Глава 2

Представление знаний

2.1 Введение

Настоящая и следующая главы посвящены **логике**. Большинство людей считают, что слово “логичный” означает “обоснованный”. Таким образом, если человек рассуждает логично, то его рассуждения обоснованны, поэтому он не допускает поспешных выводов. Но прежде чем перейти к изучению искусственного интеллекта и экспертных систем на основе логики, необходимо дать более строгое определение терминов. Все, кто использует компьютеры, знают, что преимущество компьютеров состоит в том, что они точно выполняют данные им задания. Но и недостаток компьютеров заключается в том, что они действуют в точном соответствии с инструкциями. В настоящее время экспертные системы применяются для оценки кредитоспособности; определяют, должна ли быть проведена ревизия компании налоговым управлением; обеспечивают бесперебойное функционирование атомных электростанций; а также являются ключевыми элементами другой столь же важной деятельности, поэтому необходимо обеспечить, чтобы эти системы действовали чрезвычайно логично и не были подвержены двусмысленным толкованиям. С формальной точки зрения логика — это наука о формировании действительных логических выводов. Это означает, что при наличии необходимого количества истинных фактов вывод всегда должен быть истинным. С другой стороны, если логический вывод недействителен, это означает, что на основании истинных фактов получено ложное заключение (но не стоит пытаться это доказать инспектору дорожной службы, который остановит вас за превышение скорости).

Необходимо также четко провести различие между **формальной логикой** и **неформальной логикой**. Отличительная особенность неформальной логики состоит в том, что ею пользуются в обычной жизни и особенно в адвокатской практике при попытке выиграть в словесном состязании, например, в судебном

разбирательстве. В последнем случае такое состязание обычно не принимает вид ожесточенного обмена репликами, который заканчивается перестрелкой, а выглядит как юридическое доказательство в зале суда, в ходе которого слова, несущие эмоциональную нагрузку, используются для того, чтобы убедить присяжных в том, какая из противоборствующих сторон имеет лучшего адвоката, и тем самым выиграть дело. Сложное логическое доказательство представляет собой цепь логических выводов, в которой одно заключение ведет к другому, и т.д. В суде построение такой цепи может привести к одному из нескольких заключений, которое становится основой судебного решения о виновности, невиновности, невменяемости или необходимости дополнительного расследования, после чего объявляется приговор.

В **формальной логике**, называемой также **символической логикой**, исключительную важность имеет то, как осуществляется логический вывод и как учитываются другие факторы, которые обеспечивают доказательство истинности или ложности окончательного заключения допустимым способом. Идеальным примером компьютерной программы, осуществляющей недействительный символический логический вывод, может служить программа, содержащая программную ошибку. (К счастью, пользователи, потратившие сотни долларов на обновление операционной системы своего компьютера с переходом на новейшую версию, всегда могут получить немедленное удовлетворение, отправив изготовителю ОС отчет об ошибке.) Логика нуждается также в **семантике**, позволяющей придать смысл символам. В формальной логике используется семантика, не основанная на применении слов, несущих эмоциональную нагрузку, как в следующем примере: “Вы предпочитаете пепси-колу или кока-колу?” Вместо этого область применения семантики в формальной логике ограничивается направлением, подобным выбору осмысленных имен для переменных в обычном программировании.

В настоящей главе приведено вводное описание логики, а также даны основные сведения о наиболее широко используемых способах представления знаний. Тематика представления знаний (Knowledge Representation — KR) уже давно считается одним из основных направлений работ в области искусственного интеллекта, поскольку выбор правильного способа представления знаний является не менее значимым фактором, от которого зависит успешное создание системы, чем разработка самого́ программного обеспечения, в котором используются эти знания. С тематикой представления знаний тесно связана не менее важная тематика представления данных, которая рассматривается в такой области компьютерных наук, как проектирование баз данных. Безусловно, базы данных в основном рассматриваются как репозитарии текущих данных, таких как данные инвентарного учета товарно-материальных запасов на складах, данные о кредиторской задолженности, дебиторской задолженности и т.д., а не знаний, но в настоящее время многие компании проводят активную деятельность в направлении **анализа скрытых закономерностей в данных** для извлечения знаний.

Анализ скрытых закономерностей в данных направлен на использование **архивных данных**, находящихся в **хранилищах данных**, для предсказания будущих тенденций. Например, компания может заняться изучением своих данных о сбыте в последние пять лет за декабрь месяц для прогнозирования того, какие товары и в каком количестве следует запасти на складах. Например, специалисты компании с помощью анализа скрытых закономерностей в данных могут обнаружить, что в декабре хорошо продаются рождественские открытки, а поздравления ко дню Святого Валентина мало кого интересуют. Безусловно, этот пример немного надуман, а несколько более реалистичным примером может служить обнаружение того, что одежда в красных и зеленых тонах лучше продается зимой, а не весной (поскольку красный и зеленый цвета ассоциируются с Рождеством), а одежда в коричневых, оранжевых и желтых тонах находит более активный сбыт осенью. Несомненно, об этом догадываются и администраторы магазинов, но анализ скрытых закономерностей в данных может использоваться для получения количественных оценок того, сколько предметов одежды должна закупить торговая компания и когда следует объявить их распродажу в связи с окончанием сезона. Безусловно, истинная ценность анализа скрытых закономерностей в данных состоит в том, что он позволяет обнаружить тенденции, неочевидные для человека, но доступные обнаружению путем анализа огромных объемов исторических данных, которые хранятся в архиве компании. В процессе анализа скрытых закономерностей в данных применяются не только классические статистические методы, но и такие методы искусственного интеллекта, как искусственные нейронные системы, генетические алгоритмы, эволюционные алгоритмы и экспертные системы, не только отдельно взятые, но и в различных комбинациях [94].

Выбор правильных способов представления знаний имеет очень важное значение для экспертных систем по двум причинам. Первая причина состоит в том, что экспертные системы рассчитаны на использование представления знаний определенного типа, основанного на **правилах логики**, называемых способами **логического вывода**. Обычно под термином **умозаключение** подразумевается получение логических заключений на основании фактов. К сожалению, люди не очень хорошо справляются с указанной задачей, поскольку им свойственно путать семантику с самим процессом формирования рассуждений, поэтому им не всегда удается прийти к действительному заключению. Наглядные примеры подобных недостатков часто обнаруживаются при изучении предвыборных плакатов, применяемых политическими противниками. Основой логики, которая используется политиками, являются методы убеждения, не базирующиеся на фактах, а построенные на ненадежных сведениях или применяющие одни и те же факты для достижения полностью противоположных выводов.

Формирование логических выводов — это формальный термин, используемый для обозначения рассуждений специального типа, которые не опираются на семантику (т.е. в них не учитывается смысл слов). Разумеется, в реальном мире невоз-

можно обойтись без учета семантики, но экспертные системы проектируются для проведения рассуждений на основе логики, поэтому не должны подвергаться влиянию той эмоциональной окраски, которая может быть внесена в рассуждения под влиянием семантики. Цель логического вывода состоит в достижении действительного заключения на базе фактов с использованием доказательства в допустимой форме. Еще раз отметим, что в логике термин *доказательство* обозначает формальный способ, в котором применяются факты и правила логического вывода для обоснования действительного заключения. С другой стороны, просматривая телевизионное шоу, в котором тележурналист проводит оживленную дискуссию с человеком, представленным как отставной генерал, бывший дипломат, отстраненный от дел присяжный заседатель или другой подобный деятель, можно быть вполне уверенным в том, что целью этого шоу является не достижение действительных логических заключений, а привлечение интереса телезрителя.

Итак, логическое рассуждение — это формирование действительных логических выводов. Как известно, в реальном мире невозможно обойтись без здравого смысла и вероятностных рассуждений, но такие формы умственной деятельности связаны с неопределенностью, поскольку в действительности ни в чем нельзя быть уверенным на все 100%. Если кто-то скажет, что сейчас небо голубое, несомненно, немного позже оно может стать серым или даже, что еще больше расходится со сказанным, приобрести зеленый оттенок! Рассуждения в условиях неопределенности — очень важная тема, которая будет обсуждаться более подробно в главах 4 и 5.

Вторая причина, по которой представление знаний является важным, состоит в том, что от правильного выбора способа такого представления зависит весь ход разработки, а также эффективность, быстродействие и удобство сопровождения системы. В этом указанное положение полностью аналогично тому положению, которое складывается в обычном программировании, где выбор правильной структуры данных имеет принципиальную значимость при разработке программы. Качественный проект программы всегда начинается с правильного выбора способа представления данных, будь то простые именованные переменные, массивы, связные списки, очереди, деревья, графы, сети или даже такие автономные внешние базы данных, как Microsoft Access, SQL Server или Oracle. В языке CLIPS в качестве средств представления знаний могут использоваться правила, конструкции `deftemplate`, объекты и факты.

В следующей главе будет описано то, как на основе знаний формируются логические выводы в целях выработки действительных заключений и каких распространенных логических ошибок следует избегать. *Логической ошибкой* называется умозаключение, которое на первый взгляд кажется логически обоснованным, но не является таковым. Соблюдение этих требований становится особенно важным в процессе приобретения знаний, в ходе которого проводится собеседование с экспертом-человеком, предоставляющим знания для разрабатываемой эксперт-

ной системы (речь об этом пойдет в главе 6). Прежде всего необходимо отделить истинные знания от семантической окраски, влияние которой может привести к недействительным заключениям. Но при этом не следует слишком много спорить с экспертом по знаниям, предъявлять ему невыполнимые требования, и тем более нельзя добиваться получения тех логических заключений, которые требуются по условиям задания, поскольку это равносильно неудачному завершению проекта!

Как показано в приложении Ж, с применением методов искусственного интеллекта для формирования рассуждений, доказательства теорем и даже обучения логике написано много компьютерных программ.

2.2 Смысл знаний

“Знания”, как и “любовь”, является одним из тех слов, смысл которого понимает каждый, но затрудняется найти ему определение. Кроме того, трудно найти двух таких людей, которые испытывали бы в любви одинаковые чувства. В действительности единственные проявления истинной любви, которые у всех людей выглядят почти одинаковыми, относятся к любимым кошкам или собакам (приносим свои извинения тем, кто любит змей и тарантулов). Как и любовь, знания имеют много толкований, которые зависят от взглядов того, кто наблюдает за их проявлениями. В качестве взаимозаменяемых со словом “знания” часто используются другие слова, такие как данные, факты и информация [83].

Как правило, в процессе поиска решения задач применяются логические рассуждения и опыт. Общим термином, которым обозначается использование опыта для решения некоторой задачи, является **эвристика**. Уже давно принято называть конкретные примеры эвристического опыта **рассуждениями на основе прецедентов**. Это — один из основных типов рассуждений, применяемых в юридической практике, медицине и автосервисе; с помощью таких рассуждений юристы, врачи и автомеханики пытаются найти решение задачи по данным о встречавшихся ранее аналогичных случаях, называемых **прецедентами** [66]. Безусловно, клиенты часто жалуются на то, что предложенное им решение на основе прецедента обходится слишком дорого, а это несправедливо, поскольку аналогичная задача уже была успешно решена ранее, но они должны представлять себе, как много приходится платить за то, чтобы создать новый прецедент в области медицины, юриспруденции или автосервиса!

Экспертная система может содержать сотни или тысячи небольших фрагментов знаний о прецедентах, на которые она опирается. Каждое правило в экспертной системе может рассматриваться как своего рода микропрецедент, который может способствовать решению задачи или использоваться в цепи логических выводов в надежде на то, что с его помощью удастся получить решение.

Наука о знаниях называется **эпистемологией**. В рамках этой науки рассматриваются характер, структура и происхождение знаний. Некоторые категории эпистемологии показаны на рис. 2.1. Кроме философских разновидностей знаний, сформулированных Аристотелем, Платоном, Декартом, Юмом, Кантом и другими учеными, на этом рисунке представлены две особые разновидности, называемые **априорными** и **апостериорными** знаниями. Латинский термин "a priori" означает предшествующий. Априорные знания предшествуют знаниям, полученным с помощью органов чувств, и не зависят от них. В частности, примерами априорных знаний является утверждение: "Все имеет свою причину" и "Сумма всех углов треугольника на евклидовой плоскости равна 180 градусам". Априорные знания рассматриваются как универсально истинные, и эти знания невозможно опровергнуть, не впадая в противоречия. К примерам априорных, неопровержимых знаний относятся логические утверждения, математические законы, а также знания, которыми владеют подростки.

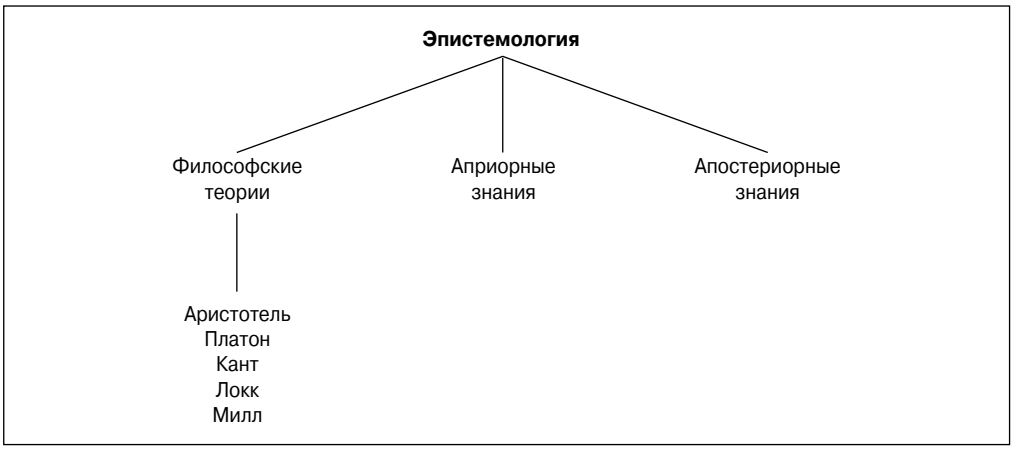

Рис. 2.1. Некоторые категории эпистемологии

Противоположными по отношению к априорным знаниям являются знания, полученные с помощью органов чувств, — апостериорные знания. Истинность или ложность апостериорных знаний, например, представленных с помощью утверждения: "На светофоре горит зеленый свет", может быть проверена на основании чувственного опыта. Но чувственный опыт не всегда может оказаться надежным, поэтому существует вероятность того, что апостериорные знания будут опровергнуты на основе новых знаний. Однако это не всегда приводит к противоречию. Например, встретив человека с карими глазами, можно поверить в то, что его глаза действительно карие. В дальнейшем, увидев, как этот человек снимает коричневые контактные линзы, после чего обнаруживаются его синие глаза, необходимо просто пересмотреть полученные ранее знания.

Знания могут дополнительно подразделяться на **процедурные**, **декларативные** и **неявные**. Типы процедурных и декларативных знаний соответствуют процедурному и декларативном подходам, которые рассматриваются в главе 1 и более подробно описаны в [10].

Процедурные знания часто называют знаниями о том, как сделать то или другое. К примерам процедурных знаний относятся знания о том, как вскипятить кастрюлю воды. Но человек, который считает, что его знаний, касающихся способов доведения воды до кипения на обычных высотах, будет достаточно, чтобы успешно вскипятить воду на любой высоте над уровнем моря, обнаруживает, что его попытка использовать знания, основанные на здравом смысле, оканчивается неудачей. Но истинной причиной этой неудачи становится не то, что рассуждения начинающего альпиниста основаны на здравом смысле, а то, что у него недостает практического опыта для проведения правильных рассуждений (см. приведенную в конце данной главы задачу, посвященную анализу процесса кипячения воды). Декларативными знаниями называют знания о том, является ли некоторое утверждение истинным или ложным. Термин *декларативный* применяется к знаниям, выраженным в форме декларативных утверждений, подобных следующему: “Не опускайте пальцы в кастрюлю с кипящей водой”. Как показано в [88], разработано много различных способов представления знаний с использованием логики.

Неявные знания иногда называют **подсознательными знаниями**, поскольку они не могут быть выражены с помощью языка. В качестве примера можно указать знания о том, как двинуть рукой. Давая наиболее общий ответ, можно было бы сказать, что движение рукой осуществляется за счет напряжения или расслабления определенных мускулов и сухожилий. Но в таком случае приходится рассматривать этот вопрос на более низком уровне и объяснять, откуда вы знаете, как следует напрягать или ослаблять свои мускулы и сухожилия. В качестве других примеров можно указать ходьбу или езду на велосипеде. А если речь идет о компьютерных системах, то знания, представленные в искусственной нейронной системе, напоминают неявные знания, поскольку обычно нейронная сеть неспособна непосредственно объяснить суть содержащихся в ней знаний, но могла бы приобрести такую способность при наличии соответствующей программы (см. раздел 1.14).

Знания играют в экспертных системах очень важную роль. В действительности можно провести аналогию с приведенным ниже классическим выражением, которое привел Николас Вирт (Nicholas Wirth) в своей оригинальной книге по языку Pascal.

$$\text{Алгоритмы} + \text{Структуры данных} = \text{Программы}$$

Применительно к экспертным системам это выражение выглядит следующим образом:

$$\text{Знания} + \text{Логический вывод} = \text{Экспертные системы}$$

Согласно определению знаний, которое используется в настоящей книге и иллюстрируется на рис. 2.2, знания входят в состав иерархии способов представления информации. На нижнем уровне этой иерархии находится шум, состоящий из информационных элементов, которые не представляют интереса и могут лишь затруднить восприятие и представление данных. На более высоком уровне находятся бесформатные данные, содержащие элементы данных, которые в принципе могут представлять определенный интерес. На следующем уровне находится информация, т.е. обработанные данные, явно представляющие интерес для пользователей. За этим уровнем следует уровень знаний, на котором представлена настолько важная информация, что ее следует надежно хранить и обеспечить выполнение над ней необходимых операций. В главе 1 знания в экспертной системе на основе правил определены как правила, которые активизируются фактами или другими правилами в целях выработки новых фактов или заключений. Заключение рассматривается как конечный продукт цепи рассуждений, называемой *логическим выводом*, но при условии, что эти рассуждения осуществляются в соответствии с формальными правилами. Процесс формирования логических выводов является существенной частью процесса функционирования экспертной системы. Сам термин **формирование логических выводов**, как правило, используется применительно к таким механическим системам, как экспертные системы. А термин *рассуждения* применяется применительно к продуктивным человеческим размышлениям.

