже сопоставлены их семантические эквиваленты — вершины. Если в конструкции между двумя такими вершинами (пусть это x_i^1 и x_j^1) имеется буквосочетание *БРАТ*, то продукция делается применимой. Ее $н$-вершинам присваиваются значения $y_1^1 := x_i^1$ и $y_2^1 := x_j^1$, которые переносятся в правую часть и формируется $[\gamma_k \perp \langle \gamma_k,\ t,\ б.\ бр,\ x_i^1,\ x_j^1 \rangle]$. В результате применения в конструкции Δ на месте $x_i^1 БРАТ x_j^1$ будет стоять одна вершина γ_k.

Порядок применения, режимы работы. Продукций может быть множество. Они упорядочиваются, образуя набор. Пусть $\mathfrak{A}$ — исходное слово (предложение). Продукции применяются к $\mathfrak{A}$ последовательно, т. е. вначале берется первая продукция и делается попытка ее применения. Если она не применима, то берется вторая и т. д. Очередная продукция набора применяется только в том случае, если не применимы все предыдущие, т. е. как в обычных грамматиках. В результате такого применения слово $\mathfrak{A}$ будет подвергаться преобразованиям. Их промежуточными состояниями будут полулинейные конструкции. Конечная цель преобразований — построение семантической сети или графа. Преобразования заканчиваются, если нет применимых продукций. При этом будем различать д в а т и п а к о н е ч н ы х с о с т о я н и й: первый — когда построена сеть (граф), второй — когда в конструкции Δ остались символы базового алфавита. Последний случай будем выделять особо. Он означает, что имеющегося набора продукций недостаточно для построения семантической сети. Требуется их пополнение, что осуществляется в специальных режимах, связанных с обучением, организацией активного диалога.

Действия применения полулинейных продукций непосредственно связаны с действиями поиска в системных знаниях. В процессе такого применения формируются семантические эквиваленты, которые могут быть уточнены на основе знаний. В частности, выше рассматривался случай, когда при применении продукций (у которых в правой части есть $н$-вершины, не входящие в левую) формируются резервные $н$-вершины, соответствующие новым, ранее неизвестным объектам. Последние должны уточняться. Здесь возможны различные р е ж и м ы работы. Один из них, когда вначале анализируется все предложение $\mathfrak{A}$, строится соответствующая сеть. И уже после этого осуществляется уточнение. Другой режим, когда попытки уточнения осуществляются постоянно, после каждого применения очередной продукции. И если возможно однозначное уточнение, то оно осуществляется. При этом уточняющими являются объекты, представленные в знаниях. Соответствующие им вершины вставляются в полулинейные конструкции. Например, если в Δ встретилось слово *ИВАН*, то уточняется, что это за человек, а соответствующая вершина замещает в Δ это слово. В результате полулинейная конструкция как бы привязывается к знаниям. Если же однозначное уточнение невозможно, то берутся новые слова из Δ, строятся их семантические эквиваленты, позволяющие более точно определить, какие объекты или субъекты имеются в виду.

Рассмотрим **пример**. Пусть входное предложение имеет вид *КАК ИВАН ИВАНОВИЧ И ИВАН НИКИФОРОВИЧ ПОССОРИЛИСЬ*. Тогда оба Ивана могут быть уточнены лишь по их отчествам, т. е. когда последним будут сопоставлены свои фрагменты. На основе них и осуществляется поиск в знаниях с уточнением — нахождением значений соответствующих $н$-вершин. При этом возможны различные с л у ч а и. Так поиск может осуществляться в рамках определенного сценария (части знаний), о котором в на-

стоящем времени идет речь. Еще один случай, когда в знаниях (с помощью весовых коэффициентов или использованием оси-времени) представлено, сколь давно упоминался тот или иной объект (субъект). И если возникли альтернативы, то из них выбирается та, которая соответствует последнему из упоминавшихся объектов. Например, если Иван Ивановичей много, то будет взята вершина, соответствующая последнему из них.

Особо стоят случаи, связанные с **появлением новых объектов**, а также с так называемыми анафорическими ссылками. Пусть ни о каком Иване Ивановиче система раньше не знала. Тогда сопоставленная ему резервная вершина просто вводится в знания вместе с соответствующими фрагментами. И каждому последующему Ивану Ивановичу (конечно, если не говорится *другой Иван Иванович*) будет сопоставляться эта вершина, что осуществляется описанным ранее способом. В случае же указателя *другой* объекту должна сопоставляться своя вершина.

При наличии а н а ф о р и ч е с к и х ссылок типа *ЭТОТ ИВАН, ТАКОЙ ИВАН* поиск с уточнением должен осуществляться не в знаниях, а в исходном предложении или полученной на основе него полулинейной конструкции Δ. Один из примеров такого поиска будет рассмотрен чуть ниже.

Язык ДВ1. Вид входных языков во многом зависит от используемых средств трансляции. Ограниченность существующих языков определяется уровнем развития таких средств. В основном они базируются на линейных грамматиках. Однако если стоит задача приближения языка к формам ЕЯ с их отображением на базу знаний, грамматик оказывается явно недостаточно. Существенные ограничения накладывают и сети Вудса (*ATN*-грамматики), а также многие другие средства. Введенные выше полулинейные продукции позволяют снять некоторые из таких ограничений. Становится возможным построение достаточно простых, но весьма интересных входных языков. Один из них был назван ДВ1 (диалоговый, первая версия). Остановимся на некоторых его особенностях.

