РОССИЙСКАЯ АССОЦИАЦИЯ
ИСКУССТВЕННОГО
ИНТЕЛЛЕКТА

Б. А. Кулик А. А. Зуенко А. Я. Фридман

АЛГЕБРАИЧЕСКИЙ ПОДХОД К ИНТЕЛЛЕКТУАЛЬНОЙ ОБРАБОТКЕ ДАННЫХ И ЗНАНИЙ

Санкт-Петербург
2010

Б.А. Кулик, А.А. Зуенко, А.Я. Фридман

АЛГЕБРАИЧЕСКИЙ ПОДХОД К ИНТЕЛЛЕКТУАЛЬНОЙ ОБРАБОТКЕ ДАННЫХ И ЗНАНИЙ

Санкт-Петербург
Издательство Политехнического университета
2010

УДК 004.657
К 90

Рецензенты:
Заместитель директора ИСА РАН, постоянный член Европейского комитета по искусственному интеллекту (ECCAI), доктор физико-математичсеских наук, профессор *Г.С. Осипов*
Ведущий научный сотрудник Санкт-Петербургского института информатики и автоматизации Российской академии наук *В.А. Дюк*

Кулик Б.А. Алгебраический подход к интеллектуальной обработке данных и знаний / Кулик Б.А., Зуенко А.А., Фридман А.Я.. – СПб.: Изд-во Политехн. ун-та, 2010. 235 с.

В книге представлен новый математический аппарат – алгебра кортежей (АК), которая относится к классу булевых алгебр и позволяет реализовать алгебраический подход к логическому анализу в системах искусственного интеллекта. В АК, в отличие от формальных систем, где основа – символьные конструкции, в качестве базового выбрано понятие “многоместное отношение” и предложены обобщения операций алгебры множеств для работы с отношениями, заданными в разных схемах. Это позволило расширить возможности существующих систем обработки данных и знаний, основанных на бинарных и реляционных отношениях. АК дает возможность унифицировать представление и анализ как данных, так и знаний, и, следовательно, решить проблему сопряжения баз данных и баз знаний в рамках одной программной системы. Алгоритмы обработки отношений, записанных в виде АК-объектов, хорошо поддаются распараллеливанию. Кроме того, разработаны дополнительные методы уменьшения трудоемкости, а в некоторых случаях – и вычислительной сложности интеллектуальных процедур. В алгебре кортежей, помимо известных методов логических исчислений, реализованы новые алгебраические методы проверки корректности следствия и поиска следствий из заданной системы аксиом. В ходе вывода учитывается внутренняя структура обрабатываемых знаний, что ускоряет решение стандартных задач логического анализа. Помимо логического вывода, АК служит для формализации широкого круга логических задач (абдуктивные и модифицируемые заключения, моделирование графов и семантических сетей, экспертных правил и т.д.).

Работа выполнена при поддержке грантов РФФИ (проекты 03-01-96142-р2003север_а, 09-07-00066), РГНФ (проект 09-02-43203а/С), проекта 2.3 Программы фундаментальных научных исследований ОНИТ РАН и проекта 4.3 "Интеллектуальные базы данных" Программы № 15 Президиума РАН.

Издание книги осуществлено за счет финансирования по проекту 4.3 "Интеллектуальные базы данных" Программы № 15 Президиума РАН.

ISBN 978-5-7422-2836-3

Предисловие

Математика – это язык!
Дж.У.Гиббс

В основе современной логики лежит математическая система, которая имеет несколько названий: формальный подход, аксиоматический метод, символическая логика, теория формальных систем. Здесь мы будем использовать последнее название (сокращенно ТФС). Этот подход начал развиваться в начале XX века, когда были открыты парадоксы теории множеств. Большинство расценило эти открытия как кризис в основаниях математики. Тогда многие математики, логики и философы решили, что только ТФС может стать защитой от парадоксов и заодно – основой всей математики и логики.

Активность защитников ТФС, среди которых были многие всемирно известные математики и философы (Б. Рассел, Л. Витгенштейн, Д. Гильберт, Дж. Пеано и др.), оказалась столь сильной, что развивавшийся в то время подход к основаниям логики на основе алгебры множеств, булевой алгебры и теории отношений стал постепенно утрачивать свое влияние. В середине XX века своеобразным каноном для приверженцев ТФС стало многотомное математическое сочинение группы известных математиков, публиковавшихся под коллективным псевдонимом Н. Бурбаки [*Бурбаки*, 1965]. В соответствии с этим каноном из оснований математики должны были исчезнуть такие "неточные", "интуитивные" понятия, как числа, пространства, геометрические фигуры, множества и т.д. По замыслу авторов этих сочинений, в основаниях математики возможны только символы и последовательности символов (предложения), слабо связанные с основными понятиями прикладной математики и предложениями на естественном языке [*Арнольд*, 2002].

Оказалось, что с помощью ТФС можно изложить не только классическую логику (к ней относятся теория доказательств, математическая логика и отчасти силлогистика), но и многочисленные варианты неклассической логики (многозначная, модальная, паранепротиворечивая, немонотонная и т.д.). В дальнейшем новые логики посыпались как из рога изобилия, и мало кого смущали следующие обстоятельства:

- многие из новых логик не имеют никакого прикладного значения и противоречат здравому смыслу;

- парадоксы, потрясшие всю математику в первой четверти XX века, так и остались необъяснимой загадкой;
- в качестве аксиом в некоторых логиках можно использовать не только бесспорные суждения, но и суждения, противоречивость которых видна невооруженным глазом (например, парапротиворечивые логики).

Язык математической логики есть частный случай ТФС. В системах искусственного интеллекта основные концепции ТФС отражены в виде ***декларативного подхода***, в котором знания выражаются в форме высказываний (или правил). В рамках этого подхода системы конструируются путем представления знаний на некотором формальном языке, а задачи решаются применением процессов логического вывода к знаниям.

Альтернативой декларативному является ***процедурный подход***, в котором правила или высказывания выражаются как алгоритмы и, в конечном итоге, в виде кода программы. В 1970-1980-х годах между приверженцами этих двух подходов происходили ожесточенные дебаты. Однако в дальнейшем многие исследователи пришли к выводу, что успешно действующие интеллектуальные системы должны сочетать в себе и декларативные, и процедурные элементы.

Развитие декларативного подхода сопровождается рядом трудностей и проблем, обусловленных его спецификой, включая следующие.

1. При использовании декларативного подхода многие задачи логического анализа необходимо сводить к задаче "выполнимость логической формулы", в которой возможны только два варианта ответа ("да" или "нет"). Этот процесс сведения, несмотря на большой объем позитивных результатов в этой области, весьма непрост. К тому же во многих случаях, когда требуется не только ответить на вопрос типа "да или нет?", но и оценить значения параметров системы или состав и число объектов, удовлетворяющих заданным условиям, он оказывается нереализуемым. Как следствие, основанные на декларативном подходе языки искусственного интеллекта стали значительно усложняться из-за необходимости их наполнения различными "недекларативными" процедурами и функциями. Также в настоящее время наблюдается тенденция использования в качестве основных языков для программирования интеллектуальных систем не специфических языков искусственного интеллекта, а процедурно ориентированных языков. При этом сохраняется разрыв между "декларативной" теорией и "процедурной" практикой.

2. Полноценный логический анализ систем включает в себя не только логический вывод, но и анализ неопределенностей и коллизий, формирование гипотез и абдуктивных заключений. Но если задачи логического вывода хоть и с трудом, но решаются формальными средствами на основе классического подхода, то для решения остальных задач привлекаются в основном неклассические логики. Спрашивается, как можно эти подходы корректно совместить в одной системе?

3. Современные интеллектуальные системы состоят из двух типов разнородных объектов: ***баз данных*** (БД) и ***баз знаний*** (БЗ). Их структуры принципиально различны, так как их построение основано на разных теоретических подходах. Представление и обработка данных (фактов, таблиц, графов, сетей, текстов и т.д.) соответствует алгебраическому подходу и часто используется при анализе данных. Что касается баз знаний, то их основные модели (предикаты, фреймы, семантические сети, правила) строятся на основе декларативного подхода. Это приводит к существенным различиям структур в системах программирования для БД и БЗ и, соответственно, большим затратам времени и средств на разработку методов сопряжения БД и БЗ в одной системе.

Традиционно к основным недостаткам алгебраического подхода в применении к задачам логического анализа относят высокую вычислительную сложность алгоритмов их решения (так называемую проблему "экспоненциальной катастрофы"). Однако полностью решить эту проблему не удается и в рамках теории формальных систем, несмотря на то, что здесь достигнуты значительные результаты в части снижения трудоемкости логических процедур.

В настоящее время попытка изложить методы логического анализа рассуждений на языке, отличающемся от языка ТФС, кажется проблематичной. Возникает вопрос: можно ли вместо символьных конструкций ТФС предложить некую новую универсальную структуру, с помощью которой можно сделать логику более понятной и более практичной?

Оказывается, такая универсальная структура уже давно имеется на вооружении математиков и специалистов по информационным технологиям. Она используется при моделировании и анализе информационных и управляющих систем, она же присутствует в качестве интерпретации во всех структурах математической логики и к тому же позволяет найти более тесную

связь логики высказываний и предикатов с основными структурами, использующимися в современной информатике. Этой универсальной структурой является ***отношение***. Примеры отношений – такие понятия, как "больше", "часть целого", "причина-следствие", "уважает (кто?, кого?)", "дети-родители" и т.д. В математической логике отношения выражаются с помощью предикатов и логических формул.

В современной информатике отношения широко используются. Математические инструменты для отображения отношений известны: это ***теория бинарных отношений*** и ***реляционная алгебра***. Первая применяется для отображения графов, семантических сетей, систем логического анализа на решетках и т.д. Вторая тесным образом связана с системами управления базами данных (СУБД). Однако с помощью бинарных отношений далеко не всегда можно выразить отношения и предикаты с размерностью более двух, а реляционная алгебра не предназначена для решения многих задач логического анализа. Кроме того, в рамках этих теорий для совокупности отношений, не являющихся подмножествами одного декартова произведения, не сохраняется соответствие с алгеброй множеств: операции алгебры множеств для такого случая не определены.

Поэтому возникает необходимость в использовании более общего математического аппарата для теории отношений, расширяющего аналитические средства и области применения такой теории. С этой целью авторами разработана математическая система, названная алгеброй кортежей (АК), которая может служить интерпретацией исчисления предикатов первого порядка. Однако, предполагаемая область применения этой теории не ограничивается только такой возможностью, она позволяет с единых позиций взглянуть на обработку структур знаний, представимых в виде предикатов, бинарных отношений (графы, семантические сети и т.д.), и структур данных, например, реляционных таблиц.

Аналитический аппарат АК основан на свойствах декартовых произведений множеств – эти свойства позволяют компактно представлять отношения, содержащие большое число элементарных кортежей. В рамках АК предложен новый подход к проверке корректности логического вывода и к методам порождения следствий. Исследованиями установлено, что, помимо новых подходов к решению задач логического вывода, с помощью АК

решаются следующие задачи:

1) моделирование и анализ модифицируемых рассуждений (гипотезы, абдукция и т.д.) и рассуждений с неопределенностями;

2) логико-семантический анализ моделируемых систем;

3) вероятностный анализ логических систем;

4) унифицированное представление данных и знаний;

5) при машинной реализации – сокращение трудоемкости алгоритмов решения сложных задач логического анализа за счет специфических свойств АК, а также за счет возможности эффективного распараллеливания алгоритмов.

Об этих и других возможностях АК подробно рассказано в настоящей книге.

Структура монографии. Книга содержит введение, пять глав и заключение. В монографии 235 машинописных страниц текста, 26 таблиц, 19 рисунков, 2 приложения и список литературы на 94 наименования.

Во **Введении** определяется позиция алгебры кортежей в семействе алгебраических систем. С математической точки зрения АК относится к алгебраическим системам, поэтому целесообразно рассмотреть основные понятия и свойства таких систем. Описаны наиболее общие классы алгебраических систем, в том числе, класс булевых алгебр, играющий важную роль в современной математической логике и вычислительной технике. Устанавливается, что АК относится к классу булевых алгебр.

В **первой главе** дается краткое введение в теоретические основы часто используемых в информационных технологиях математических структур, которые лежат в основе АК или имеют тесную смысловую связь с нею. К ним относятся алгебра множеств, алгебра логики, системы логического вывода, многоместные отношения. Данную главу можно рассматривать также как краткое введение в основы дискретной математики в объеме, необходимом для понимания последующих глав.

Во **второй главе** рассматриваются теоретические основы АК. Приводятся определения основных понятий и выводится в виде теорем ряд соотношений, с помощью которых реализуются алгоритмы выполнения основных операций на структурах АК. Минимальное сочетание основных операций АК (операции алгебры множеств и пять простых операций с атрибутами) дает возможность при определенной их комбинации выполнять многие сложные процедуры

(соединение, композиция, транзитивное замыкание, квантификация и т.д.) с многоместными отношениями и предикатами.

В **третьей главе** описываются методы уменьшения трудоемкости вычислительных операций для задач, часто встречающихся в системах логического вывода и в системах искусственного интеллекта. Эти методы основаны на специфических свойствах структур АК, в частности, их матричной (точнее, матрицеподобной) структуре.

В **четвертой главе** представлен основанный на АК принципиально новый подход к решению задач логического анализа систем и рассуждений. Свойства АК позволяют не только решать задачи логического вывода, но и задачи, выходящие за пределы дедукции, в частности задачи порождения и анализа гипотез и абдуктивных заключений, которые входят в состав так называемых модифицируемых рассуждений. Отличительная особенность предлагаемых методов решения подобных задач состоит в том, что они основаны на классических основаниях логики, то есть не используют немонотонные логики, логики умолчаний и т.д., в которых допускаются нарушения некоторых законов булевой алгебры и алгебры множеств.

В **пятой главе** приводятся примеры приложений АК, относящиеся к самым различным задачам и использующие широкий спектр методов: методы погружения структур АК в измеримое пространство, логико-вероятностный анализ, реляционная алгебра, теория бинарных отношений (в том числе, неоднородные семантические сети, формальный анализ понятий) и др.

Учитывая многообразие задач и приложений на основных объектах дискретной математики, в которых главную роль играют отношения, можно предположить, что АК может рассматриваться как математическая основа общей теории отношений.

В **заключении** изложены основные результаты работы и возможные направления дальнейших разработок по исследованной проблеме.

Авторы признательны О.В.Фридман за большую помощь в оформлении книги.

Введение

Если в прошлых веках и в начале XX века алгебра изучала лишь ограниченное число алгебраических структур, то сейчас можно дать очень общее определение алгебры – а именно: ***наука о свойствах множеств, на которых определена та или иная система операций и отношений***. В развитие такого взгляда на алгебру внес большой вклад А.И. Мальцев. В частности, он ввел понятие алгебраической системы. Благодаря работам А.И. Мальцева стало ясно, что алгебра и математическая логика – две тесно связанные между собой дисциплины.

По А.И. Мальцеву ***алгебраическая система*** определяется как объект $\boldsymbol{B} = <A, \Omega_F, \Omega_P>$, состоящий из трех множеств: непустого множества A, называемого носителем (основным множеством), множества $\Omega_F = \{F_1, F_2, \ldots, F_r\}$ функций, и множества $\Omega_P = \{P_1, P_2, \ldots, P_s\}$ предикатов. Функции и предикаты имеют определенную арность (нульместные, одноместные, двухместные и т.д.). В ряде случаев вместо термина "функция" используется термин "операция", а вместо термина "предикат" – термин "отношение". В классической работе А.И. Мальцева [*Мальцев,* 1970] каждому отношению ставится в соответствие некоторый предикат, который определяется следующим образом. Пусть заданы: 1) множество из двух элементов "истина" и "ложь" (сокращенно – множество $\{И, Л\}$), 2) декартово произведение D, образованное из n множеств, 3) некоторое отношение R – подмножество D. Элементами D и R являются кортежи из n элементов. Тогда предикатом, соответствующим отношению R, будет n-арная функция $f(x_1, x_2, \ldots, x_n)$ на множество $\{И, Л\}$, такая, что кортежу $(a_1, a_2, \ldots, a_n)$ из D соответствует константа $И$, если он принадлежит R, и константа $Л$ – в противном случае. Следовательно, между предикатом и отношением устанавливается взаимно-однозначное соответствие. Дополнением R (обозначается как $\overline{R}$) будет множество всех кортежей из D, для которых $f(x_1, x_2, \ldots, x_n) = Л$.

Иногда удобнее использовать понятие многосортной или многоосновной алгебраической системы, полагая существование нескольких основных множеств вместо одного множества A.

Итак, для задания алгебраической системы, в общем случае, необходимо

описать следующие компоненты:

1) ***носитель*** – некоторую совокупность объектов (например, числа, геометрические фигуры, слова, множества и т.д.);

2) совокупность ***отношений*** (например, больше, меньше, равно и т.д.);

3) совокупность ***операций*** (например, сложение, умножение, пересечение и др.);

4) основные соотношения (***законы***), отображающие некоторые свойства операций и отношений (например, закон коммутативности сложения и умножения, транзитивность отношения "больше", законы де Моргана и т.д.).

Алгебраическая система, у которой отсутствуют операции ($\Omega_F = \varnothing$), называется ***моделью*** или ***реляционной системой***. ***Алгеброй*** называется алгебраическая система, у которой $\Omega_P = \varnothing$. Отношения в алгебрах, в отличие от моделей, определяются с помощью операций или представлены как операции. Например, решетка в теоретических работах представлена как алгебра, в которой отношение порядка ($\leq$) задается с помощью операций *inf* (точная нижняя грань) и *sup* (точная верхняя грань): $a \leq b$ тогда и только тогда, когда $inf(a, b) = a$ или $sup(a, b) = b$. Однако в прикладных исследованиях нередко отношения в таких алгебрах являются первичными или используются наряду с операциями. Например, при моделировании решеток используются сети или графы, и результаты операций *inf* и *sup* вычисляются с помощью операций на сетях. Любая алгебра (например, группа, кольцо, поле или булева алгебра) содержит, по крайней мере, одно отношение – равенство как запись результата выполнения операции.

В зависимости от конкретного набора операций и отношений, а также их свойств, алгебраические системы могут быть сгруппированы в определенные классы. Рассмотрим некоторые фундаментальные классы алгебр, моделей, а также тесно связанные с ними понятия, которые потребуются для дальнейшего изложения. Здесь лишь заметим, что между моделями и алгебрами зачастую могут быть установлены взаимно-однозначные соответствия, то есть с помощью внешне различных алгебраических структур часто описываются одни и те же математические идеи.

В качестве примера моделей рассмотрим ***частично упорядоченные множества*** (ч.у.м., посеты, *y*-множества) $<X, \leq>$, которые можно представить как частный случай графов, где для элементов носителя задано ***отношение***

частичного порядка со свойствами рефлексивности, транзитивности и антисимметричности. Примерами ч.у.м. являются числа, упорядоченные по величине; слова с лексикографическим порядком; множества, упорядоченные по включению; целые числа с отношением делимости и т.д. Последний пример считается универсальным, так как интерпретирует все возможные типы и разновидности ч.у.м.

Формально отношение частичного порядка определяется как заданное на множестве X бинарное отношение со следующими свойствами:

1) ***рефлексивности***: $a \le a$;

2) ***транзитивности***: если $a \le b$ и $b \le c$, то $a \le c$;

3) ***антисимметричности***: из $a \le b$ и $b \le a$ следует $a = b$,

где a, b и c – произвольные элементы X.

Любое исходное ч.у.м. можно представить как ориентированный граф, в котором дуга $a \to b$ между парой элементов означает $a \le b$. Однако не любой ориентированный граф является представлением ч.у.м. Чтобы ориентированный граф представлял правильное ч.у.м., необходимо и достаточно, чтобы в нем не было циклов.

Используя понятие частично упорядоченного множества, можно описать еще один фундаментальный класс алгебраических систем – решетки. Для этого введем несколько вспомогательных определений.

Пусть X – ч.у.м. с частичным порядком $\le$. Элемент x называется ***нижней гранью*** для a и b, если $x \le a$ и $x \le b$. Аналогично, y называется ***верхней гранью*** для a и b, если $a \le y$ и $b \le y$. Элемент x называется ***точной нижней гранью*** для a и b, если x – нижняя грань элементов a и b и для любой другой их нижней грани v выполняется $v \le x$. Обозначение: $x = inf(a, b)$. Элемент y называется ***точной верхней гранью*** для a и b, если y – верхняя грань элементов a и b и для любой другой их верхней грани u выполняется $y \le u$. Обозначение: $y = sup(a, b)$.

Решеткой называется ч.у.м. $<X, \le>$, для любых двух элементов a, b которого существует точная нижняя грань $inf(a, b) \in X$ и точная верхняя грань $sup(a, b) \in X$.

Другими словами, если в модель ч.у.м. добавить две двухместные операции *inf* и *sup*, то получим известную алгебраическую систему, которая называется решеткой.

Примечательно, что решетка может быть представлена и как "чистая" алгебра $<B, \cap, \cup>$, если положить, что $a \cap b = inf(a, b)$, $a \cup b = sup(a, b)$, а отношение $\leq$ заменить следующим образом: $a \leq b \Leftrightarrow a \cap b = a$ или $a \cup b = b$ (знак $\Leftrightarrow$ здесь означает "равносильно"). Далее приведем формальное определение решетки через операции $\cap, \cup$ (в некоторых источниках эти операции обозначаются как $\wedge$ и $\vee$ соответственно).

Алгебра $<B, \cap, \cup>$, где B – непустое множество элементов решетки, а $\cap, \cup$ – двухместные операции (***пересечение*** и ***объединение***), называется ***решеткой***, если для всех a, b, c из B выполняются следующие требования:

1) ***замкнутость***: множество B содержит $a \cap b$, $a \cup b$;

2) ***свойства идемпотентности***: $a \cap a = a$, $a \cup a = a$;

3) ***коммутативные законы***: $a \cap b = b \cap a$, $a \cup b = b \cup a$;

4) ***ассоциативные законы***: $a \cap (b \cap c) = (a \cap b) \cap c$,

$$a \cup (b \cup c) = (a \cup b) \cup c;$$

5) ***законы поглощения***: $a \cup (a \cap b) = a$, $a \cap (a \cup b) = a$.

Решетка называется ***дистрибутивной***, если соблюдается хотя бы один из следующих законов:

6) ***законы дистрибутивности***: $a \cap (b \cup c) = (a \cap b) \cup (a \cap c)$,

$$a \cup (b \cap c) = (a \cup c) \cap (a \cup c).$$

Решетка называется ***ограниченной***, если:

7) $\exists 0 \in B$ (***нуль*** или ***нижняя грань решетки***) и $\exists 1 \in B$ (***единица*** или ***верхняя грань решетки***) такие, что для $\forall a \in B$ выполняется $0 \cap a = 0$, $1 \cup a = 1$, $a \cup 0 = a$, $a \cap 1 = a$.

Решетки сами по себе часто встречаются в разных прикладных задачах, но еще важнее то, что понятие решетки непосредственно приводит нас к понятию булевой алгебры, значение которой для основ современной двоичной компьютерной техники трудно переоценить.

Алгебра $<B, \cap, \cup, \bar{}>$, где $\bar{}$ есть одноместная операция (***дополнение***), называется ***булевой алгеброй*** (***булевой решеткой***), если в B, помимо условий 1 – 7, выполняются следующее условие:

8) Для каждого элемента a множество B содержит элемент $\bar{a}$ (***дополнение элемента*** a) такой, что $a \cup \bar{a} = 1$, $a \cap \bar{a} = 0$.

Другими словами, булева алгебра является дистрибутивной ограниченной

решеткой, в которой для каждого элемента существует дополнение.

Приведем некоторые дополнительные важные свойства булевых алгебр:

9) ***свойство совместимости***: $a \cap b = a \Leftrightarrow a \cup b = b$;

10) ***двойственность***, ***законы де Моргана***: $\overline{a \cap b} = \bar{a} \cup \bar{b}$, $\overline{a \cup b} = \bar{a} \cap \bar{b}$;

11) $\bar{\bar{a}} = a$;

12) $\bar{0} = 1$, $\bar{1} = 0$;

В любой решетке можно естественным образом ввести частичный порядок. Положим по определению $a \le b$ равносильно $a \cap b = a$, что ввиду свойства 9 равносильно $a \cup b = b$. Ниже приводятся решеточные свойства объединения и пересечения, а также некоторые другие свойства, использующие отношение ≤:

13) $a \le (a \cup b)$, $b \le (a \cup b)$;

14) $a \le c$ и $b \le c$, то $(a \cup b) \le c$;

15) $a \cap b \le a$, $(a \cap b) \le b$;

16) $c \le a$ и $c \le b$, то $c \le (a \cap b)$;

17) $0 \le a$, $a \le 1$;

18) ***закон контрапозиции***: $a \le b \Leftrightarrow \bar{b} \le \bar{a}$.

Большой список литературы по различным вариантам аксиоматических определений булевой алгебры приведен в [*Сикорский,* 1969].

К булевым алгебрам относятся:

1) алгебра множеств (алгебра Кантора, алгебра классов, алгебра подмножеств, алгебра частей множества) $<P(M), \cap, \cup, \bar{}>$, причем M – верхняя грань, $\varnothing$ – нижняя грань, $\subseteq$ – естественный частичный порядок, $P(M)$ – булеан (множество всех подмножеств) множества M;

2) алгебра Буля (алгебра логики, алгебра высказываний, алгебра B_2) $<\{0,1\}, \wedge, \vee, \neg>$, причем 1 – верхняя грань, 0 – нижняя грань.

Своеобразным эталоном в классе булевых алгебр является алгебра множеств, о чем свидетельствует следующая теорема.

Теорема (Стоуна). Всякая конечная булева алгебра B изоморфна алгебре множеств. [*Столл,* 1968].

С помощью булевых алгебр можно интерпретировать многие известные логические исчисления (системы). Так, например, алгебра множеств была

создана для обоснования силлогистики Аристотеля, а алгебра логики служит интерпретацией исчисления высказываний.

Как уже упоминалось, в настоящее время не существует полноценного аналога алгебры множеств для случая, когда в качестве множеств рассматриваются отношения, несмотря на то, что очевидна необходимость разработки общей теории многоместных отношений, обеспечивающей поддержку операций алгебры множеств над отношениями с различными схемами.

В качестве такого математического аппарата в настоящей работе предлагается ***алгебра кортежей***, которая развивает существующую теорию отношений и относится к классу булевых алгебр. Отношение, выраженное в виде предиката – необходимая составная часть практически всех алгебраических систем. Как будет показано далее, при определенных условиях сама по себе произвольная совокупность отношений есть алгебраическая система, в которой справедливы все законы и соотношения алгебры множеств.

Глава 1. Основные математические структуры

… если между людьми возникают разногласия,
достаточно было бы только сказать
"Вычислим!", чтобы … стало ясно, кто прав.
Г.В. Лейбниц

Приводится обзор средств логического анализа, некоторые из них исторически сыграли важную роль в формировании этого направления, другие применяются в современных программных системах в таких областях, как искусственный интеллект и системный анализ. Подобные средства можно разделить на две категории: 1) системы на основе формального подхода; 2) алгебраические системы. В теории формальных систем, помимо классических логик (силлогистика, логика высказываний, логика предикатов и т.д.), активно развиваются неклассические логики (логика умолчаний, немонотонная логика и т.п.). Неклассические логики часто используются при моделировании и анализе модифицируемых рассуждений (рассуждений с гипотезами и абдуктивными заключениями), однако авторы придерживаются мнения, что на компьютере такого рода задачи целесообразно решать алгебраически с применением операций и законов алгебры множеств. Последующие главы посвящены подробному изложению этого подхода к логическому анализу.

В качестве алгебраических систем, обеспечивающих решение широкого круга задач логического анализа, рассматриваются алгебра множеств и алгебра логики (алгебра высказываний), которые относятся к классу булевых алгебр [*Скорняков*, 1982] (см. Введение), а также теория отношений.

1.1. Алгебра множеств

В настоящее время алгебра множеств редко используется как инструмент логического анализа. Во многом это обусловлено подрывом доверия к правомерности "множественного" подхода после того, как на рубеже XIX и XX были сформулированы парадоксы теории множеств. Обнаружение таких парадоксов породило у многих логиков и математиков уверенность в том, что "наивное" понятие множества противоречиво в принципе. Здесь мы будем придерживаться точки зрения, согласно которой парадоксы, связанные с

множествами, обусловлены не самим предметным подходом, а тем, что они формулируются с не заметными для многих нарушениями логики. При детальном рассмотрении оказывается, что многие (а, возможно, и все) парадоксы – это тонко замаскированная игра словами.

Основными понятиями алгебры множеств считаются ***множество*** и ***элемент***. Соотношение между ними называется ***отношением принадлежности*** и обозначается знаком "$\in$". Запись $b \in A$ переводится с символического языка как "b есть элемент множества A" или "элемент b принадлежит множеству A". Если известны все элементы множества (например, a, b и c), то общепринята такая запись множества:

$A = \{a, b, c\}$.

В этом случае элементы множества принято заключать в фигурные скобки. Множества можно задать двумя способами: с помощью ***формулировки характерных признаков*** (например, множество K содержит неотрицательные четные числа, не превышающие числа 8) или ***перечислением элементов*** (например, $K = \{0, 2, 4, 6, 8\}$).

В современной математике нет четкого определения отношения принадлежности. В алгебре множеств этой неопределенности можно избежать, если считать, что оно связывает два разных типа объектов ("элемент"$\in$"множество"), но ни в коем случае не должно быть связи типа "множество"$\in$"множество". Заметим, что именно эта нечеткость в определении отношения принадлежности приводит к парадоксам теории множеств.

Между множествами устанавливается другое, на первый взгляд, похожее, но в то же время принципиально отличающееся отношение – ***включение***, структурные свойства которого в современной математике определены достаточно четко и однозначно. Рассмотрим его более подробно. Допускаются два разных варианта этого отношения:

"$\subset$" – строго включено;

"$\subseteq$" – включено или равно.

Запись $A \subset B$ означает, что множество A включено в множество B, т.е**. *все элементы множества A являются одновременно элементами множества B***, причем невозможно равенство этих множеств. Запись $A \subseteq B$ означает, что множество A включено во множество B, но они могут быть равными.

Изображение отношения включения ($A \subset B$) с помощью диаграмм Эйлера показано на рис. 1.1.

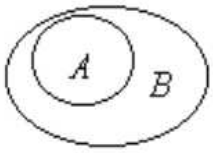

Рис. 1.1. Отношение строгого включения

Если множества заданы с помощью перечисления элементов, то отношение включения (или невключения) одного множества в другое можно легко установить, если сравнить элементы этих множеств. Например, если заданы три множества $P = \{a, b, c, d, e\}$; $Q = \{b, d, a\}$; $R = \{a, c, f\}$, то, сравнивая элементы этих множеств, можно утверждать, что $Q \subset P$, но отношение $R \subset P$ для данных множеств неверно, так как элемент f из множества R не принадлежит множеству P. В случаях, когда число элементов слишком велико или бесконечно, отношение включения можно иногда установить, используя свойства этих множеств. Например, если даны два бесконечных множества положительных целых чисел N_2 (числа, кратные 2) и N_6 (числа, кратные 6), то нетрудно доказать, не прибегая к поэлементной проверке, что $N_6 \subset N_2$.

Рассмотрим еще одно отношение между множествами – отношение ***равенства***. Множества равны, если у них одни и те же элементы. Для доказательства равенства двух множеств, особенно в случаях, когда у них большое или бесконечное число элементов, используется следующее утверждение.

Множества A и B ***равны***, если справедливо как $A \subseteq B$, так и $B \subseteq A$.

Заметим, что в теории множеств это утверждение – доказанная теорема. Если множества связаны отношениями $A \subset B$ или $A \subseteq B$, то A называют ***подмножеством*** B. Среди всех возможных подмножеств произвольного множества A обязательно содержится также и само множество A. Другими словами, для любого множества A всегда справедливо $A \subseteq A$.

Еще один интересный и полезный частный случай множества – ***пустое множество***, не содержащее никаких элементов (обозначается "$\varnothing$"). Его математический смысл раскрывается в следующем предложении, которое можно отнести к аксиомам алгебры множеств: ***Пустое множество включено в любое множество***.

В математических рассуждениях, когда надо доказать, что при заданных условиях множество X не существует (или существует), сводят доказательство существования к доказательству отношения $X = \varnothing$ (или $X \neq \varnothing$). Часто такой метод позволяет намного упростить доказательство.

Если множество задано перечислением элементов, то можно записать ***совокупность всех подмножеств*** этого множества. Например, для множества $A = \{a, b, c\}$ такая совокупность состоит из восьми подмножеств:

$\varnothing$, $\{a\}$, $\{b\}$, $\{c\}$, $\{a, b\}$, $\{a, c\}$, $\{b, c\}$, $\{a, b, c\}$.

Доказано простое соотношение, позволяющее узнать общее количество всех возможных подмножеств множества, содержащего ровно N элементов. Оказывается, что для любого N это количество равно 2^N. Например, для нашего множества $A = \{a, b, c\}$ число всех возможных подмножеств равно 2^3.

Во многих рассуждениях используется некоторый набор множеств. Такой набор называется в алгебре множеств системой множеств. В систему множеств при этом, помимо пустого множества, включается и ***универсум***, т.е. множество, для которого все множества системы множеств являются подмножествами.

Система множеств есть некоторая совокупность подмножеств заданного множества, принятого за универсум. Например, для множеств планет, комет, звезд и т.д. в качестве универсума можно принять множество астрономических объектов. Для универсума нет общепринятых обозначений. Далее мы будем обозначать его символом ***U***.

Перейдем к операциям. Начнем с операции дополнения, которая может быть определена только тогда, когда для системы множеств задан универсум.

Определение 1.1. ***Дополнением*** множества A называется множество $\overline{A}$, содержащее все элементы универсума, которые не являются элементами множества A.

В логике дополнению множества соответствует связка "не". Например "не красный" – любой возможный цвет, кроме красного. Обычно дополнение множества обозначается с помощью черты над символьным обозначением этого множества. Например, $\overline{R_{ik}}$ обозначает дополнение множества R_{ik}.

Пример 1.1. Пусть $\boldsymbol{U} = \{a, b, c, d\}$ и $P = \{a, c\}$. Тогда $\overline{P} = \{b, d\}$.

Определение 1.2. ***Пересечение*** множеств A и B – это множество C, которое содержит все элементы, принадлежащие одновременно как A, так и B.

Операция пересечения множеств обозначается символом "$\cap$". Символически определение 1.2 можно записать как формулу

$C = A \cap B$.

Например, пересечение множества всех студентов данного вуза и множества всех участников КВН – это множество всех студентов данного вуза, участвующих в КВН. Другой пример: пересечение множества всех чисел, делящихся на 2, и множества всех чисел, делящихся на 3, есть множество всех чисел, делящихся на 6.

В логике операции пересечения соответствует логическая связка "И" (обозначается как $\wedge$ или &). Если речь идет об объектах со свойствами P или Q, то логическая формула $P \wedge Q$ означает, что речь идет только об объектах, которым присущи оба этих свойства.

Пример 1.2. Пусть $A = \{a, b, c, d\}$ и $P = \{a, c, f\}$. Тогда $A \cap P = \{a, c\}$.

Возможна ситуация, когда два множества не содержат общих элементов. Тогда пересечение этих множеств – пустое множество. Например, для $Q = \{a, c,\}$ и $R = \{b, d\}$ $Q \cap R = \varnothing$.

Определение 1.3. ***Объединением*** множеств A и B называется множество C, все элементы которого принадлежат хотя бы одному из этих множеств.

Операция объединения множеств обозначается символом "$\cup$". Символически определение 1.3 можно записать как формулу $C = A \cup B$.

В логике операции объединения соответствует логическая связка "ИЛИ" (обозначается "$\vee$"). Если речь идет об объектах со свойствами P или Q, то логическая формула $P \vee Q$ означает, что речь идет только об объектах, которым присуще хотя бы одно из этих свойств. Причем допускается, что объекты, обладающие обоими свойствами, также относятся к этому классу объектов.

Пример 1.3. Пусть $A = \{a, b, c, d\}$ и $P = \{a, c, f\}$. Тогда $A \cup P = \{a, b, c, d, f\}$.

Операции дополнения, пересечения и объединения являются ***основными*** операциями алгебры множеств. Ниже приведены операции, которые называются ***производными операциями***. Это означает, что их можно выразить с помощью основных операций.

Определение 1.4. ***Разность*** множеств A и B есть множество $C = A \setminus B$, которое содержит только элементы множества A, не принадлежащие

одновременно множеству B.

Пример 1.4. Пусть $A = \{a, b, c, d\}$ и $B = \{a, c, f\}$. Тогда $A \setminus B = \{b, d\}$.

Для разности множеств справедливо следующее соотношение:

$A \setminus B = A \cap \overline{B}$.

Если в примере 1.4 задать универсум, например, $\boldsymbol{U}=\{a, b, c, d, e, f\}$, то нетрудно убедиться в справедливости этого равенства:

$\overline{B} = \{b, d, e\}$; тогда $A \setminus B = A \cap \overline{B} = \{b, d\}$.

В то же время операцию дополнения можно выразить с помощью операции разности: $\overline{A} = \boldsymbol{U} \setminus A$. На рис. 1.2 соответствующие операции над множествами изображены с помощью диаграмм Эйлера. Серым цветом показаны результаты операций.

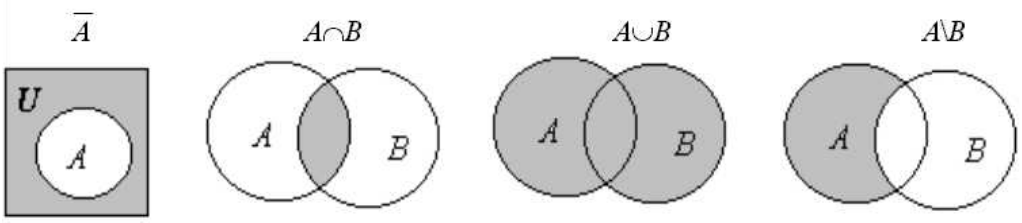

Рис. 1.2. Операции алгебры множеств

Здесь хотелось бы обратить внимание на одно важное обстоятельство. Для множеств A и B, у которых нет общих элементов, справедливы следующие соотношения:

$A \cap B = \varnothing$; $A \subseteq \overline{B}$; $B \subseteq \overline{A}$.

Их можно наглядно отобразить с помощью такой диаграммы (рис. 1.3).