Рис. 2.2. Пирамида знаний

Искусственная нейронная система не формирует логические выводы. Вместо этого такая система осуществляет поиск в целях обнаружения в данных основополагающих образов, которые не являются очевидными для человека. По существу искусственная нейронная система представляет собой классификатор образов. Например, способность человека к чтению основана на возможностях распознавания образов, заложенных в нейронной сети мозга, которая была обучена распознаванию образов букв. В другом участке мозга эти образы преобразуются в слова, которые человек мысленно слышит во время чтения, поскольку именно так про-

исходит обучение чтению детей — путем озвучивания слов, произносимых по буквам. В качестве эксперимента можно, например, попытаться перевернуть книгу или газету на 180 градусов и приступить к чтению. Большинство людей на первых порах испытывают затруднения при чтении перевернутого текста, но способны переобучить свою нейронную сеть, применяемую во время чтения, чтобы она распознавала буквы, представленные в необычном ракурсе. Еще в одном классическом психологическом эксперименте людям предлагали носить очки, которые переворачивают изображение, воспринимаемое глазами. Но через несколько дней мозг у таких людей адаптируется, и они начинают снова видеть мир так, как будто его изображение не перевернуто. А в действительности хрусталики в глазах человека проектирует изображение на сетчатку в перевернутом виде, а мозг снова его переворачивает, поэтому мир воспринимается в нормальном виде.

Еще с одним примером гибкости нейронных сетей можно ознакомиться, попытавшись приступить к чтению книги, повернутой на какой-то угол, скажем, на 30° [84]. При этом человек не теряет способности к чтению, просто чтение происходит немного медленнее. Эта задача становится сложнее после поворота книги на 180°. А после определенной практики некоторые люди приобретают способность читать перевернутый текст так же быстро, как и обычный, что показывает поразительную приспособляемость нейронных сетей человека. Аналогичным образом искусственная нейронная система, обученная распознаванию букв, расположенных под различными углами, приобретает способность к распознаванию и чтению текста, который демонстрируется в условиях, отличающихся от обычных. Чем больше число различных вариантов поворота, на примере которых была обучена сеть, тем быстрее и точнее она будет работать. Кроме того, обучение искусственной нейронной системы чтению различных стилей почерка позволяет использовать эту систему для чтения рукописных текстов, оформленных с помощью еще более разнообразных стилей, по аналогии с тем, как люди читают рукописи различных авторов.

Термином **факты** обозначается информация, рассматриваемая как надежная. Экспертные системы формируют логические выводы с использованием фактов. Факты, ложность которых будет продемонстрирована в дальнейшем, могут быть исключены с помощью средств поддержания истинности системы CLIPS; в ходе этого автоматически изымаются все выводы, правила и другие факты, сформированные на основании ложного факта. Кроме того, экспертные системы могут выполнять следующие действия: во-первых, отделять данные от шума, во-вторых, преобразовывать данные в информацию и, в-третьих, преобразовывать информацию в знания. В экспертной системе, которая рассчитана на получение фактов, чрезвычайно опасно использовать бесформатные данные, поскольку надежность полученных в результате заключений может оказаться полностью неприемлемой. Безусловно, и в экспертных системах оправдывается поговорка: “Мусор на входе — мусор на выходе”, которой руководствуются программисты. Но, безусловно,

кто-то может сознательно принять решение о применении информационного мусора для поддержки конкретного рабочего списка правил.

В качестве примера применения представленных выше понятий рассмотрим следующую последовательность из 24 цифр:

137178766832525156430015

При отсутствии знаний об этой последовательности она может показаться просто проявлением шума. Но если есть основания полагать, что эта последовательность имеет смысл, или это достоверно известно, то указанная последовательность рассматривается как данные. При определении того, что́ является данными и что́ является шумом, вполне можно руководствоваться старой рекомендацией, касающейся сельского хозяйства: “Сорняком следует считать все, что выросло вопреки вашему желанию”.

Определенные знания могут относиться к тому, как нужно преобразовывать данные в информацию. Например, следующий алгоритм показывает, как обработать приведенные выше данные для получения информации:

```
Разбить представленные цифры на пары.
Игнорировать те из полученных двухзначных чисел, которые
   меньше 32.
Подставить вместо оставшихся двухзначных чисел символы
   кода ASCII.
```

Применение этого алгоритма к приведенным выше 24 цифрам приводит к получению следующей информации:

```
GOLD 438+
```

После этого к полученной информации можно применить знания. Например, допустим, что существует такое правило:

```
IF цена на золото меньше 500
   и цена растет (+)
THEN
   покупать золото
```

Очевидно, что на рис. 2.2 это явно не показано, но **экспертные знания** представляют собой специализированную разновидность знаний и навыков, которыми обладают эксперты. Экспертные знания могут относиться к показанным на этом рисунке уровням знаний, метазнаний и мудрости. Хотя весьма специализированные знания можно найти в таких общедоступных источниках информации, как книги и статьи, недостаточно просто прочитать книгу, чтобы стать экспертом. Например, в медицинских учебниках можно найти подробную информацию о том, как следует проводить хирургические операции. Но вряд ли найдется такой человек, который согласится подвергнуться хирургической операции на головном

мозге, если кто-то постучится в его дверь и заявит, что окончил заочные курсы и готов предложить свои услуги по сниженным ценам. На это вряд ли может даже соблазнить бесплатный набор ножей Ginsu-2000 (немного попорченный после проведения очередной хирургической операции на головном мозге).

Экспертные знания — это неявные знания и навыки эксперта, которые должны быть извлечены и преобразованы в явные с тем, чтобы их можно было представить в экспертной системе. Причина, по которой знания являются неявными, состоит в том, что истинный эксперт владеет этими знаниями настолько хорошо, что они превратились в его вторую натуру и не требуют размышлений. В качестве примера можно указать, что после окончания медицинского училища практиканты служат в лечебном учреждении примерно один год, работая, как правило, 80 или больше часов в неделю, до тех пор, пока не приобретают способность выполнять медицинские процедуры даже без размышлений. Разумеется, такая организация обучения специалистов подвергалась критике, но она позволяет усваивать знания настолько глубоко, что они становятся второй натурой. (Безусловно, пациентам иногда перед проведением очередной процедуры не мешает также спросить практиканта, как долго ему удалось поспать на этой неделе.) Почти на самом верхнем уровне иерархии (см. рис. 2.3) над уровнем знаний находится уровень **метазнаний**. Префикс **мета** означает “свыше” или “дальше”.

Метазнания представляют собой знания об обычных и экспертных знаниях. Безусловно, экспертная система может быть спроектирована с учетом знаний о нескольких различных проблемных областях, но, как правило, это нежелательно, поскольку в результате система становится менее качественно определенной. Опыт показывает, что наиболее успешно работают такие экспертные системы, применение которых ограничивается наименьшей проблемной областью из всех возможных. Например, если экспертная система спроектирована для выявления заболеваний, вызванных бактериями, то нет смысла применять ее также для диагностирования неисправностей в автомобилях. В качестве практического примера можно указать, что сами врачи специализируются только в одной небольшой области, а не во всей медицине. Даже семейные врачи (в качестве которых чаще всего выступают терапевты) направляют своих пациентов в случае необходимости к соответствующему специалисту.

В экспертных системах *онтология* представляет собой метазнания, которые описывают все, что известно о рассматриваемой предметной области. В идеальном случае онтология должна быть представлена в формальном виде для того, чтобы можно было легко обнаруживать несовместимости и несоответствия. Для построения онтологий может применяться целый ряд бесплатных и коммерческих инструментальных средств. Построение онтологии должно быть закончено до реализации экспертной системы, поскольку в противном случае может потребоваться пересматривать правила по мере поступления дополнительной информации о данной предметной области, что приводит к повышению издержек, увеличению

продолжительности разработки и возрастанию вероятности появления программных ошибок.

Например, экспертная система может иметь базы знаний о ремонте легковых автомобилей компании GM (General Motors), джипов (Sport Utility Vehicle — SUV) GM и дизельных грузовиков GM. В зависимости от того, к какому типу относится автомобиль, требующий ремонта, должна использоваться соответствующая база знаний. В связи с необходимостью снижения потребности в памяти и повышения быстродействия было бы неэффективно держать в памяти одновременно все базы знаний, поскольку в процессе эксплуатации rete-сети непрерывно происходит модификация в памяти всех правил, находящихся в сети. Кроме того, могут возникать конфликты, если антецеденты какого-либо правила для грузовиков и легковых автомобилей содержат одинаковый шаблон, а заключения являются различными. Например, если датчик измерения уровня топлива показывает, что бак пуст, то экспертная система для легковых автомобилей может дать указание: "Заполнить бак бензином", а экспертная система для грузовиков — указание: "Заполнить бак дизельным топливом". Такая путаница, в результате которой бензобак легкового автомобиля будет заполнен дизельным топливом или бак грузовика — бензином, весьма нежелательна. Кроме того, работа экспертной системы замедляется при возрастании количества правил в системе, поскольку rete-сеть становится больше. А метазнания могут использоваться для принятия решения о том, какая база знаний должна быть загружена в память, а также служить в качестве общего руководства по проектированию и сопровождению самой экспертной системы и ее онтологии.

Наконец, вершиной всех знаний является **мудрость**, рассматриваемая в ее философском толковании. Мудрость — это метазнания, позволяющие определять наилучшие цели в жизни и находить пути их достижения. Например, одно из правил мудрости можно выразить следующим образом:

```
IF мне удастся заработать достаточно денег для того, чтобы
   моя семья ни в чем не нуждалась
THEN я уволюсь с работы и буду наслаждаться жизнью
```

Объем работ по искусственному интеллекту на основе инженерии знаний, достигающих уровня мудрости, постоянно возрастает. Однако человечество и так сформулировало чрезвычайно много мудрых мыслей, но прислушиваются к ним лишь единицы. В настоящей книге мы ограничимся рассмотрением систем, основанных на знаниях, и оставим проблематику создания систем на основе мудрости для политических деятелей и других экспертов того же рода.

2.3 Продукции

До настоящего времени предложен целый ряд различных методов представления знаний. К ним относятся правила, семантические сети, фреймы, сценарии, логика, концептуальные схемы и др. В частности, было предложено много языков представления знаний, таких как классический язык KL-ONE (Knowledge Language ONE — язык знаний номер один) и происходящий от него язык на основе фреймов CLASSIC [8]. Кроме того, были предложены многие другие языки, включая языки на основе визуальных средств.

Как было описано в главе 1, в качестве основы базы знаний в экспертных системах широко используются правила, поскольку преимущества правил намного перевешивают их недостатки.

Одной из формальных систем обозначений, применяемых для определения продукций, является нормальная форма Бэкуса–Наура (Backus-Naur form — BNF). Эта система обозначений представляет собой **метаязык**, применимый для определения **синтаксиса** любого языка. Синтаксис определяет форму, а **семантика** обозначает смысл. Метаязык — это язык, предназначенный для описания других языков. Поскольку префикс "мета" обозначает "свыше" или "дальше", метаязык находится на более высоком уровне по сравнению с обычным языком.

Языки подразделяются на несколько типов, таких как естественные языки, логические языки, языки математики, компьютерные языки и т.д. Ниже приведено продукционное правило, в котором система обозначений на основе нормальной формы Бэкуса–Наура используется для формулировки простого правила английского языка, согласно которому предложение (*sentence*) состоит из подлежащего (*subject*), сказуемого (*verb*), за которыми следует знак пунктуации, обозначающий конец предложения (*end-mark*).

```
<sentence>::= <subject> <verb> <end-mark>
```

В этом правиле **угловые скобки** (<>) и символ `::=` представляют собой символы метаязыка, а не определяемого языка. Символ `::=` означает "определено как" и представляет собой эквивалент стрелки (→), применяемой в нормальной форме Бэкуса–Наура. Как описано в главе 1, в продукционных правилах используется стрелка.

Термы, заключенные в угловые скобки, принято называть **нетерминальными символами** (nonterminal). Нетерминальный символ — это переменная, представляющая другой терм. В свою очередь, в качестве подобного "другого" терма может использоваться нетерминальный символ или **терминальный символ** (terminal). Терминальный символ не может быть заменен каким-либо иным символом и поэтому представляет собой константу.

Нетерминальный символ `<sentence>` является специальным, поскольку относится к числу **начальных символов**, применяемых для определения других

символов. В определениях языков программирования начальный символ обычно носит имя <program>. Ниже приведено продукционное правило, которое указывает, что предложение состоит из подлежащего, за которым следует сказуемое, а в конце находится маркер конца предложения.

```
<sentence> → <subject> <verb> <end-mark>
```

А следующие правила позволяют раскрывать значения нетерминальных символов, поскольку указывают терминальные символы, которыми они могут быть заменены. В этом метаязыке **вертикальная черта** означает "или".

```
<subject> → I | You | We
<verb> → left | came
<end-mark> → . | ? | !
```

Все возможные предложения языка, т.е. продукции, могут быть произведены путем последовательной замены каждого нетерминального символа соответствующими ему нетерминальными или терминальными символами, взятыми из правой части правил, до тех пор, пока не будут устранены все нетерминальные символы. Некоторые продукции рассматриваемого языка приведены ниже.

```
I left.
I left?
I left!
You left.
You left?
You left!
We left.
We left?
We left!
```

Ряд терминальных символов называется **строкой** символов языка. Если строка может быть получена из начального символа путем замены нетерминальных символов с применением определяющих их правил, то строка называется действительным **предложением**. Например, такие строки, как "We", "WeWe" и "leftcamecame", являются действительными строками языка, но не действительными предложениями.

Грамматикой называется полный набор продукционных правил, который однозначно определяет язык. Безусловно, приведенные выше правила определяют некоторую грамматику, но она является очень ограниченной, поскольку допускает производство слишком малого количества возможных продукций. Более сложная грамматика может, в частности, предусматривать использование прямых дополнений, как показано в следующих продукциях:

```
<sentence> → <subject> <verb> <object> <end-mark>
<object> → home | work | school
```

Хотя эта грамматика является действительной, она также слишком проста для практического использования. А ниже показана грамматика, в большей степени применимая на практике, в которой в целях упрощения был исключен маркер конца.

```
<sentence> → <subject phrase> <verb> <object phrase>
<subject phrase> → <determiner> <noun>
<object phrase> → <determiner> <adjective> <noun>
<determiner> → a | an | the | this | these | those
<noun> → man | eater
<verb> → is | was
<adjective> → dessert | heavy
```

Нетерминальный символ `<determiner>` используется для замены конкретного члена предложения. Используя эту грамматику, можно вырабатывать предложения, подобные приведенным ниже.

```
the man was a dessert eater
an eater was the heavy man
```

В качестве графического представления предложения, декомпонованного на все терминальные и нетерминальные символы, применявшиеся для вывода этого предложения, используется **дерево синтаксического анализа**, или **дерево вывода**. На рис. 2.3 показано дерево синтаксического анализа для предложения "`the man was a heavy eater`" (этот человек любил поесть). Это предложение является действительным, а строка "`man was a heavy eater`" не представляет собой действительное предложение, поскольку в ней отсутствует определяющее слово (`determiner`) в группе подлежащего (`subject phrase`). Предпринимая попытку определить, соответствует ли оператор программы действительному синтаксису языка, компилятор создает дерево синтаксического анализа.

Дерево, приведенное на рис. 2.3, показывает, что предложение "`the man was a heavy eater`" может быть получено из начального символа путем применения подходящих продукционных правил. Ниже показаны этапы процесса создания рассматриваемого предложения; **двойные стрелки** (=>) указывают на результаты применения продукционных правил.

```
<sentence> => <subject phrase> <verb> <object phrase>
<subject phrase> => <determiner> <noun>
<determiner> => the
<noun> => man
<verb> => was
<object phrase> => <determiner> <adjective> <noun>
<determiner> => a
<adjective> => heavy
<noun> => eater
```

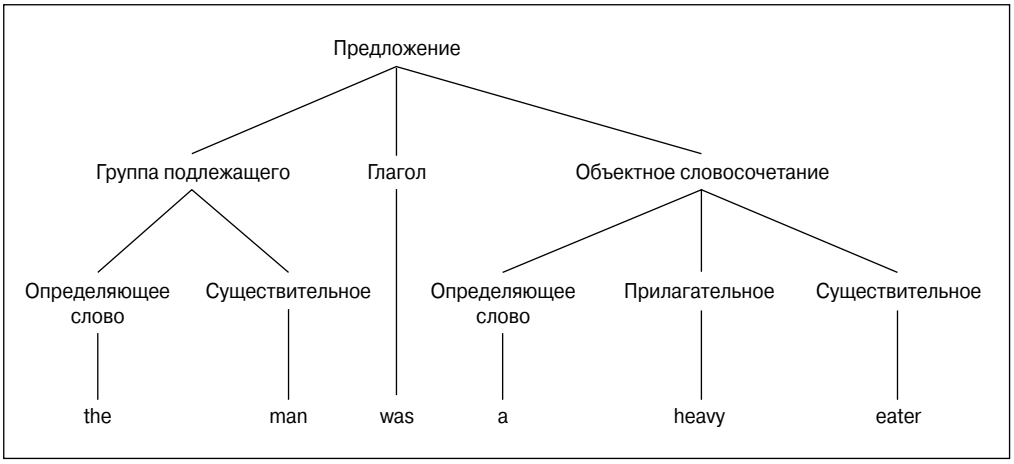

Рис. 2.3. Дерево синтаксического анализа предложения

Альтернативный способ использования продукционных правил состоит в формировании действительных предложений путем подстановки всех допустимых терминальных символов вместо нетерминальных символов, как было описано выше. Безусловно, смысл имеют не все создаваемые при этом продукции, в частности "`the man was the dessert`" (разумеется, речь не идет о том, что каннибалы нашли бы смысл в этом предложении).

Для распознавания структуры предложения очень хорошо подходят конечные автоматы (Finite State Machine — FSM). Например, конечные автоматы используются в компиляторах, которые транслируют исходный код компьютерных языков для **синтаксического анализа** и разбиения исходного кода на меньшие смысловые единицы, называемые **лексемами**. В таких приложениях, как компиляторы, транслирующие исходный код в код на языке ассемблера, и программы, применяемые для точного распознавания речи, невозможно обойтись без синтаксического анализа и использования конечных автоматов. А со времени появления языка Java термин **компилятор** приобрел более широкое значение и стал обозначать средство трансляции исходного кода не только в код ассемблера, но и в независимый от платформы байт-код (эти функции выполняет компилятор javac). Впервые такая возможность появилась с введением в действие программного обеспечения языка Pascal в 1970-х годах, поскольку байт-код этого языка мог эксплуатироваться на любом микропроцессоре. Программное обеспечение, позволяющее преобразовывать код на языке Pascal в байт-код, правильнее считать интерпретатором, а не компилятором, поскольку этот байт-код не содержит команд языка конкретного компьютера, подобных командам языка ассемблера. Но интерпретаторы обладают меньшим быстродействием по сравнению с компиляторами, поэтому разработчи-

ки языка Pascal решили, что лучшей рекламой для программного обеспечения служит его обозначение как компилятора, а не интерпретатора.