ДВ1 предназначен для ввода в систему информации, выраженной на ЕЯ. Каждое предложение ЕЯ вначале подвергается **предварительной переработке** (что может делаться автоматически или вручную). Перед каждым словом ставится префикс, отражающий грамматическую форму слова. Так, слово *СКЛАДА* будет переработано в (*КОГО, М, ЕД*) *СКЛАД*, где *КОГО* отражает падеж слова, *М* указывает на мужской род, а *ЕД* — единственное число (в одном из вариантов пара *М, ЕД* заменяется на *ОН*, пара *М, МН* — на *ОНИ* и т. д.). Само слово записывается в форме, которая при различных словоупотреблениях должна быть одна и та же. Аналогичные префиксы ставятся и перед словосочетаниями, а также самостоятельными группами слов, которые ограничиваются скобками [...]. Перед глаголами, наречиями, ... ставятся свои префиксы.

Рассмотрим пример. Остановимся вначале на случае, когда используются наиболее простые префиксы, т. е. не учитывается род, число и некоторые другие категории. Так, предложение *ДАВЛЕНИЕ МАССЫ ГАЗА ЗАВИСИТ ОТ АБСОЛЮТНОЙ ТЕМПЕРАТУРЫ* записывается в следующем виде [(*ЧТО*) *ДАВЛЕНИЕ* (*ЧЕГО*) [(*ЧТО*) *МАССА* (*ЧЕГО*) *ГАЗ*] (*ЧТО ДЕЛАЕТ*) *ЗАВИСИТ* (*ОТ ЧЕГО*) [(*КАКОЙ*) *АБСОЛЮТНЫЙ* (*ЧТО*) *ТЕМПЕРАТУРА*]]. Подобные формы записи строятся по достаточно простым правилам. Нужно только уметь формулировать вопросы на каждое слово или словосочетание. Соответствующие вопросительные местоимения

и будут префиксами. При этом предлоги будут входить в такие префиксы.

Отображение указанных форм **в базу знаний** (на уровень СС) осуществляется полулинейными продукциями рассмотренного ранее вида. Остановимся на них более подробно. Введем сокращенную запись *об — класс объектов, от — класс отношений, св — класс свойств*. Будем пользоваться следующими обозначениями *о*-вершин: *св — иметь свойство, б.мас — быть массой, б. дав — быть давлением, б. темп — быть температурой, абс* — соответствует свойству *абсолютный, зав* — отношению *зависит*. Тогда продукции, осуществляющие отображение, будут иметь вид

$ГАЗ \to [y^1_1 | об \perp \langle \_, t, \in, y^1_1, 'газ' \rangle]$;

$МАССА \to б.мас | отн$;

$ДАВЛЕНИЕ \to б.дав | отн$;

$ТЕМПЕРАТУРА \to [y^1_1 | об] \cdot \langle \_, t, б.темп, y^1_1, y^1_2 \rangle]$;

$АБСОЛЮТНЫЙ \to абс | св$;

$ЗАВИСЕТЬ \to зав | отн$;

$[(ЧТО) y^1_1 | отн | (ЧЕГО | y^1_2 | об] \to [y^1_3 | об \perp \langle \_, y, y^1_1, y^1_3, y^1_2 \rangle]$;

$[(КАКОЙ)\ y^1_1 | св | (ЧТО)\ y^1_2 | об] \to [y^1_2 | об \perp \langle \_, t, св, y^1_2, y^1_1 \rangle]$;

$[(ЧТО) y^1_1 | об | (ЧТО\ ДЕЛАЕТ) y^1_2 | отн | (ОТ\ ЧЕГО) y^1_3 | об] \to$
$\to [[\beta_1 \perp \langle \beta_1, t, y^1_2, y^1_1, y^1_3 \rangle]$,

где $y^1_1 | об |$ — сокращенное обозначение $[y^1_1 \perp \langle \_, t, \in, y^1_1, об \rangle]$. В результате применения продукций к исходной форме будет получен граф

$[x_1 \perp \langle \_, t, \in, x_1, 'газ' \rangle] \circ [x_2 \perp \langle \_, t, б, мас, x_1, x_2] \circ [x_3 \perp \langle \_, t, б.дав, x_2, x_3 \rangle] \circ [x_4 \perp \langle \_, б.темп, x_4, x_5 \rangle] \circ [x_4 \perp \langle \_, t, св, x_4, абс \rangle] \circ [\gamma_1 \perp \langle \gamma_1, t, зав, x_3, x_4 \rangle]$,

который представляет семантический эквивалент предложения (уровня СС), см. рис. П1. Его *н*-вершины $x_1, ..., x_5$ будут сформированы в процессе применения продукций как резервные. Они соответствуют неопределенным объектам. При необходимости может быть сделана попытка их уточнения, т. е. о каком газе идет речь (x_1: =?), о температуре чего говорится в предложении (x_5: =?). Такое уточнение сводится к нахождению значений *н*-вершин, для чего используются знания.