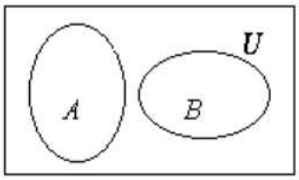

Рис. 1.3. Множества, не имеющие общих элементов

Законы алгебры множеств есть, по сути, теоремы, которые выводятся из основных определений и аксиом. Часто приводятся 26 или 28 законов алгебры множеств. Ниже мы приведем без доказательства лишь некоторые из них, необходимые для ясного понимания дальнейшего. Пусть A, B, C – некоторые произвольные множества в универсуме $\boldsymbol{U}$. Тогда законами алгебры множеств являются следующие соотношения между ними.

1. $\overline{\overline{A}} = A$.

Пример 1.5. Пусть $\boldsymbol{U} = \{a, b, c, d\}$ и $P = \{a, c\}$. Тогда $\overline{P} = \{b, d\}$ и $\overline{\overline{P}} = \{a, c\} = P$.

В алгебре множеств это соотношение (двойное дополнение) носит название ***закон инволюции***. В логике этот закон известен под названием ***закон отрицания отрицания*** (или закон двойного отрицания): не (не-A) – то же самое, что и A.

2. $A \cap \overline{A} = \varnothing$ (множество и его дополнение не имеют общих элементов).

В логике этому закону соответствует ***закон непротиворечия*** (утверждение и его полное отрицание логически несовместимы).

3. $A \cup \overline{A} = \boldsymbol{U}$.

В логике этому закону соответствует ***закон исключенного третьего*** (совмещение любого утверждения и его полного отрицания не допускает присутствия какого-либо третьего промежуточного варианта).

Следующие соотношения характеризуют более подробно свойства пустого множества и универсума:

4. $\overline{\varnothing} = \boldsymbol{U}$; 5. $\overline{\boldsymbol{U}} = \varnothing$

6. $A \cap \varnothing = \varnothing$; 7. $A \cup \varnothing = A$;

8. $A \cap \boldsymbol{U} = A$; 9. $A \cup \boldsymbol{U} = \boldsymbol{U}$.

Еще одна группа законов алгебры множеств связывает друг с другом отношения включения и равенства:

10. Если $A \subseteq B$, то справедливы следующие соотношения:

10a. $A \cap B = A$; 10b. $A \cup B = B$;

10c. $\overline{A} \cup B = \boldsymbol{U}$; 10d. $A \cap \overline{B} = \varnothing$.

Соотношение 10d можно выразить также с помощью операции разности множеств:

10e. Из $A \subseteq B$ следует $A \setminus B = \varnothing$.

Следующие законы в логике и алгебре множеств называются ***законами де Моргана***:

11a. $\overline{A \cap B} = \overline{A} \cup \overline{B}$; 11b. $\overline{A \cup B} = \overline{A} \cap \overline{B}$.

Законы дистрибутивности декларируют правила раскрытия скобок в некоторых выражениях алгебры множеств:

12. $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$;

13. $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$.

И, наконец, приведем два закона, которые определяют основные свойства отношения включения.

14a. Если $A \subseteq B$ и $B \subseteq C$, то $A \subseteq C$ (***закон транзитивности включения***).

14b. Если $A \subseteq B$, то справедливо также и $\overline{B} \subseteq \overline{A}$ (***закон контрапозиции***).

В математической литературе приводятся разные способы обоснования этих законов. Многие из них весьма сложны для понимания. Здесь мы рассмотрим относительно простой способ, который называется ***комбинаторным***.

Пусть необходимо вывести некоторые законы для двух множеств X и Y. Рассмотрим диаграмму Эйлера (рис. 1.4), на которой изображены эти множества и объемлющий их универсум $\boldsymbol{U}$.

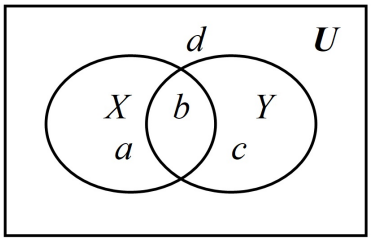

Рис. 1.4. Пересекающиеся множества

Выделим на диаграмме участки (в данном случае множества) a, b, c и d, которые не имеют внутренних границ, т.е. выполним ***разбиение*** нашего универсума на непересекающиеся друг с другом множества. Такое разбиение позволяет представить имена этих множеств как элементы, из которых состоят универсум $\boldsymbol{U}$ и множества X и Y. Тогда для них справедливы соотношения:

$\boldsymbol{U} = \{a, b, c, d\}$; $X = \{a, b\}$; $Y = \{b, c\}$.

Примем полученные соотношения в качестве исходных данных и докажем для этого общего представления данной системы один из законов де Моргана $\overline{X \cap Y} = \overline{X} \cup \overline{Y}$. Тогда получим:

1) $X \cap Y = \{b\}$;

2) $\overline{X \cap Y} = \{a, c, d\}$;

3) $\overline{X} = \{c, d\}$;

4) $\overline{Y} = \{a, d\}$;

5) $\overline{X} \cup \overline{Y} = \{a, c, d\}$;

6) сравнивая 2 и 5, заключаем, что $\overline{X \cap Y} = \overline{X} \cup \overline{Y}$, что и требовалось доказать.

Для более полного доказательства комбинаторным способом необходимо учесть все ситуации, когда некоторые из множеств a, b, c и d – пустые. В нашей модели это означает отсутствие соответствующих элементов в универсуме. Например, если множество b пусто, то исходные данные для множества имен будут такими:

$\boldsymbol{U} = \{a, c, d\}$; $X = \{a\}$; $Y = \{c\}$.

Докажем для этого случая справедливость закона де Моргана.

1) $X \cap Y = \varnothing$;

2) $\overline{X \cap Y} = \{a, c, d\}$;

3) $\overline{X} = \{c, d\}$;

4) $\overline{Y} = \{a, d\}$;

5) $\overline{X} \cup \overline{Y} = \{a, c, d\}$;

6) сравнивая 2 и 5, заключаем, что $\overline{X \cap Y} = \overline{X} \cup \overline{Y}$, что и требовалось доказать.

Закон транзитивности (14a) интуитивно понятен. Рассмотрим обоснование закона контрапозиции (14b). Схема на рис. 1.4 не подходит, так как между множествами X и Y нет соотношения включения. Однако, если убрать из универсума элемент a, найдем то, что нужно: $X \subseteq Y$. Данный результат можно выразить в виде следующих равенств: $\boldsymbol{U} = \{b, c, d\}$; $X = \{b\}$; $Y = \{b, c\}$.

Приступим к доказательству. Для этого вычислим

1) $\overline{X} = \{c, d\}$;

2) $\overline{Y} = \{d\}$;

3) сравнивая $\overline{X}$ и $\overline{Y}$, получим $\overline{Y} \subseteq \overline{X}$, что и требовалось доказать.

Рассмотрим несколько важных для анализа рассуждений законов, которые в совокупности не встречаются в литературе по алгебре множеств. Объединим их общим названием ***законы сохранения***. Эти законы можно интерпретировать как обобщение в алгебре множеств свойства монотонности.

Понятие ***монотонности*** в настоящее время связывают с системами логического вывода. В классическом варианте математической логики

монотонность является неотъемлемым свойством логического вывода. Это свойство может не соблюдаться в некоторых вариантах неклассической логики (например, в логике умолчаний и в немонотонной логике [*Тейз,* 1990]). Свойство монотонности заключается в следующем: если из некоторого множества посылок Г выводится предложение *B*, то при добавлении во множество Г любой посылки (например, *A*) предложение *B* также должно выводиться.

В алгебре множеств этому свойству логического вывода соответствует следующий закон:

Если $G \subseteq B$, то $(G \cap A) \subseteq B$, где *A* – любое множество.

Доказательство этого закона таково. Из определения пересечения следует, что $(G \cap A) \subseteq G$. Тогда по закону транзитивности из $(G \cap A) \subseteq G$ и $G \subseteq B$ следует $(G \cap A) \subseteq B$, что и требовалось доказать.

Рассмотрим законы сохранения.

1) ***Закон сохранения равенства***: если $X = Y$, то для любого множества *Z* справедливы следующие равенства:

$X \cap Z = Y \cap Z$ и $X \cup Z = Y \cup Z$.

2) ***Законы сохранения для нестрогого включения***: если $X \subseteq Y$, то для любого множества *Z* справедливы следующие соотношения:

(i) $X \cap Z \subseteq Y \cap Z$;

(ii) $X \cup Z \subseteq Y \cup Z$.

Первый закон сохранения кажется очевидным (возможно, поэтому его и не рассматривают в литературе по алгебре множеств и булевой алгебре). Докажем закон сохранения нестрогого включения.

Для доказательства используем следующий закон алгебры множеств: $A \subseteq B$ равносильно $A \cap \overline{B} = \varnothing$. Начнем с соотношения (i). Докажем, что $(X \cap Z) \setminus (Y \cap Z) = \varnothing$.

$$(X \cap Z) \setminus (Y \cap Z) = (X \cap Z) \cap \overline{Y \cap Z} = X \cap Z \cap (\overline{Y} \cup \overline{Z}) = (X \cap Z \cap \overline{Y}) \cup (Y \cap Z \cap \overline{Z}).$$

Множество $X \cap Z \cap \overline{Y}$ равно $\varnothing$, так как по условию $X \subseteq Y$, в силу чего $X \cap \overline{Y} = \varnothing$, а множество $X \cap Z \cap \overline{Z}$ равно $\varnothing$, так как $Z \cap \overline{Z} = \varnothing$. Следовательно, $(X \cap Z) \setminus (Y \cap Z) = \varnothing$ и соответственно $X \cap Z \subseteq Y \cap Z$.

Теперь перейдем к доказательству соотношения (ii), что равносильно доказательству $(X \cup Z) \setminus (Y \cup Z) = \varnothing$.

$(X \cup Z) \setminus (Y \cup Z) = (X \cup Z) \cap \overline{Y \cup Z} = (X \cup Z) \cap \overline{Y} \cap \overline{Z} = (X \cap \overline{Y} \cap \overline{Z}) \cup (Z \cap \overline{Y} \cap \overline{Z})$.

Нетрудно убедиться, что оба "слагаемых" полученного объединения тождественно равны пустому множеству, из чего следует $X \cup Z \subseteq Y \cup Z$.

Интересно отметить, что для строгого включения законы сохранения не выполняются. Имеются исключения, которые можно выявить с помощью теорем 1.1 и 1.2.

Теорема 1.1. Если $X \subset Y$, то для произвольного множества Z строгое включение $(X \cap Z) \subset (Y \cap Z)$ выполняется, если и только если $(Y \setminus X) \cap Z \neq \varnothing$.

Доказательство. Строгое включение $(X \cap Z) \subset (Y \cap Z)$ выполняется, если и только если $(Y \cap Z) \setminus (X \cap Z) \neq \varnothing$. Преобразуем разность $(Y \cap Z) \setminus (X \cap Z)$:

$(Y \cap Z) \setminus (X \cap Z) = (Y \cap Z) \cap \overline{X \cap Z} = Y \cap Z \cap (\overline{X} \cup \overline{Z}) = (Y \cap Z \cap \overline{X}) \cup (Y \cap Z \cap \overline{Z})$.

Вторая компонента объединения множеств $(Y \cap Z \cap \overline{Z})$, безусловно, равна $\varnothing$. Что касается первой компоненты $(Y \cap Z \cap \overline{X})$, то после преобразования она оказывается равной условию теоремы. *Конец доказательства.*

Теорема 1.2. Если $X \subset Y$, то для произвольного множества Z строгое включение $(X \cup Z) \subset (Y \cup Z)$ выполняется, если и только если $(Y \setminus X) \cap \overline{Z} \neq \varnothing$.

Доказательство. Строгое включение $(X \cup Z) \subset (Y \cup Z)$ выполняется, если и только если $(Y \cup Z) \setminus (X \cup Z) \neq \varnothing$. Преобразуем разность $(Y \cup Z) \setminus (X \cup Z)$:

$(Y \cup Z) \setminus (X \cup Z) = (Y \cup Z) \cap \overline{X \cup Z} = (Y \cup Z) \cap (\overline{X} \cap \overline{Z}) = (Y \cap \overline{X} \cap \overline{Z}) \cup (Z \cap \overline{X} \cap \overline{Z})$.

Вторая компонента объединения множеств $(Z \cap \overline{X} \cap \overline{Z})$, безусловно, равна $\varnothing$. Что касается первой компоненты $(Y \cap \overline{X} \cap \overline{Z})$, то после преобразования она оказывается равной условию теоремы. *Конец доказательства.*

Нетрудно убедиться, что все законы алгебры множеств соответствуют приведенным ранее свойствам булевой алгебры и аксиомам (A_1) – (A_5). Более краткая система аксиом алгебры множеств приводится в известной книге Р. Куранта и Г. Роббинса [*Курант*, 1947]. В этой аксиоматической системе для обозначения операции дополнения используется апостроф ('), который размещается после соответствующего символа или выражения. Исходных аксиом всего три:

1) $A \cup B = B \cup A$;

2) $(A \cup B) \cup C = A \cup (B \cup C)$;

3) $(A' \cup B')' \cup (A' \cup B)' = A$.

Тогда операция $\cap$ и соотношение $A \subseteq B$ определяются так:

$A \cap B$ обозначает множество $(A' \cup B')'$;

$A \subseteq B$ соответствует равенству $A \cup B = B$.

Указанных соотношений оказывается достаточно, чтобы вывести все остальные законы алгебры множеств.

Аналогичная математическая система, но уже для аксиоматического построения булевой алгебры, приведена в [*Мендельсон*, 1984]. В этой аксиоматической системе также вначале вводятся только две операции (пересечение и дополнение) и три аксиомы:

(i) $X \cap Y = Y \cap X$;

(ii) $X \cap (Y \cap Z) = (X \cap Y) \cap Z$;

(iii) $X \cap Y' = Z \cap Z'$ тогда и только тогда, когда $X \cap Y = X$.

Но этих аксиом явно недостаточно для отображения всех свойств булевой алгебры, тем более что в них используются только две операции из трех. Поэтому дополнительно вводятся следующие обозначения:

Выражение $(X' \cap Y')'$ обозначается как $X \cup Y$ – нетрудно видеть в этом "обозначении", с помощью которого вводится третья операция, закон де Моргана.

Выражение $Z \cap Z'$ для любого Z обозначается как 0, а выражение $Z \cup Z'$ для любого Z – как 1. В этом "обозначении" неявно вводятся законы непротиворечия и исключенного третьего, и заодно – значения истинности.

Запись $X \cap Y = X$ предлагается обозначать как $X \leq Y$. Здесь неявно вводится отношение нестрогого включения, которое в литературе по булевой алгебре часто обозначается символом "≤".

В заключение отметим, что, несмотря на общие черты, алгебра множеств имеет и некоторые особенности, отличающие ее от других булевых алгебр и обусловливающие спектр ее приложений, а именно:

1) В алгебре множеств имеется отношение принадлежности, которое отсутствует практически во всех остальных булевых алгебрах. Отношение принадлежности неявно присутствует лишь в исчислении предикатов в качестве переменных и констант.

2) В алгебре множеств определено отношение строгого включения, не являющееся отношением частичного порядка.

1.2. Алгебра логики

Со времен Аристотеля формальная логика существовала только в вербальной форме. Дж. Буль описал формальную логику в виде алгебры, то есть в виде системы обозначений и правил, применимых к всевозможным объектам, от множеств (см. предыдущий подраздел) до предложений. Пользуясь этой системой, он смог закодировать высказывания (утверждения, истинность или ложность которых требовалось доказать) и манипулировать ими подобно тому, как в математике манипулируют числами. В настоящем разделе рассматривается алгебра логики, которая изучает операции над высказываниями, и показывается взаимосвязь между ее законами и законами алгебры множеств.

Пусть задана система, содержащая некоторые объекты, которые мы назовем ***атомами***, причем каждый атом может иметь только одно из двух значений, допустим, 0 и 1. Из атомов можно составлять выражения, используя скобки и операции. К основным операциям относятся:

$\neg$ – ***отрицание***;

$\wedge$ – ***конъюнкция***, в логике интерпретируется как "и";

$\vee$ – ***дизъюнкция***, в логике интерпретируется как "или";

$\supset$ – ***импликация***, в логике интерпретируется как "влечет" или "если …, то".

Из операций и атомов можно составлять выражения. Правила составления правильных выражений следующие:

(i) любой атом является правильным выражением;

(ii) если A и B – правильные выражения, то $(\neg A)$, $(A \wedge B)$, $(A \vee B)$, $(A \supset B)$ – также правильные выражения;

(iii) другие сочетания атомов, операций и скобок правильными выражениями не являются.

Таким образом, в силу определения правильны следующие выражения: A, B, $(\neg B)$, $(\neg(A \wedge B))$, $(\neg((A \wedge B) \vee (A \supset (\neg B))))$ и т.д. Иногда некоторые скобки можно опускать, если при этом однозначно восстанавливается последовательность операций. Например, последнее правильное выражение можно записать как $\neg((A \wedge B) \vee (A \supset \neg B))$.

Каждое правильное выражение также может принимать значения из множества $\{0, 1\}$. Тогда значения выражений определяются значениями

входящих в них атомов и последовательностью операций. Для приведенных выше операций эти значения определяются с помощью следующих таблиц:

Таблица 1.1

A	$\neg A$
0	1
1	0

Таблица 1.2

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

Таблица 1.3

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

Таблица 1.4

A	B	$A \supset B$
0	0	1
0	1	1
1	0	0
1	1	1

Определение логических связок "и", "или", "если ..., то" с помощью этих таблиц устраняет некоторые неоднозначности, которые встречаются при использовании связок в естественных рассуждениях. Кроме того, такие определения позволяют установить тесную связь логических операций (или связок) с операциями алгебры множеств. Пусть заданы элемент t и некоторые множества X и Y, и атомы A и B обозначают соответственно высказывания $t \in X$ и $t \in Y$. Тогда нетрудно обосновать, что табл. 1.2 для конъюнкции соответствует ситуации, когда $t \in (X \cap Y)$, а табл. 1.3 для дизъюнкции – ситуации, когда $t \in (X \cup Y)$. Табл. 1.4 в алгебре множеств соответствует выражение $t \in (\overline{X} \cup Y)$.

Что касается таблицы для отрицания, то ее можно интерпретировать как соответствие дополнению множеств, но при некоторых допущениях. Дело в том, что отрицанием высказывания $t \in X$ традиционно считается высказывание $t \notin X$. Но это высказывание не во всех случаях эквивалентно высказыванию $t \in \overline{X}$, поскольку для $t \notin X$ не исключен случай, когда элемент t не принадлежит даже универсуму рассматриваемой системы множеств. Таким образом, при ложном высказывании $t \in X$ возможны два случая:

1) высказывание $t \in \overline{X}$ истинно,

2) высказывание $t \in \overline{X}$ ложно – это означает, что элемент t не является элементом принятого для данной системы универсума.

Чтобы сохранить однозначность отрицания, необходимо рассматривать только такие системы, в которых второй вариант отрицания невозможен, т.е.

справедливо $t \notin X = t \in \overline{X}$ или, равносильно, t в любом случае принадлежит универсуму.

При переходе к формальным логическим системам атомы интерпретируются как высказывания, правильные выражения называются "правильно построенными формулами" или ***ппф***, а вместо множества $\{0, 1\}$ возможных значений атомов и формул используется соответственно множество значений {ложно, истинно}. Установлено, что при такой замене, а также при выборе соответствующих аксиом получается исчисление высказываний. Заметим, что в формальной логике термин "операция" употреблять не принято, вместо него употребляется термин "логическая связка".

На основе табличного определения операций можно также с помощью таблиц определять значения формул. В качестве примера рассмотрим, как "вычисляются" значения формулы $\neg((A \vee B) \wedge (A \supset \neg B))$. Для этого так же, как и в приведенных выше таблицах, необходимо перечислить все возможные сочетания значений атомов, входящих в формулу, и затем последовательно выписывать, соблюдая последовательность операций, значения входящих в исходное выражение подформул (табл. 1.5). Такие таблицы называются истинностными таблицами.

Таблица 1.5

A	B	$A \vee B$	$\neg B$	$A \supset \neg B$	$(A \vee B) \wedge (A \supset \neg B)$	$\neg((A \vee B) \wedge (A \supset \neg B))$
0	0	0	1	1	0	1
0	1	1	0	1	1	0
1	0	1	1	1	1	0
1	1	1	0	0	0	1

Сравнивая значения разных формул, можно установить отношение эквивалентности между ними. В качестве примера сравним табличным способом две формулы $A \supset B$ и $\neg B \supset \neg A$, равенство которых соответствует закону контрапозиции в логике и в алгебре множеств (табл. 1.6):

Таблица 1.6

A	B	$A \supset B$	$\neg B$	$\neg A$	$\neg B \supset \neg A$
0	0	1	1	1	1
0	1	1	0	1	1
1	0	0	1	0	0
1	1	1	0	0	1

Нетрудно убедиться, что 3-я и 6-я колонки табл. 1.6, соответствующие сравниваемым формулам, совпадают, что свидетельствует о равенстве формул.

Табличным способом, в частности, можно доказать, что импликацию можно выразить через дизъюнкцию и отрицание:

$$A \supset B = \neg A \vee B. \qquad (1.1)$$

В алгебре логики, помимо описанных выше четырех основных операций ($\neg, \vee, \wedge, \supset$), определено еще 12 различных логических связок. Эти дополнительные операции можно задать с помощью таблиц, аналогичных тем, которые были приведены выше (таблицы 1.1 – 1.4). Для того, чтобы выразить все функции алгебры логики, достаточно использовать лишь некоторые из них. Все остальные функции можно получить с помощью соответствующих формул (например, импликацию можно выразить с помощью отрицания и дизъюнкции, используя формулу $X \supset Y = \neg X \vee Y$). Множество логических функций, достаточное для выражения всех остальных функций алгебры логики, называется ***функционально полным базисом***. Например, таким базисом является множество логических функций $\{\neg, \vee, \wedge\}$. Алгебра логики, заданная в этом базисе, обозначается обычно $B_2 = <\{0,1\}, \wedge, \vee, \neg >$. Как правило, для описания законов алгебры логики выбирается именно этот базис, причем сами законы можно разделить на две группы: 1) основные равносильности, соответствующие законам булевой алгебры; 2) равносильности, выражающие дополнительные логические операции через базисные.

Для иллюстрации законов второй группы приведем еще две дополнительные часто используемые операции. Это разделительное "ИЛИ" или ***строгая дизъюнкция*** (обозначается символом "$\oplus$"), и ***эквивалентность*** (обозначается символами "$\equiv$" или "$\sim$"). Их табличные определения даны в

таблицах 1.7 и 1.8.

Таблица 1.7

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Таблица 1.8

A	B	$A \sim B$
0	0	1
0	1	0
1	0	0
1	1	1

Тогда, если возникает необходимость выразить дополнительные операции $\oplus$ и $\sim$, используя только основные, то применяются следующие формулы:

$$A \oplus B = (A \vee B) \wedge (\neg A \vee \neg B) = (A \wedge \neg B) \vee (\neg A \wedge B) \quad (1.2)$$

$$A \sim B = (A \supset B) \wedge (B \supset A) = (A \wedge B) \vee (\neg A \wedge \neg B) \quad (1.3)$$

Проверить правильность этих преобразований можно табличным способом.

Возможно построить и другие функционально полные базисы. Так, установлено, что в базисе $\{\neg, \vee, \wedge\}$ одна из операций – "лишняя", поскольку к функционально полным относятся базисы $\{\neg, \vee\}$ и $\{\neg, \wedge\}$. Для обоснования этого достаточно использовать один из законов де Моргана. Например, если выбрать в качестве базиса множество $\{\neg, \vee\}$ логических функций, то конъюнкция двух произвольных атомов или формул выражается с помощью отрицания и дизъюнкции, если использовать равенство $X \wedge Y = \neg(\neg X \vee \neg Y)$.

Одним из функционально полных является базис $\{\neg, \supset\}$. Если выразить импликацию через дизъюнкцию и отрицание ($X \supset Y = \neg X \vee Y$), то этот базис сводится к функционально полному базису $\{\neg, \vee\}$. В базисе $\{\neg, \supset\}$ в последние годы принято строить аксиоматику ***исчисления высказываний*** – одного из разделов математической логики, изучающего общезначимые формулы

Формулы называются ***тождественно истинными*** (или общезначимыми, или тавтологиями), если при всех значениях входящих в них атомов они принимают значение 1 (истинно), и ***тождественно ложными***, если при всех значениях входящих в них атомов они принимают значение 0 (ложно). Если формула не тождественно ложна, то это ***выполнимая*** формула. Если каждому атому, входящему в формулу, присвоить значение истинности (например, $A = 0$,

$B = 1$, $C = 1$), то набор этих равенств составляет (элементарную) ***подстановку*** данной формулы. Если при этом будет установлено, что для некоторой подстановки формула истинна, то такая подстановка называется ***выполняющей подстановкой***.

Например, для формулы $\neg B \supset \neg A$ система равенств $A = 1$, $B = 0$ есть подстановка, но это не выполняющая подстановка (см. табл. 1.6).

Алгебру логики можно рассматривать как интерпретацию исчисления высказываний, поскольку она позволяет устанавливать тождественную истинность логических формул не с помощью символьных преобразований, а на основе алгебраических операций (уже упомянутый табличный способ). Табличным способом нетрудно проверить, что к тождественно истинным формулам относятся $A \vee \neg A$ и $B \supset (A \supset B)$, а к тождественно ложным – $A \wedge \neg A$ и $(B \supset A) \wedge (\neg A \wedge B)$. Кроме проверки общезначимости формул исчисления высказываний, табличный способ может применяться и для доказательства законов алгебры множеств. Рассмотрим это подробнее.

Для установления соответствия между законами алгебры множеств и алгебры логики необходимо вначале сопоставить операции алгебры множеств и логические связки алгебры логики:

Операции алгебры логики	$\overline{\ }$ (дополнение)	$\cap$	$\cup$
Логические связки алгебры множеств	$\neg$ (отрицание)	$\wedge$	$\vee$

Теперь соответствие между формулами алгебры множеств и выражениями алгебры логики можно установить с помощью следующих правил:

1) все законы алгебры множеств, выраженные как равенство некоторого выражения универсуму, соответствуют общезначимой формуле алгебры логики (например, $A \cup \overline{A} = U$ соответствует общезначимой формуле);

2) все законы алгебры множеств, выраженные как равенство некоторого выражения пустому множеству (например, $A \cap \overline{A} = \varnothing$) соответствуют тождественно ложной формуле алгебры логики;

3) если закон алгебры множеств устанавливает включение между двумя множествами, заданными определенными выражениями, то замена символа "$\subseteq$" символом импликации ($\supset$) в соответствующей формуле алгебры логики приводит к тому, что эта формула становится общезначимой;

4) законы алгебры множеств, выраженные в форме "если A, то B", записываются в алгебре логики как импликация ($A \supset B$), при этом полученная формула общезначима.

В качестве примера выполним табличное доказательство одного из рассмотренных в предыдущем разделе законов сохранения алгебры множеств: если $X \subseteq Y$, то для любого множества Z справедливо $X \cap Z \subseteq Y \cap Z$.

С учетом приведенных выше правил соответствия этот закон в алгебре логики выражается формулой $(X \supset Y) \supset ((X \wedge Z) \supset (Y \wedge Z))$. Ее общезначимость доказана табличным методом в табл. 1.9.

Нетрудно убедиться, что закон сохранения нестрогого включения подтверждается с помощью табличного метода в алгебре логики, однако его доказательство оказалось более трудоемким, чем доказательство с использованием законов алгебры множеств. Для использования табличного метода доказательства необходимо экспоненциальное число возможных вариантов: если число атомов в формуле равно n, то при ее анализе табличным методом потребуется рассмотреть и сопоставить 2^n вариантов, что соответствует числу строк в полной таблице.

Таблица 1.9

X	Y	Z	$X \supset Y$	$X \wedge Z$	$Y \wedge Z$	$(X \wedge Z) \supset (Y \wedge Z)$	$(X \supset Y) \supset ((X \wedge Z) \supset (Y \wedge Z))$
0	0	0	1	0	0	1	1
0	0	1	1	0	0	1	1
0	1	0	1	0	0	1	1
0	1	1	1	0	1	1	1
1	0	0	0	0	0	1	1
1	0	1	0	1	0	0	1
1	1	0	1	0	0	1	1
1	1	1	1	1	1	1	1

Несмотря на кажущуюся простоту, табличный метод обладает одним существенным недостатком: когда число переменных в формуле превышает 60, его использование для решения логических задач практически невозможно даже с помощью современной вычислительной техники. Число необходимых

вариантов в этом случае выражается формулой 2^n, что означает известную многим исследователям логического вывода "экспоненциальную катастрофу". Было изобретено немало методов борьбы с этой катастрофой [*Гэри*, 1982]. Многие из них основаны на законах алгебры логики и алгебры множеств. Некоторые из методов приведены в главе 3. В настоящее время известно много частных случаев "труднорешаемых" задач логики и дискретной математики, решение которых не требует экспоненциального перебора вариантов. Но общего алгоритма, позволяющего при решении всех задач логического вывода избежать экспоненциальной катастрофы, пока не найдено. И даже не доказано существование или не существование такого алгоритма.

Проведенный выше краткий обзор определений алгебры логики показывает, что в алгебре логики и в алгебре множеств много общего, по крайней мере, законы этих систем однозначно соответствуют друг другу при замене соответствующих обозначений и терминов. Сходство этих систем подтверждается уже упомянутой теоремой Стоуна.

1.3. Булевы алгебры и системы логического вывода

1.3.1. Формальные системы

Описанные выше алгебраические системы имеют множество самостоятельных приложений. Например, алгебра логики применяется при исследовании релейно-контактных схем, а алгебра множеств – универсальное средство описания различного рода алгоритмов. Однако, в рамках формального подхода алгебраические системы в целом и булевы алгебры в частности традиционно воспринимаются лишь как модели или интерпретации некоторых формальных систем (алгебра множеств – как модель силлогистики Аристотеля, алгебра логики – как интерпретация исчисления высказываний). Возможно, в какой-то степени это объясняется историческими этапами формирования математики. Так, алгебра множеств была создана в процессе многовековых поисков математических оснований для силлогистики.

Рассмотрим понятия формальной системы и ее интерпретации более детально.

Математическая логика развивалась в XX веке как система, претендующая на то, чтобы выразить средствами ТФС все, имеющее отношение к

доказательствам. Формальная система содержит всего четыре компонента:

алфавит – совокупность символов: буквы, скобки, специальные знаки;

синтаксические правила, позволяющие составлять из символов алфавита правильные цепочки символов (формулы, предложения);

аксиомы – выбранная совокупность правильных предложений;

правила (логического) ***вывода***, позволяющие выводить из аксиом теоремы и следствия.

Правила вывода в ТФС сформулированы таким образом, чтобы из символьных конструкций (предложений), которые в данной системе выполняют роль аксиом или теорем, можно было получать новые символьные конструкции, которые уже понимаются как следствия или новые теоремы.

Другими словами, если заданы некоторый алфавит, множество формул (синтаксических правил их составления), множества аксиом и правил вывода, то тем самым задана некоторая ***формальная система F*** [*Поспелов,* 1989]:

$$\boldsymbol{F} = <T, L, Q, R>. \qquad (1.4)$$

Здесь T – конечное множество символов, базовых элементов, кирпичиков, не расчленяемых на более простые. Примерами таких элементов служат буквы (графемы) или детали в детском конструкторе. Единственное требование к элементам множества T состоит в том, что для любого элемента за конечное число шагов можно узнать, принадлежит он T или нет, а также отличить одни элементы от других, отождествляя одинаковые элементы.

L – множество правил грамматики (синтаксические правила), применение которых к символам из T позволяет конструировать правильно построенные формулы. Так, из графем возникают линейно упорядоченные сочетания, называемые словами, предложениями, текстами; из деталей детского конструктора возникают более сложные образования, в которых отдельные элементы набора скреплены.

Множество Q включает выделенные синтаксически правильные формулы, принимаемые без доказательств, к которым предъявляются специфические требования – полнота, непротиворечивость, независимость и т.д. Они называются аксиомами.

К основным аксиомам формальной системы (они называются логическими аксиомами) могут быть добавлены аксиомы, описывающие некоторую

предметную область. Они называются собственными или нелогическими аксиомами. Формальная система с нелогическими аксиомами называется ***формальной теорией***. Например, если к аксиомам исчисления предикатов добавить аксиомы, описывающие арифметику или теорию групп, то получим формальную теорию – формальную арифметику или формальную теорию групп.

Наконец, R представляет собой совокупность процедур (они называются ***правилами вывода***), с помощью которых можно получать одни синтаксически правильные предложения из совокупности других.

Вывод формулы B из $A_1, A_2, ..., A_n$ – это последовательность формул
$F_1, F_2, ..., F_m,$ (1.5)

где $F_m = B$, а любая F_i $(1 \le i \le m\text{-}1)$ – либо аксиома, либо A_i $(1 \le i \le n)$, либо непосредственное следствие из $F_1, F_2, ..., F_{m\text{-}1}$ по одному из правил вывода.

В этом случае говорят, что B ***выводима*** из $A_1, A_2, ..., A_n$ (или является следствием $A_1, A_2, ..., A_n$). Элементы последовательности $A_1, A_2, ..., A_n$ называются ***посылками вывода.***

Доказательство формулы в формальной теории – вывод из пустого множества формул, то есть вывод, в котором в качестве посылок используются только аксиомы. Формула, для которой существует доказательство, называется ***доказуемой формулой*** или ***теоремой***.

Применяются два варианта записи логического вывода. Первый вариант записи: $\frac{A_1, A_2, ..., A_n}{B}$ используется в основном в тех случаях, когда формула B непосредственно выводима из формул $A_1, A_2, ..., A_n$, с помощью какого-то одного правила вывода.

Второй вариант записи $A_1, A_2, ..., A_n \vdash B$ используется, как правило, в более общем случае. Символом "$\vdash$" (иногда вместо него пишут символ "$\Rightarrow$") обычно обозначается ***отношение выводимости***. Если знак "$\vdash$" находится перед формулой (например, $\vdash B$), значит, B – выведенная теорема. Для большей точности этот знак сопровождается нижним индексом, который указывает на имя теории, в которой осуществляется вывод, например, $\vdash_L$ обозначает отношение выводимости в теории L.

В качестве примера формальной системы можно привести силлогистику

Аристотеля, подробно рассмотренную ниже. Ее элементы T – это, например, имена терминов, а также символы **A**, **E**, **O**, **I**. Синтаксические правила образуют из элементов нормальные формы представления высказываний *Asp*, *Esp* и т.п. Исходные аксиомы – законы силлогистики: закон тождества $(\vdash Ass)$, закон противоречия $(\vdash \neg(Asp \wedge Esp))$, закон исключенного третьего $(\vdash (Isp \vee Osp))$. Наконец, к правилам вывода относится таблица получения заключений в правильных модусах при наличии посылок для этих заключений.

Формализация всякой содержательной теории начинается с выбора символов формальной теории, ***языка теории***. Один из наиболее изученных формальных языков – язык исчисления предикатов первого порядка. В современной логике он играет ключевую роль и служит одним из основных языков представления знаний в искусственном интеллекте. Алфавит T этого языка включает:

1. Счетное множество букв: $z, y, x, \ldots$; которое будем называть множество символов для обозначения переменных языка.

2. Счетное множество букв: $a, b, c, \ldots$; которое будем называть множество символов для обозначения констант языка.

3. Счетное множество прописных букв $P, Q, \ldots$; для обозначения предикатных символов языка.

4. Счетное множество строчных букв $f, g, \ldots$; для обозначения функциональных символов.

5. Символы для обозначения логических связок ($\wedge$, $\vee$, $\rightarrow$, $\neg$).

6. Символы для обозначения кванторов ($\forall$, $\exists$).

7. (,) – скобки.

Предикатные буквы $P, Q, \ldots$ и функциональные символы $f, g, \ldots$ могут быть n-местными. Грамматика языка исчисления предикатов первого порядка приведена, например, в [*Колмогоров,* 1982; *Мендельсон*, 1984; *Новиков*, 1973]. Пусть дана формальная система ***F*** с языком исчисления предикатов первого порядка и алгебраическая система $\boldsymbol{B} = \langle A, \Omega_F, \Omega_P \rangle$. Зададим отображение ***I***, которое всякому константному символу из алфавита T системы ***F*** ставит в соответствие некоторый элемент $a \in A$ из алгебраической системы ***B***, всякому n-местному предикатному символу – n-местное отношение из Ω_P, всякому n-местному функциональному символу – n-местную операцию из Ω_F. Тогда ***I*** называется ***интерпретирующим отображением***, а алгебраическая система ***B***

– ***моделью*** или ***интерпретацией*** формальной системы ***F.*** Аналогично может быть введено понятие интерпретации для любой формальной системы.

Формальные системы часто ассоциируются с ***дедуктивными системами*** или ***исчислениями*** [*Маслов,* 1986]. При этом неявно предполагается, что алгебраический подход не предназначен или, в лучшем случае, плохо приспособлен для моделирования логического вывода. Современная математическая логика построена по строгим канонам чистого исчисления и, хотя найдены соответствия между многими разделами логики и алгебраическими системами, однако сам по себе алгебраический подход в настоящее время в теоретических исследованиях по классической логике применяется крайне редко. Он используется, в основном, в прикладных исследованиях, но отнюдь не в исследованиях, которые, по мнению многих специалистов, причисляются к исследованиям фундаментальным. Такая установка, хотя и следует из тенденции развития современной логики, но, по мнению авторов, не является безусловной истиной.

При организации процедур вывода в программных системах на основе формального подхода, помимо перечисленных во Введении, возникают следующие проблемы [*Поспелов,* 1989]:

1. Трудности понимания поступающих в систему сообщений, истолкования их в терминах, понятных на уровне внутреннего языка.

2. Трудности проверки поступающего сообщения на согласованность содержащейся в нем информации с той информацией, которая ранее хранилась в памяти системы.

3. Трудности организации поиска, не опирающегося на какую-либо цель, или при известной цели (в случае доказательства теоремы) не опирающегося на какие-либо соображения о путях движения по дереву вывода.

4. Трудности, связанные с прекращением процедур вывода и формированием отрицательного ответа на поставленный перед системой вопрос о выводимости.

Пункты 3 и 4 показывают, что с помощью формальных систем так и не удалось окончательно решить проблему "экспоненциальной катастрофы", несмотря на достигнутые значительные результаты по снижению трудоемкости процедур логического вывода.