С оперативной демонстрационной версией конечного автомата, предложенной компанией Xerox, можно ознакомиться по адресу `http://www.xrce.xerox.com/competencies/content-analysis/fsCompiler/fsinput.html`. С примерами таких конечных автоматов, как автомат по продаже безалкогольных напитков, можно ознакомиться, перейдя по ссылкам, которые представлены по адресу `http://www.xrce.xerox.com/competencies/content-analysis/fsCompiler/fsexamples.html`. Практически исчерпывающий справочник по конечным автоматам приведен по адресу `http://odur.let.rug.nl/alfa/fsa_stuff/`.

Безусловно, конечные автоматы успешно действуют применительно к грамматике с сокращенным множеством символов, например, если в число этих символов входят цифры от `0` до `9` или буквы алфавита, но при их использовании в таких областях, как распознавание речи, где могут обнаруживаться неоднозначности, возникают проблемы. Например, рассмотрим следующие два предложения:

```
(1) No one has let us read
    и
(2) No one has lettuce red
```

В предложении (1) группа людей жалуется, что им не дают возможности читать, а в предложении (2) говорится о том, что ни у кого нет красного салата. В одном из удобных способов, позволяющих однозначно отличить друг от друга такие похожие слова и словосочетания, как “`lettuce`” и “`let us`”, “`read`” и “`red`”, используется **скрытая марковская машина** (Hidden Markov Machine — HMM), в которой вероятности присваиваются действиям, осуществимым в конечном автомате. Скрытая марковская машина позволяет рассмотреть всю структуру предложения или проанализировать дополнительные предложения, чтобы выявить правильный контекст (либо прочитав всю книгу, в которой встретилось предложение, либо проведя поиск названий овощей в каталогах овощных магазинов). Безусловно, экспертные системы не являются наиболее подходящим вариантом программного обеспечения, с помощью которого могла бы быть создана скрытая марковская машина, но они могут использоваться в качестве внешнего интерфейса для системы распознавания речи, для того чтобы в дальнейшем такая экспертная система, как CLIPS, могла применяться для инициирования соответствующих правил.

В действительности в языке CLIPS предусмотрена возможность легко связывать с помощью интерфейса код на языке C или C++, поэтому программное обеспечение скрытой марковской машины, написанное на C или C++, может вызываться полностью аналогично любой другой функции, определяемой ключевым словом языка CLIPS. Одной из сильных сторон языка CLIPS является то,

что этот язык — расширяемый, и пользователь может легко вводить новые ключевые слова на этапе компиляции для достижения наибольшей эффективности. Кроме того, благодаря наличию объектно-ориентированных средств языка COOL для расширения возможностей языка CLIPS с использованием всей мощи средств множественного наследования могут применяться объекты. Другое программное обеспечение, такое как искусственные нейронные системы, генетические алгоритмы и прочие программы, написанные на C или C++, может вводиться либо в левые части правил, для инициирования правил, либо в правые части правил, для обеспечения вывода. Из программы на языке CLIPS можно, например, обращаться к синтезатору речи, эффекторам и актюаторам робота, определив с помощью ключевых слов соответствующие функции. Исходный код языка CLIPS представлен в общее пользование, а это означает, что компиляция вызовов новых ключевых слов в программе CLIPS может осуществляться без потери быстродействия, которое обнаруживается при использовании других экспертных инструментальных средств, для которых исходный код не предоставлен.

2.4 Семантические сети

Семантические сети, или просто сети, — это классический способ представления пропозициональной информации, используемый в искусственным интеллекте (`http://www.pcai.com/web/T6110H2/R6.o1.h8/pcai.htm`), поэтому семантические сети иногда называют **пропозициональными сетями**. Как было описано выше, пропозициональным утверждением (или высказыванием) называется предложение, которое может быть либо истинным, либо ложным, такое как “все собаки — млекопитающие” и “треугольник имеет три стороны”. Высказывания имеют форму декларативных знаний, поскольку в них утверждаются факты. С точки зрения математики семантическая сеть представляет собой помеченный ориентированный граф. Высказывание всегда является либо истинным, либо ложным и называется **атомарным**, так как его истинностное значение не подлежит дальнейшей декомпозиции. Здесь термин *атомарный* применяется в классическом смысле, который применялся для обозначения неделимого объекта в трудах древнегреческих ученых. В отличие от этого, нечеткие утверждения, описанные в главе 5, не должны обязательно быть только истинными или только ложными.

Семантические сети впервые были разработаны для исследований в области искусственного интеллекта как способ описания человеческой памяти и языка Квиллианом (Quillian) в 1968 году. Квиллиан использовал семантические сети для анализа смысла слов в предложениях. (Обратите внимание на то, что выявление смысла предложения не сводится к синтаксическому анализу и разбиению предложения на лексемы и лексические структуры по такому принципу, который осуществляется с использованием конечных автоматов и скрытых марковских ма-

шин, рассматриваемых в предыдущем разделе.) С тех пор семантические сети успешно применялись для решения многих задач, связанных с представлением знаний. Понимание смысла, достигаемое с помощью семантических сетей, позволяет выйти за рамки возможностей программного обеспечения простых экспертных систем или искусственного интеллекта для устранения неоднозначности. В следующей главе, посвященной логическому выводу, будет показано, как экспертные системы выводят из фактов заключения, которые могут использоваться другими правилами в цепи логического вывода до тех пор, пока не будет достигнуто действительное заключение. А если не учитывается семантика, то работа экспертных систем может оканчиваться неудачей, как и размышления человека, сбитого с толку из-за обнаружившейся неоднозначности.

Структура семантической сети отображается графически с помощью **узлов** и соединяющих их **дуг**. Узлы иногда именуются **объектами** а дуги — **связями**, или **ребрами**.

Связи в семантической сети применяются для представления отношений, а узлы, как правило, служат для представления физических объектов, концепций или ситуаций. На рис. 2.4, *а* показана обычная сеть (фактически ориентированный граф), в которой связи обозначают авиационные маршруты, проложенные между городами. Узлы обозначены кружками, а связи — линиями, соединяющими узлы. Стрелки показывают направление, или ориентацию, самолетов, совершающих полеты (именно поэтому граф, на котором указаны направления, называется *ориентированным*). На рис. 2.4, *б* связи показывают отношения между членами некоторой семьи. Отношения имеют для семантических сетей исключительно важное значение, поскольку предоставляют базовую структуру для организации знаний. Знания, заданные без учета отношений, превращаются просто в коллекцию несвязанных фактов. А если заданы отношения, то знания приобретают связную структуру, исследование которой позволяет выводить логическим путем другие знания. Например, на основании рис. 2.4, *б* можно сделать вывод, что Анна и Билл — бабушка и дедушка Джона, даже несмотря на то, что на этом рисунке нет явно заданной связи, обозначенной как “`grandfather-of`”.

Семантические сети иногда называют **ассоциативными сетями**, поскольку одни узлы в таких сетях ассоциированы, или связаны, с другими. В действительности в оригинальной работе Квиллиана человеческая память моделировалась как ассоциативная сеть, в которой понятия были представлены в виде узлов, а связи показывали, как соединены эти понятия друг с другом. Согласно указанной модели, если происходит стимуляция одного узла в результате чтения слов в предложении, то происходит активизация связей этого узла с другими узлами, и такая активность распространяется по сети. Если же достаточную активизацию получает другой узел, то в обладающем сознанием разуме всплывает концепция, представленная этим узлом. Например, очевидно, что человек знает тысячи слов,

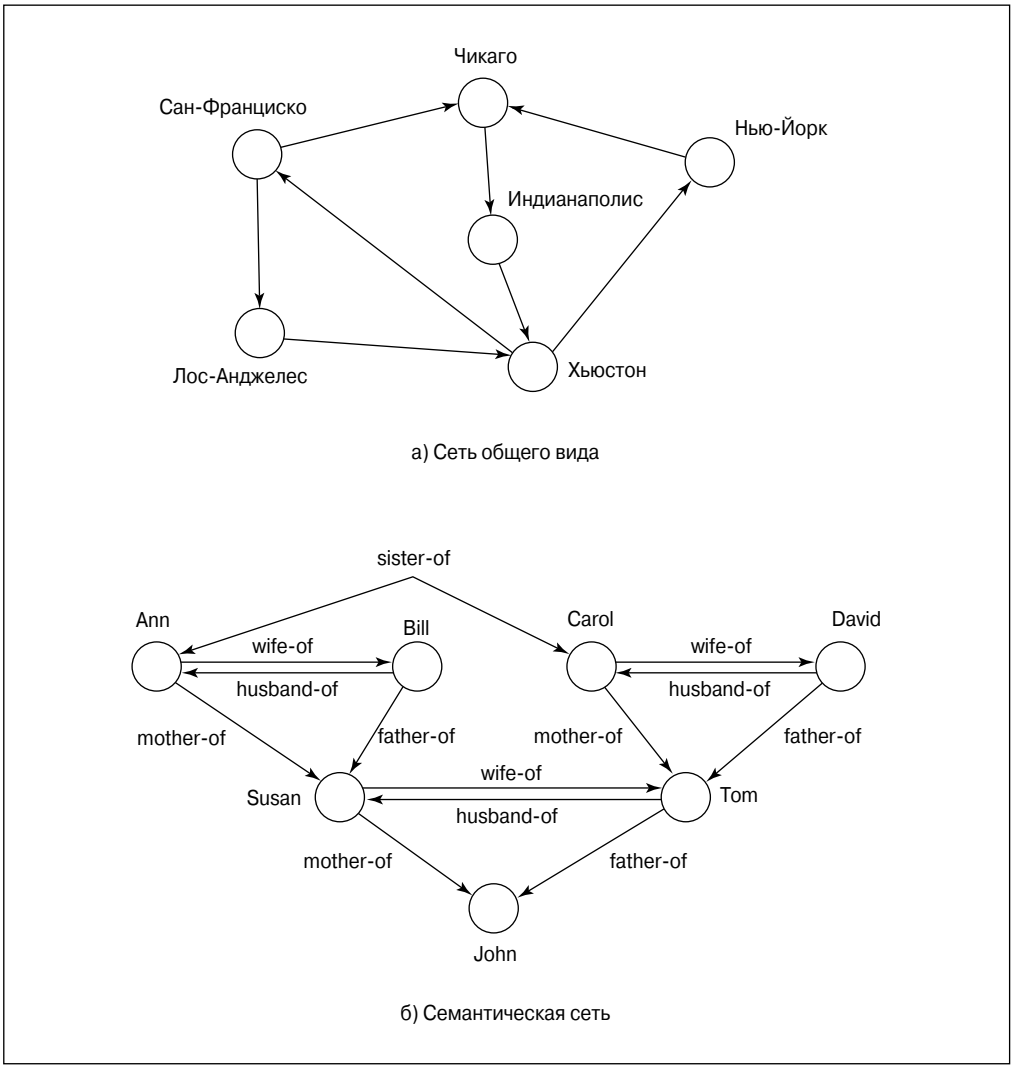

Рис. 2.4. Примеры двух типов сетей

но в его сознании отражаются только слова того конкретного предложения, которое он читает.

Как оказалось, во многих разных способах представления знаний особенно полезными являются отношения определенных типов. Поэтому при изучении различных задач принято использовать именно эти типы, а не определять каждый раз новые отношения. К тому же применение общепринятых типов позволяет другим людям проще понять, что описано в незнакомой для них сети.

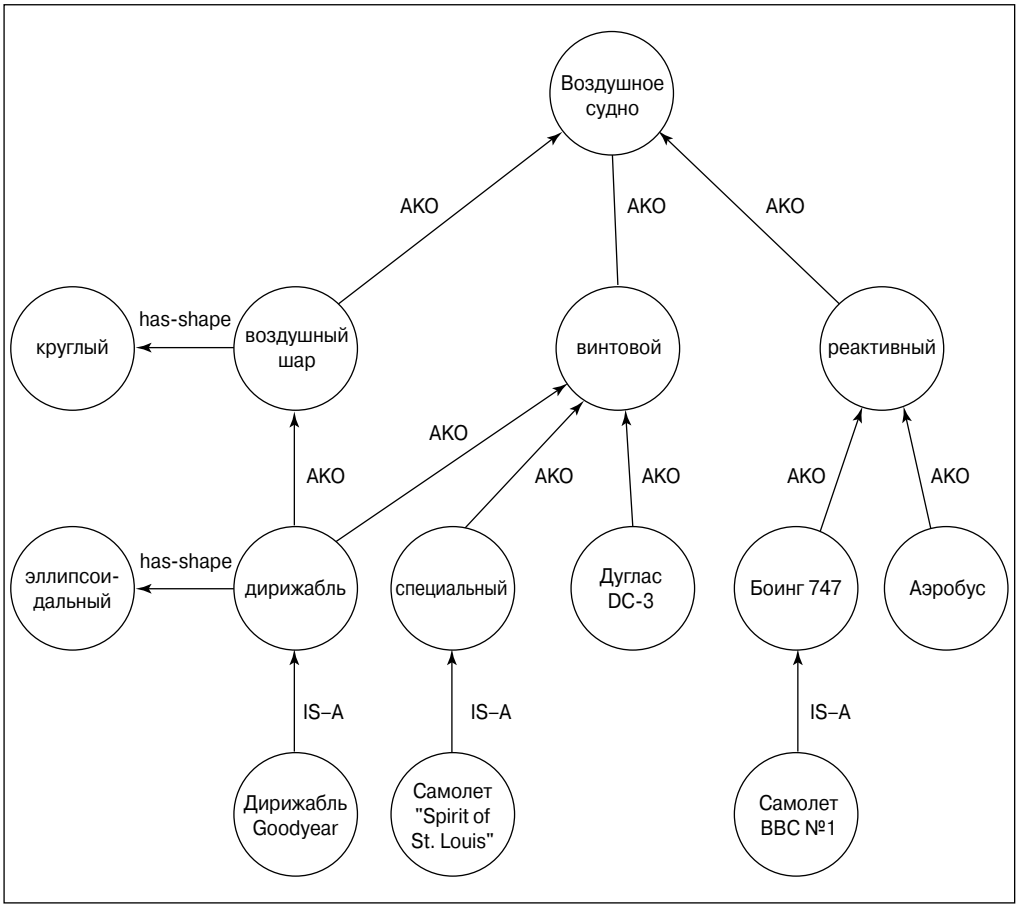

Рис. 2.5. Семантическая сеть со связями IS-A и A-KIND-OF (AKO)

К двум таким связям широко используемых типов относятся связи **IS-A** и **A-KIND-OF** (связь последнего типа иногда записывается как **AKO**). На рис. 2.5 показана семантическая сеть, в которой используются такие связи. На данном рисунке связь IS-A означает “является экземпляром” данного класса и указывает на конкретный экземпляр некоторого класса. В этом контексте понятие **класса** близко к математическому понятию множества, поскольку служит для обозначения группы объектов. Тем не менее множество может содержать элементы любого типа, а объекты в классе связаны определенными отношениями друг с другом, т.е. не могут быть взяты произвольно. Например, может быть определено множество, состоящее из следующих элементов:

$$\{3, \text{eggs}, \text{blue}, \text{tires}, \text{art}\}$$

Но элементы этого множества не связаны какими-либо обычными отношениями. С другой стороны, в семантической сети может быть задан класс, состоящий из самолетов, поездов и автомобилей. Объекты этого класса связаны друг с другом, поскольку все они представляют собой некоторые средства транспорта.

В данном случае связь AKO используется для обозначения отношения одного класса с другим. Связь AKO не может служить для обозначения отношений конкретных объектов; для этого предназначена связь IS-A. Связь AKO определяет отношение между отдельным классом и родительским классом (или классами), применительно к которому этот отдельный класс является дочерним.

Указанные отношения можно трактовать и так, что связь AKO определяет отношение между одними **родовыми** узлами и другими родовыми узлами, а связь IS-A определяет отношение между экземпляром, или **отдельным объектом**, и родовым классом. Следует отметить, что на рис. 2.5 более общие классы находятся в верхней части, а более конкретные классы — в нижней части. Более общий класс, на который указывает стрелка AKO, называется **суперклассом**. Если суперкласс имеет связь AKO, указывающую на другой узел, то он вместе с тем представляет собой класс суперкласса, на который указывает стрелка AKO. Иной способ представления таких отношений заключается в том, что связь AKO направлена от **подкласса** к классу. Вместо связи AKO иногда используется связь **ARE**, в которой ARE читается как обычное слово “are” (есть).

Все объекты класса должны иметь один или несколько общих **атрибутов**. Каждый атрибут имеет **значение**. Комбинация атрибута и значения называется **свойством**. Например, дирижабль имеет такие атрибуты, как размеры, вес, форма и цвет. Значением атрибута формы является эллипсоид. Иными словами, дирижабль имеет такое свойство, как эллипсоидальная форма. В семантических сетях можно также найти связи других типов. Связь **IS-A** определяет значение. Например, самолет, на котором летает Президент, приобретает атрибут “Самолета ВВС номер один”. А если Президент летит на вертолете, такой вертолет IS-A (является) “Вертолетом ВВС номер один”. Связь **CAUSE** выражает причинные знания. Например, горячий воздух CAUSE (становится причиной) того, что воздушный шар поднимается.

Рекламный дирижабль компании Goodyear является дирижаблем, а дирижабли имеют эллипсоидальную форму; из этого следует, что дирижабль Goodyear является эллипсоидальным. Повторение характеристик узла в его потомках называется **наследованием**. Если нет более определенного свидетельства, позволяющего утверждать обратное, то предполагается, что все элементы некоторого класса наследуют все свойства суперклассов этого класса. Например, воздушные шары имеют круглую форму. Но класс дирижаблей имеет связь, указывающую на узел, обозначающий эллипсоидальную форму, поэтому явно заданная связь рассматривается как имеющая более высокий приоритет. Наследование представляет собой полезное средство в представлении знаний, поскольку позволяет избавиться от

необходимости повторно указывать общие характеристики. Связи и наследование являются основой эффективных способов представления знаний, поскольку дают возможность показывать многие сложные отношения с помощью нескольких узлов и связей. Ниже в этой главе будут описаны объектно-ориентированные возможности языка CLIPS, которые реализуются с использованием встроенного в него языка COOL. Эти два языка позволяют создавать экспертные системы, используя правила, объекты или комбинации правил и объектов, связанных полноценными отношениями множественного наследования.

2.5 Тройки “объект–атрибут–значение”

Одна из проблем, связанных с применением семантических сетей, состоит в том, что не предусмотрены стандартные определения для имен связей. Например, в некоторых книгах связи IS-A используются для представления и общих, и индивидуальных отношений. Это означает, что связь IS-A применяется для представления такого же смысла, какой представляют обычные слова “is a” (является), а также может применяться вместо связи AKO.