Предыдущий пример намеренно упрощен, чтобы проиллюстрировать основную идею. В реальных случаях требуются префиксы, отражающие с достаточной точностью грамматические категории слов. Соответственно усложняются продукции. С помощью последних удается в достаточной степени учитывать информацию, касающуюся соотнесенности объектов к классам и др. За счет нее удается «бороться» с **полисемией** глаголов. Например, в предложениях *ВЫБИВАТЬ БЕЛЬЕ* и *ВЫБИВАТЬ КВАРТИРУ* глаголы *ВЫБИВАТЬ* означают различные действия, которым на уровне СС должны быть сопоставлены свои *о*-вершины. Такое сопоставление осуществляется продукциями, в левых частях которых представлена принадлежность к классам объектов. Реализуются так называемые с е м а н т и ч е с к и е п р е д п о ч т е н и я, где классы определяют значения слов.

Естественно, подобные продукции будут применимы только к таким конструкциям, в которых имеется **информация о классах**. Последняя должна

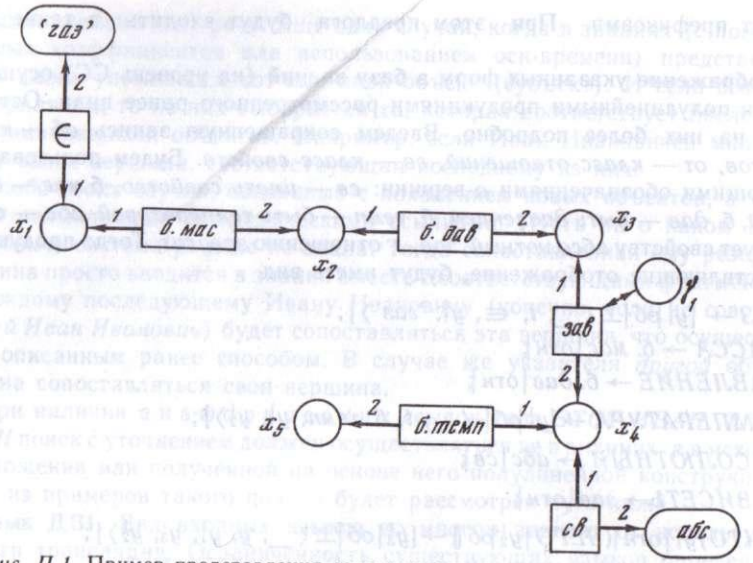


Рис. П.1. Пример представления семантического эквивалента предложения

быть выявлена и **добавлена** к предложению или полученной из него конструкции Δ. Это может осуществляться в процессе уточнения и нахождения значений *н*-вершин. Вернемся к ранее рассмотренному примеру рис. П1. Пусть на основе знаний было найдено $x_2 := a_1$. Это означает, что речь идет о категории *массы*, которой в знаниях сопоставлена вершина a_1. Вся окрестность последней как бы переносится на x_2. В результате уточняется и информация о классах, т. е. что речь идет о весе (массе), а не о количестве (сравните — масса народу). Становится возможным более дифференцированное применение последующих продукций.

Полулинейные продукции могут служить для работы с так называемыми **эллиптическими конструкциями**, т. е. восстанавливать то, что дается по умолчанию. При этом такое восстановление может осуществляться двумя способами. Первый — за счет обязательных знаний, где предполагается, что вначале предложение преобразуется на уровень СС. И второй способ — восстановление, дополнение информации с помощью продукций в процессе преобразования на уровень СС. Некоторые отношения навязываются формами предложений. Например, *ДОМ ИЗ КИРПИЧА* — предполагается отношение *состоять из*. При этом здесь большую роль играют не только грамматические формы слов, но и семантические классы, например, *ШАПКА НА МЕХУ* (отношение *состоять из*) и *ШАПКА НА ГОЛОВЕ* (*находится на*). Выявление подобных отношений должно осуществляться в процессе анализа форм, для чего могут использоваться полулинейные продукции вида

$$[(ЧТО)y_1|изд|(НА\ \ ЧЕМ)y_2|материал|] \rightarrow [y_1 \perp \langle \_\_,\ t,\ сост,\ y_1,\ y_2 \rangle|.$$

Данная продукция анализирует словосочетания форм (*ЧТО*) (*НА ЧЕМ*), где первое слово означает *изделие*, а второе — *материал*. Продукция применяется к конструкциям Δ предложений, в которых уже введена информация о се-

мантических классах. В результате применения восстанавливается отношение *состоять из* (*сост*).

В рамках описанного подхода удается осуществлять отображение (на уровень СС) множества связанных предложений. При этом **местоимения** могут **отождествляться** с обозначаемыми ими объектами достаточно простым способом. Пусть на вход системы поступило два предложения $\mathfrak{A}$ и $\mathfrak{B}$. Вначале берется первое из них и строится полулинейная конструкция Δ^1, где уже представляются упоминаемые объекты (последние могут быть уточнены на основе знаний). Затем берется второе предложение. Пусть в $\mathfrak{B}$ встретилось местоимение *ЕМУ*. С помощью продукций оно преобразуется к виду (*КОМУ, М, ЕД*) $x_j|об|$ и делается попытка нахождения значения x_j, но уже на основе Δ^1. При этом учитывается лишь часть префикса (__, *М, ЕД*), т. е. падеж не важен. Если такое значение найдено (пусть $x_j := x_i$), то x_i может быть поставлено на место x_j. В результате слово *ЕМУ* будет замещено на (*КОМУ, М, ЕД*) $x_i|об|$, где x_i — вершина, соответствующая подразумеваемому объекту (под *ЕМУ*).