Как показано в последующих главах, при более углубленном исследовании

алгебраического подхода в логике оказывается, что в его рамках решаются перечисленные во Введении проблемы моделирования и анализа рассуждений, задачи снижения трудоемкости интеллектуальных процедур. Также во многих случаях существенно упрощается, по сравнению с формальным подходом, разработка алгоритмов решения задач логического анализа, включая алгоритмы логического вывода. Кроме того, если в формальной системе аксиомы и правила вывода принимаются "на веру", поскольку не могут быть проверены средствами самой системы, то алгебраические методы позволяют обосновать правильность этих конструкций.

Далее кратко описываются наиболее популярные дедуктивные системы, уточняется их взаимосвязь с соответствующими алгебраическими системами.

1.3.2. Силлогистика Аристотеля и алгебра множеств

До начала XIX столетия основным инструментом логического анализа была созданная еще Аристотелем силлогистика [*Аристотель,* 1978]. Объектом исследования силлогистики является ***категорический силлогизм,*** в котором содержится три суждения, из них два суждения являются ***посылками***, а третье – ***заключением***. Каждое суждение силлогизма должно быть одним из четырех предусмотренных Аристотелем типов. К ним относятся:

A (общеутвердительное): Все *S* есть *P*;

E (общеотрицательное): Все *S* не есть *P*;

I (частноутвердительное): Некоторые *S* есть *P*;

O (частноотрицательное): Некоторые *S* не есть *P*.

Других типов суждений в категорическом силлогизме не предусмотрено. В каждом суждении утверждается определенное соотношение между ***субъектом*** (*S*) и ***предикатом*** (*P*). Деление на субъекты и предикаты в суждениях относительно – в суждениях в качестве субъектов могут использоваться термины, которые в других суждениях играют роль предикатов, и наоборот.

Если рассматривать термины *S* и *P* как множества, то каждый тип суждений можно представить как соотношения между множествами [*Колмогоров,* 1982]:

A: $S \subseteq P$ или $S \setminus P = \varnothing$;

E: $S \subseteq \overline{P}$ или $S \cap P = \varnothing$;

I: $S \cap P \neq \varnothing$;

O: $S \cap \overline{P} \neq \varnothing$.

Логический вывод в силлогистике производится в соответствии с определенными правилами. Приведем краткое описание одной из самых распространенных систем таких правил. Каждое суждение состоит из двух терминов, независимо от того, употребляется ли этот термин с логическим квантором "некоторые" или с квантором "все" (или "каждый"). Причем квантор может применяться только к левому термину суждения. Отрицание нельзя употреблять в левом термине (субъекте), хотя этот запрет можно обойти, если представить термин с отрицанием как позитивный термин. Например, заменив термин "неядовитый" позитивным термином *X*, термин "ядовитый" надо будет обозначить как не-*X*.

В двух посылках категорического силлогизма содержится три термина. Один из терминов должен встречаться в каждой посылке – он называется "***средним***" термином. Два других термина называются "***меньшим***" и "***большим***". Например, в посылках

"Все животные передвигаются сами",

"Планеты не животные"

термин "животные" есть средний термин. Считается, что больший термин – это тот, который содержится в первой посылке и не является средним, а меньший термин – тот, который содержится во второй посылке и не является средним. В наших посылках больший термин – "передвигающиеся сами", а меньший термин – "планеты".

В заключении силлогизма должно быть два термина, из которых субъектом является меньший термин, а предикатом – больший. Другими словами, заключение силлогизма допускается только в следующей форме: левая часть заключения должна содержать меньший термин, а правая – больший.

Каждое из трех суждений категорического силлогизма может быть любым из четырех возможных типов суждений. Отсюда ясно, что число возможных сочетаний вариантов категорического силлогизма может быть 4^3, т.е. 64. Но

этим не исчерпывается разнообразие вариантов. Необходимо учитывать, что существуют еще четыре фигуры силлогизма, в каждой из которых может быть 64 различных варианта. Итого получается 256, но далеко не все из этих сочетаний – правильные.

Рассмотрим теперь, что подразумевается под ***фигурой силлогизма***. Обозначим буквами *P*, *M* и *S* соответственно больший, средний и меньший термины силлогизма. Фигура силлогизма определяется порядком расположения терминов в посылках. В традиционной логике фигуры принято изображать с помощью диаграмм, приведенных на рис. 1.5. Например, приведенные выше посылки о планетах и животных относятся к первой фигуре силлогизма, так как порядок расположения терминов соответствует диаграмме этой фигуры на рис. 1.5.

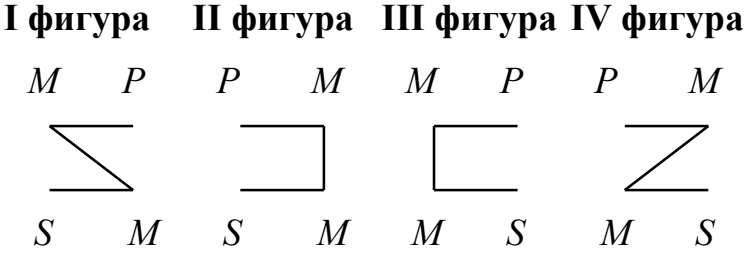

Рис. 1.5. Фигуры силлогизма

Еще одно важное понятие Аристотелевой силлогистики – ***модусы силлогизма***. Если взять какой-либо силлогизм, и каждое содержащееся в нем суждение заменить обозначением его типа (**A**, **E**, **I** или **O**), то получим слово из трех символов, которое и есть обозначение модуса. Например, модус *EAO* означает, что в силлогизме первая посылка относится к типу **E**, вторая – к типу **A**, а заключение – к типу **O**. При этом модусы различаются не только по сочетаниям букв, но и по принадлежности к определенной фигуре. Например, модус *AEE* правилен для второй и четвертой фигур и неправилен – для первой и третьей.

Таких модусов, как уже было сказано ранее, возможно 256. Из них правильны только 24 (19 основных и 5 дополнительных). Дополнительные правильные модусы можно получить из тех основных модусов, у которых заключение относится к типам **A** (общеутвердительное) или **E** (общеотрицательное). Тогда соответствующие частные типы заключений (соответственно **I** или **O**) считаются правильными. Например, если заключение

имеет вид "Все *X* не есть *Y*", то по правилам силлогистики суждение "Некоторые *X* не есть *Y*" также правильно.

Если определен модус соответствующего силлогизма и его принадлежность определенной фигуре, то для распознавания правильности или неправильности его заключения можно воспользоваться приведенным ниже списком правильных модусов.

1-я фигура	**2-я фигура**	**3-я фигура**	**4-я фигура**
AAA (Barbara)	*AEE (Camestres)*	*AII (Datisi)*	*AEE (Camenes)*
EAE (Celarent)	*EAE (Cesare)*	*EIO (Ferison)*	*EIO (Fresison)*
AII (Darii)	*AOO (Baroko)*	*IAI (Disamis)*	*IAI (Dimaris)*
EIO (Ferio)	*EIO (Festino)*	*OAO (Bocardo)*	*AAI (Bramalip)*
		AAI (Darapti)	*EAO (Fesapo)*
		EAO (*Felapton*)	

Здесь каждый модус, помимо обозначения, имеет название, в котором гласные буквы соответствуют сокращенному имени модуса. Например, имя "*Barbara*" определяет модус *AAA* первой фигуры.

С помощью данного списка можно вывести из двух посылок правильное заключение. Покажем, как выполняется эта процедура для приведенного выше силлогизма. Мы уже определили, что модус этого силлогизма относится к первой фигуре. Начальные буквы этого модуса *AE*, но среди правильных модусов первой фигуры нет таких, которые начинались бы с этих букв. Следовательно, в соответствии с правилами силлогистики Аристотеля вывести какое-либо заключение из этих посылок невозможно.

Таким образом, переход к заключению происходит чисто механически. Надо только определить по виду выбранных посылок номер фигуры, а затем обратиться к таблице правильных силлогизмов, в которой по номеру фигуры и типам посылок (первые две буквы в обозначении модуса) находится тип заключения (третья буква в обозначении модуса).

Попытки найти математическое обоснование силлогистики предпринимались с давних пор. Сомнение в правильности некоторых модусов силлогизма выражали многие известные ученые. Лейбниц в первой построенной им системе в качестве терминов использовал целые числа, причем

число, соответствующее предикату, должно было без остатка делиться на число, соответствующее субъекту. Но обосновать с помощью этой системы все правильные модусы силлогистики Лейбницу не удалось. Алгебра множеств появилась первоначально также как математический аппарат для обоснования правильности модусов и сформировавшейся процедуры логического вывода: категорические силлогизмы проверялись с помощью диаграмм Венна [*Кузичев,* 1968; *Поспелов*, 1989] или законов алгебры множеств.

Если в рассуждении содержится более двух посылок, то оно относится к ***полисиллогизмам*** или к ***соритам***. Полисиллогизм можно также решать с помощью механического применения правил категорического силлогизма или на основе диаграмм Венна, последовательно выбирая некоторые пары исходных посылок или промежуточных заключений. Однако анализ полисиллогизмов таким способом крайне трудоемок и во многих случаях, когда структура полисиллогизма не соответствует некоторым канонам, не приводит к однозначному результату.

В современной теории логического вывода принято, что правильность или неправильность следствия не зависит от порядка расположения исходных суждений. Но, если мы переставим посылки в приведенном выше примере, то по правилам силлогистики Аристотеля получим другой результат.

На рубеже XIX и XX столетий начала интенсивно развиваться математическая логика, которая была основана уже на других предпосылках. Силлогистика при этом существовала как бы отдельно от нее. В 1957 г. в Оксфорде была издана ставшая широко известной книга Я. Лукасевича "Аристотелевская силлогистика с точки зрения современной формальной логики". На русском языке эта книга была издана в 1959 г. [*Лукасевич,* 1959]. В ней силлогистика представлена как аксиоматическая система и показано, что некоторые аксиомы силлогистики несовместимы с аксиомами исчисления предикатов в математической логике. В России исследования Я. Лукасевича в этом направлении продолжили В.А. Смирнов с учениками. В результате было установлено [*Смирнов,* 1987], что в исчислении предикатов невозможно доказать следующие утверждения, получаемые по правилам силлогистики:

1) из суждения "Все *S* есть *P*" следует суждение "Некоторые *S* есть *P*", а из суждения "Все *S* не есть *P*" следует суждение "Некоторые *S* не есть *P*";

2) суждения "Все S есть P" и "Некоторые S не есть P" не могут быть оба истинными;

3) суждения "Некоторые S есть P" и "Некоторые S не есть P" не могут быть оба ложными;

4) модусы *Darapti*, *Felapton*, *Fesapo*, *Bramalip*.

Оказалось, что основная причина этих несоответствий такова: традиционная силлогистика не учитывает того, что некоторые субъекты и предикаты суждений могут представлять собой пустые множества. Рассмотрим первое из перечисленных несоответствий, но сформулируем его на языке алгебры множеств – это позволит значительно упростить анализ. Тогда вместо утверждения «Все S есть P» получим $S \subseteq P$, а вместо "Некоторые S есть P" – $S \cap P \neq \varnothing$. Если допустить, что $S = \varnothing$, то $S \subseteq P$ подтверждается, но неравенство $S \cap P \neq \varnothing$ оказывается ошибочным.

Доказано, что при устранении указанной причины несоответствий за счет введения соответствующих поправок в формальные представления типов суждений силлогистика оказывается погруженной в математическую логику [*Смирнов,* 1987]. Для иллюстрации приведем один из возможных вариантов такого "перевода" типов суждений в формулы математической логики [*Гладкий*, 2001]:

$$\begin{aligned}
&\mathbf{A}\colon\ (\exists y S(y) \wedge \exists z P(z)) \wedge \forall x(S(x) \supset P(x));\\
&\mathbf{E}\colon\ (\exists y S(y) \wedge \exists z P(z)) \wedge \forall x(S(x) \supset \neg P(x));\\
&\mathbf{I}\colon\ (\exists y S(y) \wedge \exists z P(z)) \wedge \exists x(S(x) \wedge P(x));\\
&\mathbf{O}\colon\ (\exists y S(y) \wedge \exists z P(z)) \wedge \exists x(S(x) \wedge \neg P(x)).
\end{aligned} \qquad (1.6)$$

В первой части этих формул $(\exists y S(y) \wedge \exists z P(z))$ утверждается, что меньший (S) и больший (P) термины силлогизма не могут представлять пустые множества. Как видим, "перевод" не уменьшает сложность математического представления силлогистики. Более того, исследования, направленные на поиски формальных представлений силлогистики, значительно усложнили саму процедуру логического вывода и анализа рассуждений [*Гладкий*, 2001]. Далее опишем самые популярные в настоящее время системы логического вывода.

1.3.3. Логические исчисления и их интерпретация

В математической логике наиболее распространены три системы логического вывода: 1) исчисления гильбертовского типа, которые были предложены в работах Д. Гильберта и В. Аккермана [*Гильберт,* 1947]; 2) натуральное исчисление (или естественный вывод), разработанное логиком Г. Генценом [*Генцен*, 1967], и 3) логический вывод на основе принципа резолюции, ставший широко известным после публикации статьи М. Девиса и Х. Патнема [*Davis*, 1960]. Рассмотрим особенности логического вывода для систем перечисленных типов. Каждый из типов проиллюстрируем сначала на примере исчисления высказываний, а затем – исчисления предикатов.

Исчисления высказываний гильбертовского типа в современном варианте основаны на следующих аксиомах и правилах вывода:

Аксиомы:

(A_1) $A \supset (B \supset A)$;

(A_2) $(A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C))$;

(A_3) $(\neg B \supset \neg A) \supset ((\neg B \supset A) \supset B)$.

Правила вывода:

правило подстановки: в любую аксиому или выведенную формулу можно вместо любого атома вставлять любое правильно построенное предложение;

модус поненс (MP): если A и $A \supset B$ – выводимые формулы, то B – выводимая формула.

Работу такой системы можно понять на следующем примере. Пусть необходимо вывести из приведенных выше аксиом теорему $\vdash A \supset A$. Вывод осуществляется по шагам:

1) $(A \supset ((A \supset A) \supset A)) \supset ((A \supset (A \supset A)) \supset (A \supset A))$ – подстановка в (A_2) $A \supset A$ вместо B и A вместо C;

2) $A \supset ((A \supset A) \supset A)$ – подстановка в (A_1) $A \supset A$ вместо B;

3) $(A \supset (A \supset A)) \supset (A \supset A)$ – применение *MP* к 2) и 1);

4) $A \supset (A \supset A)$ – подстановка в (A_1) A вместо B;

5) $A \supset A$ – применение *MP* к 4) и 3).

Важное место в исчислениях занимает теорема дедукции, впервые сформулированная и доказанная Ж. Эрбраном в 1930 г. Для исчисления

высказываний ее формулировка такова (приводится по книге [*Клини,* 1973]):

Теорема 1.3. (теорема дедукции).

(a) Если $A \vdash B$, то $\vdash A \supset B$;

(b) Если $A_1, ..., A_{m-1}, A_m \vdash B$, то $A_1, ..., A_{m-1} \vdash A_m \supset B$.

В исчислении предикатов в формулировку теоремы дедукции необходимо ввести некоторые дополнительные условия. Их мы рассмотрим позже.

Натуральное исчисление высказываний предназначено для того, чтобы приблизить формальное исчисление к естественному выводу. В этом исчислении состав правил вывода расширен, они имеют ясную интерпретацию на естественном языке, которая приведена в [*Гладкий,* 2001], и лежат в основе современной формальной теории доказательств [*Такеути,* 1978]. К тому же эти правила можно вывести как теоремы, используя исчисление гильбертовского типа. Для исчисления высказываний обычно применяются следующие правила вывода натурального исчисления.

Для произвольных формул A, B, C и некоторого множества формул Δ применимы следующие правила вывода:

1) $\dfrac{A, B}{A \wedge B}$ — *введение конъюнкции* (*ВК*);

2) $\dfrac{A \wedge B}{A}$; $\dfrac{A \wedge B}{B}$ — *удаление конъюнкции* (*УК*);

3) $\dfrac{A}{A \vee B}$; $\dfrac{B}{A \vee B}$ — *введение дизъюнкции* (*ВД*)

4) Если $\Delta, A \vdash C$ и $\Delta, B \vdash C$, то $\Delta, A \vee B \vdash C$ *удаление дизъюнкции* (*УД*),

5) Если $\Delta, A \vdash B$, то $\Delta \vdash A \supset B$ — *введение импликации* (*ВИ*);

6) $\dfrac{A, A \supset B}{B}$ — *удаление импликации* (*УИ*);

7) Если $\Delta, A \vdash$ *false*, то $\Delta \vdash \neg A$ — *введение отрицания* (*ВО*);

8) $A, \neg A \vdash$ *false* — *удаление отрицания* (*УО*);

9) *false* $\vdash A$ – *правило Дунса Скотта* (*ДС*);

10) $\vdash A \vee \neg A$ – *закон исключенного третьего* (*ИТ*);

11) Если A – одна из формул в составе Δ, то $\Delta \vdash A$ – *правило тривиальной выводимости* (*ТВ*).

Приведенные в скобках обозначения (*ВК*, *УК*, *ВД* и т.д.) – общеприняты в русскоязычной литературе по логическому выводу. Нетрудно видеть, например, что правило *ВИ* – один из вариантов теоремы дедукции, правило *УИ*

– правило модус поненс, правило *ДС* – формальный вариант известного логического принципа "из лжи можно вывести все, что угодно".

Обоснование аксиом исчисления высказываний легко выполнить на основе интерпретации этой системы – алгебры логики (см. п. 1.2.). Множество формул исчисления высказываний можно отобразить во множество формул алгебры логики, где переменным уже приписываются значения 1 (истина) или 0 (ложь), а формулам – некоторые таблицы истинности. При этом взаимосвязь между формулами исчисления высказываний и алгебры логики устанавливается с помощью теоремы 1.4 [*Игошин,* 2008].

Теорема 1.4. Всякая доказуемая в формализованном исчислении высказываний формула является тождественно истинной формулой (или тавтологией) алгебры логики.

Таким образом, можно сказать, что исчисление высказываний описывает способы получения всех возможных общезначимых формул.

Если определить логические связки "$\wedge$" и "$\vee$", то из аксиом (A_1) – (A_3) можно вывести как теоремы все основные законы алгебры логики (см. [*Мендельсон*, 1984]).

Для доказательства корректности того или иного вывода $A_1, A_2, ..., A_n \vdash A_{n+1}$ можно использовать уже упомянутую теорему дедукции, которая позволяет во многих сложных случаях перейти от доказательства с помощью построения дерева вывода, не удобного для практических применений, к доказательствам с помощью алгебраических методов, в частности, к подтверждению или опровержению общезначимости логической формулы, полученной с помощью теоремы дедукции. Поясним сказанное на примере. Пусть имеется вывод: $A, B \vdash C$. Тогда, согласно теореме дедукции, имеем $A \vdash B \supset C$. Применив еще раз эту теорему, получим $\vdash A \supset (B \supset C)$. С учетом теоремы 1.4, для обоснования корректности вывода $\vdash A \supset (B \supset C)$ необходимо установить общезначимость формулы алгебры логики $A \supset (B \supset C)$.

В исчислении высказываний корректность тех или иных правил вывода также можно проверять алгебраически с помощью теоремы дедукции, подобно тому, как на основе законов алгебры множеств производится обоснование правил силлогистики (см., например, [*Поспелов,* 1989]).

Часто для доказательств с использованием алгебраических методов применяется не сама теорема дедукции, а ее следствия: теоремы 1.5 и 1.6. Эти

теоремы приведены в [*Чень,* 1983].

Теорема 1.5. Пусть даны формулы $F_1, ..., F_n$ и формула G. G есть логическое следствие $F_1, ..., F_n$ тогда и только тогда, когда формула $((F_1 \wedge ... \wedge F_n) \supset G)$ общезначима.

Теорема 1.6. Пусть даны формулы $F_1, ..., F_n$ и формула G. G есть логическое следствие $F_1, ..., F_n$ тогда и только тогда, когда формула $(F_1 \wedge ... \wedge F_n \wedge \neg G)$ противоречива.

Формально исчисление предикатов можно представить как некоторое усложнение исчисления высказываний. Переход к исчислению предикатов означает ввод двух новых обязательных условий: 1) переход к фиксированным отношениям (предикатам) и 2) введение кванторов. Это отличие принципиально даже в тех случаях, когда в моделируемой системе область определения каждой переменной содержит ровно два элемента. Правила логического вывода для исчисления предикатов не во всех случаях применимы для систем с бинарными областями значений переменных.

В исчислении предикатов аксиомы и правила логического вывода исчисления высказываний остаются без изменений, к ним только добавляются новые аксиомы и правила вывода. Их формулировка для ***исчислений предикатов гильбертовского типа*** не во всех источниках одинакова, приведем их по [*Мендельсон,* 1984].

Аксиомы:

(A_4) $\forall x_i A(x_i) \supset A(t)$, где $A(x_i)$ есть формула, содержащая свободную переменную x_i, а t — терм, свободный для x_i в $A(x_i)$; в частности, если $t = x_i$, то данная аксиома выражается как $\forall x_i A(x_i) \supset A(x_i)$.

(A_5) $\forall x_i(A \supset B) \supset (A \supset \forall x_i B)$, если в формуле A не содержится свободных вхождений переменной x.

Правило вывода (обобщения): из A следует $\forall x_i A$. Другое название этого правила – правило связывания квантором всеобщности.

Правило обобщения само по себе трудно объяснимо. Если формула A содержит свободную переменную x_i (в формулировке правила это допускается), то в случае, когда формула A такова, что $\forall x_i A$ противоречива, получается явная нелогичность. Например, из утверждения "Грибы мухоморы ядовиты" выводится утверждение "Все грибы ядовиты", но поскольку последний тезис ложен, то из этого следует, что из любой логической формулы, в которой

переменная x_i не пробегает всех своих значений, выводится ложь.

Эту несообразность в исчислении гильбертовского типа в какой-то степени исправляет теорема дедукции для исчисления предикатов [*Мендельсон,* 1984]:

Теорема 1.7. Если $\Delta, A \vdash B$ и существует вывод, построенный без применения правила обобщения к свободным переменным формулы A, то $\Delta \vdash A \supset B$.

Из формулировки теоремы ясно, что правило обобщения к посылкам применяется не во всех случаях. Получается, что необъяснимая ситуация, связанная с правилом обобщения в исчислениях гильбертовского типа, корректируется с помощью теоремы дедукции.

В ***натуральном исчислении предикатов*** такого несоответствия нет. Кванторные правила вывода в нем уже не вызывают подобных вопросов (приводятся по [*Гладкий,* 2001]):

12) Если $\Delta \vdash A(x)$ и переменная x не входит свободно ни в одну из формул Δ, то $\Delta \vdash \forall x A(x)$; *введение общности* ($B\forall$)

13) Если подстановка переменной y вместо переменной x корректна, то $\forall x A(x) \vdash A(y)$; *удаление общности* ($У\forall$)

14) Если подстановка переменной x вместо переменной y корректна, то $A(y) \vdash \exists x A(x)$; *введение существования* ($B\exists$);

15) Если $\Delta, A(x) \vdash B$ и переменная x не входит свободно ни в одну формулу из Δ, то $\exists x A(x) \vdash B$. *удаление существования* ($У\exists$)

Здесь правило вывода $B\forall$ полностью соответствует теореме дедукции для исчисления предикатов. Подробное объяснение смысла этих правил вывода и примеры можно найти в [*Гладкий,* 2001].

Считается, что формулы исчисления предикатов имеют смысл (можно судить об их выполнимости), если задана какая-либо интерпретация входящих в них символов, т.е. термов, предикатов, функций, констант и т.д. В п. 1.3.1 уже приводилось общепринятое определение понятия "интерпретация" для исчисления предикатов, согласно которому алгебраическая система $\boldsymbol{B} = <A, \Omega_F, \Omega_P>$ является интерпретацией, если:

1) каждой предметной константе соответствует некоторый элемент из A;
2) каждой переменной соответствует область определения, равная A;
3) каждой m-местной функции соответствует отображение A^m в A (операция);

4) каждому m-местному предикату соответствует некоторое отношение на A^m.

A^m – m-кратное декартово произведение множества A.

Каждой формуле исчисления предикатов равносильно некоторое (возможно, пустое) многоместное отношение, задающее ее область истинности. Состав атрибутов этого отношения соответствует составу свободных переменных в исходной формуле [*Мендельсон,* 1984].

Во многих системах приходится использовать в качестве областей изменения переменных (атрибутов) различные множества, не совпадающие друг с другом. Например, это могут быть числа, интервалы, символы, идентификаторы, логические константы и т.д. Тогда употребление канонической интерпретации, в которой все атрибуты равны одному и тому же множеству A, вызывает немалые трудности. Чтобы этого избежать, можно использовать ***многосортное исчисление предикатов***, в котором допускается применение совокупности разных ***сортов*** в качестве областей изменения различных переменных. Тогда каждому сорту приписывается уникальное множество переменных, а схемы отношений используемых предикатов и функций должны быть заданы в определенных сортах.

Разумно предположить, что выполнимость формул, у которых предикаты соединены логическими связками, например, $P(x, z) \wedge Q(y, z)$, можно вычислить алгебраически как результат операций над соответствующими многоместными отношениями. В частности, для данного примера интерпретацией исходной формулы целесообразно считать отношение, вычисляемое как $P[XZ] \cap Q[YZ]$, где $P[XZ]$ – отношение, соответствующее области истинности предиката $P(x, z)$, а $Q[YZ]$ – отношение, соответствующее области истинности предиката $Q(y, z)$. Тогда, если выражение $P[XZ] \cap Q[YZ] = \varnothing$, то формула $P(x, z) \wedge Q(y, z)$ невыполнима. Однако, запись $P[XZ] \cap Q[YZ]$ в общем случае некорректна, поскольку отношения P и Q заданы в различных схемах, а операции алгебры множеств над такими отношениями не определены.

Прежде чем приступить к изложению принципа резолюции, введем несколько определений.

В исчислении высказываний можно выделить классы формул, обладающих полезными свойствами [*Яблонский*, 1966]. Среди них наибольший интерес представляют два универсальных класса – ***дизъюнктивная нормальная форма***

(ДНФ) и ***конъюнктивная нормальная форма*** (КНФ). Кроме ДНФ и КНФ, существуют другие универсальные классы, которые здесь не рассматриваются. Классы называются универсальными потому, что любую формулу исчисления высказываний можно эквивалентно преобразовать в любой из этих классов. Чтобы задать эти классы, нужно определить несколько новых понятий.

Литералом называется формула, которая состоит либо из одного атома, либо из отрицания атома.

Конъюнкт – это формула, состоящая из одного литерала или конъюнкции произвольного числа литералов.

Дизъюнкт есть формула, состоящая из одного литерала или дизъюнкции произвольного числа литералов.

Например, формула $B \wedge \neg C \wedge D$ – конъюнкт, а $\neg A \vee B \vee \neg D$ – дизъюнкт. В то же время формулу $\neg D$ можно интерпретировать как конъюнкт или как дизъюнкт. Теперь можно перейти к определениям универсальных классов.

Дизъюнктивная нормальная форма – это дизъюнкция произвольного числа конъюнктов.

Конъюнктивная нормальная форма есть конъюнкция произвольного числа дизъюнктов.

Рассмотрим одно из интересных свойств, которое устанавливает соотношение между этими классами. Если взять произвольную формулу, выраженную как ДНФ, то ее отрицание можно выразить как КНФ с помощью следующих простых преобразований:

1) заменить все литералы в исходной формуле на противоположные (т.е. литерал X преобразуется в литерал $\neg X$, а литерал $\neg Y$ – в литерал Y);

2) заменить все присутствующие в исходной формуле логические связки "$\wedge$" на связки "$\vee$", а все связки "$\vee$" – на связки "$\wedge$".

Например, если исходная формула есть ДНФ

$$(\neg B) \vee (A \wedge B \wedge \neg C) \vee (\neg A \wedge C),$$

то после приведенных выше преобразований получим формулу

$$(B) \wedge (\neg A \vee \neg B \vee C) \wedge (A \vee \neg C),$$

представляющая отрицание предыдущей формулы в виде КНФ.

Изложенное справедливо и для отрицания формулы, выраженной в виде КНФ. Те же преобразования позволяют получить ее отрицание в виде ДНФ. Данное соотношение основано на многократном применении законов де

Моргана.

В исчислении высказываний и предикатов имеется своеобразный "полиморфизм", когда эквивалентные формулы выражаются по-разному (например, $A \vee \neg C = \neg A \supset \neg C$). Подобный "полиморфизм", хотя и в меньшей степени, может существовать и в пределах одного класса формул, в частности, для формул классов КНФ и ДНФ. Например, с помощью табличного метода можно убедиться, что две ДНФ $(A \wedge \neg B \wedge C) \vee (\neg A \wedge C)$ и $(\neg A \wedge B \wedge C) \vee (\neg B \wedge C)$ эквивалентны.

Теперь рассмотрим принцип резолюции, который применяется для автоматического доказательства теорем и сводится к определению выполнимости или невыполнимости КНФ.

Принцип резолюции имеет определенные преимущества по сравнению с другими методами логического вывода и широко применяется в машинных реализациях систем искусственного интеллекта. Он опирается на теорему 1.6. Для формул исчисления высказываний суть этого принципа состоит в следующем. Последовательность формул $F_1, ..., F_n, \neg G$ (см. теорему 1.6) путем применения тождеств алгебры логики преобразуется в некоторое множество непустых дизъюнктов (предполагается, что эти дизъюнкты соединены друг с другом связкой "$\wedge$", то есть представляют собой КНФ). Затем выбираются пары дизъюнктов, в одном из которых содержится некоторый литерал, а в другом – его отрицание. Доказано, что следствие этой пары дизъюнктов есть дизъюнкт, содержащий все ее литералы, кроме контрарных. Например, если использовать принцип резолюции к паре дизъюнктов $A \vee \overline{B} \vee C$ и $\overline{B} \vee \overline{C}$, то, удаляя из них контрарные литералы C и $\overline{C}$ и объединяя оставшиеся, получим в качестве следствия дизъюнкт $A \vee \overline{B}$.

Далее новый дизъюнкт приписывается к исходному множеству дизъюнктов, и вывод продолжается до тех пор, пока не будет получен пустой дизъюнкт (в этом случае вывод был корректным), либо окажется, что пустой дизъюнкт получить невозможно (тогда вывод не подтверждается).

Рассмотрим пример. Пусть требуется методом резолюций доказать или опровергнуть корректность вывода

$$\overline{A}, \overline{B} \vee C, A \vee \overline{B} \vee \overline{C} \vdash \overline{A} \wedge \overline{B}.$$

Применив теорему 1.6 и закон де Моргана к отрицанию формулы $\overline{A} \wedge \overline{B}$,

получим КНФ, которую можно представить как множество дизъюнктов:

1) $\overline{A}$;

2) $\overline{B} \vee C$;

3) $A \vee \overline{B} \vee \overline{C}$;

4) $A \vee B$.

Далее, применяя принцип резолюции, получаем новые дизъюнкты, которые добавляем к исходным:

5) $\overline{B} \vee \overline{C}$ (по принципу резолюции из 1 и 3);

6) $\overline{B}$ (по принципу резолюции из 2 и 5);

7) B (по принципу резолюции из 1 и 4);

8) пустой дизъюнкт (по принципу резолюции из 6 и 7).

Таким образом, в данном примере правильность вывода подтверждается.

Чтобы использовать метод резолюций для формул исчисления предикатов, необходимо предварительно выполнить ряд преобразований с целью получения бескванторной КНФ. Эти методы подробно описаны в [*Чень,* 1983] и сопровождаются многочисленными примерами.

Итак, в рамках формального подхода процедура логического вывода часто сводится к задаче анализа выполнимости КНФ, решения которой породили целое направление в математической логике и в Computer science. Речь идет о направлении, которое называется "вычислительная сложность алгоритмов".

1.3.4. Сложность алгоритмов логического вывода (задача "выполнимость КНФ")

Уже на первых этапах разработки программных систем, в частности систем искусственного интеллекта, начали появляться задачи, которые даже при небольшом числе исходных данных были практически нереализуемыми из-за громадного объема требуемых вычислительных операций. Возникла проблема классификации задач по степени сложности. Первые результаты в этом направлении были получены Г.С. Цейтиным в конце 50-х годов [*Яновская,* 1959], но по ряду причин не стали широко известными. Исходным пунктом дальнейших исследований по теории сложности вычислений стали работы М.О. Рабина [*Rabin,* 1960], Ю. Хартманиса и Р.Е. Стирнза [*Hartmanis,* 1967], в

которых достаточно четко обозначилась постановка проблемы. Интенсивное развитие этой теории началось после опубликования в 1971 и 1972 гг. двух статей С.А. Кука [*Cook,* 1971] и Р.М. Карпа [*Karp,* 1972].

Кратко результат работы С.А. Кука заключается в следующем. В основу теории *NP*-полноты положен класс задач *NP* (Nondeterministically Polynomial). Он состоит из задач распознавания, то есть задач, где возможны только два варианта ответа ("да" или "нет"), и содержит не только задачи, решаемые за время, выраженное полиномом фиксированной степени от длины входного предложения, но и задачи, для которых полиномиальный алгоритм решения в общем случае неизвестен. В своей статье С.А. Кук доказал, что любую задачу класса *NP* можно при использовании рациональной системы кодирования представить в виде некоторой КНФ, при этом преобразование условий любой задачи класса *NP* в КНФ занимает полиномиальное время. Таким образом, основной *NP*-полной задачей, стала считаться задача проверки выполнимости заданной КНФ (сокращенно SAT). Суть проблемы, которая носит название ***выполнимость КНФ***, заключается в том, что требуется построить универсальный алгоритм, с помощью которого для любой КНФ можно было бы определить, выполнима данная формула или нет (имеется ли хотя бы один набор значений переменных, при котором формула является истинной).

Поиск быстрого универсального алгоритма решения этой задачи сопровождался заодно поиском формулировок этой задачи не только на языке самой математической логики, но на языках многих других областей дискретной математики. Оказалось, что многие практически значимые задачи теории графов, алгебры множеств, теории автоматов, теории автоматического управления и т.д. могут быть с помощью определенных преобразований сведены к задаче выполнимости.

Теперь, чтобы доказать, что некоторая задача *T* относится к классу *NP*-полных задач, необходимо: 1) доказать ее принадлежность классу *NP*; 2) выбрать известную *NP*-полную задачу и доказать, что эта задача может быть за полиномиальное время преобразована в *T*. В работе С.А. Кука было также доказано, что к классу *NP*-полных задач помимо SAT относятся еще две задачи класса *NP*. Р.М. Карп в своей статье дополнил этот список еще двадцатью широко известными задачами.

Аналогичные результаты независимо и немного позднее были получены в

СССР Л.А. Левиным [*Левин,* 1973], который ввел класс универсальных переборных задач. Этот класс аналогичен классу *NP*-полных задач и, по мнению С.А.Кука [*Cook,* 1971], является более сильным понятием. Однако интерпретация Л.А.Левина осталась незамеченной.

В последующее десятилетие усилиями многочисленных исследователей класс *NP*-полных задач расширился до нескольких тысяч [*Гэри,* 1982], а к известным классам сложности добавились новые классы [*Стокмейер,* 1989]. Было установлено, что значительная часть прикладных систем, в которых применяются методы искусственного интеллекта, содержит задачи, относящиеся к классу *NP*-полных. В связи с этим были обозначены следующие основные пути преодоления "проклятия размерности":

1) поиск приближенных методов решения *NP*-полных задач (эвристики, правдоподобный вывод, интервальные оценки, теория нечетких множеств и нечеткого вывода);

2) поиск подклассов *NP*-полных задач, для которых решение возможно с помощью алгоритмов полиномиальной сложности (представление сложных структур в виде иерархических, использование в системах логического вывода хорновских дизъюнктов и т.д.);

3) использование нейросетевых систем.

Однако основная проблема теории сложности вычислений – возможно ли решение всех задач класса *NP* с помощью алгоритма полиномиальной сложности – остается нерешенной. В частности, известно немало точных алгоритмов решения задачи SAT. Но, в общем случае, для решения этой задачи требуется огромное число операций, требующих таких вычислительных ресурсов (времени или памяти), которые не может обеспечить даже современная вычислительная техника. Количественная оценка требуемых для выполнения алгоритма ресурсов называется ***вычислительной сложностью алгоритма*** решения задачи.

В настоящее время открыто несколько классов вычислительной сложности [*Гэри*, 1982], которая оценивается как функция от объема входных условий задачи. Для задачи "выполнимость КНФ" такой объем есть общее число литералов в заданной КНФ (пусть это будет число *N*). Один из простых с точки зрения реализации классов называется ***полиномиальным***, его вычислительная сложность определяется как полином от *N* с фиксированными коэффициентами

и степенями. В этом случае при возрастании N число необходимых для выполнения алгоритма операций возрастает, но интенсивность этого возрастания считается вполне приемлемой даже в том случае, когда максимальная степень полинома равна 10. Вычислительная сложность многих задач оценивается полиномом первой (линейная вычислительная сложность) или второй степени.

Более трудным считается класс, в котором функция вычислительной сложности выражена экспонентой. В этом случае вычислительная сложность алгоритма называется ***экспоненциальной,*** и вполне возможна ситуация, когда для алгоритма такой сложности даже при $N \approx 50$ время решения на современных компьютерах будет исчисляться неделями или месяцами. Задачи с экспоненциальной оценкой вычислительной сложности называются ***NP-полными***, и "выполнимость КНФ" относится к этому классу задач. Уже найдено немало классов КНФ, для которых алгоритм решения имеет полиномиальную сложность. Например, доказано [*Krom*, 1967], что если в некоторой КНФ каждый дизъюнкт содержит не более двух литералов, задача выполнимости всегда может быть решена с помощью алгоритма полиномиальной сложности. Но для общего случая полиномиального алгоритма пока что не найдено. К тому же пока не известно, существует ли вообще такой алгоритм.

Задачи, выходящие по вычислительной сложности за пределы *NP*-полных задач, входят в класс **#*P*-полных** [*Гэри*, 1982], т.е. задач перечисления. В частности, если требуется не просто установить выполнимость КНФ (найти хоть одну выполняющую подстановку), а указать количество всех ее выполняющих подстановок, то задача реализуется, в общем случае, алгоритмами со сложностью выше экспоненциальной и относится к #*P*-полным.