Еще одной широко применяемой связью является HAS-A, которая устанавливает отношение между классом и подклассом. Связь HAS-A направлена в сторону, противоположную по отношению к связи AKO, и часто используется для обозначения отношения между одним объектом и частью другого объекта, например, как показано ниже.

```
автомобиль HAS-A двигатель
автомобиль HAS-A шины
автомобиль IS-A автомобиль производства компании Ford
```

Более определенно можно указать, что связь IS-A устанавливает отношение между значением и атрибутом, а связь HAS-A — между объектом и атрибутом.

Такие три понятия, как *объект*, *атрибут* и *значение*, встречаются вместе настолько часто, что иногда обнаруживается возможность создать упрощенную семантическую сеть с использованием только этих понятий. Чтобы охарактеризовать все знания, представленные в семантической сети, можно воспользоваться **тройкой “объект–атрибут–значение”** (**object-attribute-value — OAV**), или просто **триплетом**; такие триплеты использовались в экспертной системе MYCIN, предназначенной для диагностирования инфекционных заболеваний. Представление в виде тройки OAV является удобным способом оформления списка имеющихся знаний в форме таблиц, а также позволяет легко преобразовать полученную таблицу в компьютерный код по методу индукции правил. Пример таблицы с тройками OAV показан в табл. 2.1. *Индукция* — это мощный логический метод, который часто используется неправильно. В качестве классического примера можно указать такую ситуацию, когда нерадивого программиста спрашивают, по-

чему он не создает резервные копии данных, хранящихся на жестком диске своего компьютера. Обычно лентяи дают при этом ошибочный индуктивный ответ, состоящий в том, что жесткий диск на этом компьютере еще никогда не ломался, поэтому по индукции такая авария никогда не произойдет. (Такой способ "рассуждений" широкой публики очень нравится биржевым маклерам.) Дополнительные сведения об индуктивном логическом программировании приведены в [5].

Таблица 2.1. Таблица со списком троек "объект–атрибут–значение" (OAV)

Объект	Атрибут	Значение
apple	color	red
apple	type	mcintosh
apple	quantity	100
grapes	color	red
grapes	type	seedless
grapes	quantity	500

Тройки "объект–атрибут–значение" лежат в основе особенно полезного способа представления фактов и шаблонов, применяемых для согласования с фактами в антецеденте правила. Семантическая сеть для подобной системы состоит из узлов, представляющих объекты, атрибуты и значения, соединенных связями HAS-A и IS-A. Если должен быть представлен только один объект и применять отношения наследования не требуется, то может оказаться приемлемым еще более простое представление, называемое **парой "атрибут–значение"** (**attribute-value — AV**), или просто **парой**.

2.6 Язык PROLOG и семантические сети

Семантические сети могут быть легко преобразованы в программу на языке PROLOG. Например, ниже приведены операторы PROLOG, которые представляют некоторую часть отношений из семантической сети, приведенной на рис. 2.5.

```
is_a(goodyear_blimp,blimp).
is_a(spirit_of_st_louis,special).
has_shape(blimp,ellipsoidal).
has_shape(balloon,round).
```

Обратите внимание на то, что конец оператора PROLOG обозначается точкой (.).

Основные сведения о языке PROLOG

Каждый из приведенных выше операторов PROLOG представляет собой **предикативное выражение** (или просто предикат), поскольку эти операторы основаны на логике предикатов. Однако PROLOG нельзя назвать настоящим языком логики предикатов, поскольку он является компьютерным языком с исполняемыми операторами. В языке PROLOG предикативное выражение состоит из имени предиката, такого как `is_a`, за которым следуют от нуля или больше параметров, заключенных в круглые скобки и разделенных запятыми. Ниже приведены некоторые примеры предикатов PROLOG. Комментарии обозначаются точками с запятой и игнорируются машиной PROLOG.

```
color(red).                ; то, что цвет — красный, является фактом
mother(pat,ann).           ; Пэт — мать Энн
parents(jim,ann,tom)       ; Джим и Энн — родители Тома
surrogatemother(pat,tom).  ; Пэт — суррогатная мать Тома
```

Назначение предикатов с двумя параметрами проще понять, представив себе, что они состоят из имени предиката, за которым следует первый параметр. С другой стороны, смысл предикатов, имеющих больше двух параметров, необходимо определять явно, как показано в примере предиката `parents`. А при использовании семантических сетей возникает другая сложность, поскольку они позволяют представлять в основном только двоичные отношения в силу того, что линия, обозначающая связь, имеет только два конца. Таким образом, предикат `parents` нельзя представить графически с помощью одного ориентированного ребра, поскольку в определении предиката имеются три параметра. А попытка реализовать идею, согласно которой Том и Энн должны быть представлены в одном родительском узле, а Джим — в другом, приводит к новым осложнениям. Дело в том, что при этом исключается возможность использовать родительский узел для определения других бинарных отношений, таких как `mother-of`, поскольку Тома никак нельзя считать матерью.

Предикаты могут быть также представлены с помощью таких отношений, как IS-A и HAS-A, например, следующим образом:

```
is_a(red,color).
has_a(john,father).
has_a(john,mother).
has_a(john,parents).
```

Обратите внимание на то, что эти предикаты `has_a` не выражают тот же смысл, как и раньше, поскольку отец, мать и в целом родители Джона явно не

названы. Для того чтобы указать имена отца, матери и отдельно каждого из родителей, необходимо добавить несколько дополнительных предикатов, как показано ниже.

is_a(tom,father).
is_a(susan,mother).
is_a(tom,parent).
is_a(susan,parent).

Но даже эти дополнительные предикаты не выражают тот же смысл, что и первоначальные предикаты. Например, мы знаем, что Джон имеет отца, а Том является отцом, но из этого не следует, что Том — отец Джона.

Все приведенные выше операторы фактически представляют собой описания фактов в языке PROLOG. Программы на языке PROLOG состоят из фактов и правил, заданных в общей форме **целей**:

$$p\text{:-}\ p_1, p_2, \ldots, p_N.$$

В этом операторе терм p представляет собой голову правила, а термы p_k выполняют роль **подцелей**. Как правило, выражение, применяемое в форме такого оператора, является хорновским выражением, которое утверждает, что цель p, заданная в голове выражения, выполняется тогда и только тогда, когда выполнены все подцели. Исключение из этого правила возникает только в том случае, если используется специальный предикат, обозначающий отказ. Предикат с обозначением отказа является удобным, поскольку в классической логике нелегко доказать истинность отрицания, а язык PROLOG основан на классической логике. Отрицание рассматривается как невозможность найти доказательство, и поэтому процесс выполнения программы может оказаться очень длинным и сложным, если количество потенциальных согласований достаточно велико. Благодаря использованию **оператора отсечения** и специального предиката `failure` процесс доказательства истинности отрицания становится более эффективным, поскольку процедура поиска доказательства сокращается.

Запятые, разделяющие подцели, представляют логическую операцию `AND`, а символ `:-` интерпретируется как знак операции `IF`. Если в операторе задана только голова выражения, а правая часть отсутствует (как в приведенном ниже примере), то считается, что голова выражения имеет истинное значение.

$$p.$$

Именно поэтому предикаты, подобные следующим, рассматриваются как факты, следовательно, должны быть истинными:

color(red).
has_a(john,father).

Еще один способ трактовки понятия факта состоит в том, что факт рассматривается как безусловное заключение, которое не зависит от чего-либо иного, поэтому для его определения операция `IF` (или :-) не требуется. В отличие от этого, правила PROLOG требуют применения операции `IF`, поскольку представляют собой условные выражения, истинность которых зависит от одного или нескольких условий. В качестве примера можно привести следующие правила определения предиката `parent`:

$$\text{parent}(X, Y)\text{:- father}(X, Y).$$
$$\text{parent}(X, Y)\text{:- mother}(X, Y).$$

Эти правила означают, что X является родителем Y, если X — отец Y или если X — мать Y. Аналогичным образом, предикат `grandparent` (дедушка или бабушка) может быть определен таким образом:

$$\text{grandparent}(X, Y)\text{:- parent}(X, Z)\text{,parent}(Z, Y).$$

С другой стороны, предикат `ancestor` (предок) можно определить, как показано ниже.

(1) $\text{ancestor}(X, Y)\text{:- parent}(X, Y).$
(2) $\text{ancestor}(X, Y)\text{:- ancestor}(X, Z)\text{,ancestor}(Z, Y).$

Как принято в настоящей книге, обозначения (1) и (2) используются для идентификации отдельных компонентов примера.

Обеспечение поиска в языке PROLOG

Как правило, в системе, предназначенной для выполнения операторов PROLOG, применяется интерпретатор, хотя в некоторых системах может вырабатываться откомпилированный код. Общая структура системы PROLOG показана на рис. 2.6. Пользователь взаимодействует с системой PROLOG, вводя запросы в форме предикатов и получая ответы. **Предикативная база данных** содержит предикаты, представленные в виде правил и фактов, которые были в нее введены и в результате сформировали базу знаний. Интерпретатор предпринимает попытки определить, следует ли предикат запроса, введенный пользователем, из базы данных. Если запрос сформулирован как факт и этот факт находится в базе данных, интерпретатор возвращает ответ `yes`, а если этот факт в базе данных отсутствует — возвращает ответ `no`. С другой стороны, если запрос сформулирован в виде правила, то интерпретатор предпринимает попытки выполнить подцели этого правила, проводя **поиск в глубину**, как показано на рис. 2.7. Для сравнения на этом рисунке показан также ход **поиска в ширину**, хотя такой режим работы в системе PROLOG обычно не применяется.

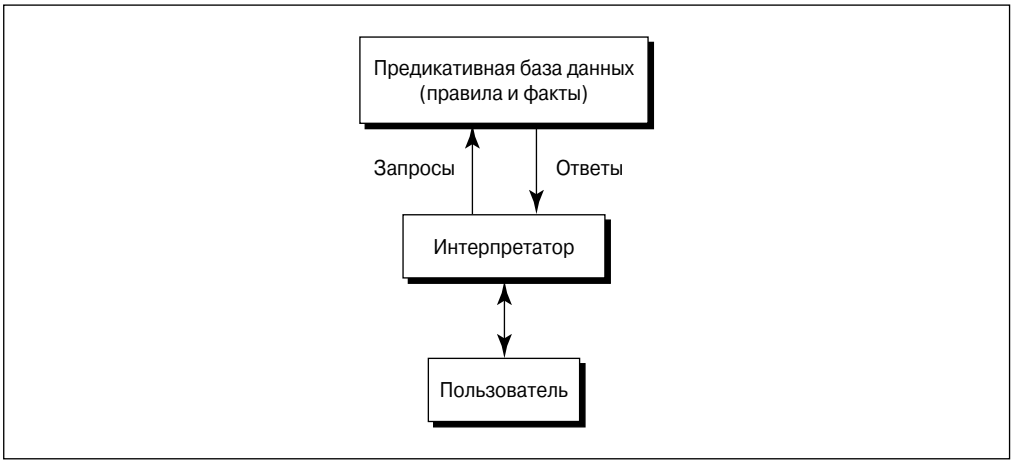

Рис. 2.6. Общая организация системы PROLOG

При поиске в глубину поиск в дереве начинается от корня, продолжается настолько далеко, насколько это возможно, после чего происходит возврат. Еще одна отличительная особенность поиска в системе PROLOG состоит в том, что он происходит слева направо. А поиск в ширину предусматривает полную обработку узлов на текущем уровне перед переходом на следующий, более низкий уровень.

В качестве примера поиска цели в системе PROLOG рассмотрим, как происходит обработка приведенных выше правил (1) и (2), относящихся к предикату `ancestor`. Предположим, что на данном этапе введены следующие факты:

(3) parent(ann,mary).
(4) parent(ann,susan).
(5) parent(mary,bob).
(6) parent(susan,john).

А теперь допустим, что системой PROLOG передан следующий запрос для определения того, является ли Энн предком Сьюзен:

:- ancestor(ann,susan).

Признаком того, что это — **запрос**, является отсутствие в правиле головы. Это означает, что система PROLOG должна доказать данное условие. Тремя типами хорновских выражений, применяемых в системе PROLOG, являются факты, правила и запросы. Условие может быть доказано, только если оно представляет собой заключение некоторого экземпляра выражения. Безусловно, для этого должно быть доказуемым само выражение, и такое доказательство осуществляется путем доказательства условий выражения.

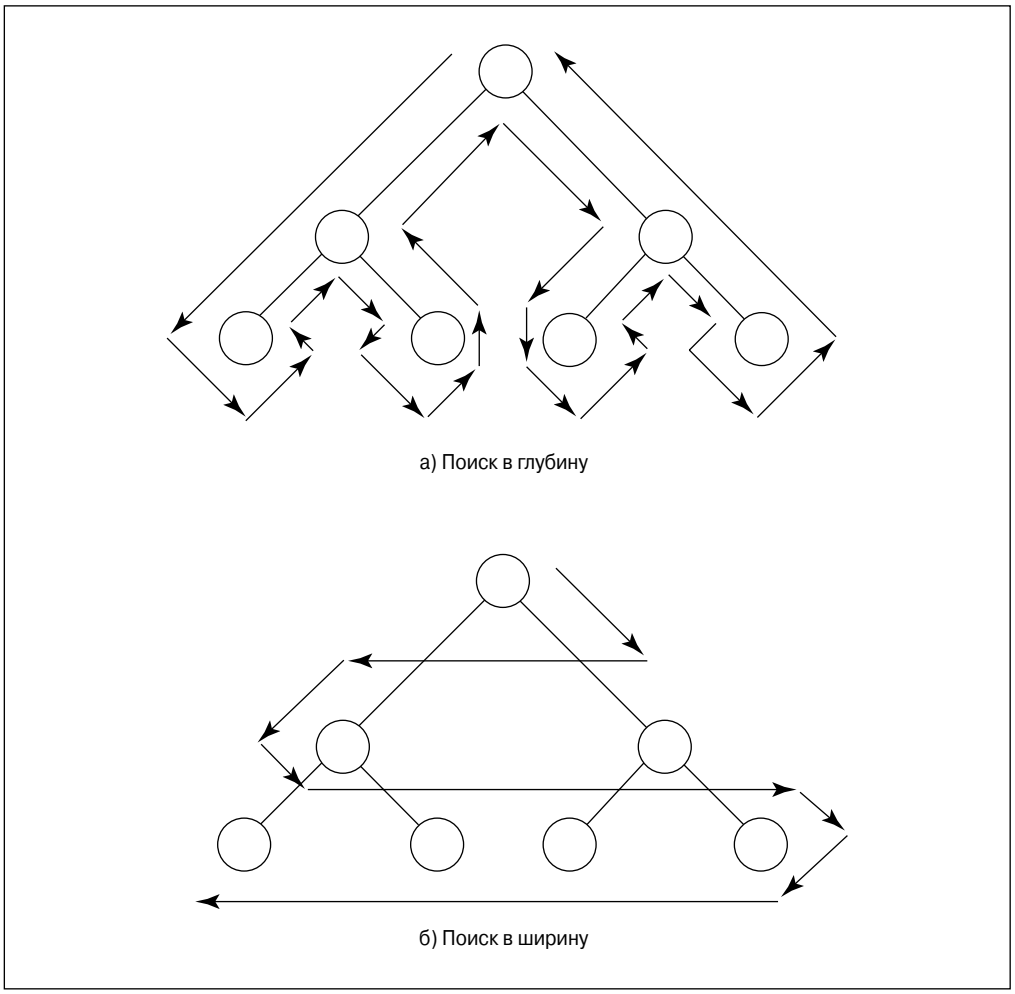

Рис. 2.7. Поиск в глубину и поиск в ширину применительно к дереву произвольной формы

После того как система PROLOG принимает этот входной запрос, начинается поиск выражения, голова которого согласуется с входным шаблоном ancestor(ann,susan). Такая процедура называется **сопоставлением с шаблоном** и полностью аналогична сопоставлению с шаблоном, в котором участвуют факты и антецеденты продукционного правила. Поиск начинается с первого введенного оператора, (1), который находится в **начале** списка операторов, и происходит в направлении к последнему оператору, (6), находящемуся в **конце** списка. При этом обнаруживается возможность согласования с первым правилом предиката `ancestor`, с правилом (1). Переменная X согласуется с `ann`, а переменная

Y — с `susan`. Поскольку теперь голова предиката согласована, система PROLOG предпринимает попытку согласовать тело правила (1), что приводит к созданию подцели `parent`(ann,susan). Затем система PROLOG осуществляет попытку согласовать эту подцель с другими выражениями и в конечном итоге согласует ее с фактом (4), parent(ann,susan). На этом все цели, подлежащие согласованию, исчерпываются, и система PROLOG отвечает "`yes`", поскольку первоначальный запрос является истинным.

В качестве еще одного примера предположим, что введен следующий запрос:

:- ancestor(ann,john).

Согласуется правило (1) предиката `ancestor`, в результате чего переменной X присваивается значение `ann`, а переменной Y — значение `john`. После этого система PROLOG предпринимает попытки согласовать тело правила (1), parent(ann,john), с каждым из операторов `parent`. Однако все попытки согласования оканчиваются неудачей, поэтому обнаруживается, что тело правила (1) не может быть истинным. Поскольку тело правила (1) не является истинным, голова этого правила также не может быть истинной.

Поскольку правило (1) не может быть истинным, система PROLOG затем предпринимает попытку проверить истинность правила (2) предиката `ancestor`. Переменной X присваивается значение `ann`, а переменной Y — значение `john`. Система PROLOG осуществляет попытки доказать, что голова правила (2) является истинной, доказывая то, что обе подцели правила (2) являются истинными. Это означает, что требуется доказать истинность обеих подцелей, ancestor(ann,Z) и ancestor(Z,john). Система PROLOG предпринимает попытки согласования подцелей в направлении слева направо, начиная с подцели ancestor(ann, Z). Проходя по списку выражений, начиная сверху, система PROLOG обнаруживает, что эта первая подцель согласуется с правилом (1), и поэтому пытается доказать истинность тела правила (1), parent(ann, Z). А после того как поиск снова начинается с верхней части списка, обнаруживается первое согласование этого тела с оператором (3), поэтому система PROLOG присваивает переменной Z значение `mary`. Теперь система PROLOG пытается согласовать вторую подцель правила (2), представленную в виде ancestor(mary,john). Эта подцель согласуется с правилом (1), и поэтому PROLOG пытается выполнить тело этого оператора, parent(mary,john). Однако в операторах от (3) до (6) факт parent(mary,john) отсутствует, поэтому данная попытка поиска оканчивается неудачей.

Затем система PROLOG пересматривает свое предположение, согласно которому переменной Z должен быть присвоено значение `mary`. Поскольку этот вариант оказался неприменимым, система пытается найти приемлемое значение. Еще одна возможность осуществляется с помощью правила (4), согласно которому переменной Z присваивается значение `susan`. Показанный здесь метод возврата в целях опробования другого пути поиска, после того как предыдущая попытка

воспользоваться некоторым путем окончилась неудачей, называется **перебором с возвратами** и часто используется для решения задач.