В рассмотренном примере отождествление осуществлялось на уровне семантических эквивалентов. Такое же отождествление может сводиться к подстановкам на уровне слов. Пусть в $\mathfrak{A}$ и $\mathfrak{B}$ использован вариант записи префиксов с помощью местоимений. Пусть в $\mathfrak{A}$ имелось слово *ИВАНА*, которое записывается в виде (*КОГО, ОН*) *ИВАН*. Тогда местоимение ЕМУ из $\mathfrak{B}$ преобразуется в (*КОМУ, ОН*) χ, где χ соответствует неизвестному слову. Поиск этого слова сводится к анализу предыдущих префиксов. В $\mathfrak{A}$ ищется такой из них, в котором есть местоимение *ОН*. Если префикс найден, то стоящее за ним слово просто ставится на место χ. В данном примере на место χ будет поставлено *ИВАН*. В дальнейшем обычном способом (см. выше) одним и тем же словам ставится в соответствие один семантический эквивалент — вершина.

Аналогичная методика может использоваться для отождествления местоимений типа *КОТОРЫЙ, КОТОРОГО*, ... Они преобразуются в ту же запись (*КТО, ОН*) χ, (*КОГО, ОН*) χ,...

Отождествление указательных местоимений типа *ЭТОТ, ТАКОЙ, ДАННЫЙ* и др. лучше осуществлять на уровне семантических эквивалентов. Дело в том, что вслед за этими местоимениями могут стоять слова, которые не встречались ранее в тексте и которые указывают на класс. Отождествление возможно, только когда выявлены эти классы, т. е. когда построены полулинейные конструкции Δ^i. Рассмотрим методику отождествления на простом примере.

Пусть в первом предложении имелось слово *ИВАН*, которому в Δ^1 была сопоставлена вершина $x_i^1|человек| \perp \langle \_\_,\ t,\ б.и,\ 'Иван',\ x_i^1 \rangle$. Пусть во втором предложении говорится *ЭТОМУ ЧЕЛОВЕКУ*. Данное словосочетание с помощью продукций преобразуется к виду $(КОМУ)x_j^1|человек|$ и делается попытка нахождения значения x_j^1 на основе Δ^1. При этом в процессе поиска префикс (*КОМУ*) не учитывается (он нужен для анализа форм с указанным словосочетанием). Итак, указательные местоимения *ЭТОТ, ТАКОЙ, ДАННЫЙ* как бы говорят, что искать соответствующие объекты нужно не в знаниях, а на основе предыдущей информации Δ^1 (или уже построенной части Δ^2). В результате будет получено $x_j^1 := x_i^1$, что равносильно отождествлению объектов.

Семантические трансляторы. Как уже говорилось, лингвистические процессоры могут служить для построения трансляторов, в которых предпола-

гается отображение выражений входного языка на уровень СС (языка-посредника) с последующей обработкой на этом уровне. Не следует путать их с так называемыми семантическими ориентированными транслято-рами, нашедшими распространение в системах программирования и отображающими входные выражения в машинные коды (или на операторы какого-либо алгоритмического языка, с которого уже имеется транслятор). Для такого отображения достаточно существующих грамматик. Но на уровне машинных кодов легко реализуются лишь функции выполнения команд (операторов) ЭВМ. В то же время для повышения степени благожелательности системы необходимы многие другие функции, реализацию которых лучше осуществлять на специальном уровне СС (см. § 2.4).

Ниже будет рассматриваться методика использования полулинейных продукций для преобразования выражений формальных (в том числе алгоритмически) языков на уровень СС. Такие выражения гораздо проще предложений ЕЯ — в них нет грамматических форм, порядок слов достаточно жестко фиксирован и т. д. И естественно, здесь должны быть применимы введенные ранее средства и методики работы с ЕЯ, что будет проиллюстрировано ниже на типовых примерах.

Остановимся вначале на **алгебраических языках**. Выражения этих языков строятся из элементов базового алфавита, куда входят константы A_i, имена переменных X_j, знаки операций $+, \times, ..., \wedge, \vee, ...$ и имена функций F_k. В общем случае константами могут быть как числа, так и символы t, f и цепочки символов — слова. Такие константы отображаются на уровень СС с помощью продукций вида

$$A_i \to a_i | m_p |, \tag{6}$$

где a_i — o-вершина, сопоставленная A_i, а m_p представляет тип константы или класс (если константы лишь одного типа, то o-вершина m_p может отсутствовать). Продукции применяются к исходному выражению и осуществляют замещения, равносильные обычному кодированию, но с дополнительным указанием на класс, например *класс целых чисел* $a_i | ц.\ чс |$ и т. д.

Аналогичные продукции служат для отображения знаков операций и имен функции, например

$$+ \to + | ap.\ on |, \tag{7}$$

где $+$ в левой части обозначает сам знак, а в правой — соответствующую ему o-вершину. Последняя входит во фрагмент, представляющий *принадлежность к классу арифметических операций (ap. on)*. Другая продукция

$$F_k \to f_k | m_g | \tag{8}$$

служит для замещения имени функции F_k на сопоставленную ему o-вершину f_k, окрестность которой представляет принадлежность к классу M_g, например к классу *тригонометрических функций*.