Несмотря на многочисленные привлекательные свойства аксиоматического подхода и его большую популярность (особенно среди теоретиков), он обладает, по крайней мере, одним существенным недостатком. Дело в том, что алгоритмы, используемые в этих системах, не обладают свойством детерминированности – в нетривиальных случаях перед каждым этапом построения доказательства необходимо сделать выбор среди различных путей, т.е. ответить на вопросы: "какие применить правила вывода и к каким

аксиомам или выведенным ранее предложениям?" или "какие пары дизъюнктов необходимо взять, чтобы применить к ним принцип резолюции?". Неправильный выбор часто ведет к тупику или к слишком длинному доказательству. Детерминированность здесь достигается за счет того, что доказательство строится по принципу дерева вывода: просматриваются подряд все ветви этого дерева до тех пор, пока не получится положительный результат (в теории алгоритмов такой метод называется поиском с возвращением или методом ветвей и границ [*Рейнгольд,* 1980]). Если же формула невыводима (теорема недоказуема), а мы этого заранее не знаем, то для получения результата приходится просматривать все ветви дерева вывода, что достигается только алгоритмами экспоненциальной сложности. Как следствие, возрастают сложность программного обеспечения, предназначенного для логического анализа систем, и затраты вычислительных ресурсов.

1.4. Отношения в математике и информационных системах

Ранее мы рассмотрели язык предикатов – универсальный язык представления знаний в системах искусственного интеллекта, а также привели обзор некоторых систем логического вывода и указали на их взаимосвязь с соответствующими алгебраическими системами, относящимися к классу булевых алгебр. Было установлено, что при реализации процедур логического вывода возможны две альтернативы: 1) организовывать вывод в рамках некоторого исчисления путем построения дерева вывода; 2) алгебраически устанавливать выводимость логических формул или обосновывать корректность вывода.

В рамках современной логики вторая альтернатива, как правило, серьезно не рассматривается, поскольку считается, что она порождает более трудоемкие алгоритмы. Однако теория формальных систем, пришедшая на смену алгебраическому подходу, не предложила принцип решения проблемы экспоненциальной катастрофы, и вычислительная сложность процедур вывода осталась по-прежнему высокой.

С появлением вычислительной техники вновь встал вопрос о том, какой подход лучше применять при компьютерной реализации задач логического анализа: алгебраический или основанный на теории формальных систем. Легко понять, что вычислительная техника, построенная на аппарате булевых алгебр,

более приспособлена для алгебраических методов. Этим объясняется, например, популярность реляционной алгебры в управлении данными.

В отличие от алгебраических операций, символьные преобразования, лежащие в основе ТФС, плохо приспособлены для их распараллеливания. Учитывая, что интерпретациями большинства систем логического вывода являются те или иные булевы алгебры, естественно предположить, что именно на основе алгебраического подхода целесообразно (по возможности) строить программное обеспечение, предназначенное для решения логических задач.

Часто в прикладных задачах, где исследуется некоторые процессы, необходимо не просто ответить на вопрос об их осуществимости или неосуществимости, но и оценить параметры, при которых они осуществимы. Средствами ТФС задачи оценки параметров трудно реализуемы. Как уже упоминалось, большинство типичных для ТФС задач сводятся к выполнимости КНФ, но не к оценке ее выполняющих подстановок. В терминах же алгебраических систем не всегда удается даже описать требуемые методы логического анализа.

По мнению авторов, для развития алгебраических методов логического анализа необходимо найти математическую структуру, которая бы позволила представлять основные виды данных и знаний подобно тому, как они представляются на языке предикатов в рамках ТФС. "Алгебраический аналог" предиката – многоместное отношение. Как показано далее, многие системы знаний можно представить математически не только посредством определенного искусственного языка, но и в виде совокупности отношений, с которыми выполняются специальные операции, сопоставления и преобразования. Некоторые из этих преобразований можно непосредственно соотнести с отдельными видами логического анализа рассуждений. В истории математики проблема связи логики и теории отношений возникала неоднократно. Так, немецкий математик И. Юнг в своей работе, опубликованной в 1638 году, показал, что классическая силлогистика не в состоянии охватить некоторые методы логического анализа, используемые в научных рассуждениях. Юнг пытался построить теорию несиллогистических выводов, которую в настоящее время можно рассматривать как часть теории отношений [*Стяжкин,* 1967]. Английский математик и логик А. де Морган (1806–1871) в ряде своих работ рассматривал связь логики и теории отношений

и по праву считается родоначальником современной теории отношений. Он не только систематизировал и открыл некоторые законы логики (один из законов носит его имя), но и впервые определил операции с отношениями (композиция, обращение и др.) Однако общая теория отношений, пригодная для задач логического анализа, пока полностью не разработана. В частности, как было показано в п. 1.3, операции пересечения и объединения не определены для многоместных отношений, заданных в различных схемах, то есть недостаточно исследована связь алгебры множеств с теорией многоместных отношений.

Под теорией отношений в современной математике и информационных системах подразумевается либо теория бинарных отношений, либо теория многоместных отношений, в основе которой лежит реляционная алгебра.

В любом случае используется классическое определение отношения через декартово произведение множеств.

Заметим, что в системах искусственного интеллекта активно применяются методы моделирования естественных рассуждений на основе теории бинарных отношений. Как показано в последующих главах, создание общей теории многоместных отношений позволяет значительно расширить область применения алгебраического подхода при решении задач логического анализа, давая новые возможности логического вывода и не ограничиваясь только им.

1.4.1. Понятие "отношение" и декартово произведение множеств

Объектами алгебры множеств могут быть не только видимые или воображаемые предметы, но и разнообразные математические структуры: числа, точки, линии, интервалы, аналитические функции и т.д. Часто встречаются последовательности из двух, трех и т.д. элементов. Такие последовательности в математике называют *n*-местными последовательностями, *n*-ками, кортежами. Будем использовать термин ***элементарный кортеж***. Количество элементов элементарного кортежа называется ***размерностью*** этого кортежа. Понятно, что элементарные кортежи одной размерности могут отличаться типами элементов.

Множество однотипных элементарных кортежей одной и той же размерности *n* называют ***многоместным*** (или *n*-местным) ***отношением***. Широко используемая разновидность многоместных отношений – таблицы, например, таблицы, в которых содержатся сведения о персонале какой-либо

фирмы, или таблицы, отображающие дискретные (дискретизированные) функции либо отношения типа "больше", "предшествует", "является потомком" и т.д. Например, отношение "меньше" для множества $\{1, 2, 3\}$ целых чисел можно представить как множество пар чисел $\{(1, 2), (1, 3), (2, 3)\}$ или как табл. 1.11:

Таблица 1.11

1	2
1	3
2	3

В виде многоместных отношений можно представить многие предложения естественного языка. Например, предложение "Петров взял в библиотеке книгу Шолохова "Тихий Дон"" математически можно выразить как элемент (кортеж) отношения "Взятые в библиотеке книги" (ВК): *ВК* (*Фамилия читателя*, *Автор книги*, *Название книги*).

Способ представления текстов отношениями позволяет применять логический анализ с помощью средств и методов математической логики. При этом используется специальная терминология: имя отношения (в нашем примере ВК) и его структура (схема, т.е. число и состав атрибутов этого отношения) в математической логике называется ***предикатом***. Более точно, предикатом является не само отношение, а отображение некоторого универсального отношения на множество $\{\mathbf{T}, \mathbf{F}\}$ логических констант **T** (истинно) и **F** (ложно). Например, если универсум содержит пары (a, c), (a, d), (b, c) и (b, d), а отношение P – только две пары (a, c) и (b, c), то предикату $P(x, y)$ соответствует табл. 1.12.

Таблица 1.12

Универсум		**Значения истинности**
X	***Y***	
a	*C*	**T**
a	*D*	**F**
b	*C*	**T**
b	*D*	**F**

Для моделирования и анализа многоместных отношений часто используется структура, которая называется "декартово произведение

множеств" (прямое произведение множеств). Дадим ее определение.

Пусть задана некоторая совокупность из n одинаковых или различных множеств $X, Y, ..., Z$. Тогда ***декартово произведение*** этих множеств есть совокупность всех возможных n-местных элементарных кортежей, у которых на первом месте стоит элемент множества X, на втором – элемент множества Y, ..., а на последнем – элемент множества Z. Декартово произведение множеств $X, Y, ..., Z$ обозначается $X \times Y \times ... \times Z$.

Декартово произведение n одинаковых множеств X записывается как X^n. Для совокупности множеств, именованных одинаковыми символами с разными индексами (например, $S_1, S_2, ..., S_n$), используется символ многократного произведения $\prod$, в этом случае декартово произведение $S_1 \times S_2 \times ... \times S_n$ можно записать как $\prod_{i=1}^{n} S_i$.

Рассмотрим несколько примеров декартовых произведений. Для двух множеств $X = \{a, b\}$, $Y = \{b, c\}$ декартово произведение имеет вид:

$X \times Y = \{(a, b), (a, c), (b, b), (b, c)\}$.

Здесь множество $X \times Y$ содержит ***пары*** элементов, у которых (в отличие от множеств) порядок строго определен (т.е. их нельзя менять местами). Чтобы отличить (упорядоченные) пары от множеств, их заключают не в фигурные, а в простые круглые скобки.

Исходные множества могут быть заданы непрерывными интервалами, в этом случае декартовым произведением является область, ограниченная соответствующим прямоугольником на плоскости. Например, на рис. 1.6 затемненный прямоугольник вместе со своими границами изображает декартово произведение отрезков AB и CD, находящихся на разных координатных осях. Отрезки можно представить как бесконечную совокупность точек. Каждая из точек имеет координату – действительное число, равное расстоянию от начала координат до этой точки. Тогда элементы декартова произведения – всевозможные пары чисел, которые обозначают координаты точек, расположенных на плоскости в пределах этого прямоугольника. Ясно, что число таких пар бесконечно.

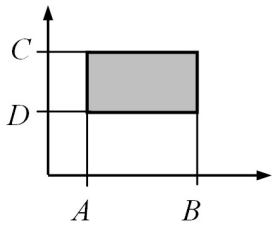

Рис. 1.6. Декартово произведение отрезков

Возможны более сложные структуры, формируемые с помощью декартова произведения (рис. 1.7 и 1.8). На рис. 1.7 структурой из 4-х затемненных прямоугольников отображается декартово произведение множеств интервалов $\{AB, CD\}\times\{EF, GH\}$. На рис. 1.8 совокупность горизонтальных отрезков на координатной плоскости, обозначенных жирными линиями, изображает декартово произведение множества $\{AB, CD\}$ отрезков одной координатной оси на множество $\{F, G, H\}$ точек другой оси. Если в качестве элементов декартовых произведений использовать только точки (например, $\{A, B, C\}\times\{G, H\}$), то на плоскости само декартово произведение будет отображаться как множество точек. Аналогичные структуры можно строить не только на плоскости, но и в трехмерном, четырехмерном и т.д. пространствах.

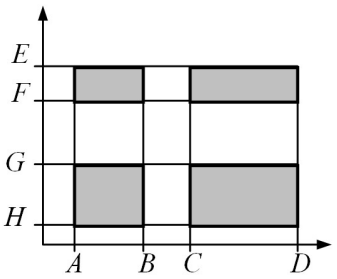

Рис. 1.7. Декартово произведение интервалов

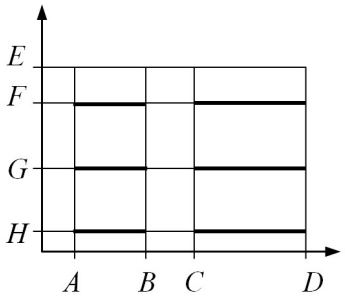

Рис. 1.8. Декартово произведение интервалов и точек

Последние два примера показывают, что декартово произведение можно определить не только в сугубо дискретных системах, но и в системах, содержащих непрерывные точечные множества.

Если декартово произведение формируется из N конечных множеств, то

оно содержит конечное число кортежей, и его можно представить в виде таблицы, содержащей N столбцов; в каждой строке этой таблицы будет элементарный кортеж данного декартова произведения. В математике с декартовыми произведениями тесно связано определение понятия "отношение":

Определение 1.5. ***N-местным отношением*** называется подмножество элементарных кортежей некоторого декартова произведения из N множеств.

Например, отношение "меньше" на множестве чисел {1, 2, 3} (табл. 1.11) с точки зрения этого определения является подмножеством элементарных кортежей, выбранных из декартова произведения $\{1, 2, 3\}\times\{1, 2, 3\}$.

Рассмотрим количественные соотношения на декартовых произведениях. Пусть X – некоторое множество, тогда $|X|$ обозначает количество элементов или ***мощность*** этого множества. Количество всех элементов (т.е. элементарных кортежей) декартова произведения будет в точности равно произведению мощностей всех используемых в этом произведении множеств:

$$|X\times Y\times...\times Z| = |X|\cdot|Y|\cdot...\cdot|Z|. \qquad (1.7)$$

Например, если заданы множества $P = \{a, b, c\}$; $Q = \{a, d, f\}$ и $R = \{a, b, c, f\}$, то их декартово произведение $P\times Q\times R$ будет содержать $3\times3\times4 = 36$ элементарных кортежей.

Приведем основные свойства декартовых произведений [*Бурбаки*, 1965]. Подробные сведения об этих свойствах и об их практическом применении содержатся также в [*Мелихов*, 1971].

1. ***Пересечение*** декартовых произведений.

Если даны декартовы произведения $X_1\times X_2\times...\times X_n$ и $Y_1\times Y_2\times...\times Y_n$, то их пересечение вычисляется как:

$$(X_1\times X_2\times...\times X_n)\cap(Y_1\times Y_2\times...\times Y_n)=(X_1\cap Y_1)\times(X_2\cap Y_2)\times...\times(X_n\cap Y_n).$$

Таким образом, надо сначала вычислить пересечение пар множеств, находящихся на соответствующих "местах" исходных декартовых произведений, а затем сформировать декартово произведение полученных множеств. Например, пусть заданы следующие декартовы произведения:

$\{a, c, d\}\times\{b, c\}\times\{b, c, f\}$ и

$\{a, b, c, d\}\times\{a, c, d\}\times\{b, c, d\}$.

В соответствии с приведенным правилом вычисления пересечения декартовых произведений получим:

$(\{a, c, d\}\times\{b, c\}\times\{b, c, f\}) \cap (\{a, b, c, d\}\times\{a, c, d\}\times\{b, c, d\}) =$

$= (\{a, c, d\} \cap \{a, b, c, d\})\times(\{b, c\} \cap \{a, c, d\})\times(\{b, c, f\} \cap \{b, c, d\}) =$

$= \{a, c, d\}\times\{c\}\times\{b, c\}.$

Нетрудно убедиться, что результат пересечения содержит 6 элементарных кортежей:

(a, c, b);

(a, c, c);

(c, c, b);

(c, c, c);

(d, c, b);

(d, c, c).

Тот же результат можно получить, если развернуть первое декартово произведение (в нем содержится 18 элементарных кортежей), затем второе (в нем 36 элементарных кортежей), а потом выбрать из этих двух совокупностей одинаковые элементарные кортежи. Но вычислять результат таким путем намного сложнее.

2. ***Пустое декартово произведение***

Если при вычислении пересечения декартовых произведений окажется, что в полученном декартовом произведении хотя бы одно множество – пустое, то результатом будет ***пустое*** декартово произведение. Например,

$\{a, c, d\}\times\{b, c\}\times\{b, c, f\} \cap \{a, b, d\}\times\{a, d\}\times\{b, c, d\} = \{a, d\}\times\varnothing\times\{b, c\} = \varnothing.$

В некоторых структурах (например, когда декартовы произведения используются в графах) это правило может не соблюдаться. Но здесь мы будем в основном рассматривать логические структуры, для которых правило пустого декартова произведения – аксиома.

3. ***Проверка включения декартовых произведений***

Если при сравнении двух декартовых произведений

$A = (X_1\times X_2\times...\times X_n)$ и $B = (Y_1\times Y_2\times...\times Y_n)$

окажется, что для всех "мест" соблюдаются соотношения

$(X_1 \subseteq Y_1), (X_2 \subseteq Y_2),..., (X_n \subseteq Y_n),$

то справедливо $A \subseteq B$, т.е. все элементарные кортежи декартова произведения A обязательно принадлежат также и декартову произведению B. В противном случае отношение $A \subseteq B$ не соблюдается. Например, если заданы два

декартовых произведения

$\{a, c, d\}\times\{b, c\}\times\{b, c, f\}$ и $\{a, d\}\times\{b, c\}\times\{b, f\}$,

легко убедиться, что второе из них включено в первое, так как каждая компонента второго декартова произведения есть подмножество соответствующей компоненты первого.

4. ***Объединение*** декартовых произведений

В общем случае объединение двух декартовых произведений нельзя представить как одно декартово произведение. Однако, если вычислить декартово произведение тем же методом, что и при вычислении их пересечения, но вычислять не пересечение, а объединение соответствующих компонент, то получим декартово произведение, которое содержит все элементарные кортежи исходных декартовых произведений. Оно также может включать элементарные кортежи, не входящие ни в одно из исходных декартовых произведений. Формула этого соотношения такова:

$$\prod_{i=1}^{n} X_i \cup \prod_{i=1}^{n} Y_i \subseteq \prod_{i=1}^{n} (X_i \cup Y_i). \qquad (1.8)$$

Если нужно вычислить число элементов в объединении двух декартовых произведений, то расчет производится по следующей формуле. Пусть даны декартовы произведения A и B. Тогда

$|A \cup B| = |A| + |B| - |A \cap B|$.

Пример. Пусть $A = \{a, c, d\}\times\{b, c\}\times\{b, c, f\}$ и $B = \{a, b, d\}\times\{b, c\}\times\{b, c, d\}$.

Тогда $A \cap B = \{a, d\}\times\{b, c\}\times\{b, c\}$; $|A| = 18$; $|B| = 18$; $|A \cap B| = 8$; $|A \cup B| = 18 + 18 - 8 = 28$.

Существуют случаи, когда в соотношении (1.8) знак включения можно заменить знаком равенства. Таких случаев всего два.

Первый случай. Если для двух декартовых произведений A и B соблюдается $A \subseteq B$, то справедливо $A \cup B = B$. Например,

$\{a, c, d\}\times\{b, c\}\times\{b, c, f\} \cup \{a, d\}\times\{b, c\}\times\{b, f\} = \{a, c, d\}\times\{b, c\}\times\{b, c, f\}$,

так как проверка показывает, что второе декартово произведение включено в первое.

Второй случай. Если в двух декартовых произведениях все "места", за исключением одного, содержат равные множества, то в соотношении (1.8) знак включения заменяется на знак равенства. Например,

$$\{a, d\}\times\{b, c\}\times\{b, c, d\} \cup \{a, d\}\times\{b, f\}\times\{b, c, d\} = \{a, d\}\times\{b, c, f\}\times\{b, c, d\},$$

так как в этих декартовых произведениях отличаются друг от друга только множества на втором "месте", а все остальные "места" содержат равные множества.

5. ***Разность*** декартовых произведений.

Пусть даны два декартовых произведения $\prod_{i=1}^{n} X_i$ и $\prod_{i=1}^{n} Y_i$. Тогда

$$\prod_{i=1}^{n} X_i \setminus \prod_{i=1}^{n} Y_i = ((X_1 \setminus Y_1)\times X_2\times\ldots\times X_n) \cup (X_1\times(X_2 \setminus Y_2)\times\ldots\times X_n) \cup \ldots \cup (X_1\times X_2\times\ldots\times(X_n \setminus Y_n)).$$

Структуру этой формулы легче понять, если расположить объединяемые в правой части подформулы в виде столбца:

$$\begin{vmatrix} X_1 \setminus Y_2 & \times & X_2 & \times\ldots\times & X_n \\ X_1 & \times & X_2 \setminus Y_2 & \times\ldots\times & X_n \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ X_1 & \times & X_2 & \times\ldots\times & X_n \setminus Y_n \end{vmatrix}.$$

Эти и некоторые другие свойства декартовых произведений используются в алгебре кортежей, основные понятия и методы которой рассматриваются в главе 2.

Прежде чем описывать конкретные разделы теории отношений, а также их применения в программных системах, подчеркнем, что над отношениями, сформированными в одном декартовом произведении (имеющими одну схему), также могут выполняться операции алгебры множеств. Для такой совокупности отношений подтверждаются все основные тождества булевых алгебр, если декартово произведение рассматривать как универсум, а отношения – как его подмножества. Получается полная аналогия с алгеброй множеств. Однако при переходе к отношениям с разными схемами соответствие между системой многоместных отношений и алгеброй множеств теряется: для этого случая, в частности, не определены операции пересечения и объединения.

1.4.2. Бинарные отношения

Теория бинарных отношений имеет большую теоретическую базу и находит широкое применение в практических приложениях. Многие алгоритмы в современных информационных технологиях построены на основе этой теории [*Кристофидес*, 1978; *Липский*, 1988; *Рейнгольд*, 1980; *Kuznetsov,* 2002]. В частности, исследование свойств бинарных отношений (симметричности, транзитивности, рефлексивности) лежит в основе методов приобретения знаний интеллектуальными системами [*Осипов*, 2009]. В некоторых книгах, например, в [*Непейвода*, 1997; *Новиков*, 2002], под теорией отношений подразумевают теорию бинарных отношений.

Бинарное отношение – это отношение, заданное как некоторое подмножество декартова произведения двух множеств, или подмножество пар (a_i, b_j) декартова произведения $A \times B$, где $a_i \in A$; $b_j \in B$.

Широко известны частные случаи бинарных отношений – графы и частично упорядоченные множества. Бинарное отношение можно задать списком пар или (в случае, когда $A = B$) матрицей смежности порядка $n \times n$, где n – мощность множества A.

Нередко бинарные отношения типа $R \subseteq A \times B$ представляют как ***соответствие*** между элементами множества A и элементами множества B. Тогда множество A называется ***областью отправления***, а B – ***областью прибытия*** соответствия. Соответствие, в котором каждому элементу области отправления сопоставляется не более одного элемента в области прибытия, называется ***функциональным соответствием*** или ***отображением***.

Для совокупности бинарных отношений, заданных на одном декартовом произведении, выполняются все соотношения и операции алгебры множеств (пересечение, объединение, дополнение, включение, равенство). Помимо операций алгебры множеств, в них определены еще три операции: ***соединение, композиция*** и ***обращение***. Первые две операции возможны и для случаев, когда эти отношения определены на разных декартовых произведениях. Унарная операция обращения – это операция, при выполнении которой элементы всех пар отношения меняются местами. Более подробно перечисленные операции будут рассмотрены ниже (см. главу 2) как примеры использования алгебры кортежей при моделировании бинарных отношений и соответствий.

1.4.3. Отношения в реляционной алгебре

Одна из распространенных математических систем, предназначенных для моделирования и анализа многоместных отношений, – это разработанное Э.Ф. Коддом реляционное исчисление, которое лежит в основе современных систем управления базами данных (СУБД) [*Codd*, 1970; *Codd*, 1972]. На основе реляционного исчисления была создана реляционная алгебра, позволяющая выполнять следующие операции с отношениями и внутри отношений: 1) объединение; 2) пересечение; 3) вычитание; 4) декартово произведение; 5) выборка; 6) проекция; 7) соединение; 8) деление. Первые четыре операции определяются как теоретико-множественные операции, остальные – как реляционные. Связь этих операций с логическим анализом систем легко прослеживается, в частности, операция "проекция" соответствует применению квантора существования, а операция "деление" – квантора всеобщности.

В основе реляционного исчисления лежат некоторые аналитические средства исчисления предикатов. Область интерпретации этого исчисления – множество кортежей всех включенных в конкретную СУБД отношений. Чтобы обеспечить в БД работу не только с кортежами, но и с атрибутами, в реляционное исчисление вводится понятие индексированной переменной. Для этого разработана теория нормализации отношений [*Codd*, 1971; *Fagin*, 1971], существенно облегчающая поиск информации в базах данных.

Одним из основных языков для СУБД является SQL, с его помощью осуществляется логический вывод новых фактов на основе существующих [*Дейт*, 1999; *Ульман*, 2000]. Однако выразительные и аналитические средства СУБД существенно беднее средств и методов математической логики хотя бы потому, что в них не используется дополнение отношения, а методы логического вывода существенно упрощены. Для расширения этих возможностей были разработаны дедуктивные базы данных [*Чери*, 1992], которые отличаются от реляционных тем, что в них определены рекурсивные процедуры. Но такое расширение не позволяет применять значительную часть аналитических средств математической логики. Итак, современную теорию баз данных нельзя рассматривать как общую теорию многоместных отношений.

1.4.4. Отношения в логике и искусственном интеллекте

Во многих современных теоретических исследованиях по искусственному интеллекту [*Вагин*, 2004; *Тейз*, 1990; *Смолин*, 2004; *Представление* знаний, 1989] предлагается в основном декларативный подход к разработке приложений. В математической логике термин "отношение" употребляется редко, однако оказывается, что многие структуры логики и искусственного интеллекта представимы в виде многоместных отношений. В математической логике предикаты изоморфны отношениям [*Новиков*, 1973]. Покажем, что любую логическую формулу можно отобразить в виде многоместного отношения.

Рассмотрим произвольную формулу исчисления высказываний или предикатов. Любая формула имеет некоторое, возможно, пустое или бесконечное, множество выполняющих подстановок. Каждой выполняющей подстановке можно сопоставить определенный кортеж элементов, порядок расположения которых соответствует порядку свободных переменных данной формулы. Тогда множество всех выполняющих подстановок такого вида есть отношение, изоморфное данной формуле. Разумеется, подобное представление логических формул во многих случаях практически нереализуемо из-за чрезмерно большого или бесконечного числа выполняющих подстановок, но сейчас речь идет о теоретическом аспекте представления формул с помощью математического подхода, в основе которого лежит теория отношений.

Рассмотрим вкратце основные структуры баз знаний искусственного интеллекта. К ним относятся семантические сети, фреймы и системы продукций. Формально ***семантические сети*** эквивалентны ориентированным графам с помеченными вершинами и ребрами [*Искусственный* интеллект, 1990; *Представление* знаний, 1989]. Имена ребер соответствуют именам отношений, а вершины – элементам семантической сети. С учетом этого любую семантическую сеть можно представить как совокупность связанных по смыслу бинарных отношений. В правилах вывода на семантических сетях используются операции, которые соответствуют композиции или соединению этих отношений.

Фреймы реализуют более сложные (по сравнению с семантическими сетями) структуры. Каждый фрейм состоит из некоторого множества ***слотов***, а каждый слот содержит ***значение*** слота. Значением слота могут быть

индивидные константы, множества слотов (другие фреймы) или правила [*Искусственный* интеллект, 1990; *Смолин*, 2004; *Представление* знаний, 1989]. Если фрейм содержит только простые слоты, т.е. слоты, содержащие только индивидные константы, то он представляется обычной таблицей или многоместным отношением, в котором имя фрейма соответствует имени отношения, а имена слотов – именам атрибутов.

Если значение некоторого слота есть множество слотов, то это отношение (фрейм) имеет атрибут (слот), являющийся многоместным отношением, и этот фрейм представляет собой суперпозицию многоместных отношений. Во многих случаях фреймы можно отобразить как сеть взаимосвязанных многоместных отношений. Правила или продукции, которые во фреймах присутствуют в качестве значений некоторых слотов или используются в продукционных системах, эквивалентны логическим формулам (см. п. 1.3), представимым (по крайней мере, теоретически) как многоместные отношения.

Задачи семантического анализа текста, перевода текстов с одного языка на другой и т.д. практически невозможны без преобразования предложений в отношения. Например, предложение "Ракета "Союз-ФГ" стартовала в воскресенье в 11.01 мск с космодрома Байконур" можно выразить как элемент отношения

СТАРТОВАТЬ (Объект действия, время действия, место действия).

Одна из основных задач искусственного интеллекта заключается в автоматизации работы со знаниями, представленными в виде текстов [*Башмаков*, 2005; *Искусственный* интеллект, 1990]. С этой проблемой тесно переплетается задача перевода текстов с одного языка на другой [*Башмаков*, 2005]. Решение такой задачи состоит из следующих этапов: 1) лексический анализ; 2) морфологический анализ; 3) синтаксический анализ и 4) семантический анализ. Для выполнения семантического анализа необходимо перевести текст во "внутреннее представление", т.е. представить в виде семантической сети [*Башмаков*, 2005], которая, как было сказано выше, есть взаимосвязанная совокупность бинарных отношений.

Изложенное показывает, что имеются предпосылки для разработки теоретического подхода к построению и анализу интеллектуальных систем, в котором более широко применяется понятие "отношение".

Глава 2. Теоретические основы алгебры кортежей

Чтобы разобраться в бесконечном, надо
сперва различать, а затем связывать.
И.В. фон Гете, "Фауст"

Из обзора, приведенного в главе 1, следует, что общая теория отношений пока полностью не разработана. В теории бинарных отношений и реляционной алгебре многоместное отношение трактуется как подмножество элементарных кортежей, входящих в некоторое декартово произведение. Это классическое определение позволяет оперировать с отношениями как с обычными множествами при условии, что отношения сформированы в одном и том же декартовом произведении *D*. Но соответствие алгебры многоместных отношений алгебре множеств теряется при переходе к совокупностям отношений, заданных в различных декартовых произведениях, поскольку для них не удается определить операции объединения и пересечения. Также в алгебре многоместных отношений существуют операции соединения и композиции, для которых нет эквивалента в алгебре множеств.

Рассматриваемая в этой и последующих главах алгебра кортежей реализует общую теорию многоместных отношений, пригодную для задач логического анализа. В качестве основной структуры АК выступает не элементарный кортеж, а декартово произведение множеств, которому в исчислении предикатов соответствует конъюнкция предикатов или формул, попарно не содержащих совпадающих переменных. По сути, алгебра кортежей представляет собой алгебру декартовых произведений и их дополнений: каждый кортеж, в зависимости от его принадлежности к определенному классу АК-объектов, интерпретируется либо как декартово произведение множеств, либо как дополнение некоторого декартова произведения множеств.

2.1 Основные термины и структуры

Алгебра кортежей была разработана для моделирования и анализа многоместных отношений [*Кулик*, 1993; 2008]. В отличие от реляционной алгебры, применяющейся для формализации БД, АК позволяет использовать все средства логического моделирования и анализа систем, которые входят в

математическую логику (логический вывод, проверка правильности следствия, анализ гипотез, абдуктивный вывод и т.д.). Свойства АК основаны на свойствах декартова (прямого) произведения множеств и соответствуют законам математической логики. В АК при получении промежуточных результатов нет необходимости представлять структуры как множества элементарных кортежей. Все операции выполняются с компонентами атрибутов (множествами) или кортежами компонент.

В математике ***кортежем*** называется последовательность некоторых объектов.

Определение 2.1. ***Алгебра кортежей*** – это алгебраическая система, носитель которой – совокупность многоместных отношений, выраженных в специфических структурах (элементарный кортеж, *C*-кортеж, *C*-система, *D*-кортеж, *D*-система), называемых ***АК-объектами***.

Таким образом, помимо элементарного кортежа, в АК определены еще четыре структуры, с помощью которых сжато представляются множества элементарных кортежей. В АК выполняются все операции и проверяются все соотношения алгебры множеств. Для обеспечения изоморфизма между этими системами в АК операции и отношения алгебры множеств обобщены на случай, когда многоместные отношения заданы в различных декартовых произведениях.

Кроме операций, типичных для алгебры множеств, в АК добавлено 5 операций с атрибутами:

1) переименование атрибутов;

2) перестановка атрибутов и соответствующих им столбцов;

3) обращение АК-объектов;

4) добавление нового фиктивного атрибута (+*Atr*);

5) элиминация атрибута из АК-объекта (–*Atr*).

Под ***атрибутом*** здесь понимается имя некоторого свойства системы или ее части, представленное множеством непосредственно заданных или вычисляемых значений (***доменом***). Возможны случаи, когда одному и тому же домену соответствует несколько атрибутов. Множество таких атрибутов будем называть ***сортом***. В математической логике доменам атрибутов соответствуют области определения переменных, в информационных системах – шкалы признаков.

Если задано некое отношение, то его определяющий признак – совокупность присущих ему атрибутов. Последовательность этих атрибутов в АК называется ***схемой отношения***. Универсум отношения есть декартово произведение доменов атрибутов, представленных в его схеме.

АК-объекты, сформированные в одной и той же схеме отношения, называются ***однотипными***. Универсум для совокупности однотипных отношений называется ***частным универсумом***.

Обозначения АК-объектов содержат собственно имя, в некоторых случаях дополняемое последовательностью имен атрибутов (заключается в прямые скобки), определяющих схему отношения, в которой задан этот АК-объект. Например, $R[XYZ]$ означает, что АК-объект R задан в схеме отношения $[XYZ]$.

Размерность АК-объекта определяется как размерность его матричного отображения (т.е. число строк и столбцов) независимо от мощности компонент, из которых он сформирован.

Эквивалентными являются АК-объекты, у которых совпадают множества содержащихся в них элементарных кортежей. Другими словами, два АК-объекта P и Q эквивалентны, если выполняется как $P \subseteq Q$, так и $Q \subseteq P$.

Определение 2.2. ***Элементарный кортеж*** – это последовательность элементов, каждый из которых есть элемент домена соответствующего атрибута из схемы отношения.

Например, если задан элементарный кортеж $T[XYZ] = (a, b, c)$, то подразумевается, что $a \in X$, $b \in Y$ и $c \in Z$. В АК элементарные кортежи принадлежат множествам (отношениям), выраженных другими типами АК-объектов.

Остальные типы АК-объектов формируются из множеств, представляющих собой подмножества доменов соответствующих атрибутов и называемых ***компонентами***. Среди компонент особую роль играют две ***фиктивные*** компоненты:

1) ***полная компонента*** – "∗" – равна домену соответствующего (по месту ее расположения в кортеже) атрибута, используется в C-кортежах и C-системах;

2) ***пустое множество*** – "∅" – используется в D-кортежах и D-системах.

Далее будет показано, что за счет использования фиктивных компонент (фиктивных атрибутов) в АК имеется возможность приводить отношения с различными схемами к одному типу, что позволяет применять к ним теоретико-

множественные операции. Существенное отличие предлагаемого метода ввода фиктивных атрибутов от известных состоит в том, что новые данные вводятся в многоместные отношения не поэлементно, а как множества. Это, как показано в главе 3, существенно уменьшает трудоемкость вычислительных алгоритмов и объем памяти для представления структур.

Определение 2.3. ***C-кортежем*** называется заданный в определенной схеме отношения кортеж компонент.

Префиксы "*C*" и "*D*" в названиях структурных типов АК-объектов выбраны не случайно. Это сокращенные обозначения логических понятий "конъюнкция" (conjunction) и "дизъюнкция" (disjunction), которые отображают структуры, соединенные логическими связками "И" (конъюнкция) и "ИЛИ" (дизъюнкция). При сопоставлении со структурами математической логики можно обосновать (см. п. 2.4), что *C*-кортежи соответствуют конъюнкции некоторых структур, а *D*-кортежи – дизъюнкции.

C-кортеж эквивалентен некоторому множеству элементарных кортежей – это множество можно перечислить, если вычислить декартово произведение компонент *C*-кортежа. Для изображения *C*-кортежей используются прямые скобки. Например, $R[XYZ] = [A\ B\ C]$ означает, что $A \subseteq X$, $B \subseteq Y$, $C \subseteq Z$ и $R[XYZ] = A \times B \times C$.

Чтобы отличить *C*-кортеж от обычных (элементарных) кортежей, содержащих не множества, а элементы, будем заключать его в прямые скобки. Пусть с помощью декартовых произведений заданы отношения

$R[XYZ] = \{b, d\} \times \{f, h\} \times \{a, b\}$ и $Q[YZ] = \{f, g\} \times \{a, c\}$.

Эти отношения можно записать как *C*-кортежи:

$R[XYZ] = [\{b, d\}\ \{f, h\}\ \{a, b\}]$,

$Q[YZ] = [\{f, g\}\ \{a, c\}]$.

Поскольку каждый *C*-кортеж есть декартово произведение множеств, он содержит некоторое множество элементарных кортежей. Например,

$$Q[YZ] = [\{f, g\}\ \{a, c\}] = \{f, g\} \times \{a, c\} = \begin{vmatrix} (f,a) \\ (f,c) \\ (g,a) \\ (g,c) \end{vmatrix}.$$

C-кортеж, у которого имеется хотя бы одна пустая компонента, пуст. *C*-кортеж со всеми полными компонентами эквивалентен некоторому частному

универсуму. В АК, если речь идет о моделях исчисления высказываний или предикатов, эти утверждения приняты в качестве аксиомы, для которой имеется соответствующая интерпретация, основанная на свойствах декартовых произведений.

Определение 2.4. ***C-системой*** называется множество однотипных *C*-кортежей, которые записываются в виде матрицы, ограниченной прямыми скобками. Строки этой матрицы содержат *C*-кортежи.

C-система эквивалентна множеству элементарных кортежей. Это множество равно объединению множеств элементарных кортежей, принадлежащих соответствующим *C*-кортежам. Например, *C*-систему $Q[XYZ] = \begin{bmatrix} A_1 & B_1 & C_1 \\ A_2 & B_2 & C_2 \end{bmatrix}$ можно представить как множество элементарных кортежей, вычисляемое по формуле

$$Q[XYZ] = (A_1 \times B_1 \times C_1) \cup (A_2 \times B_2 \times C_2).$$

Здесь и далее в этом подразделе будем рассматривать примеры АК-объектов, заданные в пространстве $\boldsymbol{S} = X \times Y \times Z$, где $X = \{a, b, c, d\}$, $Y = \{f, g, h\}$, $Z = \{a, b, c\}$. Например, для $\boldsymbol{S}$ можно определить некоторое отношение $R[XYZ]$ как *C*-систему

$$R[XYZ] = \begin{bmatrix} \{a,d\} & * & \{b,c\} \\ \{b,d\} & \{f,h\} & \{a,c\} \\ \{b,c\} & \{g\} & \{b\} \end{bmatrix}.$$

Компонента "$*$" в первом *C*-кортеже соответствует атрибуту Y и в силу этого равна множеству $\{f, g, h\}$. Полученную структуру можно представить как множество элементарных кортежей, если последовательно "разворачивать" каждый из *C*-кортежей, содержащихся в *C*-системе, во множество элементарных кортежей. При этом надо учитывать, что одинаковые элементарные кортежи могут содержаться в разных *C*-кортежах, подобное дублирование надо исключать. В *C*-системе Q такими "неудобными" являются первый и второй *C*-кортежи, так как результатом их пересечения является непустой *C*-кортеж $[\{d\}\ \{f, h\}\ \{c\}]$. Это означает, что элементарные кортежи (d, f, c) и (d, h, c) содержатся в каждом из этих *C*-кортежей.