После выбора варианта присваивания переменной *Z* значения `susan` система PROLOG предпринимает попытку доказать, что выражение ancestor(susan,john) является истинным. Согласно правилу (1), для этого необходимо доказать истинность тела parent(susan,john). И действительно, факт (3) согласуется, поэтому указанный запрос обозначается как истинный, так как доказано, что его условия являются истинными:

:- ancestor(ann,john)

Обратите внимание на то, что управляющая структура PROLOG относится к типу марковского алгоритма, в котором последовательность поиска в ходе сопоставления с шаблонами обычно определяется тем, в каком порядке введены хорновские выражения. В этом состоит отличие системы PROLOG от экспертных систем, основанных на правилах, которые обычно следуют принципу Поста (согласно этому принципу, ход поиска не зависит от того, в каком порядке введены правила).

Система PROLOG обладает многими другими особенностями и возможностями, которые не упоминаются в настоящей книге. С точки зрения разработчиков экспертных систем, особенно важными возможностями PROLOG являются перебор с возвратами и сопоставление с шаблонами. К тому же важно то, что язык PROLOG имеет декларативный характер, поскольку это означает, что в качестве исполняемой программы непосредственно применяется спецификация программы.

2.7 Трудности, связанные с использованием семантических сетей

Безусловно, семантические сети могут оказаться весьма полезным средством представления знаний, но они имеют целый ряд ограничений, в частности, связанных с отсутствием стандартов именования связей, о чем было сказано выше. В результате возникают затруднения при попытке понять, для чего фактически была создана какая-то конкретная сеть, и действительно ли эта сеть была создана правильно. С указанной проблемой именования связей сопряжена проблема именования узлов. Например, если в сети имеется узел, обозначенный как “chair” (“кресло”), остается неизвестным, обозначает ли это слово одно из указанных ниже понятий, или что-то другое.

```
данное конкретное кресло
класс всех кресел
```

```
понятие кресла
кресло председателя собрания
```

Семантическая сеть может применяться для представления **категорических знаний** (т.е. знаний, которые определены однозначно) только при том условии, что строго определены сами имена связей и узлов. Безусловно, аналогичные проблемы обнаруживаются и при использовании языков программирования. Но, к счастью, указанная задача уже решена с введением в действие **расширяемого языка разметки** (**eXtensible Markup Language — XML**) и онтологий. Язык XML проявил себя как бесценное средство реализации стандартного способа условного представления формальной семантики в виде конструкций языков всех возможных типов. Кроме того, разработана версия XML, основанная на правилах, а также созданы языки разметки для математики, музыки, пищевой промышленности и многих других областей, в которых для обработки информации применяются компьютеры. Благодаря использованию языка XML и онтологий система World Wide Web (сокращенно WWW, или W3) постепенно превращается из хранилища данных в коллекцию информации, доступной для чтения с помощью компьютера, имеющую гораздо более широкую область применения. В связи с этим Web теперь все чаще именуют **семантической системой Web** (**Semantic Web**), а не просто системой Web, поскольку в этой системе в настоящее время реализуются новые способы представления смыслового значения информации.

Еще одна проблема, связанная с использованием семантических сетей, состоит в том, что при осуществлении поиска узлов возникает комбинаторный взрыв, особенно если ответ на запрос является отрицательным. Дело в том, что если запрос должен выработать отрицательный результат, то может оказаться, что требуется выполнить поиск по многим или даже по всем связям в сети. А как показано в описании задачи коммивояжера, приведенном в главе 1, количество связей равно факториалу от количества узлов в сети за вычетом единицы, если все узлы сети связаны друг с другом. Безусловно, такая степень связности встречается не во всех представлениях задач, но возможность комбинаторного взрыва нельзя исключить.

Семантические сети были первоначально предложены для использования в качестве модели ассоциативной памяти человека. В этой модели каждый узел имеет связи с другими узлами и выборка информации происходит в результате распространения активности по узлам сети. Но в мозгу человека должны также быть предусмотрены другие механизмы, поскольку для него не требуется много времени, чтобы ответить на вопрос: “Есть ли футбольная команда на Плутоне?” В мозгу человека имеется приблизительно 10^{11} нейронов и примерно 10^{15} связей. Если бы все его знания были представлены только с помощью семантической сети, то потребовалось бы чрезвычайно продолжительное время для формулировки отрицательного ответа на вопросы, подобные указанному выше вопросу о футбольной

команде, поскольку для поиска ответа пришлось бы ввести в действие все 10^{15} связей.

Кроме того, семантические сети не соответствуют требованиям логики, поскольку не позволяют применять для определения знаний такие способы, которые являются применимыми к логике. Как будет описано ниже в данной главе, логические способы представления позволяют определить какое-то конкретное кресло, некоторые кресла, все кресла, отсутствие кресел и т.д. Еще одна проблема состоит в том, что семантические сети остаются неприменимыми для эвристических методов, поскольку в сети невозможно представить эвристическую информацию, касающуюся того, как можно эффективно провести поиск в сети. **Эвристика** — это эмпирическое правило, которое может помочь найти решение, но, в отличие от алгоритма, не гарантирует нахождение решения. В искусственном интеллекте возможность применения эвристик очень важна, поскольку типичные задачи искусственного интеллекта настолько сложны, что алгоритмическое решение для них не существует или слишком неэффективно для практического использования. Единственной стандартной стратегией управления, встроенной в сеть, которая может помочь в решении задач, является наследование, но не все задачи могут быть представлены с использованием такой структуры.

Для устранения недостатков, присущих семантическим сетям, было опробовано много подходов. Некоторые введенные в них усовершенствования были основаны на логике, а другие усовершенствования, в которых предпринимались попытки закрепления процедур за узлами, были основаны на использовании эвристик. Такие процедуры должны были вызываться на выполнение после того, как узел становится активизированным. Но созданные в результате системы позволяли достичь лишь незначительного расширения возможностей за счет ухудшения естественной выразительности семантических сетей. По результатам анализа всех этих усилий был сделан вывод, что семантические сети, как и любые другие инструментальные средства, должны использоваться в тех областях, для которых они приспособлены в наибольшей степени, т.е. для представления бинарных отношений, поэтому не следует их загромождать, пытаясь превратить в универсальное инструментальное средство.

2.8 Схемы

Семантическая сеть может служить примером **поверхностной структуры знаний**. Поверхностность знаний в семантической сети обусловлена тем, что все знания в ней представлены в виде связей и узлов. **Структура знаний** аналогична структуре данных в том, что составляет упорядоченную коллекцию знаний, а не просто разрозненные данные. С другой стороны, **глубокую структуру знаний** имеют причинные знания, которые объясняют, почему происходит то или

иное событие. Например, может быть создана медицинская экспертная система на основе поверхностных знаний, содержащая примерно такие правила:

```
IF у пациента имеется высокая температура
THEN пациент должен принять аспирин
```

Но в подобной системе отсутствует фундаментальные знания о биохимии человеческого организма и о том, почему прием аспирина влечет за собой снижение температуры. Это правило вполне могло быть определено следующим образом:

```
IF у одного из присутствующих имеется розовая обезьяна
THEN он должен принять в подарок холодильник
```

Иными словами, знания в такой экспертной системе являются поверхностными, поскольку они основаны на синтаксисе, а не на семантике, а в этой синтаксической конструкции любые два словесных выражения могут быть заменены на `X` и `Y`, как в следующем правиле:

```
IF у какого-то человека имеется (X)
THEN он должен принять (Y)
```

Обратите внимание на то, что в этом правиле (`X`) и (`Y`) — не переменные, а подстановочные символы, заменяющие любые два словесных выражения. С другой стороны, врачи обладают причинными знаниями, поскольку учились много лет в медицинском институте и приобрели опыт лечения пациентов. Если же способ лечения, применяемый как обычно, оказывается неэффективным, врачи могут использовать свои способности к рассуждению, чтобы найти вариант, альтернативный по отношению к выбранному с помощью обычно применяемого ими метода, основанного на прецедентах. Иными словами, эксперт знает, когда можно нарушить правила.

Структура семантической сети слишком проста, поэтому с ее помощью невозможно представить знания многих типов, применяемые в реальном мире. Для лучшего представления сложных структур знаний требуются более сложные структуры представления. В искусственном интеллекте для описания более сложных структур знаний, чем семантические сети, используется термин **схема** (schema). Термин *схема* заимствован из психологии, где он обозначает непрерывную организацию знаний или реакций живого существа, вырабатываемых в ответ на стимулы. Это означает, что живые существа, изучая причинные отношения между стимулом и результатом, стремятся повторно испытать приятные стимулы и избежать неприятных.

Например, при осуществлении таких действий, как еда и питьё, складываются **сенсорно-двигательные схемы**, окрашенные ощущениями удовольствия, которые обеспечивают координацию информации, полученной от органов чувств, с необходимыми моторными (мускульными) движениями, с помощью которых осуществляется процесс еды и питья. Человек не обязан задумываться над этими

неявными знаниями, чтобы узнать, как выполнять эти действия, к тому же очень трудно точно объяснить, как они осуществляются, вплоть до уровня управления мускулами. А еще более значительные затруднения возникают при попытке объяснить схему, которой подсознательно руководствуется человек во время езды на велосипеде. Попробуйте объяснить словами чувство равновесия!

Еще одним важным типом схем являются **концептуальные схемы**, с помощью которых в мозгу человека складываются концепции. Например, допустим, что человек овладел концептуальным представлением о том, что́ такое животное. Но большинство людей, которых просят объяснить, что представляет собой животное, скорее всего, будут описывать его в таких терминах, что животное — существо с четырьмя ногами, заросшее мехом. К тому же, безусловно, сложившаяся у человека концепция животного будет зависеть от того, вырос ли он в сельской местности, в городе или на берегу реки. Тем не менее все люди имеют в своем сознании, складывающемся из концепций, определенные **стереотипы**. Безусловно, в обыденной речи термин *стереотип* может иметь отрицательную окраску, но в искусственном интеллекте он обозначает типичный пример. Поэтому для большинства людей стереотипом животного может быть собака.

Концептуальная схема — это абстракция, в которой конкретные объекты классифицируются по их общим свойствам. Например, если вы увидите рисунок с надписью “Искусственный фрукт”, на котором изображен небольшой круглый объект красного цвета с зеленым черенком, то, по-видимому, распозна́ете его как искусственное яблоко. Этот объект обладает свойствами принадлежности к яблокам, которые могут быть поставлены в соответствие с концептуальной схемой яблока.

Концептуальная схема настоящего яблока может охватывать такие общие свойства яблок, как размеры, цвет, вкус, назначение и т.д. Эта схема не будет включать подробных сведений о том, где именно было сорвано яблоко, на каком грузовике оно доставлено в супермаркет и как зовут человека, положившего его на полку. Эти подробности не имеют значения при анализе тех свойств, из которых складывается абстрактная концепция яблока. Кроме того, следует учитывать, что, например, слепой человек может иметь совсем другую концептуальную схему яблока, в которой наибольшую важность имеет текстура поверхности и запах.

Сосредоточение на общих свойствах объекта позволяет упростить проведение рассуждений об этих свойствах, поскольку при этом не приходится отвлекаться на незначащие подробности. Как правило, схемы имеют структуру, внутреннюю по отношению к применяемым в них узлам, а семантические сети не имеют такой структуры. Все знания о некотором узле, применяемом в семантической сети, заключены в надписи на этом узле. Семантическая сеть аналогична такой структуре данных, применяемой в компьютерных науках, в которой ключ поиска одновременно представляет собой данные, хранящиеся в узле. А схема аналогична структуре данных, в которой узлы содержат записи. Каждая запись может содержать данные, записи или указатели на другие узлы.

2.9 Фреймы

Одним из типов схем, который используется во многих приложениях искусственного интеллекта, является **фрейм**. Еще один тип схемы представляет собой **сценарий**, который по существу является упорядоченной во времени последовательностью фреймов. Фреймы предложены для использования в качестве метода понимания особенностей зрения, естественных языков и других областей и предоставляют удобную структуру для описания объектов, типичных для какой-то конкретной ситуации, в частности, стереотипов объектов. Фреймы являются особенно полезным средством моделирования знаний, основанных на здравом смысле, а эта область знаний с большим трудом поддается освоению с помощью компьютеров. Безусловно, семантические сети по существу можно рассматривать как двумерное представление знаний, а фреймы добавляют третье измерение, поскольку позволяют использовать узлы, имеющие внутреннюю структуру. В качестве таких структур могут применяться простые значения или другие фреймы.

Основная характерная особенность фрейма состоит в том, что он представляет взаимосвязанные знания по узкой теме, значительная часть которых — это знания, заданные по умолчанию. Система фреймов может оказаться весьма подходящей для описания механического устройства, например автомобиля. Такие компоненты автомобиля, как двигатель, корпус, тормозная система и т.д., должны рассматриваться в сочетании, чтобы можно было получить общее представление о связях между этими компонентами. Дополнительные сведения о компонентах могут быть получены с помощью изучения структуры фреймов. Безусловно, автомобили разных моделей отличаются друг от друга, но большинство автомобилей, как правило, имеют такие аналогичные компоненты, как колеса, двигатель, корпус и передаточный механизм. В этом состоит отличие фрейма от семантической сети, поскольку сеть, как правило, используется для общего представления знаний. Тем не менее, как и в случае семантических сетей, стандарты определения систем на основе фреймов отсутствуют. Для работы с фреймами создан целый ряд языков специального назначения, таких как FRL, SRL, KRL, KEE, HP-RL; кроме того, предусмотрены такие усовершенствованные средства работы с фреймами языка LISP, как LOOPS и FLAVORS.

Фрейм аналогичен структуре записи на языке высокого уровня, таком как C, или атому со списком свойств в языке LISP. В свою очередь, полям и значениям записи соответствуют такие компоненты фрейма, как **слоты** и **заполнители** слотов. Фрейм по существу представляет собой группу слотов и заполнителей, которые определяют объект, рассматриваемый в качестве стереотипа. Фрейм с описанием автомобиля показан на рис. 2.8. В терминах представления в виде тройки “объект–атрибут–значение” автомобиль — это объект, имя слота — атрибут, а заполнитель — значение.

Слоты	Заполнители
manufacturer	General Motors
model	Chevrolet Caprice
year	1979
transmission	automatic
engine	gasoline
tires	4
color	blue

Рис. 2.8. Фрейм с описанием автомобиля

Большинство фреймов не являются настолько простыми, как показано на рис. 2.8. Фреймы обнаруживают свою значимость в составе иерархических систем фреймов и в условиях применения наследования. Использование фреймов в виде заполнителей слотов и ввод в действие отношения наследования позволяет создавать очень мощные системы представления знаний. В частности, как оказалось, экспертные системы на основе фреймов являются очень полезным средством представления причинных знаний, поскольку хранимая в них информация организована с учетом причин и следствий. В отличие от этого, экспертные системы, основанные на правилах, обычно опираются на неорганизованные знания, которые не являются причинными.

Некоторые инструментальные средства, основанные на фреймах, такие как KEE, позволяют хранить в слотах самые разнообразные элементы. В частности, слоты фреймов могут хранить правила, графику, комментарии, отладочную информацию, вопросы для пользователей, гипотезы, касающиеся той или иной ситуации, или другие фреймы.

Фреймы обычно применяются для представления либо универсальных, либо специальных знаний. На рис. 2.9 показан универсальный фрейм, предназначенный для представления концепции собственности, которой может владеть человек.

Заполнителями могут быть значения, такие как название собственности в слоте `name`, или ряд значений, например, как в слоте `types`. Сами слоты могут также содержать процедуры, закрепленные за слотами и называемые **процедурными вложениями**. Сами процедуры, как правило, подразделяются на три типа. Процедура типа **if-needed** выполняется, если требуется значение заполнителя, но первоначально это значение не присутствует в слоте или является неприменимым значение **default**, предусмотренное по умолчанию. Во фреймах значения,

Слоты	Заполнители
name	property
specialization_of	a_kind_of object
types	(car, boat, house) if-added: Процедура ADD_PROPERTY
owner	default: government if-needed: Процедура FIND_OWNER
location	(home, work, mobile)
status	(missing, poor, good)
under_warranty	(yes, no)

Рис. 2.9. Универсальный фрейм, предназначенный для описания концепции собственности

предусмотренные по умолчанию, являются исключительно важными, поскольку позволяют моделировать некоторые аспекты функционирования мозга. Применяемые по умолчанию значения соответствуют ожиданиям человека в отношении некоторой ситуации, которые складываются на основании опыта. А после того как обнаруживается новая ситуация, осуществляется модификация наиболее подходящего фрейма, что позволяет проще приспособиться к ситуации. Люди не начинают с пустого места, сталкиваясь с каждой новой ситуацией. Вместо этого они модифицируют в существующих фреймах заданные по умолчанию значения или другие заполнители. Заданные по умолчанию значения часто используются для представления знаний, основанных на здравом смысле. **Знания, основанные на здравом смысле**, могут рассматриваться как общеизвестные знания. Мы используем здравый смысл, если недоступны знания, более соответствующие конкретной ситуации.

Процедура типа **if-added** вызывается на выполнение, если в слот должно быть введено какое-либо дополнительное значение. Например, в слоте `types` процедура `if-added` вызывает на выполнение в случае необходимости процедуру `ADD-PROPERTY` для добавления собственности нового типа. Например, эта процедура может быть вызвана после приобретения драгоценностей, телевизора, стереомагнитофона или собственности другого типа, поскольку указанные значения не содержатся в слоте `types`.

Процедура типа **if-removal** вызывается на выполнение каждый раз, когда возникает необходимость удалить из слота какое-либо значение. В частности, проце-

дура такого типа выполняется, если данные, представленные каким-то значением, устаревают.

Заполнители слотов могут также содержать отношения по такому же принципу, который применяется при специализации слотов. На рис. 2.9–2.11 показано, как используются отношения `a-kind-of` и `is-a` для создания иерархических связей между фреймами. Фреймы, представленные на рис. 2.9 и 2.10, являются универсальными, а фрейм, приведенный на рис. 2.11, является конкретным, поскольку представляет собой экземпляр фрейма с описанием определенного автомобиля. Разрабатывая эти фреймы, мы руководствовались соглашением, что отношения `a-kind-of` являются универсальными, а отношения `is-a` — конкретными.