Несколько другим способом отображаются имена переменных (X_j). Обычно с помощью таких имен выражаются соотношения, в которых принимают участие не сами переменные, а их значения. В связи с этим имена должны отображаться на семантические эквиваленты, представляющие эти значения, что осуществляется с помощью продукций

$$X_j \to [x_i^l | m_p | \perp \langle \_,\ t,\ и.\ зн,\ 'X_j',\ x_i^l \rangle], \tag{9}$$

где o-вершина 'X_j' соответствует имени переменной, $н$-вершина x_i^l — ее значению, а o-вершина *и.зн* — отношению *иметь значение*. Путем применения такой продукции к $\mathfrak{A}$ будут осуществляться необходимые замещения.

Итак, для каждого символа алфавита имеется своя продукция, которая отображает этот символ на уровень СС (их число может быть значительно уменьшено использованием обобщенных продукций, что пока оставим в стороне). Синтаксические знаки (скобки, запятые и др.), которые указывают на связи, на данном этапе не отображаются. Такое отображение осуществляется своими продукциями, выявляющими соотношения, см. ниже.

В результате применения введенных продукций к исходному выражению $\mathfrak{A}$ будет получена полулинейная конструкция Δ. Например, для арифметического выражения $((X_1+5)\times X_2)$ такая конструкция будет иметь вид

$$\begin{aligned}&(([x_1^l | ц.чс | \perp \langle \_,\ t,\ и.\ зн,\ 'X_1',\ x_1^l \rangle] + | ap.\ on | '5' | ц.чс |) \times \\ &\times | ap.\ on | [x_2^l | ц.\ чс | \perp \langle \_,\ t,\ и.\ зн,\ 'X_2',\ x_2^l \rangle]),\end{aligned} \tag{10}$$

где имеется девять позиций. На первых двух стоят открывающиеся скобки, на последующих трех — вершины x_1^l, $+$, '5', затем — закрывающаяся скобка, на последующих двух — вершины $\times$, x_2^l и на последней позиции — закрывающаяся скобка. Все остальные знаки — окрестности вершин (считается, что они не занимают позиций). Отметим, что в конструкции (10) еще есть отдельные символы алфавита — синтаксические знаки (скобки).

Для **выявления самостоятельных частей** исходного выражения (называемых термами) используются полулинейные продукции следующего вида:

$$(y_1^l | m_p | y_2^l | m_g | y_3^l | m_p |) \to [y_4^l | m_p | \perp \langle \_,\ t,\ y_2^l,\ y_1^l,\ y_3^l,\ y_4^l \rangle], \tag{11}$$

где в левой части имеется всего пять позиций — на первой стоит открывающаяся скобка, затем — $н$-вершины y_1^l, y_2^l, y_3^l и на последней — закрывающаяся скобка. O-вершины m_p и m_g представляют классы. Пусть, к примеру, m_p соответствует *классу целых чисел (ц. чс)*, а m_g — *арифметических операций (ap. on)*. Тогда путем применения продукции (11) к (10) будет получен семантический граф, рис. П2, представляющий само арифметическое выражение. При этом продукция будет применима два раза. При первом применении (на базе y_4^l) будет сформирована резервная $н$-вершина x_3^l, соответствующая (X_1+5), а при втором — x_1^l, соответствующая значению всего арифметического выражения.

Выше был рассмотрен лишь пример. В общем случае возникает необходимость в продукциях вида (11), у которых в левой части отсутствуют скобки, а на месте y_2^l стоит вершина $\times$ (или $+$,...). Продукции упорядочиваются в соответствии с приоритетом операций. Тогда становится возможным анализ выражений, записанных в бесскобочной символике. Далее, необходимы продукции, анализирующие равенства, тождества и строящие соответствующие семантические графы. Такие продукции рассмотрены в [25]. С помощью них любое выражение языков арифметических функций, булевой алгебры, логики предикатов, теоретикомножественных соотношений может быть преобразовано на уровень СС.

При работе с **языками программирования** требуются дополнительные наборы продукций, анализирующих разного рода служебные слова, (*go to, if, then*,...) и выявляющих законченные конструкции. В языках программи-

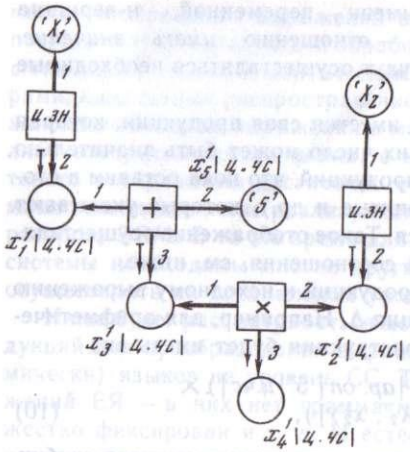


Рис.. П.2. Граф, представляющий арифметическое выражение $((X_1 + 5) \times X_2)$

рования таких конструкций достаточно большое количество. С помощью соответствующих продукций они отображаются на уровень СС. Полученные фрагменты дополняют общий граф (сеть, см. § 2.4). При этом особую роль играют описатели переменных, массивов, функций. Они отображаются на продукции, которые дополняют общий набор. Например, по описанию *ЦЕЛЫЙ X1* строится продукция $X1 \rightarrow x_1 | ц.\ чс |$, которая в дальнейшем используется для сопоставления идентификаторами *X1* вершины x_1. В общем случае такое сопоставление может осуществляться лишь внутри блоков (тогда продукции привязываются к блокам — «сценариям»).