Теорема 2.1. Алгебра кортежей для однотипных АК-объектов изоморфна алгебре множеств.

Для обоснования этого положения в данном разделе рассмотрены особенности реализации операций (пересечение, объединение, дополнение) и отношений (включения и равенства) сначала для *C*-объектов (*C*-кортежей, *C*-систем), а затем для *D*-объектов (*D*-кортежей, *D*-систем). После чего, в подразделе 2.2 показаны способы преобразования *C*-объектов в *D*-объекты и наоборот.

Операции с однотипными АК-объектами, а также проверка соотношений включения или равенства осуществляется на основании теорем 2.2-2.8 (для *C*-объектов) и 2.9-2.15 (для *D*-объектов). Многие теоремы приводятся здесь без доказательств, так как их формулировка в терминах АК соответствует известным свойствам декартовых произведений.

Пусть заданы однотипные *C*-кортежи

$P = [P_1\ P_2\ \dots\ P_n]$ и $Q = [Q_1\ Q_2\ \dots\ Q_n]$.

Теорема 2.2. $P \cap Q = [P_1 \cap Q_1\ \ P_2 \cap Q_2\ \ \dots\ \ P_n \cap Q_n]$.

Примеры: $[\{b, d\}\ \{f, h\}\ \{a, b\}] \cap [\ *\ \{f, g\}\ \{a, c\}] = [\{b, d\}\ \{f\}\ \{a\}]$;

$$[\{b, d\}\ \{f, h\}\ \{a, b\}] \cap [\ *\ \{g\}\ \{a, c\}] = [\{b, d\}\ \varnothing\ \{a\}] = \varnothing.$$

Теорема 2.3. $P \subseteq Q$, если и только если $P_i \subseteq Q_i$ для всех $i = 1, 2, \dots, n$.

Теорема 2.4. $P \cup Q \subseteq [P_1 \cup Q_1\ \ P_2 \cup Q_2\ \dots\ P_n \cup Q_n]$, причем равенство возможно лишь в двух случаях:

(i) $P \subseteq Q$ или $Q \subseteq P$;

(ii) $P_i = Q_i$ для всех соответствующих пар компонент, за исключением одной пары.

Заметим, что в алгебре кортежей в соответствии с определением 2.4 во всех случаях справедливо равенство $P \cup Q = \begin{bmatrix} P_1 & P_2 & \dots & P_n \\ Q_1 & Q_2 & \dots & Q_n \end{bmatrix}$.

Теорема 2.5. Пересечение двух однотипных *C*-систем равно *C*-системе, содержащей все непустые пересечения каждого *C*-кортежа первой *C*-системы с каждым *C*-кортежем второй *C*-системы.

Пример. Пусть в пространстве ***S*** заданы *C*-системы:

$$R_1[XYZ] = \begin{bmatrix} \{a,b,d\} & \{f,h\} & \{b\} \\ \{b,c\} & * & \{a,c\} \end{bmatrix},\ R_2[XYZ] = \begin{bmatrix} \{a,d\} & * & \{b,c\} \\ \{b,d\} & \{f,h\} & \{a,c\} \\ \{b,c\} & \{g\} & \{b\} \end{bmatrix}.$$

Нужно вычислить их пересечение. Сначала вычисляем пересечение всех

пар *C*-кортежей, содержащихся в разных *C*-системах:

$[\{a, b, d\}\ \{f, h\}\ \{b\}] \cap [\{a, d\} * \{b, c\}] = [\{a, d\}\ \{f, h\}\ \{b\}]$;

$[\{a, b, d\}\ \{f, h\}\ \{b\}] \cap [\{b, d\}\ \{f, h\}\ \{a, c\}] = \varnothing$;

$[\{a, b, d\}\ \{f, h\}\ \{b\}] \cap [\{b, c\}\ \{g\}\ \{b\}] = \varnothing$;

$[\{b, c\} * \{a, c\}] \cap [\{a, d\} * \{b, c\}] = \varnothing$;

$[\{b, c\} * \{a, c\}] \cap [\{b, d\}\ \{f, h\}\ \{a, c\}] = [\{b\}\ \{f, h\}\ \{a, c\}]$;

$[\{b, c\} * \{a, c\}] \cap [\{b, c\}\ \{g\}\ \{b\}] = \varnothing$.

Затем из непустых *C*-кортежей формируем *C*-систему:

$$R_1 \cap R_2 = \begin{bmatrix} \{a,d\} & \{f,h\} & \{b\} \\ \{b\} & \{f,h\} & \{a,c\} \end{bmatrix}.$$

Даже этот сравнительно несложный пример показывает некоторые возможности уменьшения трудоемкости алгоритмов за счет использования АК: тот же результат можно получить, если предварительно перевести исходные *C*-системы в множества элементарных кортежей. Но при этом трудоемкость вычислений значительно увеличится, так как *C*-система *P* содержит 24 элементарных кортежа, *C*-система *Q* – 20, а *C*-система $P \cap Q$ – 8.

Теорема 2.6. Объединение двух однотипных *C*-систем равно *C*-системе, содержащей все *C*-кортежи операндов.

Например,

$$\begin{bmatrix} \{a,b,d\} & \{f,h\} & \{b\} \\ \{b,c\} & * & \{a,c\} \end{bmatrix} \cup \begin{bmatrix} \{a,d\} & \{f,h\} & \{b\} \\ \{b\} & \{f,h\} & \{a,c\} \end{bmatrix} = \begin{bmatrix} \{a,b,d\} & \{f,h\} & \{b\} \\ \{b,c\} & * & \{a,c\} \\ \{a,d\} & \{f,h\} & \{b\} \\ \{b\} & \{f.h\} & \{a,c\} \end{bmatrix}.$$

В некоторых случаях после вычисления объединения *C*-систем можно сократить общее число кортежей в полученной *C*-системе, если использовать условия (i) или (ii) теоремы 2.4. Можно убедиться, что в последней *C*-системе третий *C*-кортеж включен в первый, а четвертый – во второй. Значит, результат будет равен первой *C*-системе из левой части равенства.

Для операций пересечения и объединения отношений можно было бы ограничиться тремя определенными ранее типами АК-объектов. Но для использования операции дополнения этого недостаточно. Поэтому требуется определить еще три структуры.

Определение 2.5. ***Диагональная C-система*** – *C*-система размерности $n \times n$, у которой все недиагональные компоненты равны полной компоненте.

Например, $\begin{bmatrix} \{a,c\} & * & * \\ * & \{f,g\} & * \\ * & * & \{b,c\} \end{bmatrix}$ – диагональная *C*-система.

Определение 2.6. ***D-кортеж*** – это отношение, равное диагональной *C*-системе, записанное в виде кортежа диагональных компонент и ограниченное перевернутыми квадратными скобками.

Например, $\begin{bmatrix} \{a,c\} & * & * \\ * & \{f,g\} & * \\ * & * & \{b,c\} \end{bmatrix} =]\{a, c\}\ \{f, g\}\ \{b, c\}[,$

где в правой части равенства *D*-кортеж. Отсюда ясно, что любой *D*-кортеж можно при необходимости "развернуть" в диагональную *C*-систему.

D-кортеж с хотя бы одной полной компонентой равен некоторому частному универсуму. *D*-кортеж со всеми пустыми компонентами эквивалентен пустому множеству

Чтобы вычислить дополнение *C*-кортежей и *C*-систем, необходимо найти дополнение каждой компоненты. Дополнение $\overline{P_j}$ любой компоненты P_j АК-объекта определяется как дополнение относительно домена соответствующего ей атрибута. Например, если задан *C*-кортеж $R[XYZ] = [A\ B\ C]$, то $\overline{A} = X \setminus A$, $\overline{B} = Y \setminus B$ и $\overline{C} = Z \setminus C$.

Теорема 2.7. Для произвольного *C*-кортежа $P = [P_1\ P_2\ \dots\ P_n]$ его дополнение равно *D*-кортежу $\overline{P} =]\overline{P_1}\ \overline{P_2}\ \dots\ \overline{P_n}[$.

Формулировка данной теоремы соответствует операции разности декартовых произведений, выраженной в структурах АК (см. подраздел 1.4). Рассмотрим примеры. Пусть в пространстве $\boldsymbol{S} = X \times Y \times Z$ задан *C*-кортеж $T = [\{b, d\}\ \{f, h\}\ \{a, b\}]$. Тогда

$$\overline{T} =]X \setminus \{b, d\}\ Y \setminus \{f, h\}\ Z \setminus \{a, b\}[= \begin{bmatrix} \{a,c\} & * & * \\ * & \{g\} & * \\ * & * & \{c\} \end{bmatrix}.$$

Вычислим дополнение для случая, когда в *C*-кортеже имеются фиктивные компоненты. Пусть $T_1 = [\{b, d\} * \{a, b\}]$. Тогда $\overline{T_1} =]\{a, c\} \varnothing \{c\}[$. Здесь в

D-кортеже появилась фиктивная компонента "∅". При преобразовании *D*-кортежа с фиктивными компонентами в диагональную *C*-систему *C*-кортежи, содержащие эти фиктивные компоненты, будут пустыми и поэтому могут быть удалены из *C*-системы. Например,

$$\overline{T_1} =]\{a, c\}\ \varnothing\ \{c\}[= \begin{bmatrix} \{a,c\} & * & * \\ * & \varnothing & * \\ * & * & \{c\} \end{bmatrix} = \begin{bmatrix} \{a,c\} & * & * \\ * & * & \{c\} \end{bmatrix}.$$

Оказывается, *D*-кортеж не только позволяет компактно отобразить диагональные *C*-системы, но самостоятельно используется во многих операциях АК.

Используя *D*-кортежи, можно сформировать еще одну структуру АК – *D*-систему.

Определение 2.7. ***D-система*** – структура, состоящая из множества однотипных *D*-кортежей, равная пересечению множеств элементарных кортежей, содержащихся в этих *D*-кортежах.

Изображение *D*-системы аналогично изображению *C*-системы, только вместо обычных прямых скобок используются перевернутые.

Теорема 2.8. Дополнение *C*-системы есть *D*-система той же размерности, в которой каждая компонента равна дополнению соответствующей компоненты в исходной *C*-системе.

Доказательство. Пусть дана *C*-система *P*, содержащая множество $\{P_1, P_2, \ldots, P_n\}$ *C*-кортежей. Это означает, что $P = P_1 \cup P_2 \cup \ldots \cup P_n$. При вычислении дополнения по закону де Моргана получим $\overline{P} = \overline{P_1} \cap \overline{P_2} \cap \ldots \cap \overline{P_n}$. Тогда справедливость данной теоремы следует из теоремы 2.7. *Конец доказательства.*

Например, дополнение *C*-системы $F[XYZ] = \begin{bmatrix} \{a,b,d\} & \{f,h\} & \{b\} \\ \{b,c\} & * & \{a,c\} \end{bmatrix}$,

заданной в пространстве ***S***, можно вычислить как *D*-систему

$$\overline{F} = \rceil \begin{matrix} X \setminus \{a,b,d\} & Y \setminus \{f,h\} & Z \setminus \{b\} \\ X \setminus \{b,c\} & Y \setminus * & Z \setminus \{a,c\} \end{matrix} \lfloor = \rceil \begin{matrix} \{c\} & \{g\} & \{a,c\} \\ \{a,d\} & \varnothing & \{b\} \end{matrix} \lfloor .$$

Теперь опишем возможные операции и соотношения для *D*-объектов. Пусть заданы однотипные *D*-кортежи $P =]P_1\ P_2\ \ldots\ P_n[$ и $Q =]Q_1\ Q_2\ \ldots\ Q_n[$.

Теорема 2.9. Для произвольного *D*-кортежа $P =]P_1\ P_2\ \ldots\ P_n[$ его дополнение равно *C*-кортежу $\overline{P} = [\overline{P_1}\ \ \overline{P_2}\ \ \ldots\ \ \overline{P_n}\ ]$.

Теорема 2.10. Дополнение *D*-системы есть *C*-система той же размерности, в которой каждая компонента равна дополнению соответствующей компоненты в исходной *D*-системе.

Обобщая теоремы 2.7-2.10, можно сказать, что дополнение произвольного АК-объекта *P* эквивалентно АК-объекту *Q* альтернативного класса и той же размерности, в котором каждая компонента Q_{ij} равна дополнению соответствующей компоненты P_{ij} в *P*.

Имеется исключение, когда дополнение может быть непосредственно выражено АК-объектом того же класса. Если *C*-кортеж или *D*-кортеж имеют только одну нефиктивную компоненту, то для вычисления дополнения достаточно в исходной структуре заменить нефиктивную компоненту на ее дополнение. Например, дополнением *C*-кортежа $[*\ A\ *]$ является *C*-кортеж $[*\ \overline{A}\ *]$, который в свою очередь равен *D*-кортежу $]\varnothing\ \overline{A}\ \varnothing[$.

Теорема 2.11. $]P_1 \cap Q_1\ \ P_2 \cap Q_2\ \ldots\ P_n \cap Q_n[\subseteq P \cap Q$, причем равенство возможно лишь в двух случаях:

(i) $P \subseteq Q$ или $Q \subseteq P$;

(ii) $P_i = Q_i$ для всех соответствующих пар компонент, за исключением одной пары.

Доказательство. Пусть $D =]P_1 \cap Q_1\ \ P_2 \cap Q_2\ \ldots\ P_n \cap Q_n[$. По закону контрапозиции утверждение теоремы справедливо тогда и только тогда, когда выполняется $\overline{P \cap Q} \subseteq \overline{D}$ или $\overline{P} \cup \overline{Q} \subseteq \overline{D}$, где $\overline{P}$, $\overline{Q}$, $\overline{D}$ – *C*-кортежи $[\overline{P_1}\ \ \overline{P_2}\ \ \ldots\ \ \overline{P_n}]$, $[\overline{Q_1}\ \ \overline{Q_2}\ \ \ldots\ \ \overline{Q_n}\ ]$ и $[\overline{P_1} \cup \overline{Q_1}\ \ \overline{P_2} \cup \overline{Q_2}\ \ \ldots\ \ \overline{P_n} \cup \overline{Q_n}\ ]$, соответственно. Согласно теореме 2.4, соотношение $\overline{P} \cup \overline{Q} \subseteq \overline{D}$ справедливо, причем равенство возможно лишь в двух случаях: (i) $\overline{P_i} \subseteq \overline{Q_i}$ или $\overline{Q_i} \subseteq \overline{P_i}$; (ii) $\overline{P_i} = \overline{Q_i}$ для всех соответствующих пар компонент за исключением одной пары. Снова используя закон контрапозиции, условия (i) и (ii) можно записать в более привычном виде, а именно так, как в формулировке теоремы. *Конец доказательства.*

В соответствии с определением 2.7 всегда справедливо равенство

$$P \cap Q = \begin{bmatrix} P_1 & P_2 & \ldots & P_n \\ Q_1 & Q_2 & \ldots & Q_n \end{bmatrix}.$$

Теорема 2.12. $P \subseteq Q$, если и только если $P_i \subseteq Q_i$ для всех $i = 1, 2, \ldots, n$.

Доказательство. Рассмотрим дополнения операндов:

$\overline{P} = [\overline{P_1}\ \ \overline{P_2}\ \ \ldots\ \ \overline{P_n}]$ и $\overline{Q} = [\overline{Q_1}\ \ \overline{Q_2}\ \ \ldots\ \ \overline{Q_n}]$.

По закону контрапозиции $P \subseteq Q$, если и только если $\overline{Q} \subseteq \overline{P}$. Из свойств *C*-кортежей следует, что $\overline{Q} \subseteq \overline{P}$, если и только если $\overline{Q_i} \subseteq \overline{P_i}$ для всех *i*. Соответственно, $P_i \subseteq Q_i$ для всех *i*, из чего следует справедливость предложения. *Конец доказательства.*

Теорема 2.13. $P \cup Q =]P_1 \cup Q_1\ \ P_2 \cup Q_2\ \ldots\ P_n \cup Q_n[$.

Доказательство этой теоремы очевидно, если вспомнить, что *D*-кортежи *P* и *Q* – это диагональные *C*-системы. Если итоговый *D*-кортеж содержит хотя бы одну компоненту, равную домену соответствующего атрибута, то результат равен частному универсуму.

Теорема 2.14. Пересечение двух однотипных *D*-систем равно *D*-системе, содержащей все *D*-кортежи операндов.

Теорема 2.15. Объединение *D*-систем *P* и *Q* эквивалентно *D*-системе, состоящей из всех не равных универсуму *D*-кортежей, образующихся при выполнении операций объединения каждого *D*-кортежа из *P* с каждым *D*-кортежем из *Q*.

Нетрудно видеть, что соотношения между *C*-объектами (*C*-кортежи и *C*-системы) и *D*-объектами (*D*-кортежи и *D*-системы) соответствуют законам двойственности де Моргана. В силу этого они названы ***альтернативными классами.***

Отметим, что все теоремы текущего раздела соответствуют алгоритмам полиномиальной сложности. Однако, если необходимо выполнить операции или проверить соотношения для пары АК-объектов, которые относятся к альтернативным классам, то сложность этих алгоритмов в общем случае значительно возрастает, поскольку для их реализации требуется преобразование одного из АК-объектов в альтернативный класс. Лишь в немногих случаях можно обойтись без таких преобразований, то есть непосредственно анализировать структуры, относящиеся к альтернативным классам. В частности, для проверки включения одного АК-объекта в другой на

основе следующих теорем также можно построить полиномиальные алгоритмы.

Теорема 2.16. Для *C*-кортежа $P = [P_1\ P_2\ \ldots\ P_n]$ и *D*-кортежа $Q =]Q_1\ Q_2\ \ldots\ Q_n[$ справедливо $P \subseteq Q$, если и только если по крайней мере для одного i соблюдается $P_i \subseteq Q_i$.

Доказательство. Каждый *D*-кортеж эквивалентен *C*-системе, содержащей n *C*-кортежей, каждый из которых состоит из всех полных компонент, за исключением Q_i. Следовательно (учитывая, что *C*-система есть объединение *C*-кортежей), для того, чтобы соблюдалось $P \subseteq Q$, необходимо выполнение $P_i \subseteq Q_i$ по крайней мере для одного i. Действительно, если один из *C*-кортежей преобразованного в *C*-систему *D*-кортежа Q равен $[* * \ldots\ Q_i \ldots *]$ и $P_i \subseteq Q_i$, то $P \subseteq [* * \ldots\ Q_i \ldots *]$ и, следовательно, $P \subseteq Q$. Докажем достаточность. Предположим, что условие $P_i \subseteq Q_i$ не соблюдается для всех i. Докажем, что в таком случае $P \subseteq Q$ невозможно. Из предположения следует, что для каждого i существует $R_i = P_i \setminus Q_i \neq \varnothing$. Значит, для всех i выполняется $R_i \subseteq P_i$ и $R_i \subseteq \overline{Q_i}$. Тогда существует непустой *C*-кортеж $R = [R_1\ R_2\ \ldots\ R_n]$, такой что $R \subseteq P$ и $R \subseteq \overline{Q}$, из чего, в свою очередь, следует невозможность $P \subseteq Q$. *Конец доказательства.*

Теорема 2.17. Для *C*-кортежа (или *D*-кортежа) P и *D*-системы Q справедливо $P \subseteq Q$, если и только если для каждого *D*-кортежа Q_j из Q выполняется $P \subseteq Q_j$.

Доказательство. Утверждение следует из того, что *D*-система есть пересечение множеств элементарных кортежей, содержащихся в *D*-кортежах, входящих в ее состав. Поскольку P включено в каждый *D*-кортеж, то оно включено также в их пересечение и, следовательно, в *D*-систему. *Конец доказательства.*

Следующее утверждение в силу его тривиальности приводится без доказательства.

Теорема 2.18. Для *C*-системы P и *D*-системы Q справедливо $P \subseteq Q$, если и только если каждый *C*-кортеж из P включен в каждый *D*-кортеж из Q.

Далее рассмотрим алгоритмы, регламентирующие упомянутые преобразования АК-объектов.

2.2. Преобразования АК-объектов в альтернативные классы

Приведенные выше операции и алгоритмы проверки включения определены не для всех возможных сочетаний классов АК-объектов. В частности, не приведены алгоритмы пересечения *D*-системы и *C*-системы. Кроме того, пока что не предусмотрены алгоритмы проверки включения АК-объектов $P \subseteq Q$, когда P – *D*-система или Q – *C*-система. Следовательно, необходимо предложить алгоритмы преобразования АК-объектов в альтернативные классы. Эти алгоритмы, в общем случае, не полиномиальны по вычислительной сложности.

Теорема 2.19. Каждый *C*-кортеж (*D*-кортеж) P преобразуется в эквивалентную ему диагональную *D*-систему (*C*-систему).

Диагональная *D*-система (по аналогии с диагональной *C*-системой) есть выраженная матрицей размерности $n \times n$ *D*-система, у которой все недиагональные компоненты равны фиктивной компоненте "$\varnothing$".

Например, *D*-кортеж $]A \varnothing B C[$, где A, B, C – непустые компоненты, преобразуется в *C*-систему $\begin{bmatrix} A & * & * & * \\ * & * & B & * \\ * & * & * & C \end{bmatrix}$, а *C*-кортеж $[A\ B * C]$ – в *D*-систему $\begin{bmatrix} A & \varnothing & \varnothing & \varnothing \\ \varnothing & B & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & C \end{bmatrix}$.

Ясно, что алгоритмы преобразования *C*-кортежей и *D*-кортежей в альтернативные классы не требуют для своей реализации алгоритмов экспоненциальной сложности. Трудоемкость алгоритмов существенно возрастает при преобразовании в альтернативные классы *C*-систем и *D*-систем. Следующие два предложения в силу их тривиальности приводятся без доказательства.

Теорема 2.20. *D*-система P, содержащая m *D*-кортежей, эквивалентна *C*-системе, которая равна пересечению m *C*-систем, полученных с помощью преобразования каждого *D*-кортежа из P в *C*-систему.

Важность данной теоремы с теоретической точки зрения состоит в том, что она определяет существенную в ряде доказательств и обоснований

возможность представить любую структуру АК в виде *C*-кортежа или *C*-системы.

Теорема 2.21. *C*-система *P,* содержащая *m* *C*-кортежей, эквивалентна *D*-системе, которая равна объединению *m* *D*-систем, полученных с помощью преобразования каждого *C*-кортежа из *P* в *D*-систему.

Рассмотрим преобразование *D*-системы $P = \begin{bmatrix} \{a,c\} & \{d\} & \{b,d\} \\ \varnothing & \{a,d\} & \{a,c\} \\ \{b,c\} & \varnothing & \{b\} \end{bmatrix}$ в *C*-систему. Преобразуя в соответствии с теоремой 2.19 каждый *D*-кортеж в *C*-систему, получим промежуточный результат:

$$P = \begin{bmatrix} \{a,c\} & * & * \\ * & \{d\} & * \\ * & * & \{b,d\} \end{bmatrix} \cap \begin{bmatrix} * & \{a,d\} & * \\ * & * & \{a,c\} \end{bmatrix} \cap \begin{bmatrix} \{b,c\} & * & * \\ * & * & \{b\} \end{bmatrix}.$$

Вычислим в соответствии с теоремой 2.5 пересечение первых двух *C*-систем (получающиеся при этом пустые *C*-кортежи, которые затем удаляются из *C*-системы, здесь и далее для экономии места не показаны):

$$\begin{bmatrix} \{a,c\} & * & * \\ * & \{d\} & * \\ * & * & \{b,d\} \end{bmatrix} \cap \begin{bmatrix} * & \{a,d\} & * \\ * & * & \{a,c\} \end{bmatrix} = \begin{bmatrix} \{a,c\} & \{a,d\} & * \\ \{a,c\} & * & \{a,c\} \\ * & \{d\} & * \\ * & \{d\} & \{a,c\} \\ * & \{a,d\} & \{b,d\} \end{bmatrix}.$$

Теперь найдем пересечение полученной *C*-системы с оставшейся:

$$P = \begin{bmatrix} \{a,c\} & \{a,d\} & * \\ \{a,c\} & * & \{a,c\} \\ * & \{d\} & * \\ * & \{d\} & \{a,c\} \\ * & \{a,d\} & \{b,d\} \end{bmatrix} \cap \begin{bmatrix} \{b,c\} & * & * \\ * & * & \{b\} \end{bmatrix} = \begin{bmatrix} \{c\} & \{a,d\} & * \\ \{a,c\} & \{a,d\} & \{b\} \\ \{c\} & * & \{a,c\} \\ \{b,c\} & \{d\} & * \\ * & \{d\} & \{b\} \\ \{b,c\} & \{d\} & \{a,c\} \\ \{b,c\} & \{a,d\} & \{b,d\} \\ * & \{a,d\} & \{b\} \end{bmatrix}.$$

Нетрудно убедиться, что преобразование АК-объектов в альтернативный класс дает возможность обеспечить полноту всех операций с АК-объектами и проверок соотношений между ними, не прибегая к представлению АК-объектов в форме множеств элементарных кортежей. Однако уже из последнего примера

видно, что преобразование АК-объекта в альтернативный класс во многих случаях весьма трудоемко. В главе 3 будут рассмотрены некоторые методы уменьшения этой трудоемкости.

Приведенные выше соотношения и преобразования позволяют получить полный спектр операций алгебры множеств и проверок включения и эквивалентности для любых пар АК-объектов в заданном универсуме. Отсюда следует фундаментальное предложение, сформулированное в теореме 2.1.

Итак, любая система отношений с одинаковыми схемами атрибутов представляет собой обычную алгебру множеств, в которой каждое отношение рассматривается как множество элементарных кортежей. В алгебре кортежей такой системе множеств соответствует совокупность однотипных АК-объектов. Ранее в главе были сформулированы в виде теорем, определений и предложений алгоритмы выполнения операций алгебры множеств с АК-объектами, а также алгоритмы проверки включения одного АК-объекта в другой. Эти алгоритмы систематизированы и приведены в Приложении.

Как уже упоминалось, в АК можно непосредственно выполнять операции алгебры множеств с АК-объектами, если они однотипны (т.е. имеют одну и ту же схему отношения). Однако АК также дает возможность производить аналогичные операции над многоместными отношениями, которые имеют разные типы. Предварительно отношения приводятся к одной схеме с помощью перечисленных ранее (см. определение 2.1) операций с атрибутами. Следующий подраздел показывает, как это реализуется.

2.3. Операции с атрибутами, операции соединения и композиции, обобщенные операции

Универсальный характер АК обусловлен тем, что в ней, помимо операций алгебры множеств с АК-объектами, дополнительно вводятся еще пять вышеупомянутых операций с атрибутами, которые существенно расширяют аналитические возможности АК. Рассмотрим эти операции более подробно, а также проиллюстрируем их применение при выполнении операций соединения и композиции отношений.

Переименование атрибутов допускается только для атрибутов, относящихся к одному сорту. Эта операция используется при замене

переменных, в частности, в алгоритмах вычисления транзитивного замыкания графа.

Перестановка атрибутов – операция, при выполнении которой в матрице АК-объекта меняются местами столбцы, а в схеме отношения соответственно изменяется порядок атрибутов.

При одновременной перестановке столбцов АК-объекта и соответствующих им атрибутов содержание отношения не изменяется. Эта операция необходима для того, чтобы привести АК-объекты с одинаковыми, но расположенными в разном порядке атрибутами, к такому виду, при котором можно выполнять операции алгебры множеств.

Например, *C*-система $P[XYZ] = \begin{bmatrix} \{a,b,d\} & \{f,h\} & \{b\} \\ \{b,c\} & * & \{a,c\} \end{bmatrix}$ при перестановке атрибутов преобразуется в эквивалентную *C*-систему $P[YXZ] = \begin{bmatrix} \{f,h\} & \{a,b,d\} & \{b\} \\ * & \{b,c\} & \{a,c\} \end{bmatrix}$.

Обращение отношений. В случае бинарных отношений перестановка столбцов без перестановки атрибутов позволяет получить отношение, обратное исходному. Так, отношение $G[XY] = \begin{bmatrix} \{a\} & \{a,b\} \\ \{b,c\} & \{a,c\} \end{bmatrix}$ при перестановке столбцов преобразуется в обратное отношение $G^{-1}[XY] = \begin{bmatrix} \{a,b\} & \{a\} \\ \{a,c\} & \{b,c\} \end{bmatrix}$. При обращении АК-объекта все элементарные кортежи (s, t) исходного отношения преобразуются в обратные – (t, s). Если элементарный кортеж содержит одинаковые элементы (например, (b, b)), то при обращении он не изменяется.

Добавление фиктивного атрибута (*+Atr*) осуществляется, если добавляемый атрибут отсутствует в схеме отношения АК-объекта (АК-объекты с повторяющимися атрибутами допустимы, но здесь не рассматриваются). При выполнении этой операции одновременно в схему отношения добавляется имя нового атрибута, а в структуру вводится на соответствующем месте новый столбец с фиктивными компонентами, при этом в *C*-кортежи и в *C*-системы добавляется фиктивные компоненты “*”, а в *D*-кортежи и *D*-системы – фиктивные компоненты “∅”.

Элиминация атрибута (*–Atr*) осуществляется так: из АК-объекта

удаляется столбец, а из его схемы отношения – соответствующий атрибут. Но, в отличие от предыдущей операции, логический смысл элиминации зависит от того, к какому классу объектов она применяется.

Семантика операций "+*Atr*" и "–*Atr*" будет изложена ниже в п. 2.4. Они применяются, в частности, при вычислении ***соединения*** и ***композиции*** двух разнотипных отношений, заданных АК-объектами. Эти операции часто используются при решении разнообразных задач на графах, в базах данных, в интеллектуальных системах [*Кулик*, 1996; 2007 b]. В логическом анализе операции соединения соответствует конъюнкция двух предикатов с отличающимся составом переменных, например, $P(x, y) \wedge Q(x, z, v)$, а операции композиции – конъюнкция предикатов под квантором существования, например, $\exists x(P(x, y) \wedge Q(x, z, v))$.

Простейший пример применения этих операций приведен в следующей задаче. Пусть дано отношение РОДИТЕЛИ[*X*, *Y*], в котором первый элемент пары обозначает родителя, а второй – его (или ее) ребенка. В результате соединения отношения РОДИТЕЛИ с самим собой получается трехместное отношение "персоны – их дети – их внуки", а в результате композиции этого отношения с самим собой образуется двухместное отношение "персоны – их внуки".

В общем случае можно выполнять операции соединения и композиции отношений для любых пар АК-объектов. Пусть заданы две структуры $R_1[\boldsymbol{V}]$ и $R_2[\boldsymbol{W}]$, где $\boldsymbol{V}$ и $\boldsymbol{W}$ – множества атрибутов, причем $\boldsymbol{V} \neq \boldsymbol{W}$. Эти множества можно разложить на непересекающиеся подмножества $\boldsymbol{X}$, $\boldsymbol{Y}$, $\boldsymbol{Z}$ с помощью таких преобразований:

$\boldsymbol{X} = \boldsymbol{W} \setminus \boldsymbol{V}$; $\boldsymbol{Y} = \boldsymbol{W} \cap \boldsymbol{V}$; $\boldsymbol{Z} = \boldsymbol{V} \setminus \boldsymbol{W}$.

Тогда получим $\boldsymbol{V} = \boldsymbol{Y} \cup \boldsymbol{Z}$ и $\boldsymbol{W} = \boldsymbol{X} \cup \boldsymbol{Y}$, и заданные отношения можно переписать как $R_1[\boldsymbol{YZ}]$ и $R_2[\boldsymbol{XY}]$.

Операция ***соединения*** $R_1[\boldsymbol{YZ}] \oplus R_2[\boldsymbol{XY}]$ отношений обычно выполняется с помощью попарного сравнения всех элементарных кортежей из разных отношений. Если при сравнении элементарных кортежей окажется, что в проекции [$\boldsymbol{Y}$] они совпадают, то из двух кортежей формируется кортеж со схемой отношения [$\boldsymbol{XYZ}$], который становится одним из элементов соединения отношений. Например, имеются два элементарных кортежа $T_1 \in R_1$ и $T_2 \in R_2$,

причем

$T_1[\boldsymbol{YZ}] = (c, d, e, f, g)$; $T_2[\boldsymbol{XY}] = (a, b, c, d, e)$, где

$T_2[\boldsymbol{X}] = (a, b)$; $T_1[\boldsymbol{Z}] = (f, g)$; $T_1[\boldsymbol{Y}] = T_2[\boldsymbol{Y}] = (c, d, e)$.

Тогда результат композиции этих кортежей – элементарный кортеж $T_3[\boldsymbol{XYZ}] = (a, b, c, d, e, f, g)$.

В АК операция соединения отношений существенно упрощается, ее можно выполнить без попарного сравнения всех элементарных кортежей по формуле:

$$R_1[\boldsymbol{YZ}] \oplus R_2[\boldsymbol{XY}] = +\boldsymbol{X}(R_1) \cap +\boldsymbol{Z}(R_2) \qquad (2.1)$$

Операция ***композиции*** $R_1[\boldsymbol{YZ}] \circ R_2[\boldsymbol{XY}]$ отношений вычисляется после вычисления соединения. Для этого нужно их каждого элементарного кортежа, принадлежащего соединению отношений, изъять проекцию $[\boldsymbol{Y}]$. Например, композицией двух приведенных выше кортежей T_1 и T_2 – элементарный кортеж $T_4[\boldsymbol{XZ}] = (a, b, f, g)$.

В алгебре кортежей композиция отношений определяется по формуле:

$$R_1[\boldsymbol{YZ}] \circ R_2[\boldsymbol{XY}] = -\boldsymbol{Y}(+\boldsymbol{X}(R_1) \cap +\boldsymbol{Z}(R_2)) = -\boldsymbol{Y}(R_1 \oplus R_2), \qquad (2.2)$$

при условии, что $(R_1 \oplus R_2)$ является *C*-кортежем или *C*-системой.

Рассмотрим пример. Пусть в пространстве $\boldsymbol{S}$ заданы АК-объекты

$$R_1[YZ] = \begin{bmatrix} \{f\} & \{a,b\} \\ \{g,h\} & \{a,c\} \end{bmatrix}; \; R_2[XY] = \begin{bmatrix} \{a\} & \{g,h\} \\ \{b,c\} & \{f\} \end{bmatrix}.$$

По формуле (2.1) найдем их соединение. Для этого добавим фиктивный атрибут X в R_1 и фиктивный атрибут Z в R_2, после чего вычислим их пересечение:

$$R_1 \oplus R_2 = \begin{bmatrix} * & \{f\} & \{a,b\} \\ * & \{g,h\} & \{a,c\} \end{bmatrix} \cap \begin{bmatrix} \{a\} & \{g,h\} & * \\ \{b,c\} & \{f\} & * \end{bmatrix} = \begin{bmatrix} \{b,c\} & \{f\} & \{a,b\} \\ \{a\} & \{g,h\} & \{a,c\} \end{bmatrix}.$$

Далее ищем по формуле (2.2) их композицию. Для этого надо элиминировать из $R_1 \oplus R_2$ атрибут Y:

$$R_1 \circ R_2 = \begin{bmatrix} \{b,c\} & \{a,b\} \\ \{a\} & \{a,c\} \end{bmatrix}.$$

В случаях, когда требуется найти соединение отношений, которые выражены как *D*-системы, можно воспользоваться формулой (2.1). При этом

надо учесть особенности операции $+Atr$ для D-систем (в добавляемом фиктивном атрибуте должны быть пустые компоненты) и операции пересечения D-систем. Однако для операции композиции в этом случае формула (2.2) не подходит, так как элиминация атрибута из D-системы означает навешивание квантора $\forall$, но не $\exists$ (см. п. 2.4). Для вычисления композиции во всех случаях справедлива формула

$$R_1[\boldsymbol{YZ}] \circ R_2[\boldsymbol{XY}] = \exists \boldsymbol{y}(+\boldsymbol{X}(R_1) \cap +\boldsymbol{Z}(R_2)) = \exists \boldsymbol{y}(R_1 \oplus R_2), \qquad (2.3)$$

Если при вычислении соединения АК-объектов получается D-система, то перед элиминацией атрибутов ее необходимо преобразовать в эквивалентную C-систему или использовать метод квантификации для D-систем, который рассматривается в главе 3.

Назовем операции и отношения алгебры множеств с АК-объектами с предварительным добавлением недостающих фиктивных атрибутов ***обобщенными операциями и отношениями*** алгебры множеств в АК и обозначим их соответственно $\cap_G$, $\cup_G$, $\setminus_G$, $\subseteq_G$, $=_G$ и т.д. С учетом этого формулу 2.1 можно записать как

$$R_1[\boldsymbol{YZ}] \oplus R_2[\boldsymbol{XY}] = R_1[\boldsymbol{YZ}] \cap_G R_2[\boldsymbol{XY}].$$

Операции $\cap_G$, и $\cup_G$ полностью соответствуют логическим операциям $\wedge$ и $\vee$. Отношение $\subseteq_G$ в АК эквивалентно ***отношению выводимости*** в исчислении предикатов. Отношение $=_G$ означает, что структуры равны при условии, что они приведены к одной схеме отношения путем добавления атрибутов. Это обстоятельство позволяет использовать принципиально новый подход к построению процедур логического вывода и проверок выводимости, который рассматривается в главе 4.

Далее раскроем взаимосвязь АК с логическими исчислениями.

2.4. Логические исчисления и АК

2.4.1. АК как интерпретация логических исчислений

Рассмотрим систему, с помощью которой можно показать, что АК является алгебраической интерпретацией некоторых известных логических исчислений.

Пусть задана система S с множеством возможных состояний $Y = \{t_1, t_2, ..., t_r\}$. Кроме того, в состав S входит некоторое множество V узлов или подсистем $V = \{V_1, V_2, ..., V_n\}$. В свою очередь, каждому узлу V_i соответствует некоторое множество $X_i = \{c_1^i, c_2^i, ..., c_{k_i}^i\}$ его состояний, где k_i – число состояний, в котором может находиться узел V_i, а множества X_i – произвольные конечные или бесконечные множества. Наглядно соотношения в этой системе можно представить с помощью схемы, показанной на рис. 2.1.