Слоты	Заполнители
name	car
specialization_of	a-kind-of property
types	(sedan, sports, convertible)
manufacturer	(GM, Ford, Toyota)
location	mobile
wheels	4
transmission	(manual, automatic)
engine	(gasoline, hybrid gas/electric)

Рис. 2.10. Фрейм с описанием автомобиля — универсальный субфрейм с описанием типа собственности

Системы фреймов проектируются таким образом, чтобы более универсальные фреймы находились ближе к вершине иерархии. Предполагается, что специализация фреймов применительно к конкретным случаям может осуществляться путем модификации применяемых по умолчанию значений и создания более конкретных фреймов. С помощью фреймов может быть предпринята попытка моделировать объекты реального мира, используя универсальные знания для описания большинства атрибутов того или иного объекта и более конкретные знания — для описания частных случаев. Например, обычно принято рассматривать птиц как теплокровных, покрытых перьями живых существ, способных летать. Но некоторые птицы, такие как пингвины и страусы, не могут летать. Эти типы представляют более конкретные классы птиц, а их характеристики могут обнаруживаться на более низких уровнях иерархии фреймов по сравнению с такими птицами, как канарейки или

Слоты	Заполнители
name	John's car
specialization_of	is_a car
manufacturer	GM
owner	John Doe
transmission	automatic
engine	gasoline
status	good
under_warranty	yes

Рис. 2.11. Экземпляр фрейма с описанием определенного автомобиля

дрозды. Иными словами, верхние уровни в иерархии фреймов с описанием птиц представляют характеристики, в большей степени относящиеся ко всем птицам, а нижние уровни обозначают нечеткие границы между объектами реального мира. Объект, обладающий всеми типичными характеристиками, принято называть **прототипом**; этот термин буквально означает *первичный тип*.

Кроме того, классификация фреймов может проводиться на основании того, в какой области они применяются. **Ситуативные фреймы** содержат знания о том, чего можно ожидать в какой-то конкретной ситуации, например, на вечеринке по случаю дня рождения. С другой стороны, **фреймы действия** содержат слоты, которые определяют действия, подлежащие выполнению в какой-то конкретной ситуации. Это означает, что заполнителями слотов являются процедуры, предназначенные для выполнения определенных действий, таких как удаление дефектной детали с конвейера. Фрейм действия представляет процедурные знания. А для описания причинно-следственных отношений могут применяться **фреймы причинных знаний**, представляющие собой сочетания ситуативных фреймов и фреймов действия.

На практике были созданы очень сложные системы фреймов, предназначенные для решения самых различных задач. Одной из наиболее впечатляющих систем, продемонстрировавшей широкие возможности применения фреймов для творческого открытия математических понятий, стала программа AM (Automated Mathematician — автоматизированный математик) Дуга Лената (Doug Lenat). В классической системе AM Лената создавались, а затем исследовались сочетания интересных новых понятий. Эта система предложила некоторые совершенно новые математические доказательства для целого ряда теорем. Другие многочисленные

примеры программ для доказательства теорем и систем совершения открытий приведены в том разделе приложения Ж, который относится к данной главе.

2.10 Трудности, связанные с использованием фреймов

Первоначально фреймы предназначались для использования в качестве способа представления стереотипных знаний. Важной особенностью любого стереотипа является то, что он имеет вполне определенные характеристики, поэтому позволяет предоставить для многих своих слотов значения, заданные по умолчанию. Наглядным примером стереотипов, вполне подходящих для представления в виде фреймов, являются математические понятия. Подход на основе фреймов имеет интуитивную привлекательность, поскольку обеспечиваемое с его помощью упорядоченное представление знаний, вообще говоря, в большей степени доступно для понимания по сравнению с логическими или продукционными системами, в которых для той же цели применяется целый ряд правил [51].

Тем не менее при использовании систем фреймов обнаруживаются существенные недостатки, связанные с тем, что в этих системах допускается неограниченная модификация или уничтожение слотов. Классическим примером этой проблемы является фрейм `elephant` с описанием слонов, который приведен на рис. 2.12.

Слоты	Заполнители
name	elephant
specialization_of	a-kind-of mammal
color	grey
legs	4
trunk	a cylinder

Рис. 2.12. Фрейм `elephant`

На первый взгляд создается впечатление, что фрейм `elephant` представляет собой приемлемое общее описание слонов. Однако предположим, что существует конкретный трехногий слон по кличке Клайд. Возможно, Клайд потерял ногу из-за несчастного случая на охоте или просто притворяется, что у него нет одной ноги, чтобы попасть на страницы настоящей книги. Но важно то, что во фрейме `elephant` содержится утверждение, будто любой слон имеет четыре ноги, а не три. Поэтому мы не можем рассматривать фрейм `elephant` как действительное определение любого слона.

Безусловно, этот фрейм можно модифицировать таким образом, чтобы он допускал в качестве исключения наличие трехногих, двуногих, одноногих или даже безногих слонов. Но в результате этого определение становится не очень качественным. Дополнительные проблемы могут возникать и при использовании других слотов. Предположим, что Клайд переболел желтухой в тяжелой форме и его кожа стала желтой. Но разве из-за этого он перестал быть слоном?

Альтернативным по отношению к использованию фрейма в качестве определения является вариант, в котором фрейм рассматривается как описание, например, типичного слона. Но такой подход приводит к возникновению других проблем, связанных с наследованием. Обратите внимание на то, что во фрейме `elephant` указано, что слон относится к категории (`a-kind-of`) млекопитающих. А поскольку теперь фреймы интерпретируются как описания типичных представителей, то наша система фреймов указывает, что типичный слон представляет собой типичное млекопитающее. Тем не менее, хотя слон — млекопитающее, его, по-видимому, нельзя считать типичным млекопитающим. Судя по количеству особей, вероятно, на звание типичного млекопитающего с большим правом могут претендовать люди, коровы, крысы и овцы.

В большинстве систем фреймов не предусмотрены способы определения неизменных слотов. А поскольку в связи с этим может подвергнуться изменениям любой слот, то свойства, наследуемые одним фреймом от другого, могут быть изменены или исключены на любом уровне иерархии. Это означает, что любой фрейм по существу является примитивным, исходным фреймом, так как нет никаких гарантий того, что заданные в нем свойства будут общими. Каждый фрейм складывается из собственных правил и поэтому каждый фрейм является изначальным. В такой неограниченной системе ни в чем действительно нельзя быть уверенным и поэтому невозможно формулировать общезначимые утверждения, подобные тем, которые были сделаны при определении понятия *слон*. Кроме того, нет возможности достаточно надежно создавать сложные объекты на основании более простых определений, например, описывать слона с тремя ногами исходя из определения обычного слона. Проблемы подобного типа возникают и при использовании семантических сетей с наследованием. Если разрешено вносить изменения в свойства любого узла, то ни в чем нельзя быть уверенным.

Однако есть и другой способ трактовки фреймов, который является очень полезным. Если понятие фрейма будет расширено таким образом, чтобы оно охватывало свойства объектов, то появляется возможность рассматривать как фрейм любой объект. В язык CLIPS встроен полный объектно-ориентированный язык, называемый COOL, поэтому CLIPS может рассматриваться как расширение языка, основанного на фреймах. С тех пор как в язык CLIPS был включен язык COOL, стали доступными все преимущества объектов. Поэтому на языке CLIPS могут создаваться объектно-ориентированные экспертные системы, позволяющие использовать правила (рассматриваемые как небольшие фрагменты знаний) и вместе

с тем предоставляющие возможность организовывать более крупные фрагменты знаний в виде объектов в целях упрощения разработки и сопровождения. Правила могут применяться для работы с фактами или объектами, и поэтому CLIPS обладает всеми преимущества двух категорий инструментальных средств экспертных систем — основанных на правилах и основанных на объектах. Благодаря применению объектов в языке CLIPS становится проще создавать, эксплуатировать и сопровождать крупные базы знаний по сравнению с такими системами, в которых предпринимаются попытки представить все знания в тысячах отдельных правил и фактов.

2.11 Логика и теория множеств

Кроме правил, фреймов и семантических сетей, для представления знаний могут использоваться символы **логики** (логика изучает правила формирования неопровержимых рассуждений). Важной частью процесса формирования рассуждений является логический вывод заключений из посылок. Развитие способов применения компьютеров для осуществления рассуждений привело к созданию **логического программирования** и к разработке таких языков, основанных на логике, как PROLOG. Кроме того, логика играет чрезвычайно важную роль в экспертных системах, поскольку в таких системах применяются машины логического вывода, проводящие рассуждения от фактов к заключениям. В действительности для обозначения систем логического программирования и экспертных систем используется общий описательный термин — **автоматизированные системы формирования рассуждений**.

Самая ранняя система формальной логики была разработана древнегреческим философом Аристотелем в IV веке до н.э. Аристотелевская логика основана на понятии **силлогизма**. Аристотель изобрел четырнадцать типов силлогизмов, а еще пять типов были предложены во времена Средневековья. Силлогизмы имеют две **посылки** и одно **заключение**, которое вытекает из посылок. Классический пример силлогизма приведен ниже.

```
Посылка. Все люди смертны
Посылка. Сократ - человек
Заключение. Сократ смертен
```

В силлогизме **посылки** служат в качестве свидетельств, из которых должно обязательно следовать заключение. Силлогизм — это один из способов представления знаний. Еще одним таким способом являются **диаграммы Венна** (рис. 2.13).

Внешний круг представляет все смертные создания, а внутренний круг, обозначающий людей, изображается полностью внутри круга, представляющего смертных, для указания на то, что все люди смертны. Поскольку Сократ — человек, то представляющий его круг полностью находится внутри круга, представ-

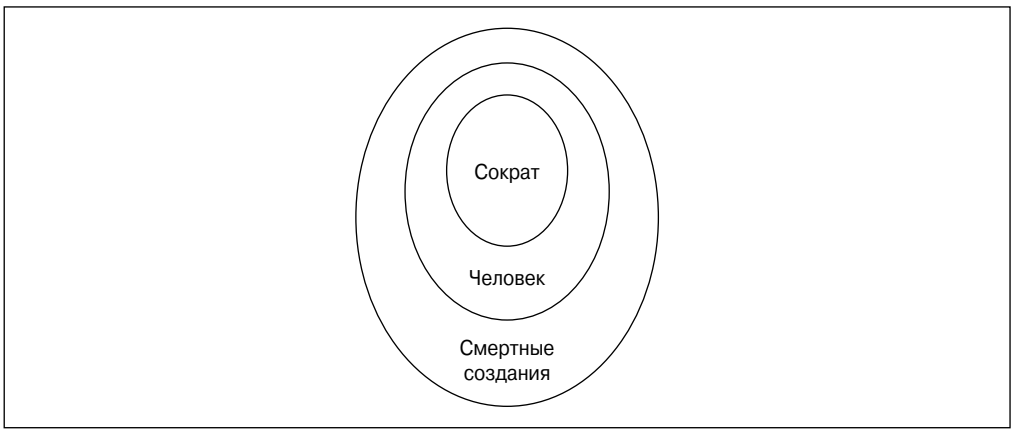

Рис. 2.13. Диаграмма Венна

ляющего людей. Строго говоря, круг, представляющий Сократа, должен выглядеть как точка, поскольку подразумевается, что круг обозначает класс. Но для удобства мы будем использовать круги при обозначении любых объектов. Заключение, что Сократ смертен, следует из того факта, что его круг находится внутри круга смертных созданий, поэтому он также должен быть смертным.

С точки зрения математики любой круг на диаграмме Венна представляет **множество**, т.е. коллекцию объектов. Объекты, принадлежащие множеству, называются **элементами** множества. Некоторые примеры множеств приведены ниже.

$$A = \{1, 3, 5\}$$
$$B = \{1, 2, 3, 4, 5\}$$
$$C = \{0, 2, 4, \dots\}$$
$$D = \{\dots, -4, -2, 0, 2, 4, \dots\}$$
$$E = \{\text{airplanes, balloons, blimps, jets}\}$$
$$F = \{\text{airplanes, balloons}\}$$

В этих примерах три точки, ..., называемые **троеточием**, указывают, что перечень термов продолжается до бесконечности.

Символ в виде греческой буквы эпсилон ($\in$) указывает, что некоторый элемент принадлежит к множеству. Например, выражение $1 \in A$ означает, что число 1 — элемент ранее определенного множества A. Если элемент не принадлежит к множеству, то для обозначения этого используется символ $\notin$, как в примере $2 \notin A$.

Если два произвольных множества, допустим, X и Y, определены так, что каждый элемент X является элементом Y, то X является **подмножеством** Y; это утверждение записывается в математической форме, как $X \subset Y$, или $Y \supset X$.

Из указанного определения подмножества следует, что каждое множество является подмножеством самого себя. А подмножество, не совпадающее с самим множеством, называется **строгим подмножеством**. Например, множество X, которое определено выше, является строгим подмножеством множества Y. В ходе рассуждений о множествах удобно считать рассматриваемые множества подмножествами **универсального множества**. При смене темы обсуждения изменяется и само универсальное множество (универсум). На рис. 2.14 показано подмножество, сформированное в результате **пересечения** двух множеств в универсуме всех автомобилей. А при обсуждении целых чисел универсумом являются все целые числа. Универсум отображается как прямоугольник, окружающий свои подмножества. Универсальные множества используются не только для удобства. Неразборчивое использование условий для определения множеств может приводить к логическим парадоксам, а универсальные множества позволяют избежать таких парадоксов.

Допустим, что A — множество всех синих автомобилей, B — множество всех автомобилей с ручной передачей, а C — множество всех синих автомобилей с ручной передачей. В таком случае можно записать следующее:

$$C = A \cap B$$

Рис. 2.14. Пересечение множеств

Здесь символ $\cap$ представляет операцию пересечения множеств. Еще один способ записи этого утверждения может быть сформулирован в терминах элементов x следующим образом:

$$C = \{x \in U \mid (x \in A) \wedge (x \in B)\}$$

В этом выражении U обозначает универсальное множество.

Символ | читается “такой, что”. Вместо символа | иногда используется двоеточие (:). Символ $\wedge$ обозначает логическую операцию AND.

Приведенное выше выражение, определяющее множество C, читается так: C состоит из элементов x универсума U, таких, что x является элементом A и x является элементом B. Логическая связка AND впервые была определена в булевой алгебре. Выражение, состоящее из двух операндов, связанных знаком логической операции AND, является истинным тогда и только тогда, когда оба операнда являются истинными. Если множества A и B не имеют общих элементов, то $A \cap B = \varnothing$, где греческая буква фи ($\varnothing$) представляет **пустое множество**, или **множество меры нуль**, $\{\}$, которое не содержит элементов. Иногда для обозначения пустого множества используется греческая буква лямбда (Λ). В качестве других обозначений для универсального множества применяется цифра 1, а пустого множества — цифра 0. Хотя пустое множество не имеет элементов, оно все еще остается множеством. В качестве примера можно привести ресторан с множеством клиентов. Если в ресторан никто не пришел, то множество клиентов пусто, но ресторан из-за этого не перестает существовать.

К другим операциям с множествами относится **объединение**, которое представляет собой множество всех элементов, принадлежащих либо множеству A, либо множеству B, либо обоим множествам одновременно:

$$A \cup B = \{x \in U \mid (x \in A) \vee (x \in B)\}$$

В этом выражении символ $\cup$ представляет операцию объединения множеств.

Символ $\vee$ обозначает логическую операцию OR.

Дополнением множества A является множество всех элементов, не принадлежащих к A:

$$A' = \{x \in U \mid \sim(x \in A)\}$$

В этом выражении штрих ($'$) обозначает дополнение множества.

Символ тильды ($\sim$) представляет логический оператор NOT.

Диаграммы Венна, представляющие основные операции над множествами, показаны на рис. 2.15.

2.12 Пропозициональная логика

Старейшим и одним из простейших типов **формальной логики** является силлогизм. Термин *формальный* означает, что логика, к которой он относится, распространяется только на форму логических утверждений, но не учитывает их значения. Иными словами, в формальной логике рассматривается только синтаксис утверждений, а не их семантика. Такое свойство логики становится чрезвычайно важным при создании экспертных системах, поскольку позволяет отделить

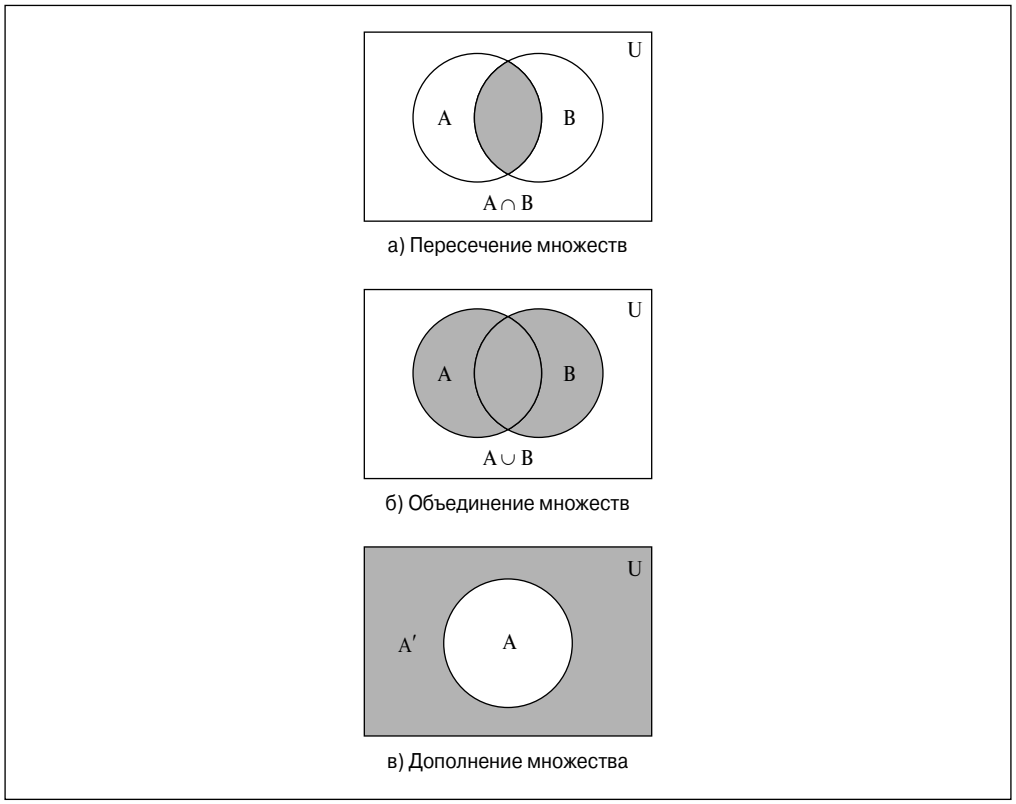

Рис. 2.15. Диаграммы Венна, представляющие основные операции с множествами

знания от рассуждений. Дело в том, что иногда высказывания, которые внешне выглядят как рассуждения, могут представлять собой знания. Например, утверждение "Президент всегда прав, поскольку он никогда не бывает неправ" внешне напоминает рассуждение в связи с наличием в нем слова "поскольку", которое связывает две части предложения. Но фактически утверждение подобного типа представляет собой **тавтологию**, поскольку, в отличие от фактов, которые могут быть в реальном мире истинными или не истинными, тавтология всегда в чисто логическом смысле истинна, так как ссылается для доказательства на саму себя. В действительности в этой тавтологии утверждается "X есть X". Но если вы не принадлежите к той же политической партии, что и Президент, то можете не согласиться с этим утверждением исходя из его семантики, а не формы, и предпочесть другую тавтологию: "Президент всегда неправ, поскольку он никогда не бывает прав".