Пусть с помощью продукций по выражению языка уже построен соответствующий граф. Тогда начинают выполняться действия, связанные с проверкой семантической правильности, допустимости, для чего используются специального вида обязательные знания (см. § 5.3). Граф может быть (по умолчанию) дополнен. Пользователю может быть указано на то, чего не хватает в выражении и др. По окончании этого этапа законченный граф может быть отображен на специальный интерпретатор (см. § 4.1) или же на систему команд какой-либо базовой машины. Для последнего отображения используются два набора продукций: сетевые продукции, осуществляющие преобразование представлений (в соответствии с системой команд базовой машины), а также полулинейные продукции, строящие последовательность команд (только эти продукции должны работать в другую сторону). Эти продукции фактически решают **задачу автоматического построения алгоритма** по структурам языка-посредника — семантическим сетям и графам. Причем с помощью последних могут представляться разного рода соотношения, по которым строится готовый алгоритм.

Итак, для отображения алгоритмических выражений на уровень СС используются собственные наборы полулинейных продукций. Однако такие наборы всегда можно пополнить продукциями, отображающими формы ЕЯ. Тогда делается возможным использование этих форм в рамках самих выражений. Процесс подобного пополнения и будет приводить к постоянному приближению формального или алгоритмического языка к ЕЯ.

Перспективы. При описанном подходе используемый входной язык какой-либо системы или машины может быть значительно приближен к ЕЯ. Однако в ряде случаев здесь могут возникать затруднения, в основном вызванные наличием линейных форм. В рассмотренных ранее примерах предполагалось, что выражения исходного языка удовлетворяют свойству проективности и порядок слов в словосочетаниях (символ в термах) фиксирован. В то же время в русском языке такой порядок достаточно произволен. В связи с этим требуются продукции, у которых в левых частях элементы не столь привязаны к своим позициям. Далее, в ряде случаев некоторые компоненты в предложениях могут быть, а могут и не быть. Возникает необходимость в продукциях с факультативными фрагментами.

Отметим еще один фактор. При работе с формами ЕЯ могут встречаться неоднозначности, которые будут приводить к различным вариантам анализа. Могут оказаться применимыми различные продукции, приводящие к различным результатам. Возникающие при этом альтернативы должны каким-то образом сохраняться и в дальнейшем устраняться. Здесь требуются соответствующие средства. Как было показано в работах [14, 24], их роль могут играть семантические сети и графы. В частности, альтернативы представляются с помощью *з*-сетей (см. § 2.2).

Еще один интересный вопрос связан со случаем **появления новых слов**. В языках программирования такое появление предваряется описателями. Однако в ЕЯ новые слова могут специальным образом не вводится, а просто появляться в контексте. Тогда для выявления (хотя бы приблизительно) их грамматической формы и семантических классов должна использоваться специальная процедура формирования продукций. Задача ставится следующим образом: нужно сформировать новую продукцию так, чтобы сделалось возможным применение уже имеющихся продукций с отображением входного предложения на уровень СС. Такая процедура будет весьма полезной и в трансляторах с алгоритмических языков, где контекст часто однозначно определяет тип переменной, ее описатель. Например, в $P1 \vee P2$ ясно, что $P1$ и $P2$ — булевы переменные.

В заключение следует отметить, что расширение, развитие линейных форм (и основанных на них внутренних языках), а также полулинейных конструкций неизбежно приводит к необходимости организации сугубо **структурного уровня**, возможность которого неизмеримо богаче. Здесь могут использоваться сетевые продукции, граф-продукции, модальные, корреляционные продукции, которые никак не привязаны к линейным формам. Хотя они прекрасно могут служить для представления, отображения таких форм (см. § 2.3 и 5.2). Собственно любая полулинейная конструкция (продукция) может быть преобразована в семантическую сеть (продукцию) достаточно простым способом — необходимо только оставшиеся в них линейные последовательности заменить на представляющие их фрагменты. Но, конечно, количество таких фрагментов может оказаться значительным (напомним, что два стоящих рядом символа представляются с помощью трех фрагментов).

Что касается обратного отображения, т. е. представления сетей (или продукций) в виде линейных форм (грамматик), то здесь возникают существенные трудности. До настоящего времени остается неясным, можно ли такое отображение сделать однозначным, какой должна быть процедура, которая возьмет семантическую сеть и всегда построит по ней одну и ту же линейную форму. Подобная форма всегда будет зависеть, с какой вершины начат осмотр сети, т. е. вначале должна быть выбрана такая вершина. Да и многие виды обработки («ведения» концептуальных моделей) чрезвычайно затрудняются, если осуществлять их на уровне линейных форм. Здесь требуется уровень структур, где становится возможным использование единых методик или механизмов. В достаточно общей постановке могут ставиться задачи обучения, обобщения, в частности, с целью формирования новых продукций (обучение по примерам), уменьшения их числа. На этом уровне может прослеживаться динамика, происходящие изменения. Для

этого в § 2.1 были введены составные продукции. На основе них система всегда сможет понять, что же на самом деле происходит, если, к примеру, ей сообщается *ТРИГГЕР ПЕРЕШЕЛ В ДРУГОЕ СОСТОЯНИЕ*, какие изменения за этим стоят. Собственно принципам унифицированной обработки на уровне структур и была посвящена настоящая работа.