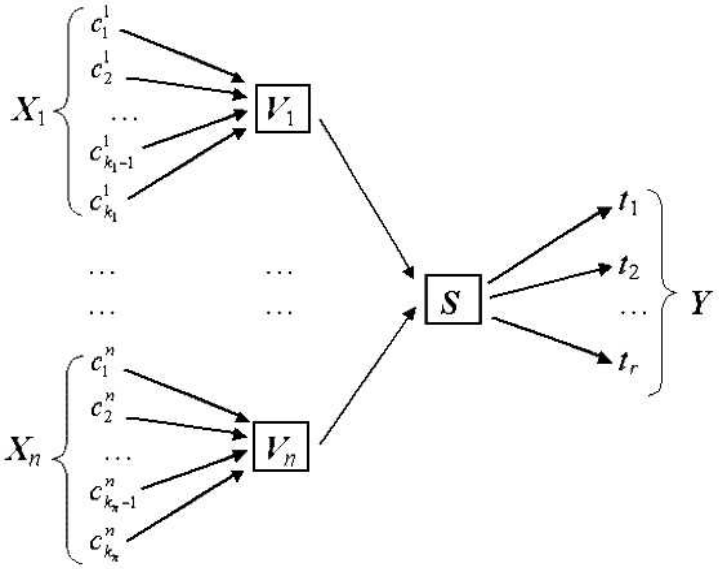

Рис. 2.1. Схема состояний системы.

Точная модель системы S дается соответствием между всеми возможными наборами состояний всех узлов и состояниями системы. Каждому состоянию $t_j \in Y$ системы соответствует некоторое множество элементарных кортежей из декартова произведения $D = X_1 \times X_2 \times \ ... \ \times X_n$. Математическая схема этого соответствия:

$$S: D \to Y. \qquad (2.4)$$

Однако модель (2.4) теряет практический смысл с увеличением числа узлов, а также с увеличением числа состояний узлов и самой системы, так как, даже в сравнительно небольших системах, число всех элементарных наборов

состояний может превзойти вычислительные ресурсы современных компьютеров. К тому же не совсем ясно, как к модели (2.4) можно применить мощные аналитические средства математической логики.

Чтобы уточнить соотношения между АК и различными вариантами исчислений, рассмотрим ряд ограничений, накладываемых на предметную область при ее представлении моделью (2.4).

Ограничение ***C1***: соответствие $D \to Y$ является однозначным (или функциональным) отображением. Это означает, что ни одному элементарному кортежу из D не может соответствовать более одного элемента из Y.

Ограничение ***C2***: множество Y содержит ровно два элемента.

Ограничение ***C3***: множества X_i не отличаются друг от друга.

Ограничение ***C4***: домены всех атрибутов и множество Y – двухэлементные множества $\{0, 1\}$.

Нетрудно обосновать следующие утверждения.

Случай 1. Система, удовлетворяющая ограничениям ***C1, C2*** и ***C3***, изоморфна некоторой модели классического исчисления предикатов. Здесь отношения, заданные в любой проекции пространства D, интерпретируются как многоместные предикаты или логические формулы. В этой системе каждая формула рассматривается как отображение соответствующего ей частного универсума на множество $\{\boldsymbol{T}, \boldsymbol{F}\}$ логических констант: если некоторый элементарный кортеж принадлежит отношению, то он одновременно является выполняющей подстановкой соответствующей логической формулы, и ему соответствует константа ***T***.

Случай 2. Система с ограничениями ***C1,*** и ***C2*** соответствует многосортному исчислению предикатов.

Случай 3. Система с ограничениями ***C1*** и ***C4*** соответствует модели исчисления высказываний.

Рассмотрим подробнее эти интерпретации, используя понятия АК.

С каждой формулой классического исчисления предикатов (случай 1) связано некоторое (возможно, пустое) многоместное отношение, состав атрибутов которого идентичен составу свободных переменных в этой формуле. Пусть переменным x, y, z и т.д. сопоставлены атрибуты $X, Y, Z, ...$, а домены этих атрибутов в точности равны множеству S. Тогда каждый предикат, например, $P(x, z)$, моделируется некоторым отношением $P[XZ]$ на декартовом

произведении $X \times Z = S \times S$. Если в этой системе необходимо представить функцию, например, $f(x, z) = y$, то используется отображение $F: X \times Z \to Y$ или, если учитывать равенство всех атрибутов, – $F: S \times S \to S$. Функцию также можно представить как отношение, например, $F[XZY]$, в котором каждому элементарному кортежу из проекции $[XZ]$ отвечает одно и только одно значение в проекции $[Y]$.

Формулы, у которых предикаты соединены логическими связками, например, $P(x, z) \supset Q(y, z)$, можно выразить как результат операций АК над отношениями. В данном случае, если предикаты $P(x, z)$ и $Q(y, z)$ представлены АК-объектами $P[XZ]$ и $Q[YZ]$, то логической формуле $P(x, z) \supset Q(y, z)$, которую можно преобразовать в формулу $\neg(P(x, z)) \vee Q(y, z)$, соответствует в АК следующее выражение:

$+Y(\overline{P[XZ]}) \cup +X(Q[XY])$ или $\overline{P[XZ]} \cup_G Q[XY]$.

Схемы отношений предикатов, входящих это выражение, могут быть различными – с помощью перестановки атрибутов и ввода фиктивных атрибутов различные отношения перед выполнением операций могут быть приведены в АК к единой схеме.

Рассмотрим один из примеров интерпретации квантора $\exists$ (более подробно моделирование кванторов в АК описано ниже). Пусть задана формула

$F = \exists y(P(x, y) \wedge Q(y, z))$

Предикаты $P(x, y)$ и $Q(y, z)$ можно представить как отношения P и Q на частных универсумах $X \times Y$ и $Y \times Z$. Согласно методам АК, пересечение этих отношений с добавленными фиктивными атрибутами образует их соединение, схема отношения которого содержит три атрибута X, Y и Z. При этом возможны два варианта: полученное отношение может быть не пустым или пустым. Отношение пусто, если в P не существует элементарного кортежа, у которого значение атрибута Y совпадает со значением этого же атрибута в каком-нибудь элементарном кортеже из Q. Если теперь к этому трехместному отношению применить квантор $\exists y$, то для пустого соединения отношений P и Q формула F ложна для всех значений X, Y и Z. При непустом соединении отношений P и Q формула F будет представлять уже не соединение, а композицию отношений.

В то же время логическое значение формулы F не изменится, если в ее интерпретации мы в композицию отношений P и Q вставим фиктивный атрибут

Y. В рамках законов логических исчислений и булевой алгебры это означает, что формула *F* заменяется на эквивалентную ей формулу $F \wedge \text{true}$. В АК логической константе true соответствует любой частный универсум, т.е. отдельный атрибут или произвольная проекция универсума $D = X \times Y \times Z \times ...$

Поясним сказанное примером. Пусть отношения *P* и *Q* заданы *C*-системами, и в результате вычисления композиции получен результат в виде *C*-системы *R*. Если добавить к этому отношению фиктивный атрибут *Y*, то получим уже другую *C*-систему.

Пусть $R[XZ] = \begin{bmatrix} \{a,c\} & \{f\} \\ \{b,c\} & \{g\} \end{bmatrix}$, $R_1[XYZ] = \begin{bmatrix} \{a,c\} & * & \{f\} \\ \{b,c\} & * & \{g\} \end{bmatrix}$. Отношения *R* и R_1 отличаются только присутствием фиктивного атрибута *Y* в R_1. С точки зрения традиционной теории отношений $R \neq R_1$. Но с точки зрения логики и АК они эквивалентны в том смысле, что для любого элементарного кортежа из этих отношений, взятого в проекции [*XZ*] (например, элементарный кортеж (*a*, *f*) из отношений *R* или R_1), можно утверждать, что значение переменной *y* для него несущественно.

Во многих системах приходится использовать в качестве областей изменения переменных (атрибутов) различные множества, не совпадающие друг с другом (случай 2). Тогда применение канонической интерпретации, в которой домены всех атрибутов равны одному и тому же множеству *S*, вызывает немалые трудности (см. п. 1.3). Чтобы этого избежать, можно использовать многосортное исчисление предикатов.

В терминах АК нетрудно выразить и исчисление высказываний (случай 3). Для этого достаточно установить для каждого атрибута только два возможных значения. Тогда доменом каждого атрибута будет множество {0, 1}.

Известно, что любую формулу исчисления высказываний можно представить как ДНФ или КНФ. Чтобы перевести в *C*-систему каждую формулу исчисления высказываний, выраженную в виде ДНФ, необходимо:

1) выделить множество всех литералов (x_i, $\overline{x_i}$), содержащихся в формуле;

2) вместо литералов x_i и $\overline{x_i}$ в формуле ввести соответственно одноэлементные множества {1} и {0};

3) в каждую конъюнкцию на соответствующих местах разместить фиктивные компоненты "*" вместо недостающих переменных;

4) расположить полученные *C*-кортежи вертикально в виде матрицы и получить *C*-систему.

Например, формула $H = (x_1 \wedge \overline{x_3}) \vee (x_2 \wedge x_3 \wedge \overline{x_4}) \vee (x_4)$ преобразуется в *C*-систему

$$R = \begin{bmatrix} \{1\} & * & \{0\} & * \\ * & \{1\} & \{1\} & \{0\} \\ * & * & * & \{1\} \end{bmatrix}.$$

ДНФ, выраженные как *C*-системы, нетрудно представить в виде множества выполняющих подстановок этой формулы. Например, *C*-кортеж [{1} * {0} *] содержит 4 выполняющие подстановки, которые можно получить, вычислив декартово произведение $\{1\}\times\{0, 1\}\times\{0\}\times\{0, 1\}$. Также несложно определить, является ли произвольно заданная подстановка выполняющей постановкой формулы. Например, подстановку $x_1 = \boldsymbol{T}$; $x_2 = \boldsymbol{F}$; $x_3 = \boldsymbol{F}$; $x_4 = \boldsymbol{T}$ для формулы *H* можно преобразовать в элементарный кортеж (1, 0, 0, 1) и убедиться, что он принадлежит первой и третьей строкам *C*-системы *R*. А это означает, что заданная подстановка – выполняющая подстановка формулы *H*. Кроме того, легко убедиться, что дополнение *R*

$$\overline{R} = \begin{bmatrix} \{0\} & \varnothing & \{1\} & \varnothing \\ \varnothing & \{0\} & \{0\} & \{1\} \\ \varnothing & \varnothing & \varnothing & \{0\} \end{bmatrix}$$

есть *D*-система, которая соответствует ДНФ, равной отрицанию формулы *H*:

$\overline{H} = (\overline{x_1} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3} \vee x_4) \wedge (\overline{x_4})$.

С учетом вышеизложенного можно утверждать, что АК является интерпретацией исчисления высказываний и исчисления предикатов. Но в отличие от традиционной интерпретации, в которой содержится только сопоставление некоторых основных формул математической логики с некоторыми отношениями, в АК имеются также аналитические средства и алгоритмическое обеспечение для практической работы с самыми разными логическими формулами и соответствующими им отношениями. Эти средства позволяют обеспечить переход от отношений, соответствующих областям истинности отдельных предикатов, входящих в некоторую формулу, к отношению, отвечающему всей формуле.

Нетрудно убедиться, что в случаях 1-3 соблюдаются законы алгебры

множеств. Можно дать еще одну логическую интерпретацию системы *S*, оставив лишь ограничение ***C1*** (при неоднозначности соответствия в системе появится неопределенность – одним и тем же набором параметров может характеризоваться более одного состояния системы). Можно рассматривать объединенное пространство $D \times Y$, среди отношений которого в разных проекциях может содержаться (или не содержаться) атрибут *Y*. Тогда множество элементарных кортежей объединенного пространства $D \times Y$ разделится на два непересекающихся множества: множество допустимых (*true*) и множество недопустимых (*false*) состояний, определяемых в соответствии с конструктивными или технологическими особенностями моделируемой системы, и соответствие

$$S_c: D \times Y \rightarrow \{\boldsymbol{T}, \boldsymbol{F}\} \qquad (2.5)$$

окажется изоморфным модели многосортного исчисления предикатов, в которой отсутствуют ограничения на число состояний узлов и системы в целом. Если представить состояния X_i узлов и состояния *Y* системы "значениями истинности" (таких значений в некоторых вариантах неклассической логики может быть более двух, вплоть до бесконечности), то модель (2.5) можно рассматривать как многозначную логику, в которой, тем не менее, справедливы все законы алгебры множеств.

Таким образом, можно сказать, что с точки зрения интерпретации АК выходит за пределы многосортного исчисления предикатов. Для этого рассмотрим модель (2.5), в которой действует ограничение ***C1***. Если отменить ограничение ***C1*** в моделях (2.4) или (2.5), то получится система, в которой не соблюдается функциональное соответствие между состояниями узлов и "значениями истинности". Для такой системы, по-видимому, невозможно подобрать соответствующую логику среди известных логик, но в то же время такая система может представлять определенные реальные сущности, и в ней также не нарушаются законы алгебры множеств.

Поскольку в современных программных системах для представления знаний чаще всего используется язык исчисления предикатов первого порядка, рассмотрим связь этой логической системы с АК более подробно.

2.4.2. Соответствие между АК и исчислением предикатов

Конъюнкция одноместных предикатов с разными переменными интерпретируется в АК *C*-кортежами, в которых отдельные атрибуты не соотносятся с многоместными отношениями. Например, логической формуле $H = P_1(x) \wedge P_2(y) \wedge P_3(z)$ может быть сопоставлен *C*-кортеж $P[XYZ] = [P_1\ P_2\ P_3]$, где $P_1 \subseteq X$; $P_2 \subseteq Y$; $P_3 \subseteq Z$.

Отрицание формулы H (***дизъюнкция*** одноместных предикатов) $\neg H = \neg P_1(x) \vee \neg P_2(y) \vee \neg P_3(z)$ моделируется *D*-кортежем $\overline{P} =]\,\overline{P_1}\ \ \overline{P_2}\ \ \overline{P_3}\,[$.

Элементарный кортеж, входящий в состав непустого АК-объекта, определяет ***выполняющую подстановку*** логической формулы. АК-объект, равный любому частному универсуму, соответствует общезначимой формуле или ***тавтологии***, непустой АК-объект – ***выполнимой формуле***, а пустой АК-объект – ***тождественно ложной формуле***.

Домены атрибутов в АК могут быть любыми, не обязательно равными друг другу множествами. Это означает, что структуры АК могут интерпретировать формулы ***многосортного исчисления предикатов***.

Далее покажем, каким образом представляются кванторы средствами АК.

Если в *C*-кортеж или в *C*-систему добавляется фиктивный атрибут, то такая процедура соотносится с известным в исчислении предикатов правилом вывода, которое называется ***правилом обобщения***. Например, если АК-объект $G[XZ] = \begin{bmatrix} \{a,c\} & * \\ \{a,c,d\} & \{b,c\} \end{bmatrix}$ представляет формулу $F(x, z)$ исчисления предикатов, то, добавив в этот АК-объект фиктивный атрибут Y, получим АК-объект $G_1[XYZ] = +Y(G[XZ]) = \begin{bmatrix} \{a,c\} & * & * \\ \{a,c,d\} & * & \{b,c\} \end{bmatrix}$, который моделирует формулу $\forall y F(x, z)$, полученную из формулы $F(x, z)$ по правилу обобщения. Для *C*-кортежей и *C*-систем это соотношение очевидно, но аналогичное преобразование для *D*-кортежей и *D*-систем нужно доказывать, так как фиктивные атрибуты, добавляемые в *D*-систему, содержат пустые фиктивные компоненты ($\varnothing$).

Теорема 2.22. Добавление нового фиктивного атрибута в *D*-кортеж или в *D*-систему соответствует правилу обобщения.

Доказательство. Пусть задан *D*-кортеж $P[X_1X_2 \ldots X_n] =]P_1\ P_2 \ldots P_n[$. Добавим в него фиктивный атрибут *Y*. Получим $Q[YX_1X_2 \ldots X_n] =]\varnothing\ P_1\ P_2 \ldots P_n[$. При преобразовании этих АК-объектов в *C*-системы получим:

$$P = \begin{bmatrix} P_1 & * & ... & * \\ * & P_2 & ... & * \\ ... & ... & ... & ... \\ * & * & ... & P_n \end{bmatrix};\quad Q = \begin{bmatrix} * & P_1 & * & ... & * \\ * & * & P_2 & ... & * \\ ... & ... & ... & ... & ... \\ * & * & * & ... & P_n \end{bmatrix}. \qquad (2.6)$$

Отсюда $Q = +Y(P) = \forall y(P)$.

Предположим, что дана *D*-система $R[X_1X_2 \ldots X_n]$. Пусть $R_1 = +Y(R) = R_1[YX_1X_2 \ldots X_n]$. В *D*-системе R_1 компонентами атрибута *Y* во всех *D*-кортежах будут "∅". При преобразовании этой *D*-системы в *C*-систему согласно теореме 2.20 на 1-м шаге получим *C*-системы с той же структурой, как у *C*-системы *Q* в (2.6). На 2-м шаге при пересечении полученных *C*-систем результаты в проекции $[X_1X_2 \ldots X_n]$ будут такими же, как и при преобразовании *D*-системы *R* в *C*-систему, а компонентами атрибута *Y* в *C*-системе R_1 будут фиктивные компоненты "*". Следовательно, $R_1 = +Y(R) = \forall y(R)$. *Конец доказательства.*

При выполнении операции $+Atr$ содержательный смысл отношения не изменяется. Это аналогично ситуации в исчислении предикатов, когда квантор $\forall x$ применяется к формуле *A*, в которой отсутствует переменная *x* [*Мендельсон*, 1984, с. 54]. Например, если задан АК-объект *P*[*YZ*], моделирующий формулу *F*(*y*, z) исчисления предикатов, то добавив в *P*[*YZ*] фиктивный атрибут *X*, получим АК-объект $+X(P[YZ])$, который интерпретирует формулу $\forall xF(y, z)$. Эта формула по смыслу равнозначна *F*(*y*, z), что позволяет отнести операцию $+Atr$ к семантически равносильным преобразованиям.

Следующие две теоремы устанавливают семантику операции $-Atr$.

Теорема 2.23. Пусть *R*[…*X*…] – *C*-система, у которой отсутствуют *C*-кортежи с пустыми компонентами в атрибуте *X*. Тогда для соответствующего этой *C*-системе предиката *P*(…, *x*, …) результат операции $-X(R)$ моделирует формулу $\exists x(P)$.

Доказательство. Пусть *R* – *C*-кортеж. Тогда равносильность $X(R) \Leftrightarrow \exists x(P)$

при условиях теоремы очевидна. Пусть R – C-система, содержащая C-кортежи $R_1, R_2, \ldots, R_n$. Это означает, что $R = R_1 \cup R_2 \cup \ldots \cup R_n$. В исчислении предикатов отношение R служит интерпретацией формулы $P = P_1 \vee P_2 \vee \ldots \vee P_n$, где P_i – формулы, моделируемые C-кортежами R_i. Применив к R операцию $-X$, получим

$-X(R) = -X(R_1) \cup -X(R_2) \cup \ldots \cup -X(R_n)$.

Правой части этого равенства соответствует следующая формула исчисления предикатов: $\exists x(P_1) \vee \exists x(P_2) \vee \ldots \vee \exists x(P_n)$, которая по правилам эквивалентного преобразования формул в математической логике равна формуле $\exists x(P_1 \vee P_2 \vee \ldots \vee P_n)$, тогда после подстановки $\exists x(P)$. *Конец доказательства.*

Теорема 2.24. Пусть $R[\ldots X\ldots]$ – D-система, у которой отсутствуют D-кортежи с компонентами "∗" в атрибуте X. Тогда для соответствующего этой D-системе предиката $P(\ldots, x, \ldots)$ результат операции $-X(R)$ моделирует формулу $\forall x(P)$.

Доказательство. Пусть $\overline{R}$ – C-система, равная дополнению D-системы R. Ясно, что $\overline{R}$ удовлетворяет условиям теоремы 2.23 и моделирует формулу $\neg P$. Предположим, что $Q = -X(\overline{R})$. Из теоремы 2.23 следует, что Q соответствует формуле $\exists x(\neg P)$. АК-объект $\overline{Q}$ равен D-системе, у которой все компоненты – дополнения компонент Q. Отсюда $\overline{Q} = -X(R)$. Выражение $\overline{Q}$ служит интерпретацией формулы $\neg(\exists x(\neg P))$, которая в силу известного в математической логике соотношения равна $\forall x(P)$. Отсюда следует, что АК-объект $-X(R)$ моделирует формулу $\forall x(P)$. *Конец доказательства.*

Следовательно, элиминация атрибута (например, X) из C-системы означает, что к этому объекту применен квантор $\exists x$, а если атрибут элиминируется из D-системы, то это эквивалентно применению квантора $\forall x$ к данному объекту. Например, пусть даны C-система и ее дополнение, выраженное как D-система:

$$Q[XYZ] = \begin{bmatrix} \{a,b,d\} & \{f,h\} & \{b\} \\ \{b,c\} & * & \{a,c\} \end{bmatrix} \text{ и } \overline{Q}[XYZ] = \begin{bmatrix} \{c\} & \{g\} & \{a,c\} \\ \{a,d\} & \varnothing & \{b\} \end{bmatrix}.$$

$$\text{Тогда } \exists x(Q[XYZ]) = \begin{bmatrix} \{f,h\} & \{b\} \\ * & \{a,c\} \end{bmatrix} \text{ и } \forall x(\overline{Q}[XYZ]) = \begin{bmatrix} \{g\} & \{a,c\} \\ \varnothing & \{b\} \end{bmatrix}.$$

Может оказаться, что после элиминации атрибута в *D*-системе полученный АК-объект будет пустым даже в том случае, когда исходная *D*-система представляла непустое отношение. Поэтому после элиминации атрибута из *D*-системы целесообразно проверить ее непустоту. Один из способов такой проверки – преобразование *D*-системы в *C*-систему. Выполним эту проверку для нашего примера.

$$\begin{bmatrix} \{g\} & \{a,c\} \\ \varnothing & \{b\} \end{bmatrix} = \begin{bmatrix} \{g\} & * \\ * & \{a,c\} \end{bmatrix} \cap [*\ \{b\}] = [\{g\}\ \{b\}].$$

В данном случае результат оказался непустым.

Рассмотренные соответствия между АК и исчислением предикатов (в общем случае, многосортным) систематизированы в виде табл. 2.1, где не отражены возможности логического вывода на структурах АК. Эта тема будет подробно рассмотрена в главе 4.

Таблица 2.1

Алгебра кортежей	***Исчисление высказываний и предикатов***
Элементарный кортеж, принадлежащий АК-объекту	Выполняющая подстановка соответствующей формулы
C-кортеж	Конъюнкция одноместных предикатов или конъюнкция в исчислении высказываний
C-система	Дизъюнктивная нормальная форма (ДНФ)
D-кортеж	Дизъюнкция одноместных предикатов или дизъюнкция в исчислении высказываний
D-система	Конъюнктивная нормальная форма (КНФ)
Непустой АК-объект	Выполнимая формула
АК-объект, равный частному универсуму	Тождественно истинная формула
АК-объект, равный $\varnothing$	Тождественно ложная формула
Операция $+Atr$ (при условии, что добавляемый атрибут отсутствует в схеме отношения АК-объекта)	Правило обобщения или равносильное по смыслу преобразование формулы
Операция $-Atr$ для *C*-кортежей и *C*-систем	Навешивание квантора $\exists_{Atr}$
Операция –Atr для *D*-кортежей и *D*-систем	Навешивание квантора $\forall_{Atr}$

Глава 3. Методы снижения трудоемкости в АК

Того, кто не задумывается о далеких трудностях,
непременно поджидают близкие неприятности.
Конфуций

При реализации алгоритмов на компьютере представление многоместных отношений как множеств элементарных кортежей зачастую приводит к повторению в памяти одних и тех же данных – компонент кортежей. При вычислениях это вызывает многократное дублирование действий.

Запись отношений в более "компактном" виде, а именно в виде АК-объектов, как показано в главе 2 (п. 2.1, пример к теореме 2.5), значительно сокращает вычислительные затраты на осуществление действий над отношениями. Кроме того, переход от элементарных кортежей к кортежам, компоненты которых – не отдельные элементы, а множества, позволяет обойтись меньшими объемами памяти при хранении данных и знаний.

Однако во многих случаях требуются дополнительные средства ускорения вычислительных процедур. Рассмотрим способы снижения трудоемкости операций над отношениями, применимые в алгебре кортежей.

Первый способ – ортогонализация – сводится к разбиению совокупности элементарных кортежей, формирующих некоторый АК-объект, на взаимно не пересекающиеся множества, что исключает дублирование одних и тех же элементарных кортежей в различных подструктурах АК-объекта. Помимо этого, ортогонализация операндов способствует уменьшению размерности результатов операций над ними.

Еще одна возможность сократить время производимых над отношениями операций состоит в использовании их матричных свойств. Исходное отношение, заданное в декартовом произведении D, предлагается разделять на блоки – отношения внутри проекций D – и обрабатывать каждый блок независимо друг от друга (распараллеливать операции), опираясь на известные

свойства декартовых произведений. Анализируя блоки, можно значительно снизить размерность исходной задачи, а в некоторых случаях – и ее вычислительную сложность.

Использование упомянутых методов позволяет применять алгебраический подход не только в системах управления БД, но и для систем БЗ, поскольку во многих случаях уменьшает трудоемкость алгоритмов, реализующих задачи логического анализа (например, задачи выполнимости КНФ).

3.1. Ортогонализация

Анализируя приведенные выше примеры *C*-систем, можно увидеть, что некоторые пары содержащихся в них *C*-кортежей имеют непустое пересечение. Иногда это сильно затрудняет расчеты. Например, возможны ситуации, когда требуется вычислить общее число элементарных кортежей, содержащихся в таких *C*-системах – приходится учитывать все возможные пересечения пар, троек и т.д. *C*-кортежей. Те же трудности возникают, если исследуются метрические свойства АК-объектов, например, их вероятностное представление. Ортогонализация как раз и предназначена для преодоления этих трудностей.

Кроме того, ортогонализация дает эффективные средства уменьшения трудоемкости алгоритмов преобразования АК-объектов в альтернативные классы и, в частности, *D*-систем в *C*-системы. Основная теорема ортогонализации была сформулирована и доказана русским логиком П.С. Порецким [*Порецкий*, 1887].

Рассмотрим основные соотношения ортогонализации, используемые в математической логике.

ДНФ называется ***ортогональной***, если любая пара ее конъюнктов не имеет общих выполняющих подстановок. ***Ортогонализация*** есть преобразование, переводящее произвольную формулу в эквивалентную ей ортогональную ДНФ. В общем случае ортогональной также можно считать любую формулу вида $H_1 \vee H_2 \vee ... \vee H_k$, если для любой пары (H_i, H_j) ее подформул соблюдается $H_i \wedge H_j = \text{false}$ при условии, что $i \neq j$. Это равносильно пустоте пересечения АК-объектов, моделирующих формулы H_i, H_j. Оценка меры ортогональной формулы, если известна мера $\mu(H_i)$ каждой из ее подформул H_i, может быть

вычислена с помощью простого суммирования мер составных частей.

Частный случай ортогональной функции – ***совершенная ДНФ*** (СДНФ), в которой каждый конъюнкт содержит столько литералов, сколько переменных в данной формуле. В АК совершенной ДНФ соответствует *C*-система, где каждый *C*-кортеж содержит в качестве компонент только одноэлементные множества. В основе существующих методов ортогонализации лежит следующее соотношение, полученное П.С. Порецким для формул исчисления высказываний:

Дизъюнкция $H_1 \vee H_2 \vee ... \vee H_k$ *эквивалентна ортогональной ДНФ вида*

$(H_1) \vee (\overline{H_1} \wedge H_2) \vee ... \vee (\overline{H_1} \wedge \overline{H_2} \wedge ... \wedge \overline{H_{k-1}} \wedge H_k)$.

Определение 3.1. *C*-система называется ***ортогональной***, если пересечение любой пары содержащихся в ней различных *C*-кортежей равно пустому *C*-кортежу.

Ортогонализация диагональной *C*-системы (*D*-кортежа) не составляет особого труда и производится согласно теореме 3.1.

Теорема 3.1. *D*-кортеж вида $]Q_1\, Q_2\, ...\, Q_{m\text{-}1}\, Q_m[$, где Q_i – произвольные компоненты, преобразуется в эквивалентную ему ортогональную *C*-систему*:*

$$\begin{bmatrix} Q_1 & * & ... & * & * \\ \overline{Q_1} & Q_2 & ... & * & * \\ & & ... & & \\ \overline{Q_1} & \overline{Q_2} & ... & Q_{m-1} & * \\ \overline{Q_1} & \overline{Q_2} & ... & \overline{Q_{m-1}} & Q_m \end{bmatrix}.$$

Доказательство. Результат следует из соотношения Порецкого и изоморфизма системы исчисления высказываний и АК, так как каждая компонента Q_i в исходном *D*-кортеже при преобразовании в *C*-систему может быть представлена только в двух вариантах (например, Q – истина и $\overline{Q_i}$ – ложь), при этом никакие другие значащие (т.е. в данном случае не фиктивные) подмножества соответствующих доменов атрибутов при преобразовании не используются. *Конец доказательства*.

Рассмотрим пример. Пусть в схеме отношения [*XYZ*], где $X = Y = Z = \{a, b, c, d\}$, задан *D*-кортеж $]\{a, c\}\ \{d\}\ \{b, d\}[$. Тогда по теореме 3.1 получим следующие равенства:

$$]\{a,\ c\}\ \{d\}\ \{b,\ d\}[= \begin{bmatrix} \{a,c\} & * & * \\ * & \{d\} & * \\ * & * & \{b,d\} \end{bmatrix} = \begin{bmatrix} \{a,c\} & * & * \\ \{b,d\} & \{d\} & * \\ \{b,d\} & \{a,b,c\} & \{b,d\} \end{bmatrix},$$

причем вторая *C*-система ортогональна.

Ортогонализация произвольных АК-объектов сводится к ортогонализации эквивалентных им *D*-систем (преобразованию *D*-систем в ортогональные *C*-системы). Для этого требуется: 1) выразить исходный АК-объект как *D*-систему, 2) представить *D*-систему как пересечение *D*-кортежей, 3) каждый *D*-кортеж преобразовать в ортогональную *C*-систему с использованием теоремы 3.1, 4) выполнить пересечение промежуточных ортогональных *C*-систем, опираясь на теорему 3.2.

Теорема 3.2. Если P и Q – ортогональные *C*-системы, то пересечение этих *C*-систем, вычисленное в соответствии с теоремой 2.5, либо пусто, либо состоит из одного *C*-кортежа, либо представляет собой ортогональную *C*-систему.

Доказательство. Каждую из исходных *C*-систем можно представить как некоторое множество элементарных кортежей. Пусть $T(P_i)$, $T(Q_j)$ – множества всех элементарных кортежей, содержащихся в *C*-кортежах P_i из P и Q_j из Q. Ясно, что системы множеств $\{T(P_i)\}$ и $\{T(Q_j)\}$ есть разбиения, так как P и Q – ортогональные *C*-системы. Полученная при пересечении *C*-система будет содержать непустые *C*-кортежи, каждый из которых состоит из множества $T(P_i) \cap T(Q_j)$ элементарных кортежей. Выберем два различных *C*-кортежа результирующей *C*-системы, эквивалентные множествам $T(P_k) \cap T(Q_l)$ и $T(P_r) \cap T(Q_s)$ элементарных кортежей (то есть вариант когда одновременно $k = r$ и $l = s$ исключается, поскольку он означает дублирование одного и того же пересечения). Не нарушая общности, достаточно рассмотреть вариант $k \neq r$ (если $k = r$, то вместо него можно рассмотреть аналогичный вариант $l \neq s$). Тогда равенство $(T(P_k) \cap T(Q_l)) \cap (T(P_r) \cap T(Q_s)) = \varnothing$ следует из того, что $T(P_k) \cap T(P_r) = \varnothing$. В результате после выполнения всех операций пересечения *C*-кортежей возможны три варианта: 1) в итоговой *C*-системе не останется ни одного непустого *C*-кортежа; 2) останется один непустой *C*-кортеж; 3) оставшиеся *C*-кортежи образуют ортогональную *C*-систему. *Конец доказательства*.

Рассмотрим пример ортогонализации для приведенной в качестве

иллюстрации к теореме 2.20 D-системы $P = \begin{bmatrix} \{a,c\} & \{d\} & \{b,d\} \\ \varnothing & \{a,d\} & \{a,c\} \\ \{b,c\} & \varnothing & \{b\} \end{bmatrix}$, имеющей те же схему отношения и состав атрибутов, что и в предыдущем примере.

Используя для каждого D-кортежа теорему 3.1, найдем

$$P = \begin{bmatrix} \{a,c\} & * & * \\ \{b,d\} & \{d\} & * \\ \{b,d\} & \{a,b,c\} & \{b,d\} \end{bmatrix} \cap \begin{bmatrix} * & \{a,d\} & * \\ * & \{b,c\} & \{a,c\} \end{bmatrix} \cap \begin{bmatrix} \{b,c\} & * & * \\ \{a,d\} & * & \{b\} \end{bmatrix}.$$

Вычислим промежуточные результаты (пустые C-кортежи не показаны):

$$\begin{bmatrix} \{a,c\} & * & * \\ \{b,d\} & \{d\} & * \\ \{b,d\} & \{a,b,c\} & \{b,d\} \end{bmatrix} \cap \begin{bmatrix} * & \{a,d\} & * \\ * & \{b,c\} & \{a,c\} \end{bmatrix} = \begin{bmatrix} \{a,c\} & \{a,d\} & * \\ \{a,c\} & \{b,c\} & \{a,c\} \\ \{b,d\} & \{d\} & * \\ \{b,d\} & \{a\} & \{b,d\} \end{bmatrix}.$$

$$\begin{bmatrix} \{a,c\} & \{a,d\} & * \\ \{a,c\} & \{b,c\} & \{a,c\} \\ \{b,d\} & \{d\} & * \\ \{b,d\} & \{a\} & \{b,d\} \end{bmatrix} \cap \begin{bmatrix} \{b,c\} & * & * \\ \{a,d\} & * & \{b\} \end{bmatrix} = \begin{bmatrix} \{c\} & \{a,d\} & * \\ \{a\} & \{a,d\} & \{b\} \\ \{c\} & \{b,c\} & \{a,c\} \\ \{b\} & \{d\} & * \\ \{d\} & \{d\} & \{b\} \\ \{b\} & \{a\} & \{b,d\} \\ \{d\} & \{a\} & \{b\} \end{bmatrix}.$$

Получен результат, который по размерности незначительно улучшил результат, вычисленный без ортогонализации – матрица C-системы сократилась лишь на одну строку. Ниже мы рассмотрим некоторые другие методы сокращения размерности и трудоемкости вычислений при выполнении этой операции. Но само по себе получение ортогональной C-системы дает и другие преимущества. Например, можно легко вычислить количество элементарных кортежей, содержащихся в исходной D-системе. Для этого достаточно подсчитать их число в каждом C-кортеже и просто сложить их. Этого нельзя было сделать без ортогонализации, так как в C-системе имелись пары C-кортежей с непустым пересечением. Так, для данного примера, если учитывать, что мера каждой фиктивной компоненты равна 4, имеем: $8 + 2 + 4 + 4 + 1 + 2 + 1 = 22$. Кроме того, если заданы вероятностные меры компонент, то вероятность всей системы вычисляется суммированием всех

произведений мер компонент, содержащихся в каждом *C*-кортеже ортогональной *C*-системы.

Рассмотрим еще одно важное соотношение, позволяющее во многих случаях существенно сократить перебор при преобразовании АК-объектов в альтернативные классы. Дело в том, что *D*-кортеж, содержащий не менее двух непустых компонент, можно представить как ортогональную *C*-систему не единственным способом. Допустим, даны равенства

$$]A\ B\ C[= \begin{bmatrix} A & * & * \\ * & B & * \\ * & * & C \end{bmatrix} = \begin{bmatrix} A & * & * \\ \overline{A} & B & * \\ \overline{A} & \overline{B} & C \end{bmatrix}.$$

Если в промежуточном АК-объекте переставить местами *C*-кортежи, то эквивалентность не нарушится, но при ортогонализации после перестановки будет получено другое изображение того же *D*-кортежа. В итоге, для *D*-кортежа]*A B C*[формируется совокупность следующих равенств:

$$]ABC[= \begin{bmatrix} A & * & * \\ * & B & * \\ * & * & C \end{bmatrix} = \begin{bmatrix} * & B & * \\ * & * & C \\ A & * & * \end{bmatrix} = \begin{bmatrix} * & B & * \\ * & \overline{B} & C \\ A & \overline{B} & \overline{C} \end{bmatrix} = \begin{bmatrix} * & * & C \\ A & * & * \\ * & B & * \end{bmatrix} = \begin{bmatrix} * & * & C \\ A & * & \overline{C} \\ \overline{A} & B & \overline{C} \end{bmatrix} = \ldots$$

Таким образом, если задан *D*-кортеж, в котором содержится k непустых компонент, то алгоритм, описывающий получение из этого *D*-кортежа всех возможных эквивалентных ему ортогональных *C*-систем, состоит из трех шагов:

1) по теореме 2.19 перевести *D*-кортеж в *C*-систему *P* с k *C*-кортежами;

2) выбрать произвольную перестановку *C*-кортежей в *P* (число таких перестановок равно $k!$) и записать *P* в порядке, соответствующем этой перестановке;

3) для каждой неполной компоненты P_{ij} (i – номер строки, j – номер столбца) преобразованной *C*-системы *P* заменить все фиктивные компоненты P_{mj} ($m > i$), находящиеся ниже P_{ij}, ее дополнениями $\overline{P_{ij}}$.

Если целенаправленно использовать эти варианты преобразования, то можно существенно уменьшить как размерность вычисляемой *C*-системы путем сокращения числа входящих в нее *C*-кортежей, так и трудоемкость вычислений за счет того, что в промежуточных вычислениях образуется

значительно больше пустых *C*-кортежей, которые не участвуют в последующих вычислениях. Этот прием особенно эффективен при преобразовании *D*-систем в *C*-системы в рамках исчисления высказываний, но и для более широкого класса АК-объектов он также позволяет иногда получить существенное уменьшение трудоемкости. Для иллюстрации повторим ранее выполненное преобразование *D*-системы *P* в *C*-систему, но при этом первый и второй *D*-кортежи преобразуем в ортогональные *C*-системы иначе.