Безусловно, для многих термин *формальная логика* звучит устрашающе, но логика не сложнее, чем алгебра. В действительности алгебра фактически пред-

ставляет собой формальную логику чисел. Например, предположим, что вас попросили решить такую задачу: в школе имеются 25 компьютеров с общим количеством микросхем памяти, равным 60. В некоторых компьютерах имеется две микросхемы памяти, в других — четыре. Каково количество компьютеров того и другого типа?

Решение этой задачи может быть записано алгебраически следующим образом:

$$25 = X + Y$$
$$60 = 2X + 4Y$$

После чего эту систему уравнения можно легко решить и получить $X = 20$ и $Y = 5$.

А теперь рассмотрим следующую задачу. На скотном дворе имеется 25 животных с общим количеством ног, равным 60. Некоторые из этих животных имеют две ноги, а другие — четыре. (*Примечание.* Наш трехногий слон Клайд находится в Африке, а не на этом скотном дворе.) Сколько имеется животных каждого типа? Вполне очевидно, что могут применяться одни и те же алгебраические уравнения, идет ли речь о компьютерах, животных или о чем-либо другом. По такому же принципу, с помощью которого алгебраические уравнения позволяют нам сосредоточиться на математических манипуляциях с символами без учета того, что они собой представляют, формальная логика позволяет сосредоточиться на рассуждениях, не погружаясь в путаницу, связанную с определением нюансов того, о каких объектах мы рассуждаем.

В качестве примера применения формальной логики рассмотрим силлогизм с бессмысленными словами сквиг и муф.

```
Посылка. Все сквиги являются муфами
Посылка. Джон - сквиг
Заключение. Джон - муф
```

Безусловно, слова сквиг и муф не имеют смысла и значения, но форма приведенного выше доказательства из-за этого не перестает быть правильной. Это означает, что рассматриваемое доказательство остается **действительным**, независимо от того, какие слова в нем используются, поскольку применяемый в нем силлогизм имеет действительную форму. Фактически действительным является любой силлогизм в следующей форме, независимо от того, какие слова будут подставлены вместо X, Y и Z:

```
Посылка. Все X суть Y
Посылка. Z есть X
Заключение. Z есть Y
```

Этот пример показывает, что в формальной логике смысл не учитывается и имеет значение только форма или внешнее представление. Логика становится

таким мощным инструментом именно благодаря используемой в ней концепции отделения формы от смысла, или семантики. В результате отделения формы от семантики появляется возможность объективно оценивать, является ли доказательство действительным, не испытывая воздействия предубеждений, вызванных семантикой. В качестве аналогии для формальной логики может рассматриваться алгебра, в которой правильность таких выражений, как $X + X = 2X$, остается неоспоримой, независимо от того, обозначает ли X целое число, количество яблок или аэропланов.

Аристотелевы силлогизмы оставались фундаментом логики до 1847 года, пока английский математик Джордж Буль (George Boole) опубликовал первую книгу с описанием **символической логики**. Безусловно, Лейбниц (Leibnitz) разработал свою собственную версию символической логики в XVII веке, но эта версия так и не получила широкого распространения. Одним из новых понятий, предложенных Булем, явилась модификация аристотелева представления, называемого **экзистенциальным значением**, согласно которому субъект рассуждений должен существовать. В соответствии с классическими аристотелевыми взглядами такое высказывание, как “все русалки хорошо плавают”, не может использоваться как посылка или следствие, поскольку русалки не существуют. А булевы представления, получившие теперь название *современных представлений*, ослабляют указанное ограничение. Важность современных представлений состоит в том, что они позволяют рассуждать о пустых классах объектов. Например, такое высказывание, как “все жесткие диски, которые отказывают, являются дешевыми”, не может использоваться в аристотелевом силлогизме, если мы не можем вести речь хотя бы об одном фактически отказавшем диске.

Еще один вклад, сделанный Булем, состоял в том, что этот ученый дал определение понятия множества **аксиом**. Формулировки аксиом состоят из символов, применяемых для представления объектов и классов, а также алгебраических операций, применяемых для манипулирования этими символами. Аксиомы представляют собой фундаментальные определения, на основании которых создаются сами логические системы, такие как математика и логика. А используя лишь одни аксиомы, можно создавать теоремы. *Теорема* — это утверждение, которое может быть доказано путем демонстрации того, что оно следует из аксиом. В период между 1910 и 1913 годами Уайтхед (Whitehead) и Расселл (Russell) опубликовали свою монументальную трехтомную работу *Principia Mathematica*, состоящую из 2000 страниц, в которой было показано, что формальная логика может служить основанием математики. Публикация указанной работы была признана важной вехой в развитии математики, поскольку Уайтхед и Расселл показали в ней, как перенести всю математику на твердое основание (а не прибегать к интуиции), полностью отказавшись от смыслового толкования арифметики и вместо этого сосредоточившись на внешних формах и манипуляциях с ними. По этой причине

математику иногда в шутку называют “способом нанесения на бумагу бессмысленных знаков”.

В 1931 году знаменитый математик Гёдель (Gödel) доказал, что не всегда возможно обосновать внутреннюю согласованность и отсутствие противоречий в формальных системах, основанных на аксиомах. Доказательство Гёделя вызвало значительное замешательство среди математиков, которые надеялись на то, что когда-либо удастся раз и навсегда доказать истинность арифметики. Но это не означает, что (как могло бы показаться неосведомленному человеку) мы фактически не можем быть уверены в правильности арифметических расчетов (следует отметить, что любой профессиональный математик, читая эти разглагольствования, найдет их весьма забавными).

Пропозициональная логика, иногда называемая **пропозициональным исчислением**, — это символическая логика, применяемая для манипулирования высказываниями. В частности, пропозициональная логика может использоваться для манипулирования **логическими переменными**, обозначающими высказывания. Безусловно, большинство людей считают, что исчисление есть нечто подобное математическим системам, которые изобретены Ньютоном и Лейбницем, но этот термин имеет более общий смысл. Аналог термина *исчисление* в английском языке (calculus) произошел от латинского слова, которое обозначало маленький камешек, используемый при выполнении расчетов. В своем наиболее общем смысле термин *исчисление* обозначает специальную систему манипулирования символами. С другой стороны, для обозначения пропозициональной логики применяются также такие термины, как **исчисление утверждений** и **исчисление высказываний**. Вообще говоря, в пропозициональной логике рассматривается определенный тип предложений естественного языка, а сами эти предложения, как показано в табл. 2.2, подразделяются на четыре основных типа.

Таблица 2.2. Типы предложений

Тип	Пример
Повелительные	Сделайте то, что я вам говорю!
Вопросительные	Что это?
Восклицательные	Это великолепно!
Повествовательные	Квадрат имеет четыре равные стороны

В пропозициональной логике рассматриваются подмножества предложений, представляющих собой повествовательные предложения, которые могут подразделяться на истинные или ложные. Очевидно, что предложение “Квадрат имеет четыре равные стороны” обладает истинностным значением *истина*, а предложение “Джордж Вашингтон был вторым по счету президентом США” обладает истинностным значением *ложь*. Предложение, истинностное значение которого

может быть определено, называется **утверждением**, или **высказыванием**. Высказывание принято называть также **закрытым предложением**, поскольку его истинностное значение не подлежит обсуждению.

Если бы авторы снабдили настоящую книгу предисловием, в котором было бы сказано "все, что содержится на этих страницах — ложно", такое предложение нельзя было бы рассматривать ни как истинное, ни как ложное. Если бы оно было истинным, то сказанное в предисловии было бы истинным, а это невозможно. Если же такое предложение не является истинным, то все, что содержится в книге, должно быть истинным; и это означает, что высказанное предложение ложно, а это также не может быть истинным. Предложения подобного типа известны как формулировки **парадокса лжеца**. Предложения, на которые невозможно дать однозначный ответ, называются **открытыми предложениями**. Например, открытым является предложение "Шпинат имеет превосходный вкус", поскольку оно является истинным для одних людей и ложным для других. Предложение "Он имеет высокий рост" также является открытым, поскольку в нем содержится местоимение "он", а не указан конкретный человек. Истинностное значение не может быть присвоено открытому предложению до тех пор, пока не станет известно конкретное лицо (или персонаж), на которое указывает это местоимение. Еще одно затруднение, возникающее при анализе рассматриваемого предложения, состоит в определении смысла выражения "высокий рост". Человек, который для одних кажется высоким, другим может не показаться таковым. С подобной неоднозначностью понятия "высокий рост" невозможно справиться с помощью пропозициональной логики или логики предикатов, но такие проблемы легко решаются с использованием нечеткой логики, которая рассматривается в главе 5.

Путем применения логических связок к отдельным высказываниям могут быть сформированы **составные высказывания**. Наиболее широко применяемые логические связки показаны в табл. 2.3.

Таблица 2.3. Широко применяемые логические связки

Связка	Значение
$\wedge$	AND; конъюнкция
$\vee$	OR; дизъюнкция
$\sim$	NOT; отрицание
$\rightarrow$	если... то...; условная операция
$\leftrightarrow$	если и только если; двусторонняя условная операция

Строго говоря, знак операции отрицания — это не связка, поскольку отрицание представляет собой унарную операцию, применяемую к одному операнду, следующему за знаком этой операции, т.е. фактически знак операции отрицания ничего не связывает. Операция отрицания имеет более высокий приоритет по сравнению

с другими операциями, поэтому нет необходимости задавать круглые скобки вокруг выражения, в котором она применяется. Таким образом, выражение $\sim p \wedge q$ означает то же, что и выражение $(\sim p) \wedge q$.

Условная операция аналогична стрелке продукционных правил в том, что также может быть представлена в форма `IF-THEN`. Например, следующее выражение:

```
IF идет дождь THEN захвати с собой зонтик
```

может быть представлено в такой форме:

$$p \rightarrow q$$

в которой применяются следующие обозначения:

p — идет дождь

q — захвати с собой зонтик

Иногда вместо символа $\rightarrow$ используется символ $\supset$. Для обозначения условной операции применяется также другой термин — **материальная импликация**.

Двусторонняя условная операция, $p \leftrightarrow q$, эквивалентна приведенному ниже выражению и является истинной только тогда, когда p и q имеют одинаковые истинностные значения.

$$(p \rightarrow q) \wedge (q \rightarrow p)$$

Таким образом, выражение $p \leftrightarrow q$ является истинным, только если p и q одновременно имеют истинное значение или ложное значение. Двусторонняя условная операция имеет такие толкования:

p если и только если q

q если и только если p

если p то q, и если q то p

Как уже было сказано выше, тавтология — это составное высказывание, которое всегда является истинным, независимо от того, истинны или ложны отдельные высказывания, входящие в его состав. С другой стороны, **противоречие** — это составное высказывание, которое всегда является ложным. А **контингентными** называются высказывания, которые не представляют собой ни тавтологию, ни противоречие. Тавтологии и противоречия называются соответственно аналитически истинными и аналитически ложными высказываниями, поскольку их истинностное значение может быть определено исключительно на основании анализа формы. Например, истинностная таблица для выражения $p \vee \sim p$ показывает, что это — тавтология, а для выражения $p \wedge \sim p$, что это — противоречие.

Если условная операция одновременно представляет собой тавтологию, то эта операция именуется **импликацией** и содержит символ $\Rightarrow$ вместо символа $\rightarrow$.

Двусторонняя условная операция, которая одновременно представляет собой тавтологию, называется **логической эквивалентностью**, или **материальной эквивалентностью**, и символически обозначается с помощью символа $\Leftrightarrow$ или $\equiv$. Два логически эквивалентных высказывания всегда имеют одно и то же истинностное значения. Например, $p \equiv \sim \sim p$.

К сожалению, не существует единственного возможного определения для импликации, поскольку количество истинностных таблиц для двух переменных, которые могут принимать истинные и ложные значения, равно 16. Поэтому в ранних экспертных системах, которые разрабатывались в 1980-х годах, применялись одиннадцать разных определений для операции импликации.

Условная операция не имеет точно такой же смысл, как и выражение `IF-THEN` в процедурном языке или в экспертной системе, основанной на правилах. В процедурных языках и в экспертных системах конструкция `IF-THEN` указывает, что если истинны условия в определении `IF`, то должны быть выполнены действия, описания которых следует за ключевым словом `THEN`. А в логике значение условной операции определяется по ее истинностной таблице. Смысл такой операции может быть переведен на естественный язык многими способами. Например, если дано следующее выражение:

$$p \rightarrow q$$

в котором p и q представляют собой любые высказывания, то данное выражение может быть переведено на естественный язык таким образом:

p влечет за собой q

если p то q

q только если p

p является достаточным для q

q если p

q является необходимым для p

Например, допустим, что через p обозначено высказывание “гражданин в возрасте 18 лет или старше”, а q обозначает высказывание “имеет право голосовать”. Условное выражение $p \rightarrow q$ может обозначать любое из следующих предложений:

```
гражданин находится в возрасте 18 лет или старше, и это
влечет за собой, что данный гражданин имеет право
голосовать если гражданин находится в возрасте 18 лет или
старше, то данный гражданин имеет право голосовать
гражданин имеет право голосовать, только если этот
гражданин находится в возрасте 18 лет или старше
гражданин находится в возрасте 18 лет или старше, и этого
достаточно, чтобы данный гражданин имел право голосовать
```

```
гражданин имеет право голосовать, если этот гражданин
находится в возрасте 18 лет или старше
гражданин должен иметь возраст 18 лет или старше,
и это условие является необходимым для того, чтобы он имел
право голосовать
```

В некоторых случаях потребовалось изменить формулировки, чтобы полученные предложения стали грамматически правильными предложениями естественного языка. В последнем предложении говорится о том, что если не выполняется требование, обозначенное как q, то не выполняется и p. Это предложение можно представить на правильном естественном языке так: “Чтобы получить возможность голосовать, необходимо иметь возраст 18 лет или старше”.

Значения бинарных логических связок показаны в табл. 2.4. Эти связки называются бинарными, так как требуют двух операндов. Связка отрицания ($\sim$) представляет собой знак унарной операции, применяемой к одному операнду, который следует за этим знаком (табл. 2.5).

Таблица 2.4. Истинностная таблица для бинарных логических связок

p	q	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$
T	T	T	T	T	T
T	F	F	T	F	F
F	T	F	T	T	F
F	F	F	F	T	T

Таблица 2.5. Истинностная таблица для связки отрицания

p	$\sim p$
T	F
F	T

Множество логических связок называется **адекватным**, если с помощью связок, взятых только из этого множества, можно представить любую истинностную функцию. К примерам адекватных множеств относятся $\{\sim, \wedge, \vee\}$, $\{\sim, \wedge\}$, $\{\sim, \vee\}$ и $\{\sim, \rightarrow\}$.

Множество, содержащее единственный элемент, называется **одноэлементным множеством**. Одноэлементными являются два из адекватных множеств. Эти множества включают связки NOT-OR (**NOR**) и NOT-AND (**NAND**). Множество со связкой NOR обозначается как $\{\downarrow\}$, а множество со связкой NAND — как $\{|\}$. Знак операции (|) называется **штрихом**, или **дизъюнкцией отрицаний**. Этот знак используется для отрицания того, что p и q одновременно являются истинными.

Таким образом, в выражении $p \mid q$ утверждается, что по меньшей мере одно из высказываний p или q является истинным. А в **операции конъюнкции отрицаний** ($\downarrow$) отрицается то, что является истинным либо p, либо q. Это означает, что в выражении $p \downarrow q$ утверждается, что p и q одновременно являются ложными.

2.13 Логика предикатов первого порядка

Безусловно, пропозициональная логика является очень полезной, но имеет целый ряд ограничений. Основная проблема состоит в том, что пропозициональная логика может применяться только с полным высказыванием. Это означает, что с ее помощью нельзя исследовать внутреннюю структуру высказывания. Пропозициональная логика не позволяет даже доказать обоснованность примерно такого силлогизма:

```
Все люди смертны
Все женщины - люди
Следовательно, все женщины смертны
```

В целях обеспечения возможности анализировать более общие случаи была разработана **логика предикатов**. В своей простейшей форме она принимает вид логики предикатов **первого порядка** и является основой таких языков логического программирования, как PROLOG. В настоящем разделе для обозначения логики предикатов первого порядка будет применяться просто термин *логика предикатов*. Пропозициональная логика — это подмножество логики предикатов.

Логика предикатов позволяет рассматривать внутреннюю структуру предложений. В частности, в ней допускается использование таких специальных слов, как “все”, “некоторые” и “ни один”, называемых **кванторами**. Эти слова очень важны, поскольку позволяют присваивать явные количественные оценки другим словам и более точно формулировать предложения. Все кванторы отвечают на вопрос “сколько” и поэтому позволяют применять более широкий круг выражений по сравнению с пропозициональной логикой.