* * *

В заключение следует отметить, что изложенный в книге подход использован в ВЦ ДВНЦ АН СССР при построении модельных образцов некоторых классов интеллектуальных систем. На основе описанных ранее семантических сетей в ИПИ АН СССР предполагается создание специальных баз знаний с их последующим использованием для поддержки так называемой беспрограммной технологии конструирования пользовательских систем, т. е. только за счет сведений, поступающих от специалистов-непрограммистов.

Хотелось бы обратить внимание читателя и на научную перспективу предложенного в книге подхода. Обычно созданию теорий предшествует построение конструктивных образцов. Собственно проекту одного из возможных образцов интеллектуальных автоматов и посвящена данная книга. Было показано, что в рамках данного образца удается найти место принципам обработки, характерным для математических построений, и проследить некоторые особенности, связанные с проявлениями человеческого интеллекта. Представляется, что именно здесь заложен ключ к более глубокому пониманию ряда дисциплин, в том числе информатики. Ведь все, что касается обработки информации, вначале осмысливается человеком, после чего переносится на различного рода устройства, конструктивные изделия.

ЛИТЕРАТУРА

1. *Амосов Н. Н.* и др. Автоматы и разумное поведение. Киев: Наук. думка, 1973. 214 с.
2. *Андерсон Д. Б., Хейз П. Ф.* Недостатки логики. М.: Мир, 1976. с. 96—100. (Кибернет. сб. Н. С.; № 13).
3. *Апресян Ю. Д.* Экспериментальное исследование семантики русского глагола. М.: Наука, 1967. 251 с.
4. *Бауэр Ф. Л., Гоос Г.* Информатика. М.: Мир, 1976, 484 с.
5. *Бурбаки Н.* Теория множеств. М.: Мир, 1965. 455 с.
6. *Вудс В. А.* Сетевые грамматики для анализа естественного языка. М.: Мир, 1976. с. 101—118. (Кибернет. сб. Н. С.; №13).
7. *Гладун В. П.* Эвристический поиск в сложных средах. Киев: Наук. думка, 1977. 127 с.
8. *Глушков В. М.* Введение в кибернетику. Киев: Изд-во АН УССР, 1964. 324 с.
9. *Дрейфус Х.* Чего не могут вычислительные машины. М.: Прогресс, 1978. 333 с.
10. *Дэйт К.* Введение в системы баз данных. М.: Наука, 1980. 463 с.
11. *Ефимов Е. Н.* Решатели интеллектуальных задач. М.: Наука, 1982. 312 с.
12. *Загоруйко Н. Г.* Методы обнаружения закономерностей. М.: Знание, 1981. 62 с.
13. *Заде Л.* Понятие лингвистической переменной и его применение. М.: Мир, 1976. 165 с.
14. *Золотов Е. В., Кузнецов И. П.* Расширяющиеся системы активного диалога. М.: Наука, 1982. 309 с.
15. *Камилов М. М., Журавлев Ю. И.* и др. Алгоритмы вычисления оценок и их применение. Ташкент: Фан, 1974. 120 с.
16. *Кейслер Г., Чен Ч.* Теория моделей. М.: Мир, 1977. 306 с.
17. Кибернетические диалоговые системы: Отчет по теме. Хабаровск, 1975. № гос. рег. 75.002.640, инв. № БИ 72240.
18. *Клини С.* Математическая логика. М.: Мир, 1973. 480 с.
19. *Клыков Ю. И.* Ситуационное управление большими системами. М.: Энергия, 1974. 136 с.
20. *Кобринский Н. Е., Трахтенброт Б. А.* Введение в теорию конечных автоматов. М.: Физматгиз, 1962. 404 с.
21. *Колмогоров А. Н., Драгалин А. Г.* Введение в математическую логику. М.: Изд-во МГУ, 1982, 119 с.
22. Краткий толковый словарь русского языка: (Для иностранцев)/Под ред. В. В. Розановой. М.: Русский язык, 1978.
23. *Кузнецов И. П.* Кибернетические диалоговые системы. М.: Наука, 1976. 300 с.
24. *Кузнецов И. П.* Механизмы обработки семантической информации. М.: Наука, 1978. 176 с.
25. *Кузнецов И. П.* Семантические сети с вершинами связи в задачах обработки структурных описаний: Препр.: ВЦ ДВНЦ АН СССР, № 14, 1983. 27 с.
26. *Кузнецов И. П., Лященко А. Н.* Об одном алгоритме конкретизации.— В кн.: Обеспечение системы ИИ. М.: МДНТП, 1978, с. 68—73.
27. *Кузнецов И. П.* О принципах работы одного класса активных диплоговых систем.— НТИ. Сер. 2, Информ. процессы и системы, 1976, № 1, с. 32—37.
28. *Кулагина О. С.* Исследования по машинному переводу. М.: Наука, 1979. 360 с.
29. *Линдсей П., Норман Д.* Переработка информации у человека. М.: Мир, 1974. 550 с.
30. *Ловицкий В. А.* Принципы построения и использования понятий в системах ИИ. Харьков: ХПИ, 1980. 103 с.
31. *Лозовский В. С.* Ситуационная и дефиниторная семантика системы представления знаний. Кибернетика, Киев, 1979, № 2, с. 98—102.
32. *Мартин Дж.* Организация без данных в вычислительных системах. М.: Мир, 1978. 615 с.