При анализе структуры исходной *D*-системы можно заметить, что в третьем столбце две компоненты $\{b,d\}$ и $\{a,c\}$ – взаимно дополнительны. *D*-кортежи, содержащие эти компоненты, преобразуем в *C*-системы и переставим в них *C*-кортежи так, чтобы данные компоненты оказались в первой строке. Выполним ортогонализацию полученных *C*-систем. Тогда в результате их пересечения получается большое число пустых *C*-кортежей. Для нашего примера:

$$P=\begin{bmatrix} * & * & \{b,d\} \\ \{a,c\} & * & \{a,c\} \\ \{b,d\} & \{d\} & \{a,c\} \end{bmatrix} \cap \begin{bmatrix} * & * & \{a,c\} \\ * & \{a,d\} & \{b,d\} \end{bmatrix} \cap \begin{bmatrix} \{b,c\} & * & * \\ \{a,d\} & * & \{b\} \end{bmatrix}.$$

Для первого пересечения имеем:

$$\begin{bmatrix} * & * & \{b,d\} \\ \{a,c & * & \{a,c\} \\ \{b,d\} & \{d\} & \{a,c\} \end{bmatrix} \cap \begin{bmatrix} * & * & \{a,c\} \\ * & \{a,d\} & \{b,d\} \end{bmatrix} = \begin{bmatrix} * & \{a,d\} & \{b,d\} \\ \{a,c\} & * & \{a,c\} \\ \{b,d\} & \{d\} & \{a,c\} \end{bmatrix}.$$

Для второго:

$$\begin{bmatrix} * & \{a,d\} & \{b,d\} \\ \{a,c\} & * & \{a,c\} \\ \{b,d\} & \{d\} & \{a,c\} \end{bmatrix} \cap \begin{bmatrix} \{b,c\} & * & * \\ \{a,d\} & * & \{b\} \end{bmatrix} = \begin{bmatrix} \{b,c\} & \{a,d\} & \{b,d\} \\ \{a,d\} & \{a,d\} & \{b\} \\ \{c\} & * & \{a,c\} \\ \{b\} & \{d\} & \{a,c\} \end{bmatrix}.$$

Нетрудно убедиться, что итоговая *C*-система, содержащая четыре *C*-кортежа, ортогональна, а по составу элементарных кортежей эквивалентна ранее полученной *C*-системе, содержащей семь *C*-кортежей. Методы и приемы эквивалентных преобразований, позволяющие уменьшать трудоемкость вычислений и размерность результата, будут более подробно описаны ниже. Вкратце методика сводится к тому, чтобы максимально увеличить количество

пустых кортежей, образующихся на первых и промежуточных этапах вычислений, путем соответствующего выбора вариантов ортогонализации *D*-кортежей в исходной *D*-системе – за счет этого существенно уменьшается число вариантов, подлежащих дальнейшей обработке.

3.2. Матричные свойства АК-объектов

Рассмотренные структуры АК (кортежи и системы кортежей) внешне напоминают матрицы. И хотя операции с АК-объектами существенно отличаются от операций алгебры матриц, они все же имеют многие свойства матричных структур. Матричные свойства АК-объектов можно использовать при разработке методов уменьшения трудоемкости вычислительных алгоритмов во многих программных системах, в том числе в системах с применением логических исчислений [*Кулик*, 1995; *Kulik*, 2010]. В данном разделе рассматриваются методы уменьшения трудоемкости при решении переборных задач (см. п. 1.3), многие из которых сводятся к задаче "Выполнимость КНФ" [*Гэри*, 1982]. Используя эти методы, во многих случаях при решении задач удается обойтись без трудоемкой операции преобразования АК-объектов в альтернативные классы.

Рассматриваемые здесь матричные свойства АК-объектов обобщают предложенные А.Д. Закревским методы анализа знаний на основе их представления в пространстве многозначных признаков [*Закревский*, 1995].

Пусть имеется произвольный *D*-кортеж $D =]s_1\ s_2\ ...\ s_n[$. Представим его как строку из *n* литер (литерами здесь являются имена компонент *D*-кортежа), которую произвольно разобьем на *R* непустых подстрок. Тогда любая подстрока отображает какой-то *D*-кортеж D_i с определенной схемой отношения. Если каждый D_i записать в схеме отношения *D*-кортежа *D*, добавив недостающие фиктивные компоненты, то такой кортеж будет состоять из компонент кортежа D_i в соответствующих атрибутах и пустых компонент во всех остальных атрибутах. Таким образом, каждой непустой компоненте *D*-кортежа D_i соответствуют пустые компоненты во всех остальных кортежах множества. Из этого, а также из теоремы 2.13, следует:

$$D = \bigcup_{i=1}^{R} D_i \qquad (3.1)$$

Например, если *D*-кортеж $]s_1\ s_2\ s_3\ s_4\ s_5\ s_6\ s_7[$ разбит на три *D*-кортежа $]s_2\ s_4\ s_6[$, $]s_1\ s_3\ s_5[$ и $]s_7[$, то после приведения их к общей схеме получим:

$]\varnothing\ s_2\ \varnothing\ s_4\ \varnothing\ s_6\ \varnothing[$;

$]s_1\ \varnothing\ s_3\ \varnothing\ s_5\ \varnothing\ \varnothing[$;

$]\varnothing\ \varnothing\ \varnothing\ \varnothing\ \varnothing\ \varnothing\ s_7[$.

Объединение этих *D*-кортежей равно исходному *D*-кортежу.

Аналогично для *C*-кортежей *C*, C_i при тех же условиях разбиения верно:

$C = \bigcap_{i=1}^{R} C_i$.

Пусть *T* – *D*-система, представляющая собой матрицу компонент размерности $M \times N$ (*M* строк и *N* столбцов). Разобьем эту *D*-систему с помощью вертикальных линий на *R* вертикальных блоков. Пусть $N(R)$ – множество $\{1, 2, ..., R\}$ индексов, а декартово произведение $(N(R))^M$ – множество последовательностей $S(R, M)$, элементами которого являются *M*-размещения с повторениями из множества $N(R)$. Так, если $N(R) = \{1, 2\}$, а $M = 3$, то $S(2, 3) = \{1, 2\}^3 = \{111, 112, 121, 122, 211, 212, 221, 222\}$.

Для этих условий справедливо следующее соотношение.

Теорема 3.3. При разбиении матрицы *D*-системы *T* размерности $M \times N$ на *R* вертикальных блоков ($R < N$) справедливо равенство

$$\boldsymbol{T} = \bigcup_{k \in S(R,M)} \left(\bigcap_{i=1}^{M} D_{ij}\right) \qquad \textbf{(3.2)}$$

где D_{ij} – *D*-кортеж, являющийся подстрокой *i*-ой строки, взятой из *j*-го блока матрицы *T*, а индекс *j* в подформуле $\bigcap_{i=1}^{M} D_{ij}$ пробегает все значения соответствующего элемента $k \in S(R, M)$.

Доказательство. Пусть *D*-кортеж D_i ($i = 1, 2, \ldots, M$) – *i*-ая строка матрицы *T*, тогда $T = \bigcap_{i=1}^{M} D_i$. При разбиении на *R* вертикальных блоков это равенство с использованием (3.1) преобразуется в равенство $T = \bigcap_{i=1}^{M} \left(\bigcup_{j=1}^{R} D_{ij}\right)$, где *D*-кортеж D_{ij} берется из подстроки *i*-ой строки, принадлежащей *j*-му блоку. Если в этом равенстве раскрыть скобки, то получим соотношение, сформулированное в

теореме. *Конец доказательства*.

Из теоремы 3.3, в частности, следует, что *D*-система непуста, если непуста хотя бы одна *D*-система $\bigcap_{i=1}^{M} D_{ij}$ при любом разбиении на вертикальные блоки.

Например, *D*-система $\begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \end{bmatrix}$ разбита линией на два вертикальных блока. По теореме ее можно представить как объединение восьми *D*-систем

$$(]t_{11} \quad \varnothing \quad \varnothing \quad \varnothing[\cup]\varnothing \quad t_{12} \quad t_{13} \quad t_{14}[) \cap (]t_{21} \quad \varnothing \quad \varnothing \quad \varnothing[\cup$$

$$]\varnothing \quad t_{22} \quad t_{23} \quad t_{24}[) \cap (]t_{31} \quad \varnothing \quad \varnothing \quad \varnothing[\cup]\varnothing \quad t_{32} \quad t_{33} \quad t_{34}[) = \begin{bmatrix} t_{11} \\ t_{21} \\ t_{31} \end{bmatrix} \cup$$

$$\begin{bmatrix} t_{11} & \varnothing & \varnothing & \varnothing \\ t_{12} & \varnothing & \varnothing & \varnothing \\ \varnothing & t_{32} & t_{33} & t_{34} \end{bmatrix} \cup \begin{bmatrix} t_{11} & \varnothing & \varnothing & \varnothing \\ \varnothing & t_{22} & t_{23} & t_{24} \\ t_{31} & \varnothing & \varnothing & \varnothing \end{bmatrix} \cup \begin{bmatrix} t_{11} & \varnothing & \varnothing & \varnothing \\ \varnothing & t_{22} & t_{23} & t_{24} \\ \varnothing & t_{32} & t_{33} & t_{34} \end{bmatrix} \cup$$

$$\begin{bmatrix} \varnothing & t_{12} & t_{13} & t_{14} \\ t_{21} & \varnothing & \varnothing & \varnothing \\ t_{31} & \varnothing & \varnothing & \varnothing \end{bmatrix} \cup \begin{bmatrix} \varnothing & t_{12} & t_{13} & t_{14} \\ t_{21} & \varnothing & \varnothing & \varnothing \\ \varnothing & t_{32} & t_{33} & t_{34} \end{bmatrix} \cup \begin{bmatrix} \varnothing & t_{12} & t_{13} & t_{14} \\ \varnothing & t_{22} & t_{23} & t_{24} \\ t_{31} & \varnothing & \varnothing & \varnothing \end{bmatrix} \cup \begin{bmatrix} t_{12} & t_{13} & t_{14} \\ t_{22} & t_{23} & t_{24} \\ t_{32} & t_{33} & t_{34} \end{bmatrix}.$$

Если непуста хотя бы одна из этих восьми *D*-систем, то непуста и исходная *D*-система.

Если в качестве блока матрицы *T* (см. формулу 3.2) выступает вектор-столбец, то его можно преобразовать в компоненту, значением которой является пересечение всех компонент вектора-столбца. В частности,

$$\begin{bmatrix} t_{11} \\ t_{21} \\ t_{31} \end{bmatrix} = t_{11} \cap t_{12} \cap t_{13.}$$

Это допустимо, так как компоненты относятся к одному атрибуту.

Возвращаясь к примеру, иллюстрирующему теорему 3.3, заметим, что если снова применить (3.2) к последнему АК-объекту разложения, то получим:

$$\begin{bmatrix} t_{12} & t_{13} & t_{14} \\ t_{22} & t_{23} & t_{24} \\ t_{32} & t_{33} & t_{34} \end{bmatrix} = \begin{bmatrix} t_{12} \\ t_{22} \\ t_{32} \end{bmatrix} \cup \ldots \cup \begin{bmatrix} t_{13} & t_{14} \\ t_{23} & t_{24} \\ t_{33} & t_{34} \end{bmatrix}.$$

Аналогично следует поступить с вновь полученным разложением.

Следовательно, непустота хотя бы одного из столбцов *D*-системы свидетельствует о непустоте всей системы.

Теорема 3.3 непосредственно пригодна для решения практических задач, но чаще применяются ее следствия, для рассмотрения которых введем некоторые обозначения и теоремы.

Конфликтующая пара *D*-системы есть непересекающаяся пара непустых компонент в одном атрибуте.

Конфликтный атрибут *D*-системы – это атрибут, в котором пересечение всех непустых компонент пусто, в противном случае атрибут называется ***бесконфликтным***.

Бесконфликтный блок *D*-системы (*BlNC*) – ее вертикальный блок, содержащий только бесконфликтные атрибуты. Соответственно вертикальный блок с конфликтными атрибутами называется ***конфликтным блоком*** (*BlC*). Из определений ясно, что в бесконфликтном блоке могут содержаться пустые компоненты, но в то же время пересечение всех непустых компонент в каждом атрибуте этого блока непусто.

Монотонным атрибутом *D*-системы является бесконфликтный атрибут, не содержащий пустых компонент.

Монотонным блоком *D*-системы (*BlM*) называется бесконфликтный блок, не содержащий *D*-кортежей со всеми пустыми компонентами.

Для *D*-системы *Q*, содержащей монотонный блок *BlM*(*Q*), построим *C*-кортеж C_{int}, в котором все компоненты проекции монотонного блока равны пересечению всех непустых компонент исходной *D*-системы *Q* в соответствующем атрибуте, а все остальные компоненты – фиктивные, т.е. равны "*".

Теорема 3.4. Если *D*-система *Q* содержит монотонный блок, то она непуста и $C_{int} \subseteq Q$.

Доказательство. По условиям теоремы $C_{int} \neq \varnothing$, и для каждого *D*-кортежа Q_i системы *Q* по теореме 2.16 соблюдается $C_{int} \subseteq Q_i$, так как в монотонном блоке всегда существует непустая компонента, такая что соответствующая компонента C_{int} включена в нее. Отсюда в силу теоремы 2.17 получаем $C_{int} \subseteq Q$. *Конец доказательства*.

В качестве примера возьмем *D*-систему

$$T = \left\rceil \begin{matrix} \{A,B\} & \{f,g\} & \{a,c,d\} \\ \varnothing & \{e\} & \{b,c\} \\ \{A\} & \{g,h\} & \varnothing \end{matrix} \right\lfloor .$$

Нетрудно убедиться, что первый и третий атрибуты в T бесконфликтны и вместе образуют монотонный блок. Соответственно $C_{int} = [\{A\} * \{c\}]$. Из теоремы 1.2 следует, что T – непуста и $C_{int} \subseteq T$, это легко проверяется с помощью теорем 2.16 и 2.17.

Рассмотрим теперь D-системы с бесконфликтными блоками, которые содержат D-кортежи со всеми пустыми компонентами. Пусть D-система Q содержит бесконфликтный блок $T_1 = BlNC(Q)$. Тогда после соответствующих перестановок строк и столбцов D-систему Q можно изобразить как блочную матрицу $T = \left\| \begin{matrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{matrix} \right\|$, где T_{11} – подматрица Q, не содержащая D-кортежей со всеми пустыми компонентами и являющаяся монотонным блоком, T_{21} – подматрица Q с D-кортежами, содержащими только пустые компоненты.

Теорема 3.5. Если в D-системе Q имеется бесконфликтный блок, то Q непуста, если и только если при разложении ее в блочную матрицу T непуста D-система, представленная блоком T_{22}.

Доказательство. Необходимость. Ясно, что одна из D-систем в формуле (3.2) из теоремы 3.3 может быть представлена блочной матрицей $\left\| \begin{matrix} T_{11} & \varnothing \\ \varnothing & T_{22} \end{matrix} \right\|$. Поскольку T_{11} монотонна, то существует непустой C-кортеж C_{11}, такой что $C_{11} \subseteq T_{11}$. Если T_{22} непуста, то в проекции, соответствующей T_{22}, имеется C-кортеж C_{22}, такой что $C_{22} \subseteq T_{22}$. При пересечении C_{11} и C_{22} образуется непустой C-кортеж C_0, так как всем нефиктивным компонентам кортежа C_{11} соответствуют фиктивные компоненты кортежа C_{22}, и наоборот. Согласно теоремам 2.16 и 2.17 $C_0 \subseteq T$. Достаточность. Предположим, что T_{22} пуста, а Q непуста. Тогда T_{22} эквивалентна D-системе той же размерности, все компоненты которой – пустые множества. Если вместо T_{22} вставить эту D-систему, то нижняя часть блочной матрицы будет содержать D-кортежи со всеми пустыми компонентами и, следовательно, соответствующая ей D-система окажется пустой. *Конец доказательства*.

Приведем пример. Дана D-система

$$Q = \left\rceil \begin{matrix} \{A,B\} & \{e,f\} & \{a,b\} & \varnothing \\ \varnothing & \{g,h\} & \varnothing & \{e\} \\ \{A\} & \{e\} & \{b\} & \{f,g\} \\ \varnothing & \{e,h\} & \varnothing & \{g,h\} \end{matrix} \right\lfloor .$$

Первый и третий столбцы в ней – бесконфликтные. Следовательно, после соответствующей перестановки столбцов и строк Q можно преобразовать в эквивалентную ей блочную матрицу:

$$Q_1 = \left\rceil \begin{matrix} \{A,B\} & \{a,b\} & \{e,f\} & \varnothing \\ \{A\} & \{b\} & \{e\} & \{f,g\} \\ \varnothing & \varnothing & \{g,h\} & \{e\} \\ \varnothing & \varnothing & \{e,h\} & \{g,h\} \end{matrix} \right\lfloor .$$

Для проверки непустоты этой D-системы достаточно проверить непустоту D-системы $T_{22} = \left\rceil \begin{matrix} \{g,h\} & \{e\} \\ \{e,h\} & \{g,h\} \end{matrix} \right\lfloor$. Первый столбец этой D-системы монотонный, следовательно, она непуста и содержит C-кортеж $C_{22} = [\{h\}\ *]$. Монотонный блок T_{11} содержит C-кортеж $[\{A\}\ \{b\}]$. Пересечение этих C-кортежей после приведения к одной схеме отношения равно C-кортежу $C_0 = [\{A\}\ \{b\}\ \{h\}\ *]$, он при перестановке атрибутов в соответствии со схемой отношения исходной D-системы Q преобразуется в C-кортеж $[\{A\}\ \{h\}\ \{b\}\ *]$, который и является выполняющей подстановкой D-системы Q. Проверить правильность этой подстановки можно с помощью теорем 2.16 и 2.17.

Из примера видно, что в одной D-системе соотношения, сформулированные в теоремах 3.4 и 3.5, можно использовать неоднократно и в ряде случаев находить решение за полиномиальное время. Очевидно, что алгоритм выявления бесконфликтных и монотонных блоков в произвольной D-системе полиномиально зависит от размерности этой D-системы. Причем оказывается, что при наличии монотонных блоков непустота D-системы проверяется алгоритмом, полиномиально зависящим от размерности матрицы D-системы, а при наличии бесконфликтных блоков можно для проверки непустоты использовать D-систему меньшей размерности.

Более подробное рассмотрение матричных свойств АК-объектов дано в следующих подразделах при описании алгоритмов решения конкретных задач, для реализации которых в общем случае требуется экспоненциальное время. В

частности, матричные свойства АК-объектов активно используются в алгоритмах решения задач проверки выполнимости КНФ и включения АК-объектов, например *C*-кортежа в *C*-систему.

3.3. Алгоритм проверки включения C-системы в C-систему

Этот алгоритм, как и его частный случай (проверка включения *C*-кортежа в *C*-систему), лежит в основе методов уменьшения трудоемкости многих не полиномиальных по вычислительной сложности алгоритмов. При общем подходе к решению этих задач требуется преобразовать вторую *C*-систему в *D*-систему (см. п. 2.2), чтобы использовать соотношения из теорем 2.16 и 2.17, которые позволяют осуществить проверку за время, полиномиально зависящее от размерности исходных АК-объектов. Однако при преобразовании АК-объекта в альтернативный класс не только требуется алгоритм из класса #*P*-полных (класс более трудный, чем *NP*), но еще может получиться АК-объект с размерностью, экспоненциальной относительно исходного АК-объекта. Рассмотрим менее трудоемкий алгоритм проверки включения *C*-кортежа в *C*-систему. Пусть заданы *C*-кортеж C и *C*-система R. Необходимо проверить выполнение соотношения $C \subseteq R$.

Алгоритм 3.1 (проверка включения C-кортежа в C-систему):

1) проверить, существует ли в R такой *C*-кортеж C_i, для которого $C \subseteq C_i$. Если существует, то выход с ответом "да";

2) вычислить $R_0 = C \cap R$;

3) если $R_0 = \varnothing$, то выход с ответом "нет";

4) если R_0 содержит один *C*-кортеж C_i, то выход с ответом "нет";

5) проверить, существуют ли в R_0 атрибуты, у которых все компоненты равны между собой и равны соответствующим компонентам *C*-кортежа C. Если такие атрибуты существуют, то удалить их из R_0 и из C (т.е. вычислить проекции этих АК-объектов по остальным атрибутам), в результате чего будут получены АК-объекты R_1 и C_1;

6) считая полученный на шаге 5 *C*-кортеж C_1 универсумом отношения R_1, вычислить *D*-систему $\overline{R_1}$.

7) проверить непустоту полученной *D*-системы $\overline{R_1}$, при возможности

используя соотношения, приведенные в разделе 3.2. Если *D*-система пуста, то ответ "да", в противном случае – "нет".

Ясно, что все шаги этого алгоритма, за исключением шага 7, полиномиальны относительно размерностей исходных объектов. Шаг 7 может потребовать экспоненциального времени выполнения, но при этом на шагах 1, 3 и 4 возможен выход из алгоритма с однозначным ответом, а на шагах 2 и 5 возможно сокращение размерности исходных АК-объектов.

Заметим, что проверка включения элементарного кортежа в *C*-систему сложностей не вызывает: для проверки достаточно найти пересечение этого кортежа с *C*-системой и, если получим пустое множество, то ответ отрицательный, в противном случае – положительный. Также несложно получить ответ, если *C*-кортеж содержит небольшое число элементарных кортежей. В то же время в системах большой размерности и в случае, когда большое число элементарных кортежей проверяемого *C*-кортежа распределены по *C*-кортежам проверяемой *C*-системы, для проверки требуется большое число операций. Приведем пример. Пусть *C*-система

$$R[XYZV] = \begin{bmatrix} \{a,d\} & \{a,b\} & \{b,c\} & \{d\} \\ \{c,d\} & \{b.d\} & \{a,b\} & * \\ \{d\} & \{a,d\} & \{b,d\} & \{a,b\} \\ \{a,b\} & \{c\} & \{b\} & * \end{bmatrix}$$

задана в универсуме $X \times Y \times Z \times V = \{a, b, c, d\}^4$ и в этой же схеме отношения сформирован *C*-кортеж $C = [\{a, b, c\} * \{b\} \{a, d\}]$. Проверить, выполняется ли включение $C \subseteq R$.

Шаг 1. Условие не выполнено.

Шаг 2. $R_0 = \begin{bmatrix} \{a\} & \{a,b\} & \{b\} & \{d\} \\ \{c\} & \{b,d\} & \{b\} & \{a,d\} \\ \{a,b\} & \{c\} & \{b\} & \{a,d\} \end{bmatrix}$.

Шаги 3 и 4. Условия не выполняются.

Шаг 5. Условие выполнено. Вычисляем проекции

$$R_1 = R_0[XYV] = \begin{bmatrix} \{a\} & \{a,b\} & \{d\} \\ \{c\} & \{b,d\} & \{a,d\} \\ \{a,b\} & \{c\} & \{a,d\} \end{bmatrix}; \; C_1 = C[XYV] = [\{a, b, c\} * \{a, d\}].$$

Шаг 6. Относительно универсума [{*a*, *b*, *c*} {*a*, *b*, *c*, *d*} {*a*, *d*}]

$$R_2 = \overline{R_1[XYV]} = \begin{bmatrix} \{b,c\} & \{c,d\} & \{a\} \\ \{a,b\} & \{a,c\} & \varnothing \\ \{c\} & \{a,b,d\} & \varnothing \end{bmatrix}.$$

Шаг 7. У *D*-системы R_2 третий атрибут бесконфликтный, поэтому в соответствии с теоремой 3.5. достаточно проверить непустоту *D*-системы $\begin{bmatrix} \{a,b\} & \{a,c\} \\ \{c\} & \{a,b,d\} \end{bmatrix}$. У этой *D*-системы второй атрибут монотонный, поэтому *D*-система непуста и, следовательно, $C \subseteq R$ не выполняется.

Пусть *D*-система *R* задана в универсуме $\{0, 1\}^n$, где *n* – число атрибутов или переменных в соответствующей формуле исчисления высказываний. В такой модели алгоритм проверки включения *C*-кортежа в *C*-систему имеет некоторые особенности по сравнению с общим алгоритмом. Пусть заданы в универсуме $\{0, 1\}^n$ *C*-кортеж *C* и *C*-система *R*.

Алгоритм 3.2 (проверка включения C-кортежа в C-систему в универсуме* {0, 1}^{n}*):

1) проверить, существует ли в *R* такой *C*-кортеж C_i, для которого $C \subseteq C_i$. Если существует, то выход с ответом "да";

2) вычислить $R_0 = C \cap R$;

3) если $R_0 = \varnothing$, то выход с ответом "нет";

4) если R_0 содержит один *C*-кортеж, то выход с ответом "нет";

5) в *C*-системе R_0 выделить проекцию R_0^*, соответствующую фиктивным атрибутам *C*-кортежа *C*;

6) проверить непустоту полученной *D*-системы $\overline{R_0^*}$. Если *D*-система пуста, то ответ "да", в противном случае – "нет".

Рассмотрим пример. Здесь для экономии места компоненты {0} и {1} записываются в АК-объектах как 0 и 1 соответственно. Пусть заданы $C = [0 * 1 * 0]$ и

$$R = \begin{bmatrix} * & 1 & * & * & 0 \\ 0 & 1 & 0 & * & * \\ * & 0 & * & 1 & 0 \\ 0 & * & 1 & 1 & * \end{bmatrix}.$$

Шаг 1. Условие не выполнено.

Шаг 2. $R_0 = \begin{bmatrix} 0 & 1 & 1 & * & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & * & 1 & 1 & 0 \end{bmatrix}$.

Шаги 3 и 4. Условия не выполнены.

Шаг 5. $R_0^* = \begin{bmatrix} 1 & * \\ 0 & 1 \\ * & 1 \end{bmatrix}$.

Шаг 6. $\overline{R_0^*} = \begin{bmatrix} 0 & \varnothing \\ 1 & 0 \\ \varnothing & 0 \end{bmatrix}$; система непуста, ответ – "нет".

Обоснование корректности алгоритма. Шаги 1, 2 и 3 очевидны. Шаг 4: если условие не выполнено на шаге 1, то для *C*-системы R_0, содержащей единственный *C*-кортеж, возможен только один вариант: $R_0 \subset C$, в силу чего неверно $C \subseteq R_0$. Шаги 5 и 6: после шага 2 во всех атрибутах R_0, совпадающих с нефиктивными атрибутами *C*, компоненты равны соответствующим компонентам *C*. В оставшихся атрибутах, чтобы выполнялось $C \subseteq R_0$, должен содержаться частный универсум. Следовательно, $\overline{R_0^*}$ пусто.

Стоит отметить, что в случае, когда *C* – элементарный кортеж или может быть разложен на небольшое число элементарных кортежей (т.е. число фиктивных атрибутов сравнительно невелико), то для проверки включения достаточно ограничиться шагом 1 алгоритмов 3.1 и 3.2.

На основе этих алгоритмов легко построить алгоритм проверки включения одной *C*-системы в другую (проверяется включение каждого *C*-кортежа первой *C*-системы во вторую).

3.4. Алгоритм решения задачи "Выполнимость КНФ"

Важность задачи "Выполнимость КНФ" в системах искусственного интеллекта обусловлена следующими причинами.

Во-первых, это основная *NP*-полная задача в теории сложности вычислений, поскольку доказано, что любая задача класса *NP* при

использовании рациональной системы кодирования может быть представлена в виде некоторой КНФ, причем преобразование условий любой задачи класса *NP* в КНФ занимает полиномиальное время.

Во-вторых, на основе теоремы 1.6 как для исчисления высказываний, так и для исчисления предикатов был сформулирован принцип резолюции, который широко применяется в машинных реализациях систем искусственного интеллекта. Согласно этому принципу формула $F_1 \wedge ... \wedge F_n \wedge \neg G$ преобразуется в КНФ, и логический вывод сводится к решению задачи выполнимости КНФ.

Таким образом, задача выполнимости КНФ занимает ведущее место в практических приложениях теории логического вывода и в теории сложности вычислений, а выявление новых классов КНФ с полиномиально распознаваемым свойством выполнимости представляется актуальным направлением исследований.

В АК любая *D*-система изоморфна некоторой КНФ, и задача выполнимости сводится к ответу на вопрос о пустоте или непустоте соответствующей *D*-системы. Один из возможных вариантов ответа на этот вопрос – преобразование *D*-системы в альтернативный класс с получением *C*-системы, описывающей множество выполняющих подстановок КНФ. Но этот вариант нерационален, особенно в случаях, когда *D*-система не пуста, так как здесь определяется множество всех элементарных кортежей, в ней содержащихся, хотя достаточно найти единственный.

Предлагается использовать следующий метод. На первом этапе с помощью соотношений, изложенных в подразделе 3.2, выявляется существование монотонных (при их наличии задача решается за полиномиальное время) или бесконфликтных (что позволяет уменьшить размерность *D*-системы) блоков. Когда эти способы исчерпаны, можно выполнить задачу преобразования *D*-системы в *C*-систему, но изменить порядок выполнения операций. Обычно вначале все *D*-кортежи исходной *D*-системы преобразуются в ортогональные *C*-системы, после чего последовательно находится пересечение этих *C*-систем. Для решения задачи выполнимости вместо поиска пересечения промежуточных *C*-систем в целом можно искать последовательное пересечение *C*-кортежей из разных *C*-систем до тех пор, пока не придем к последней *C*-системе или не прервем операцию после получения пустого *C*-кортежа. В последнем случае выбирается другая последовательность операций, и так до тех пор, пока не

доказан положительный результат или не перебраны все возможные сочетания пересечений *C*-кортежей, чтобы убедиться в пустоте исходной *D*-системы. Такой метод значительно сокращает трудоемкость алгоритма решения задачи, особенно когда осуществляется ортогонализация и используются определенные критерии, позволяющие за счет изменения порядка выполнения операций получать большое количество отрицательных результатов (пустых *C*-кортежей) на первых этапах вычислений [*Кулик*, 1993].

Здесь рассматривается еще один во многих случаях более эффективный вариант алгоритма решения задачи выполнимости [*Кулик*, 1995], где используется алгоритм проверки включения *C*-системы в *C*-систему, изложенный в предыдущем разделе.

Алгоритм 3.3. (проверка выполнимости D-системы):

Пусть R – *D*-система, в которой отсутствуют монотонные и бесконфликтные блоки и содержится множество *D*-кортежей $\{D_1, D_2, ..., D_m\}$. Очевидно, что $R = D_1 \cap D_2 \cap ... \cap D_m$.

1. Выберем какой-либо *D*-кортеж (например, D_1) и выполним следующее преобразование:

$$D_1 \cap D_2 \cap ... \cap D_m = D_1 \cap \overline{\overline{D_2} \cup ... \cup \overline{D_m}} = D_1 \setminus (\overline{D_2} \cup ... \cup \overline{D_m}).$$

Подвыражение $R_1 = \overline{D_2} \cup ... \cup \overline{D_m}$ есть *C*-система, равная дополнению *D*-системы $D_2 \cap ... \cap D_m$.

2. Выбранный *D*-кортеж D_1 преобразуем по теореме 3.1 в ортогональную *C*-систему R_0. Тогда исходную *D*-систему можно записать в виде $R_0 \setminus R_1$.

3. Для проверки выполнимости R достаточно проверить включение $R_0 \subseteq R_1$. При положительном ответе $R = \varnothing$, а при отрицательном – $R \neq \varnothing$.

Предположим, что на шаге 3 проверка включения некоторого *C*-кортежа C_i из R_0 в *C*-систему R_1 производится по алгоритму 3.1 (или 3.2) из подраздела 3.3. Если при этом достигнут последний шаг, то необходимо проверить выполнимость промежуточной *D*-системы R_k, имеющей меньшую размерность, чем исходная *D*-система R. Для проверки можно снова применить алгоритм 3.3, используя показанное выше преобразование $R_k = R_{k0} \setminus R_{k1}$.

Ясно, что при совестном применении алгоритмов 3.3 и 3.1 (или 3.2) осуществляется обход дерева решения. Процедура завершается, если

(i) проверка включения хотя бы одного C-кортежа C_j, содержащегося в промежуточной C-системе R_{k0}, в C-систему R_{k1} окажется безуспешной – в этом случае исходная D-система выполнима;

(ii) все проверки оказались успешными – D-система невыполнима.

Докажем корректность алгоритма. Если в каком-либо узле дерева решения подтверждается включение C-кортежа из R_{k0} в C-систему R_{k1}, то в одном из C-кортежей C-системы R_0 содержится по крайней мере один элементарный кортеж, не входящий в R_1. Отсюда следует, что $R_0 \setminus R_1 \neq \varnothing$ и соответственно $R \neq \varnothing$. С другой стороны, если все возможные проверки успешны, то $R_0 \setminus R_1 = R = \varnothing$.

В качестве примера определим выполнимость D-системы

$$R[XYZ] = \begin{bmatrix} \{c\} & \varnothing & \{b\} \\ \{a\} & \{b\} & \{a,c\} \\ \{c\} & \{a,c\} & \varnothing \\ \{b\} & \varnothing & \varnothing \\ \varnothing & \{a\} & \varnothing \end{bmatrix},$$

заданной в универсуме $\{a, b, c\}^3$. Пусть $R_0 =]\{c\}\ \varnothing\ \{b\}[= \begin{bmatrix} \{c\} & * & * \\ \{a,b\} & * & \{b\} \end{bmatrix}$ (с учетом ортогонализации). Тогда

$$R_1 = \begin{bmatrix} \{b,c\} & \{a,c\} & \{b\} \\ \{a,b\} & \{b\} & * \\ \{a,c\} & * & * \\ * & \{b,c\} & * \end{bmatrix}.$$

Найдем сначала включение $C_1 = [\{c\} * *]$ в R_1. Здесь уже на шаге 1 алгоритм завершается успешно, так как по теореме 2.3 C_1 включен в третий C-кортеж C-системы R_1.

Проверим включение $C_2 = [\{a, b\} * \{b\}]$ в R_1.

Шаг 1. Условие не выполнено.

Шаг 2. $R_2 = C_2 \cap R_1 = \begin{bmatrix} \{b\} & \{a,c\} & \{b\} \\ \{a,b\} & \{b\} & \{b\} \\ \{a\} & * & \{b\} \\ \{a,b\} & \{b,c\} & \{b\} \end{bmatrix}.$

Шаги 3 и 4. Условия не выполнены.

Шаг 5. Условия выполнены. $C_2[XY] = [\{a, b\}\ *]$; $R_2[XY] = \begin{bmatrix} \{b\} & \{a,c\} \\ \{a,b\} & \{b\} \\ \{a\} & * \\ \{a,b\} & \{b,c\} \end{bmatrix}$.

Шаг 6. $C_2 \setminus R_2 = \begin{bmatrix} \{a\} & \{b\} \\ \varnothing & \{a,c\} \\ \{b\} & \varnothing \\ \varnothing & \{a\} \end{bmatrix}$.

Шаг 7. Здесь уже в силу малой размерности *D*-системы R_2 можно для проверки ее выполнимости воспользоваться алгоритмом преобразования *D*-системы в *C*-систему. Тогда получим:

$$R_2 = \begin{bmatrix} \{a\} & * \\ * & \{b\} \end{bmatrix} \cap [*\ \{a, c\}] \cap [\{b\}\ *] \cap [*\ \{a\}] = \varnothing.$$

Отсюда следует, что $C_2 \subseteq R_2$, а $R_0 \subseteq R_1$ и, значит, *D*-система *R* пуста.

Рассмотрим некоторые методы ускорения алгоритма 3.3.

Проверка на сокращаемость. Сократить размерность матрицы *D*-системы *R* возможно за счет распознавания бесконфликтных проекций (см. теорему 3.4) или же включенных в нее *D*-кортежей, содержащих только одну непустую компоненту (строки матрицы с этими *D*-кортежами могут быть удалены). В таком случае размерность сокращается без разветвления дерева решения.

Проверка на полиномиальную разрешимость. Если промежуточная *D*-система разрешима за полиномиальное время, то данный узел дерева решения будем рассматривать как *лист* (т.е. завершающий узел). Допустимо применять, в частности, следующие критерии полиномиальной разрешимости:

а) существование монотонной проекции (см. теорему 3.3);

б) существование *D*-кортежа со всеми пустыми компонентами;

в) наличие во всех *D*-кортежах не более двух непустых компонент (это соответствует полиномиально разрешимой задаче 2-выполнимости [*Гэри*, 1982]);

г) возможность полиномиального преобразования в хорновскую ДНФ, то есть в ДНФ, содержащую не более одного позитивного литерала в каждом дизъюнкте. Применимы и другие известные критерии полиномиальной разрешимости [*Данцин*, 1987].

Выбор ведущего D-кортежа. Пусть алгоритмы 3.3 и 3.1 (или 3.2) применяются совместно. Тогда при выполнении шага 2 алгоритма проверки включения *C*-кортежа в *C*-систему существенное сокращение размерности результата пересечения *C*-кортежа и *C*-системы обеспечивается предварительной ортогонализацией. Уменьшение размерности результата достигается с помощью соответствующего выбора *D*-кортежа D_{k1}, преобразуемого в *C*-систему R_{k0}, т.е. *ведущего D-кортежа*. Пусть в атрибуте X_i *D*-системы R_k содержится $K_i(1)$ компонент со значением {1} и $K_i(0)$ компонент со значением {0}. Если в ведущем *D*-кортеже атрибут X_i не пуст и равен $t \in \{0, 1\}$, то D_{k1} преобразуется в ортогональную *C*-систему R_{k0} таким образом, чтобы все *C*-кортежи в R_{k0} не содержали фиктивных компонент в атрибуте X_i. В таком случае R_{k0} включает один *C*-кортеж со значением t в атрибуте X_i и $m - 1$ *C*-кортежей со значением $\neg t$ в том же атрибуте. Если, допустим, $t = 1$, то при пересечении *C*-систем $R_{k0} \cap \overline{R_{k1}}$ получаются пустыми по крайней мере $K_i(0) + (m - 1)K_i(1)$ *C*-кортежей. При учете данного свойства удается подбором ведущих *D*-кортежей и выбором метода их ортогонализации построить стратегию вычислений, существенно уменьшающую число необходимых операций за счет элиминации большого числа пустых *C*-кортежей.