2.14 Квантор всеобщности

Предложение с квантором всеобщности, или универсально квантифицированное предложение, принимает одно и то же истинностное значение при подстановке всех объектов из одной и той же области определения. **Квантор всеобщности** представляется с помощью символа $\forall$, за которым следует один или несколько параметров, соответствующих **переменным области определения**. Символ $\forall$ интерпретируется как “для каждого” или “для всех”. Например, в области определения чисел следующее выражение гласит о том, что для каждого x (где x —

число) выражение $x + x = 2x$ является истинным:

$$(\forall x)(x + x = 2x)$$

А если указанное выражение будет обозначено символом p, то приведенное выше утверждение может быть представлено еще более кратко следующим образом:

$$(\forall x)(p)$$

В качестве еще одного примера предположим, что p обозначает предложение "все собаки — животные", как показано ниже.

$$(\forall x)(p) \equiv (\forall x)(\text{если } x - \text{собака} \rightarrow x - \text{животное})$$

Противоположным по отношению к этому предложению является предложение "ни одна собака не является животным", которое записывается следующим образом:

$$(\forall x)(\text{если } x - \text{собака} \rightarrow \sim x - \text{животное})$$

Это высказывание можно также записать таким образом:

```
Каждая собака не является животным
Все собаки не являются животными
```

В качестве еще одного примера укажем, что предложение "все треугольники являются многоугольниками" записывается следующим образом:

$$(\forall x)(x \text{ является треугольником} \rightarrow x \text{ является многоугольником})$$

Это высказывание можно прочитать так: "Для всех x, если x — треугольник, то x — многоугольник". Более короткий способ записи логических высказываний, в которых участвуют предикаты, состоит в использовании **предикативных функций** для описания свойств рассматриваемого предмета. Поэтому приведенное выше логическое высказывание можно также записать следующим образом:

$$(\forall x)(\text{triangle}(x) \rightarrow \text{polygon}(x))$$

Предикативные функции обычно записываются с применением более краткой системы обозначений, в которой предикаты обозначаются прописными буквами. Например, предположим, что T обозначает треугольник, а P — многоугольник. В таком случае утверждение, касающееся треугольников, может быть записано более кратко, как показано ниже.

$$(\forall x)(T(x) \rightarrow P(x)) \text{ или } (\forall y)(T(y) \rightarrow P(y))$$

Следует отметить, что вместо фиктивных переменных x и y можно использовать любые другие переменные. В качестве еще одного примера предположим,

что H — предикативная функция, обозначающая людей, а M — функция, обозначающая смертных. В таком случае утверждение, согласно которому все люди смертны, можно записать таким образом:

$$(\forall x)(H(x) \rightarrow M(x))$$

Это утверждение читается так: для всех x, если x — человек, то x смертен. Как показано на рис. 2.16, это предложение логики предикатов может быть также представлено с помощью семантической сети. Кроме того, оно может быть выражено в терминах правил:

```
IF x - человек
THEN x смертен
```

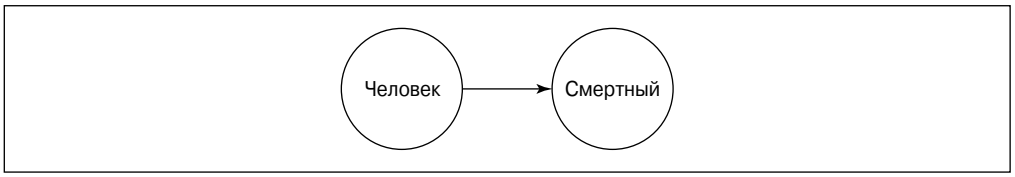

Рис. 2.16. Представление утверждения логики предикатов в виде семантической сети

Квантор всеобщности может также интерпретироваться как конъюнкция предикатов, относящихся к отдельным экземплярам. Как было указано выше, экземпляром называется конкретный случай. Например, допустим, что собака по имени Спарклер представляет собой конкретный экземпляр класса собак. Эту мысль можно выразить следующим образом:

`Dog(Sparkler)`

В данном примере `Dog` — предикативная функция, а `Sparkler` — экземпляр.

Такое предложение логики предикатов:

$$(\forall x)P(x)$$

может интерпретироваться в терминах экземпляров a_i, как показано ниже.

$$P(a_1) \wedge P(a_2) \wedge P(a_3) \wedge \cdots \wedge P(a_N)$$

В этом случае многоточие указывает, что действие предиката распространяется на все элементы данного класса. Таким образом, в приведенном выше выражении сказано, что предикат применяется ко всем экземплярам класса.

В выражениях может использоваться несколько кванторов. Например, как показано ниже, для формулировки закона коммутативности сложения для чисел требуются два квантора.

$$(\forall x)(\forall y)(x + y = y + x)$$

В этом выражении утверждается, что “для каждого x и для каждого y сумма x и y равна сумме y и x”.

2.15 Квантор существования

Квантором еще одного типа является **квантор существования**. Квантор существования определяет утверждение как истинное применительно по крайней мере к одному элементу области определения. Он представляет собой ограниченную форму квантора всеобщности (в котором утверждается, что некоторое выражение является истинным для всех элементов области определения). Квантор существования записывается как символ $\exists$, за которым следует одна или несколько переменных, например, как показано ниже.

$$(\exists x)(x \cdot x = 1)$$
$$(\exists x)(\text{elephant}(x) \wedge \text{name}(Clyde))$$

В первом из приведенных выше предложений указано, что имеется некоторое число x, результат умножения которого на самого себя равен 1. Во втором выражении сказано, что существует некоторый слон по кличке Клайд.

Квантор существования можно прочитать несколькими способами, в частности, таким образом:

```
существует
по меньшей мере один
для некоторых
имеется некоторый
некоторые
```

В качестве еще одного примера можно привести выражение, в котором указано, что все слоны имеют четыре ноги:

$$(\forall x)(\text{elephant}(x) \rightarrow \text{four-legged}(x))$$

А утверждение, что некоторые слоны имеют три ноги, записывается со знаком логической операции AND и квантором существования следующим образом:

$$(\exists x)(\text{elephant}(x) \wedge \text{three-legged}(x))$$

Так же как квантор всеобщности может быть выражен с помощью конъюнкции, квантор существования может быть выражен с помощью дизъюнкции экземпляров, a_i:

$$P(a_1) \vee P(a_2) \vee P(a_3) \vee \cdots \vee P(a_N)$$

В табл. 2.6 в качестве примеров приведены утверждения с кванторами, в которых P обозначает предложение “слоны — млекопитающие”, и их отрицания.

Числа в круглых скобках обозначают примеры, которые будут рассматриваться в последующем описании.

Таблица 2.6. Примеры отрицаемых кванторов

Пример	Значение
(1a) $(\forall x)(P)$	Все слоны — млекопитающие
(1b) $(\exists x)(\sim P)$	Некоторые слоны не млекопитающие
(2a) $(\exists x)(P)$	Некоторые слоны — млекопитающие
(2b) $(\forall x)(\sim P)$	Никакие слоны не млекопитающие

Примеры (1a) и (1b) являются отрицаниями по отношению друг к другу; таковыми являются также примеры (2a) и (2b). Обратите внимание на то, что отрицанием выражения с квантором всеобщности из примера (1a) становится выражение с отрицанием предложения P с квантором существования, как показано в примере (1b). Аналогичным образом отрицание выражения с квантором существования из примера (2a) представляет выражение с отрицанием предложения P и квантором всеобщности, как показано в примере (2b).

2.16 Кванторы и множества

Кванторы и множества могут использоваться для определения подмножеств универсального множества U, как показано в табл. 2.7.

Таблица 2.7. Некоторые выражения с множествами и их логические эквиваленты

Выражение с множествами	Логический эквивалент
$A = B$	$\forall x(x \in A \leftrightarrow x \in B)$
$A \subseteq B$	$\forall x(x \in A \rightarrow x \in B)$
$A \cap B$	$\forall x(x \in A \wedge x \in B)$
$A \cup B$	$\forall x(x \in A \vee x \in B)$
A'	$\forall x(x \in U \mid \sim(x \in A))$
U (универсальное множество)	T (истинное значение)
$\varnothing$ (пустое множество)	F (ложное значение)

Отношение, согласно которому A является строгим подмножеством B (имеющее вид $A \subset B$), означает, что все элементы множества A принадлежат к множеству B, но, в свою очередь, в множестве B имеется хотя бы один элемент, не принадлежащий к множеству A. Предположим, что E обозначает всех слонов, а M — всех млекопитающих. В таком случае следующее отношение между множествами представляет собой утверждение, что все слоны — млекопитающие, но не все

млекопитающие являются слонами:

$$E \subset M$$

Обозначив множество животных серого цвета как G, а множество животных с четырьмя ногами — как F, можно записать утверждение, что все серые и четырехногие слоны являются млекопитающими, следующим образом:

$$(E \cap G \cap F) \subset M$$

Ниже приведены некоторые примеры предложений с кванторами, в которых используются указанные обозначения.

E — слоны
R — рептилии
G — серые
F — четырехногие
D — собаки
M — млекопитающие

Никакие слоны не являются рептилиями

$$E \cap R = \varnothing$$

Некоторые слоны - серые

$$E \cap G \neq \varnothing$$

Никакие слоны не являются серыми

$$E \cap G = \varnothing$$

Некоторые слоны не являются серыми

$$E \cap G' \neq \varnothing$$

Все слоны являются серыми и четырехногими

$$E \subset (G \cap F)$$

Все слоны и собаки являются млекопитающими

$$(E \cup D) \subset M$$

Некоторые слоны являются четырехногими и серыми

$$(E \cap F \cap G) \neq \varnothing$$

Между формой представления информации с помощью множеств и логической формой имеется еще одна аналогия, которая выражается в виде законов де Моргана, как показано в табл. 2.8. Символ эквивалентности ($\equiv$), т.е. знак двусторонней условной операции, означает, что выражение, находящееся слева от него, имеет такое же истинностное значение, как и выражение, находящееся справа. Это означает, что указанные выражения эквивалентны.

Таблица 2.8. Законы де Моргана, представленные в форме с использованием множеств и логической форме

Форма с использованием множеств	Логическая форма
$(A \cap B)' \equiv A' \cup B'$	$\sim(p \wedge q) \equiv \sim p \vee \sim q$
$(A \cup B)' \equiv A' \cap B'$	$\sim(p \vee q) \equiv \sim p \wedge \sim q$

2.17 Ограничения логики предикатов

Безусловно, логика предикатов применима в ситуациях многих типов, но некоторые типы утверждений невозможно представить на основе логики предикатов с использованием кванторов всеобщности и кванторов существования. Например, в логике предикатов невозможно выразить следующее утверждение:

`Большинство учащихся в классе получили оценки отлично`

В этом утверждении квантор “большинство” означает “больше половины”.

Квантор “большинство” не может быть выражен в терминах квантора всеобщности и квантора существования. Для реализации квантора “большинство” в логике должны быть предусмотрены некоторые предикаты, обеспечивающие подсчет количества элементов, что возможно при использовании нечеткой логики, описанной в главе 5. Еще одно ограничение логики предикатов состоит в том, что она не позволяет выражать зависимости, которые могут быть истинными только иногда, но не всегда. Указанная проблема также может быть решена с помощью нечеткой логики. Однако внедрение в логическую систему средств проведения вычислений влечет за собой также появление дополнительных усложнений; к тому же в результате логика начинает в большей степени напоминать математику.

2.18 Резюме

В настоящей главе приведены основные сведения о логике, представлении знаний и некоторых методах представления знаний. Выбор правильного метода представления знаний в экспертных системах имеет очень важное значение.

Рассматривались также логические ошибки, поскольку для инженера по знаниям важно понимать правила предметной области и не путать форму представления знаний с семантикой. Если не заданы формальные правила, то экспертная система может не позволить достичь правильных заключений, что повлечет за собой губительные последствия при эксплуатации таких ответственных систем, от которых зависит жизнь и собственность людей.

С точки зрения логики знания могут классифицироваться многими способами и рассматриваться как априорные, апостериорные, процедурные, декларативные и неявные. К числу методов, широко применяемых в экспертных системах для представления знаний, относятся продукционные правила, объекты, семантические сети, схемы, фреймы и логика. Каждый из указанных подходов имеет свои преимущества и недостатки. Прежде чем приступать к проектированию экспертной системы, необходимо принять решение о том, какой из способов представления знаний может стать основой выбора подхода к решению рассматриваемой задачи. Следует не пытаться применять одно и то же инструментальное средство для решения всех задач, а выбирать каждый раз наилучшее средство. Кроме того, в данной главе речь шла о том преимуществе языка CLIPS, что он позволяет представлять знания не только с помощью объектов, но и с помощью правил. Многочисленные ссылки, с помощью которых можно ознакомиться с дополнительной информацией о логике, знаниях и логических ошибках приведены в приложении Ж. В приложениях А–В содержатся полезные справочные сведения об эквивалентностях, кванторах и свойствах множеств.

Задачи

2.1. Нарисуйте семантическую сеть для классификации компьютеров с использованием связей AKO и IS-A. Рассмотрите такие классы, как микрокомпьютеры; мэйнфреймы; суперкомпьютеры; вычислительные системы; выделенные компьютеры; компьютеры общего назначения; одноплатные компьютеры; компьютеры, реализованные в виде одной микросхемы; однопроцессорные и многопроцессорные компьютеры. Включите данные о конкретных экземплярах.

2.2. Нарисуйте семантическую сеть для классификации средств обеспечения межкомпьютерного взаимодействия с использованием связей AKO и IS-A. Рассмотрите такие классы, как локальные сети, глобальные сети, сети с маркерным кольцом, звездообразные сети, централизованные сети, децентрализованные сети, распределенные сети, сети с модемными линиями связи, телекоммуникационные сети, группы новостей и электронная почта. Включите данные о конкретных экземплярах.

2.3. Нарисуйте систему фреймов для описания здания, в котором вы проходите обучение. Предусмотрите возможность представить данные об офисах, аудиториях, лабораториях и т.д. Включите данные о конкретных экземплярах с заполненными слотами для одного фрейма каждого типа, такого как офис и аудитория.

2.4. Нарисуйте систему фреймов действия, позволяющую узнать, какие действия следует предпринять в случае аппаратного отказа конкретной компьютерной системы. Рассмотрите возможность аварийного отказа диска, источника электропитания, процессора и памяти.

2.5. Нарисуйте диаграммы Венна и обозначьте отдельные элементы каждой диаграммы как термы выражений, в которых используются описанные ниже операции с множествами. В приведенных примерах $\{1, 2\}$ — первое множество, а $\{2, 3\}$ — второе.

а) Результат операции "исключительное ИЛИ" с двумя множествами, A и B, состоит из всех элементов, которые находятся или в одном, или в другом, но не обоих множествах одновременно. Операцию "исключительное ИЛИ" называют также **симметрической разностью множеств** и обозначают знаком операции $(/)$. Например:

$$\{1, 2\}/\{2, 3\} = \{1, 3\}$$

б) Результат выполнения операции **разности множеств** с двумя множествами, которая обозначается знаком операции $(-)$, состоит из всех элементов первого множества, которые не принадлежат также ко второму множеству. Например:

$$\{1, 2\} - \{2, 3\} = \{1\}$$

2.6. Составьте истинностные таблицы и определите, какие из следующих выражений представляют собой тавтологии, противоречия или **контингентные утверждения**, и какие не относятся ни к одному из этих типов. При выполнении упражнений а) и б) вначале представьте рассматриваемые утверждения с помощью логических символов и связок.

а) Если я пройду этот курс и получу оценку "отлично", то я пройду этот курс или получу оценку "отлично".

б) Если я пройду этот курс, то получу оценку "отлично"

и

я пройду этот курс и не получу оценку "отлично".

в) $((A \wedge \sim B \rightarrow (C \wedge \sim C)) \rightarrow (A \rightarrow \sim B)$.

г) $(A \to B) \wedge (\sim B \vee C) \wedge (A \wedge \sim C)$.

д) $A \to \sim B$ (контингентное утверждение).

2.7. Два предложения логически эквивалентны тогда и только тогда, когда они имеют одно и то же истинностное значение. Таким образом, если A и B — любые утверждения, то следующее выражение с двусторонним условным оператором становится истинным при любых значениях утверждений, входящих в его состав, т.е. превращается в тавтологию:

$$A \leftrightarrow B \text{ или эквивалентность } A \equiv B$$

Определите, являются ли два приведенных ниже предложения логически эквивалентными, записав их с использованием логических символов, а также определите, показывает ли истинностная таблица двусторонней условной операции с этими предложениями, что полученное выражение представляет собой тавтологию.

`Если вы едите банановый сплит, то не можете есть торт.`
`Если вы едите торт, то не можете есть банановый сплит.`

2.8. Запишите логическое выражение, эквивалентное выражениям с соответствующими операциями разности множеств и симметрической разности множеств.

2.9. Покажите, что следующие эквивалентности соблюдаются для любых множеств A, B, C и пустого множества $\varnothing$.

а) $(A \cup B) \equiv (B \cup A)$.

б) $(A \cup B) \cup C \equiv A \cup (B \cup C)$.

в) $A \cup \varnothing \equiv A$.

г) $A \cap B \equiv B \cap A$.

д) $A \cap A' \equiv \varnothing$.

2.10. Запишите приведенные ниже утверждения в квантифицированной форме.

а) Все собаки — млекопитающие.

б) Никакая собака не является слоном.

в) Некоторые программы содержат ошибки.

г) Ни одна из моих программ не содержит ошибок.

д) Все ваши программы содержат ошибки.

2.11. **Степенны́м множеством** $P(S)$ множества S называется множество, все элементы которого представляют собой подмножества множества S. Множество $P(S)$ всегда содержит по меньшей мере такие элементы, как пустое множество $\varnothing$ и множество S.

а) Найдите степенное множество множества $A = \{2, 4, 6\}$.

б) Сколько элементов содержит степенное множество множества с N элементами?

2.12. Выполните следующие задания.

а) Запишите истинностную таблицу для выражений, представленных в табл. 2.9.

Таблица 2.9. Логические выражения, которым могут быть даны осмысленные определения

Осмысленное определение	Выражение
либо p либо q	$(p \vee q) \wedge \sim(p \wedge q)$
ни p ни q	$\sim(p \vee q)$
p если не q	$\sim q \rightarrow p$
p потому что q	$(p \wedge q) \wedge (q \rightarrow p)$
никакие p не есть q	$p \rightarrow \sim q$

б) Покажите, что $(p \vee q) \wedge \sim(p \wedge q) \equiv p/q$, где знак / обозначает операцию "исключительное ИЛИ".

2.13. Выполните следующие упражнения.

а) Запишите истинностные таблицы для операций NAND и NOR.

б) Докажите, что $\{\downarrow\}$ и $\{|\}$ — адекватные множества, представив связки $\sim$, $\wedge$ и $\vee$ в терминах связки $\downarrow$, а затем — связки $|$. Для этого составьте истинностные таблицы, чтобы показать, что следующие выражения представляют собой логические эквивалентности:

$$\sim \sim p \equiv p$$
$$(p \wedge q) \equiv (p \downarrow p) \downarrow (q \downarrow q)$$
$$\sim p \equiv p \mid p$$
$$(p \vee q) \equiv (p \mid p) \mid (q \mid q)$$

в) Поскольку $p \rightarrow q \equiv \sim(p \wedge \sim q)$, представьте выражение $p \rightarrow q$ с помощью знаков операции $\downarrow$.

г) Каковы преимущества и недостатки использования адекватных одноэлементных множеств, во-первых, с точки зрения удобства использования системы обозначений, и, во-вторых, конструирования электронных схем для микросхем?

2.14. Каковы преимущества и недостатки проектирования экспертной системы со знаниями, относящимися к нескольким предметным областям?

2.15. Объясните, почему приходится по-разному кипятить воду в высокогорной местности (например, в Денвере) и на равнине (например, в Хьюстоне), если в процессе кипячения воды готовится яйцо, которое должно быть сварено вкрутую. Относится ли эта задача к проблематике логики или физики?

2.16. Даны приведенные ниже операторы PROLOG; докажите, что Том — дедушка самого себя.

```
mother(pat,ann).              ; Пэт — мать Энн
parents(jim,ann,tom).         ; Джим и Энн — родители Тома
surrogatemother(pat,tom).     ; Пэт — суррогатная мать Тома
```