33. *Мельчук И. А.* Опыт теории лингвистических моделей смысл — текст. М.: Наука, 1974. 287 с.
34. *Мендельсон Э.* Введение в математическую логику. М.: Наука, 1976. 320 с.
35. *Минский М.* Фреймы для представления знаний. М.: Энергия, 1979. 151 с.
36. *Нариньяни А. С.* Недоопределенные модели и операции с недоопределенными значениями. Препр. Новосибирск: ВЦ СО АН СССР, № 400, 1982. 67 с.
37. *Невинс А.* Доказательство теорем планиметрии с использованием прямых рассуждений. М.: Мир, 1979. с. 390. (Кибернет. сб. Н. С.; № 16).
38. *Оре О.* Теория графов. М.: Наука, 1980. 336 с.
39. *Попов Э. В.* Общение с ЭВМ на естественном языке. М.: Наука, 1982. 360 с.
40. *Попов Э. В., Фридман Г. Ф.* Алгоритмические основы интеллектуальных роботов и искусственного интеллекта. М.: Наука, 1976. 455 с.
41. *Поспелов Д. А.* Логико-лингвистические модели в системах управления. М.: Энергоиздат, 1981. 231 с.
42. *Поспелов Д. А., Пушкин В. Н.* Мышление и автоматы. М.: Сов. радио, 1972. 224 с.
43. *Пойа Д.* Математика и правдоподобные рассуждения. М.: Изд-во иностр. лит., 1957. 535 с.
44. *Ту Дж., Гонсалес Р.* Принципы распознавания образов. М.: Мир, 1978. 410 с.
45. *Тыугу Э. Х.* Решатель вычислительных задач.— Вычисл. матем. и вычисл. физика, 1971, № 4, с. 992—1004.
46. *Тыугу Э. Х.* Решение задач на вычислительных моделях.— Вычисл. математика и мат. физика, 1970, т. 10, № 3, с. 716—733.
47. *Ульман Дж.* Основы систем баз данных. М.: Финансы и статистика, 1983. 335 с.
48. *Фу К.* Лингвистический подход к распознаванию образов.— В кн.: Классификация и кластер. М.: Мир, 1980, с. 208—247.
49. *Фу К.* Структурные методы в распознавании образов. М.: Мир, 1977. 320 с.
50. *Хендрикс Г.* О расширении применимости семантических сетей введением разбиений.— Тр. IV Междунар. конф. по искусственному интеллекту Т. 1, 1975, с. 190—206.
51. *Чень Ч., Ли Р.* Математическая логика и автоматическое доказательство теорем. М.: Наука, 1983. 224 с.
52. *Чёрч А.* Введение в математическую логику. М.: Изд-во иностр. лит., 1960. 484 с.
53. *Шенк Р.* Обработка концептуальной информации. М.: Энергия, 1980. 360 с.
54. *Шиханович Ю. А.* Введение в современную математику. М.: Наука, 1965. 376 с.
55. *Шрейдер Ю. А.* Равенство, сходство, порядок. М.: Наука, 1971. 250 с.
56. *Шуберт Л. К.* Расширение изобразительной мощности семантических сетей.— Тр. IV Междунар. конф. по искусственному интеллекту. Т. 2, 1975, с. 211—225.
57. *Brachman R. J.* On the Epistemological status of Semantic Networks.— In: Associative network, N. J., 1979, p. 3—50.
58. *Cercone N. J., Schubert L. K.* Toward a state based conceptual representation.— Proc. of 4th International Conf. of Art. Int., 1975, p. 83—90.
59. *Colmerauer A.* PROLOG in 10 figures.— In: Proc. of the Eighth Internatione Conference of Art Int. Karlsruhe, 1983, vol. 3.
60. *Crosz B. I.* Duscouse Analysis.— In: Speech Understanding Research Stanford: Research Institute, 1976.
61. *Hyvonen E.* Graph grammar approach to natural language parsing and understanding. Proceeding of the Eighth International Conference of Art. Int. Karlsruhe, 1983, vol. 2, p. 671—674.
62. *Rieger C.* An Organization of Knowleage for Problem Solving and Language Comprehension.— Art Int., 1976, N 7.
63. *Schubert L. K.* Semantic Networks — Art. Int., 1976, N 2; 8.
64. *Schubert L. K., Goebel R. G., Cercone N. T.* The structure and organization of a semantic Net Associative network. N. J., 1979, p. 121—175.
65. *Vere S. A.* Relational Production Systems.— Art. Int., 1977, N 1; N 8.

ОГЛАВЛЕНИЕ