Одно из приложений алгоритма распознавания выполнимости КНФ в исчислении высказываний – доказательство полиномиальной разрешимости некоторых не известных ранее классов КНФ. Пока что не проводились достаточно представительные исследования по оценке эффективности алгоритма 3.3 в общем случае, но для систем, изоморфных формулам исчисления высказываний, были получены некоторые интересные результаты. Доказано, что если КНФ выражена как *D*-система, где содержатся только три равномерно распределенные (с вероятностью 1/3) символа: 1, 0 и ∅, такая КНФ разрешима в среднем за полиномиальное время. Даже без применения рассмотренных выше методов ускорения среднее время решения не превосходит полинома 3-й степени от числа конъюнктов [*Кулик*, 1995].

3.5. Алгоритмы выполнения кванторных операций

Из теорем 2.23 и 2.24 следует, что операция $\exists x(R)$ легко осуществима, если АК-объект R представляет собой *C*-систему, а операция $\forall x(R)$ – если R есть

D-система. В то же время нередко встречаются примеры, где АК-объекты с кванторами не удовлетворяют этим условиям. Такие АК-объекты требуется преобразовывать в альтернативный класс, что часто приводит к значительным затратам времени и памяти. В настоящем разделе описываются алгоритмы решения подобных задач, во многих случаях позволяющие значительно сократить требуемые вычислительные ресурсы.

Пусть задана формула $\forall x(R)$, где R – C-система, содержащая атрибут X. Рассмотрим проекцию $-X(R)$. Каждому элементарному кортежу t_i в этой проекции соответствует некоторое множество элементов из атрибута X. Если в $-X(R)$ существует хотя бы один элементарный кортеж t_s, которому в R соответствует множество всех элементов домена атрибута X, то отношение $\forall x(R)$ непусто. И наоборот, если такого элементарного кортежа не существует, то $\forall x(R) = \varnothing$. Следовательно, для вычисления формулы $\forall x(R)$, где R – C-система, необходимо найти все элементарные кортежи типа t_s в проекции $-X(R)$ или убедиться в их отсутствии.

Часть задачи решается просто, если в C-системе R существуют C-кортежи с фиктивной компонентой в атрибуте X. Тогда ясно, что $\forall x(R) \neq \varnothing$, а для C-системы R_*, сформированной из этих C-кортежей, выполняется $R_* \subseteq \forall x(R)$. Для полного решения задачи необходимо в оставшейся C-системе $Q = R \setminus R_*$ найти множество элементарных кортежей из проекции $-X(Q)$, которым в Q соответствует множество всех значений атрибута X. Поиск таких кортежей основан на следующем соотношении.

Теорема 3.6. Если в C-системе Q существует атрибут X, такой что пересечение всех C-кортежей в проекции $-X(Q)$ непусто и равно C-кортежу C, то при добавлении в C атрибута X с компонентой, равной объединению всех компонент этого атрибута в Q, образуется C-кортеж C_1, такой что $C_1 \subseteq Q$.

Доказательство. При пересечении $C_1 \cap Q$ получается C-система, у которой во всех C-кортежах проекции $-X(Q)$ компоненты одинаковы и совпадают с соответствующими компонентами C-кортежа C, а в проекции $[X]$ совпадают с компонентами этого атрибута в Q. Такие C-кортежи можно объединить в один C-кортеж, у которого компонента в атрибуте X равна объединению всех компонент этого атрибута в Q. Тогда полученный C-кортеж равен C_1. Из равенства $C \cap Q = C$ следует $C \subseteq Q$. *Конец доказательства*.

Перейдем к алгоритму. В *C*-системе Q необходимо найти такие множества *C*-кортежей, у которых объединение компонент в атрибуте X равно домену этого атрибута. Перенумеруем все *C*-кортежи в Q номерами из $J = \{1, 2, ..., n\}$ и выделим проекцию $Q[X]$, которая будет представлять некоторую систему подмножеств $\{s_i\}$ множества X, где индекс $i \in J$ совпадает с номером соответствующего *C*-кортежа из Q. Ясно, что $\bigcup_{i=1}^{n} s_i \neq X$ означает невозможность существования хотя бы одного *C*-кортежа в Q, удовлетворяющего условиям поиска, так как в этом случае ни один элементарный кортеж из проекции $–X(Q)$ не может содержать в атрибуте X всех его элементов.

Если же $\bigcup_{i=1}^{n} s_i = X$, то в системе множеств $\{s_i\}$ существует некоторая совокупность минимальных покрытий. ***Минимальное покрытие*** системы $\{s_i\}$ на носителе X есть такая совокупность $\{s_k\}$ множеств из $\{s_i\}$, объединение которых равно X, и при этом удаление любого множества из $\{s_k\}$ приводит к тому, что объединение оставшихся подмножеств из $\{s_k\}$ становится строгим подмножеством X (т.е. перестает быть покрытием X).

Пусть J_k – некоторое подмножество индексов, содержащее номера всех компонент минимального покрытия $\{s_k\}$. Если из *C*-системы Q выбрать все *C*-кортежи с номерами, принадлежащими J_k, то из них можно сформировать *C*-систему Q_k, обладающую следующим свойством: если найти пересечение всех *C*-кортежей проекции $–X(Q_k)$, то в случае непустого пересечения получим *C*-кортеж C_k, который при добавлении фиктивного атрибута X удовлетворяет соотношению $C_k \subseteq Q_k$.

С учетом этого можно построить следующий алгоритм.

Алгоритм 3.4 (вычисление $\forall x(R)$ ***для C-системы***):

1) сформировать пустое множество A, в котором накапливаются *C*-кортежи для $\forall x(R)$;

2) из *C*-системы R извлечь *C*-систему R_* с фиктивными компонентами в атрибуте X и выполнить $A \cup R_*$;

3) сформировать $Q = R \setminus R_*$;

4) из проекции $Q[X]$ сформировать систему множеств $\{s_i\}$; $i \in J$;

5) если $\bigcup_{i \in J} s_i \neq X$, то конец алгоритма;

6) для системы $\{s_i\}$ найти все минимальные покрытия $\{s_k\}$ и соответствующие им множества индексов J_k;

7) для каждого J_k вычислить, используя теорему 2.2, пересечение соответствующих C-кортежей из проекции $-X(Q_k)$; если полученный C-кортеж C_k не пуст, то добавить в C_k фиктивный атрибут X и вычислить $A \cup C_k$;

8) конец алгоритма.

Шаг 6 алгоритма характеризуется экспоненциальной сложностью, так как задача поиска минимального покрытия в произвольной системе множеств относится к *NP*-полным [*Гэри*, 1982]. Однако во многих случаях алгоритм ее решения намного проще, чем прямой алгоритм, где вне зависимости от особенностей частных задач требуется заведомо экспоненциальное преобразование C-системы в D-систему. К тому же при использовании этого алгоритма решение получается в виде C-системы, непустота и состав элементарных кортежей которой выявляются непосредственно.

Рассмотрим пример использования этого алгоритма. Пусть в универсуме $X \times Y \times Z \times V = \{a, b, c, d\}^4$ задана C-система

$$R = \begin{bmatrix} \{a,d\} & \{a,b,c\} & \{b,c\} & \{d\} \\ \{c,d\} & \{b,d\} & \{a,b\} & * \\ \{b\} & \{a,d\} & \{b,d\} & \{a,b\} \\ \{a,b\} & * & \{b\} & \{b,c,d\} \end{bmatrix}.$$

Необходимо вычислить $\forall y(R)$.

Шаг 1. $A = \varnothing$.

Шаг 2. $A = [\{a, b\} * \{b\} \ \{b, c, d\}]$.

Шаг 3. $Q = \begin{bmatrix} \{a,d\} & \{a,b,c\} & \{b,c\} & \{d\} \\ \{c,d\} & \{b,d\} & \{a,b\} & * \\ \{b\} & \{a,d\} & \{b,d\} & \{a,b\} \end{bmatrix}$.

Шаг 4. $\{s_i\} = \{\{a, b, c\}, \{b, d\}, \{a, d\}\}$; $J = \{1, 2, 3\}$.

Шаг 5. $\bigcup_{i \in J} s_i = \{a, b, c, d\} = Y$ – равенство подтверждается.

Шаг 6. $\{s_m\} = \{\{a, b, \ c\}, \{b, d\}\}$; $J_m = \{1, 2\}$;

$\{s_n\} = \{\{a, b, c\}, \{a, d\}\}$; $J_n = \{1, 3\}$;

других минимальных покрытий нет.

Шаг 7. В проекции $Q[XZV] = \begin{bmatrix} \{a,d\} & \{b,c\} & \{d\} \\ \{c,d\} & \{a,b\} & * \\ \{b\} & \{b,d\} & \{a,b\} \end{bmatrix}$ вычислим

$Q_1 \cap Q_2 = [\{d\}\ \{b\}\ \{d\}]$;

$A = \begin{bmatrix} \{a,b\} & * & \{b\} & \{b,c,d\} \\ \{d\} & * & \{b\} & \{d\} \end{bmatrix}$;

$Q_1 \cap Q_3 = \varnothing$; A – без изменений.

В качестве упражнения читателю предлагается решить эту задачу с помощью прямого алгоритма и сравнить полученные ответы, а заодно и трудоемкости их выполнения.

Алгоритм для вычисления $\exists x(P)$, где P – D-система, строится как двойственный алгоритму $\forall x(R)$, где R – C-система, но можно использовать рассмотренный выше алгоритм непосредственно, если сначала применить равенство $\overline{\exists x(P)} = \forall x(\overline{P})$. В этом случае $\overline{P}$ согласно теореме 2.10 есть C-система. Тогда после получения результата операции $\forall x(\overline{P})$ по рассмотренному алгоритму вычисляется дополнение результата.

Рассмотрим теперь формулы с префиксами, содержащими более одной кванторной операции. В общем случае, когда в формуле задана последовательность кванторных приставок, то любая кванторная приставка префикса находится в зоне действия всех кванторов, расположенных левее. Например, формула $\exists x \forall y \exists z(F)$ интерпретируется как формула $\exists x(\forall y(\exists z(F)))$. Поэтому перестановки кванторных приставок в формуле, вообще говоря, искажают значение формулы. Например, если предикат $P(x, y)$ соответствует отношению "человек x назван именем y", то формула $\forall x \exists y(P(x, y))$ переводится как "каждый человек назван по крайней мере одним именем", а формула $\exists y \forall x(P(x, y))$ – как "существует имя, которым назван каждый человек".

Стоит отметить, что запрет на перестановки относится только к различающимся кванторам. Если же префикс содержит последовательности идущих подряд кванторов одного и того же типа, то перестановка кванторных приставок в пределах этих групп не меняет значения формулы. Значит, при наличии сложных префиксов с кванторами, последовательность кванторных операций должна учитывать эти особенности.

3.6. Оценка вычислительной сложности алгоритмов

Как показано в п. 2.4, структуры АК полиномиально сводимы к логическим структурам, из чего следует, что ***вычислительная сложность алгоритмов на структурах АК полностью соответствует вычислительной сложности алгоритмов решения типовых задач, заданных на логических структурах.*** Значительная часть таких задач (например, задача выполнимости КНФ), возникающих в логическом анализе при организации процедур вывода, относится по вычислительной сложности к классу *NP*-полных, то есть приводит к алгоритмам экспоненциальной сложности. Однако имеется немало частных случаев этих задач, которые решаются за полиномиальное время. Для задачи выполнимости КНФ к таковым относятся КНФ, в которых каждый дизъюнкт имеет не более двух литералов, или КНФ, содержащие только хорновские дизъюнкты. Выявление частных случаев с полиномиально распознаваемым свойством выполнимости имеет большое значение в прикладных исследованиях, поскольку способствует сокращению времени работы соответствующих алгоритмов.

Известные частные случаи могут быть интерпретированы и в структурах АК, но здесь появляются дополнительные средства снижения трудоемкости, которые были рассмотрены в предыдущих разделах.

Особенность АК состоит в том, что операции и сравнения с АК-объектами сводятся к операциям и сравнениям с множествами, являющимися компонентами этих АК-объектов. Во многих приложениях компоненты представляют собой конечные множества либо могут быть сведены к системам конечных множеств (например, системы из конечного числа интервалов, заданных на действительной числовой оси). С учетом этого для оценки вычислительной сложности алгоритмов в среде АК примем сложность операций с компонентами равной некоторой константе ***C***. Тогда можно считать, что вычислительная сложность всех операций и сравнений над АК-объектами зависит только от размерностей участвующих в операциях и сравнениях АК-объектов. В табл. 3.1 приведены различные сочетания АК-объектов, знаком "+" помечены сочетания, для которых алгоритмы выполнения соответствующих операций полиномиальны при условии, что все домены атрибутов есть простые множества (а не многоместные отношения).

Пусть размерности пары АК-объектов R_1 и R_2, приведенных к однотипной схеме отношения, равны соответственно $m_1 \times n$ и $m_2 \times n$, где n – число атрибутов, а m_1 и m_2 – число кортежей в R_1 и R_2 соответственно. Тогда очевидно, что все операции и сравнения с парами АК-объектов, предусмотренные в соотношениях теорем 2.2 – 2.18 п. 2.1, реализуются с вычислительной сложностью, не превосходящей величины $\boldsymbol{C}n^2 m_1 m_2$. Таким образом, при принятых допущениях эти операции и сравнения полиномиальны. Вычисления результатов кванторных операций в случаях, предусмотренных в теоремах 2.23 и 2.24, также являются полиномиальными и сводятся к проверке непустоты (или неполноты – для теоремы 2.24) компонент определенного столбца матрицы и к переписыванию значений компонент этого столбца.

Таблица 3.1.

Действие	***C*-кортеж**	***C*-система**	***D*-кортеж**	***D*-система**
Проверка принадлежности заданного элементарного кортежа в	+	+	+	+
Проверка включения *C*-кортежа в	+		+	+
Проверка включения *C*-системы в	+		+	+
Проверка включения *D*-кортежа в	+		+	+
Проверка включения *D*-системы в				
Кванторная операция $\forall x$	+		+	+
Кванторная операция $\exists x$	+	+	+	

Из табл. 3.1 видно, что вычислительная сложность операций зависит от класса структур используемых при этом АК-объектов. Например, проверка включения *C*-кортежа в *C*-систему выполняется в общем случае с помощью

алгоритма экспоненциальной вычислительной сложности, в то время как алгоритм проверки включения *C*-кортежа и даже *C*-системы в *D*-систему имеет полиномиальную сложность. Операция дополнения АК-объекта во всех случаях требует алгоритмов полиномиальной вычислительной сложности, но при этом система преобразуется в альтернативный класс. Операции пересечения и объединения АК-объектов, относящихся к одному классу, выполняются алгоритмами полиномиальной сложности. Трудности возникают в случаях, когда для реализации операций или сравнений требуется преобразование *C*-систем или *D*-систем в альтернативный класс.

Задача преобразования АК-объекта в альтернативный класс для случая, когда АК-объект есть *D*-система (или *C*-система), по вычислительной сложности выходит за пределы *NP*-полных задач и относится к классу #*P*-полных, то есть задач перечисления.

Если требуется преобразовать АК-объект *R* размерности $m \times n$ в альтернативный класс, то в худшем случае потребуется $\boldsymbol{C}n^m$ операций (каждая строка *R* преобразуется в систему из *n* кортежей и необходимо выполнить такой же объем операций, как при вычислении декартова произведения полученных после преобразования систем). Причем это соотношение справедливо даже если каждая компонента является подмножеством множества $\{0, 1\}$ (модели исчисления высказываний). На первый взгляд представляется, что при реализации в структурах АК задач, относящихся в исчислении высказываний и в исчислении предикатов к классам *NP*-полных, трудоемкость решения не только не уменьшается, но в ряде случаев даже увеличивается. Однако, как показывает практика, большинство задач логического анализа можно решить без преобразования АК-объектов в альтернативные классы (см., например, п. 3.3, 3.4). Если же таких преобразований не избежать, то даже в том случае, когда матрица АК-объекта содержит сравнительно небольшое число фиктивных компонент, методы уменьшения трудоемкости на основе матричных свойств АК-объектов оказываются удобными для использования. Иногда для таких "плотных" АК-объектов удается найти частные случаи, когда сложность преобразования полиномиальна [*Кулик*, 1995].

Для разреженной матрицы АК-объекта оценка $\boldsymbol{C}n^m$ заменяется оценкой $\boldsymbol{C}k^m$, где $k << n$. Путем использования вышеизложенных методов уменьшения трудоемкости и эту оценку во многих случаях удается существенно уменьшить.

Поиск и разработка новых методов снижения трудоемкости алгоритмов в АК значительно упрощается за счет большей регулярности структур АК-объектов по сравнению с формальными представлениями логических систем. В частности, матричное представление АК-объектов позволяет более естественно применять методы распараллеливания алгоритмов решения.

При реализации в АК алгоритмов задач, относящихся в соответствующих логических моделях к классам *NP*-полных, существенную роль в уменьшении их трудоемкости играет предварительная ортогонализация структур. В этом случае соответствующая организация вычислений в дереве поиска решения приводит к более интенсивному отсечению ветвей, что часто значительно уменьшает общее число вычислительных операций.

Глава 4. Логический вывод и анализ модифицируемых рассуждений в АК

… скоро люди усовершенствуют язык
до такой степени, что будут доказывать
математически верно, что дважды два - семь.
А.П. Чехов, "Огни"

Полноценный логический анализ включает в себя не только логический вывод, но и анализ неопределенностей и противоречивости, формирование гипотез и абдуктивных заключений. Если задачи логического вывода в настоящее время, хоть и с трудом, решаются формальными средствами на основе классического подхода, то для решения остальных задач привлекаются в основном неклассические логики, в частности, логика умолчаний и немонотонные логики. Совместить логический вывод и недедуктивные (в том числе, модифицируемые) рассуждения непросто, так как формальный подход, распространенный в современной логике, изначально не предназначен для логического анализа, выходящего за рамки дедукции.

В рамках предлагаемой общей теории отношений проще совместить дедуктивный и недедуктивный анализ. Для пояснения сказанного рассмотрим алгебраическую интерпретацию логического вывода.

.1. Интерпретация логического вывода

Особую роль в интерпретации логического ввода играет теорема дедукции (см. подраздел 1.3). В ней утверждается, что если из A выводимо B, то импликация $A \supset B$ – общезначимая формула. Этот результат может быть основой для следующего утверждения.

Теорема 4.1. Если для логических формул A и B имеется интерпретация в виде множеств S_A и S_B, то общезначимость импликации $A \supset B$ равносильна выполнению отношения $S_A \subseteq S_B$.

Доказательство. За основу возьмем таблицу истинности импликации (табл. 1.4. в подразделе 1.2). Пусть задана система из двух множеств S_A и S_B, а также некоторый элемент t, относительно которого можно сказать, что он принадлежит какому-то множеству, либо его дополнению. Предположим, что высказываниям "A истинно" ($A = 1$) или "B истинно" ($B = 1$) соответствуют

утверждения $t \in S_A$ и $t \in S_B$. Тогда выражениям $A = 0$ и $B = 0$ можно сопоставить утверждения $t \in \overline{S_A}$ и $t \in \overline{S_B}$. Согласно табл. 1.4 импликация $A \supset B$ истинна для пар утверждений: ($t \in \overline{S_A}$ и $t \in \overline{S_B}$); ($t \in \overline{S_A}$ и $t \in S_B$); ($t \in S_A$ и $t \in S_B$) и ложна для пары ($t \in S_A$ и $t \in \overline{S_B}$) – последнее означает ситуацию, когда не соблюдается соотношение $S_A \subseteq S_B$. Во всех остальных истинных случаях утверждения о принадлежности элемента t не противоречат соотношению $S_A \subseteq S_B$. Отсюда следует справедливость теоремы. *Конец доказательства*.

Рассмотрим интерпретацию одного из основных правил логического вывода – modus ponens. Формулируется оно следующим образом: если истинно A и истинно $A \supset B$, то B тоже истинно. Когда интерпретацией A и B являются множества S_A и S_B, то истинность импликации означает, что $S_A \subseteq S_B$. Следовательно, если некоторое множество S_A соответствует истинному утверждению, то по правилу modus ponens любое множество S_B, для которого справедливо $S_A \subseteq S_B$, также будет истинным утверждением.

Этот вывод кажется парадоксальным с учетом того, что в современной логике принято считать, что дедукция, в отличие от индукции, есть переход от общего к частному. Однако справедливость данного соотношения подтверждается не только интерпретацией правила modus ponens, но и другими обоснованиями, о которых будет рассказано ниже.

Для более сложных случаев в АК предусматривается возможность использования обобщенных операций и отношений, о которых шла речь в подразделе 2.2. Суть этих операций и отношений, обозначаемых как $\cap_G$, $\cup_G$, $\setminus_G$, $\subseteq_G$, $=_G$ и т.д., заключается в следующем. Операции и отношения алгебры кортежей и алгоритмы их выполнения определены только для однотипных АК-объектов. Если у них разные схемы отношения, то перед выполнением операций и проверок АК-объекты предварительно приводят к одинаковым схемам отношений с помощью добавления фиктивных атрибутов. Установлено, что добавление фиктивных атрибутов есть семантически равносильное преобразование. Поэтому, когда речь идет о логических формулах (например, A и B) с разным составом свободных переменных, и этим формулам соответствуют АК-объекты T_A и T_B, то формулам $A \wedge B$ и $A \vee B$ в АК сопоставляются выражения $T_A \cap_G T_B$ и $T_A \cup_G T_B$.

Аналогично, если АК-объекты (например, P и Q) имеют разные схемы отношения, то проверка отношения $P \subseteq_G Q$ означает следующую последовательность операций: сначала они приводятся к одинаковым схемам отношения за счет добавления в них недостающих фиктивных атрибутов, после чего проверяется включение с использованием стандартных алгоритмов АК, сформулированных в главе 2.

С учетом сказанного рассмотрим интерпретацию известных теорем, позволяющих свести задачи логического вывода к решению задач выполнимости. Эти теоремы сформулированы и доказаны в [*Чень*, 1983], а в данной книге идут под номерами 1.6 и 1.7 в подразделе 1.3. Их интерпретацию в АК дают теоремы 4.2 и 4.3.

Теорема 4.2. Пусть даны АК-объекты $F_1, ..., F_n$ и G. Тогда G есть следствие $F_1, ..., F_n$ тогда и только тогда, когда $(F_1 \cap_G ... \cap_G F_n) \neq \varnothing$ и $(F_1 \cap_G ... \cap_G F_n) \subseteq G$.

Теорема 4.3. Пусть даны АК-объекты $F_1, ..., F_n$ и G. Тогда G есть следствие $F_1, ..., F_n$ тогда и только тогда, когда $(F_1 \cap_G ... \cap_G F_n) \neq \varnothing$ и $F_1 \cap_G ... \cap_G F_n \cap_G \overline{G} = \varnothing$.

Таким образом, в АК выводимость основана не на правилах вывода, а на проверке включения АК-объектов или проверке пустоты при пересечении определенных отношений. По сравнению с теоремами 1.6 и 1.7, в теоремах 4.2 и 4.3 добавлено условие $((F_1 \cap_G ... \cap_G F_n) \neq \varnothing$, которое исключает ситуацию, когда "из лжи можно вывести все, что угодно" (в терминах алгебры множеств: "пустое множество включено в любое множество"). Почему-то в исчислениях гильбертовского типа считается допустимой ситуация, когда конъюнкция посылок – тождественно ложная формула, а вывод правильный (по крайней мере, в специальной литературе этот вопрос даже не обсуждается), хотя с точки зрения *естественного логического вывода* она явно абсурдна.

4.2. Формулировки задач и алгоритмы логического вывода

4.2.1. Типы задач логического вывода

В логике и системах искусственного интеллекта широко используются следующие системы логического вывода:

1) на основе правил вывода исчисления предикатов;

2) натуральный вывод [*Гладкий*, 2001] – система вывода с расширенным по сравнению с предыдущим набором правил;

3) вывод на основе принципа резолюции [*Чень*, 1983];

4) на основе специфических правил вывода, предусмотренных в конкретной системе знаний [*Russel,* 2003; *Рассел*, 2006], [*Вагин*, 2004]; эти правила позволяют формировать новые отношения с помощью соединения или композиции исходных отношений, в силу чего их можно назвать ***семантическими*** правилами.

В алгебре кортежей предлагается новая система логического вывода, использующая обобщенные операции и соотношения. Предположим, что задана система аксиом A_1, …, A_n, которые отображены в виде АК-объектов. Тогда возможно решение следующих двух типов задач.

1) ***Задача проверки правильности следствия***. Если задано предполагаемое следствие B, то доказательство производится проверкой правильности обобщенного включения

$$(A_1 \cap_G \ldots \cap_G A_n) \subseteq_G B. \qquad (4.1)$$

2) ***Задача вывода произвольных следствий***. Для решения этой задачи сначала вычисляется АК-объект $A = A_1 \cap_G \ldots \cap_G A_n$, после чего выбираются такие B_j, для которых соблюдается $A \subseteq_G B_j$. При этом можно не только использовать известные правила вывода. Ниже приводятся алгоритмы, позволяющие при известном A вычислить варианты B_j с учетом заданных ограничений.

Разумеется, логический анализ включает в себя не только эти задачи, но и такие, как проверка корректности гипотез, вывод абдуктивных заключений и т.д. Последние выходят за рамки логического вывода и, как правило, не рассматриваются в теоретических работах по математической логике. Решение таких задач средствами АК будет рассмотрено в подразделе 4.3.

4.2.2. Алгоритмы решения задачи проверки правильности следствия

Соотношение (4.1) – по сути, другая формулировка одного из соотношений теоремы 4.2. Кроме того, для его подтверждения авторами была произведена проверка всех известных в классической логике правил логического вывода. Оказалось, что (4.1) выполняется для всех правил. Исключение – формулировка правила обобщения, приводимая в некоторых монографиях по математической логике. Например, в [*Мендельсон*, 1984] это правило формулируется так:

"Правило обобщения (или связывания квантором всеобщности): *из* $\mathcal{A}$ *следует* $\forall x_1 \mathcal{A}$".

Элементарная поверка показывает, что в случае, когда x_1 содержится как свободная переменная в формуле $\mathcal{A}$, соотношение (4.1) выполняется лишь в исключительных случаях. При внимательном прочтении текста в [*Мендельсон*, 1984] можно лишь предположить, что формулировка правила подразумевает случай, когда в $\mathcal{A}$ переменная x_1 не свободна. На это указывает, в частности, приведенная на той же странице аксиома

(4): $\forall x_1 \mathcal{A}(x_1) \supset \mathcal{A}(x_1)$.

Если допустить, что в правиле обобщения x_1 есть свободная переменная в $\mathcal{A}$, то правило обобщения окажется утверждением, обратным аксиоме (4), что является логической ошибкой. Однако явного указания на это обстоятельство применительно к правилу обобщения в тексте нет.

На сомнительность правила обобщения в такой формулировке обратили внимание некоторые логики. В частности, в системе натурального вывода оно применяется лишь для случая, когда в формуле $\mathcal{A}$ отсутствует переменная x_1. Такое же требование предъявляется к операции добавления фиктивных атрибутов в АК – добавление допустимо лишь в случае их отсутствия в схеме отношения соответствующего АК-объекта.

При решении задачи проверки правильности следствия можно использовать алгоритмы выполнения операций АК и проверки правильности отношения включения [*Кулик*, 2008 b], подробно изложенные в главе 2, а для задач большой размерности применять методы уменьшения трудоемкости, рассмотренные в главе 3. Методы решения этой задачи рассмотрим на

примерах.

Пример 4.1. Докажем справедливость одного из правил вывода натурального исчисления, которое называется правилом дилеммы:

$$\frac{A \supset C, B \supset C, A \vee B;}{C}.$$

Подразумевается, что из формул над чертой выводится формула под чертой. Можно считать, что верхние формулы представляют собой посылки, а нижняя – следствие из них.

Чтобы применить аппарат АК, положим, что X_A, X_B и X_C – логические переменные с областью значений $\{0, 1\}$. Преобразуем конъюнкцию верхних формул в D-систему в схеме отношения $[X_A X_B\ X_C]$, для чего воспользуемся соотношением (1.1):

$$Up[X_A X_B X_C] = \begin{bmatrix} \{0\} & \varnothing & \{1\} \\ \varnothing & \{0\} & \{1\} \\ \{1\} & \{1\} & \varnothing \end{bmatrix}.$$

Нижняя часть правила выражается как C-кортеж $Down[X_C] = [\{1\}]$. Чтобы доказать справедливость правила методами АК, нужно проверить соотношение $Up[X_A X_B X_C] \subseteq_G Down[X_C]$.

Если, согласно приведенным в предыдущих главах алгоритмам, преобразовать $Up[X_A X_B X_C]$ в C-систему, то:

$$Up[X_A X_B X_C] = \begin{bmatrix} \{1\} & \{0\} & \{1\} \\ * & \{1\} & \{1\} \end{bmatrix}.$$

В таком случае проверку включения $Up[X_A X_B X_C] \subseteq_G Down[X_C]$ можно осуществить с помощью алгоритма полиномиальной сложности. Для этого:

1) добавим фиктивные атрибуты в $Down[X_C]$:

$Down[X_A X_B X_C] = [* * \{1\}]$;

2) проверим включение каждого C-кортежа из $Up[X_A X_B X_C]$ в $Down[X_A X_B X_C]$, что легко выполнить на основе известных соотношений АК (теорема 2.3). Тем самым подтверждается справедливость вывода.

Тот же результат можно получить, если использовать теорему 4.3. Для этого нужно в $Up[X_A X_B\ X_C]$ добавить еще один дизъюнкт, соответствующий

формуле $\neg C$, и убедиться, что полученный АК-объект пуст. Формула $\neg C$ моделируется D-кортежем $\overline{Down[X_A X_B X_C]} =]\varnothing\ \varnothing\ \{0\}[$, тогда проверяемый АК-объект будет равен D-системе

$$Up[X_A X_B X_C] \cap_G \overline{Down[X_A X_B X_C]} = \begin{bmatrix} \{0\} & \varnothing & \{1\} \\ \varnothing & \{0\} & \{1\} \\ \{1\} & \{1\} & \varnothing \\ \varnothing & \varnothing & \{0\} \end{bmatrix}.$$

Используя изложенные ранее алгоритмы вычислений, нетрудно проверить, что полученная D-система пуста. Для этого можно использовать алгоритм преобразования D-системы в альтернативный класс (подраздел 2.2).

Разработанные алгоритмы проверки правильности следствия применимы не только в рамках исчисления высказываний, но и для более широкого класса логических систем.

Пример 4.2. Данный пример приводится для иллюстрации некоторых алгоритмов логического вывода в АК. Допустим, что область значений каждой из переменных x, y, z есть множество $\{a, b, c, d\}$, а посылки интерпретируются АК-объектами, заданными в универсуме $X \times Y \times Z = \{a, b, c, d\}^3$. Пусть предполагаемым следствием является импликация "Если x равно a или c, то z равно d", а обобщенное пересечение АК-объектов, отображающих посылки, равно C-системе $R = \begin{bmatrix} \{b,d\} & \{a,c\} & \{a,d\} \\ \{a,b\} & * & \{d\} \end{bmatrix}$. Необходимо проверить корректность следствия.

Запишем предполагаемое следствие в структурах АК. Утверждение "x равно a или c", можно выразить как $P_1[X] = [\{a, \mathrm{c}\}]$, а утверждение "$z$ равно d" – как $P_2[Z] = [\{d\}]$. Соотношение $P_1[X] \supset P_2[Z]$ эквивалентно формуле $\overline{P_1[X]} \vee P_2[Z]$, которая в структурах АК имеет вид $\overline{P_1[X]} \cup_G P_2[Z]$ и может быть представлена как C-система $T[XZ] = \begin{bmatrix} \{b,d\} & * \\ * & \{d\} \end{bmatrix}$ или D-кортеж $]\{b, d\}\ \{d\}[$.

Проверку включения $R[XYZ] \subseteq_G T[XZ]$ проще выполнить, если $T[XZ]$ будет выражено как D-кортеж. Тогда можно использовать теоремы 2.16 и 2.18. Согласно теореме 4.2, следствие в данном примере корректно, поскольку соблюдается соотношение

$$\begin{bmatrix} \{b,d\} & \{a,c\} & \{a,d\} \\ \{a,b\} & * & \{d\} \end{bmatrix} \subseteq_G]\{b, d\} \varnothing \{d\}[.$$

Такой же результат дает и теорема 4.3, если вычислить $R[XYZ] \cap_G \overline{T[XZ]}$ и показать, что результат равен пустому множеству. Тогда с помощью теоремы 2.5 получим:

$$\begin{bmatrix} \{b,d\} & \{a,c\} & \{a,d\} \\ \{a,b\} & * & \{d\} \end{bmatrix} \cap_G [\{a, c\} * \{a, b, c\}] = \varnothing.$$

Значит, правильность вывода подтверждается.

Пример 4.3 [*Мендельсон*, 1984]. Проверить правильность логического вывода для следующего рассуждения: "Если Джонс не встречал этой ночью Смита, то либо Смит был убийцей, либо Джонс лжет. Если Смит не был убийцей, то Джонс не встречал Смита этой ночью, и убийство имело место после полуночи. Если убийство имело место после полуночи, то либо Смит был убийцей, либо Джонс лжет. Следовательно, Смит был убийцей".

Сначала выразим данное рассуждение на языке исчисления высказываний. Введем обозначения:

A – Джонс встречал этой ночью Смита;

B – Смит был убийцей;

C – Джонс лжет;

D – убийство имело место после полуночи.

Тогда заданное рассуждение можно сформулировать так:

- первое предложение: $\neg A \supset (B \oplus C)$;

- второе предложение: $\neg B \supset (\neg A \wedge D)$;

- третье предложение: $D \supset (B \oplus C)$;

- следствие: B.

Преобразуем первые три предложения, используя формулы (1.1) и (1.2) для того, чтобы избавиться в них от импликации и строгой дизъюнкции. Получаем:

1) $\neg A \supset (B \oplus C) = A \vee (B \wedge \neg C) \vee (\neg B \wedge C)$;

2) $\neg B \supset (\neg A \wedge D) = B \vee (\neg A \wedge D)$;

3) $D \supset (B \oplus C) = \neg D \vee (B \wedge \neg C) \vee (\neg B \wedge C)$.

Для представления этих формул АК-объектами используем универсум $X_1 \times X_2 \times X_3 \times X_4 = \{0, 1\}^4$, где $A = B = C = D = 1$ и $\neg A = \neg B = \neg C = \neg D = 0$. Тогда посылки, которые являются ДНФ, выражаются C-системами:

$$P_1 = \begin{bmatrix} \{1\} & * & * & * \\ * & \{1\} & \{0\} & * \\ * & \{0\} & \{1\} & * \end{bmatrix}; \; P_2 = \begin{bmatrix} * & \{1\} & * & * \\ \{0\} & * & * & \{1\} \end{bmatrix}; \; P_3 = \begin{bmatrix} * & * & * & \{0\} \\ * & \{1\} & \{0\} & * \\ * & \{0\} & \{1\} & * \end{bmatrix},$$

а следствие – C-кортежем $S[X_2] = [\{1\}]$.

Решить задачу можно двумя способами:

1) проверить соотношение $(P_1 \cap_G P_2 \cap_G P_3) \subseteq_G S[X_2]$;

2) определить пустоту АК-объекта $P_1 \cap_G P_2 \cap_G P_3 \cap_G \overline{S[X_2]}$.

Поскольку в любом из этих способов присутствует выражение $P_1 \cap_G P_2 \cap_G P_3$, то целесообразно вычислить соответствующий ему АК-объект. После вычисления имеем:

$$P[X_1X_2X_3X_4] = P_1 \cap_G P_2 \cap_G P_3 = \begin{bmatrix} \{1\} & \{1\} & * & \{0\} \\ * & \{1\} & \{0\} & * \\ \{0\} & \{0\} & \{1\} & \{1\} \end{bmatrix}.$$

Для проверки включения добавим недостающие атрибуты в S:

$S[X_1X_2X_3X_4] = [* \; \{1\} \; * \; *]$.

Проверка показывает, что третий C-кортеж из P не включен в S, то есть вывод неверный. Тот же результат получим, найдя пересечение $P \cap_G \overline{S[X_2]}$:

$$\begin{bmatrix} \{1\} & \{1\} & * & \{0\} \\ * & \{1\} & \{0\} & * \\ \{0\} & \{0\} & \{1\} & \{1\} \end{bmatrix} \cap_G [* \; \{0\} \; * \; *] = [\{0\} \; \{0\} \; \{1\} \; \{1\}] \neq \varnothing.$$

Предложенный подход к решению задач логического вывода позволяет по-новому осмыслить суть логического следования в классической логике. Как известно, справедливость отношения $A \subseteq B$ означает, что B есть *необходимое условие* или свойство A. Соотношение (4.1) показывает: логическое следствие корректно не только потому, что получено на основе правил вывода, смысл которых не всегда понятен, но еще и потому, что *является необходимым условием существования антецедента*.

4.2.3. Алгоритмы решения задачи вывода произвольных следствий

Такая задача в общем виде не представляет интереса, поскольку на практике поиск возможных следствий целесообразен только при наличии перечисленных далее ограничений:

1) в следствии желательно использовать лишь небольшую часть всех переменных рассуждения;

2) состав переменных в искомом следствии нередко определяется, исходя из смыслового анализа конкретной системы рассуждений.

Применение правил вывода в качестве механизма получения следствий при заданных ограничениях часто требует перебора большого числа вариантов, так как заранее невозможно предсказать порядок применения правил.

При использовании методов АК задача существенно упрощается. Когда задано множество АК-объектов $\{A_1, \ldots, A_n\}$, представляющих аксиомы (или посылки), то можно найти АК-объект $A = A_1 \cap_G \ \ldots \ \cap_G A_n$. Тогда для получения любого следствия достаточно построить АК-объект B_j такой, чтобы выполнялось соотношение $A \subseteq_G B_i$.

Рассмотрим теперь формальные правила вывода следствий. Когда A – C-система (если это не так, то можно использовать алгоритмы преобразования D-кортежей или D-систем в C-системы), то сокращение числа переменных в B_i осуществимо с помощью элиминации атрибутов из A. Ясно, что при этом соотношение $A \subseteq_G B_i$ будет выполняться.

При элиминации атрибутов из C-системы образуется проекция, свойства которой определяют дальнейшие действия по выводу следствий. Проекции могут быть ***полными***, т.е. содержащими все элементарные кортежи для их схем отношения, и ***неполными*** в противном случае. Если проекция полная, то ее следствие есть частный универсум, что равносильно общезначимой формуле и интереса не представляет. Поэтому будем учитывать только неполные проекции.

Подтверждением сказанному может служить промежуточный результат из примера 4.1:

$$Up[X_A X_B X_C] = \begin{bmatrix} \{1\} & \{0\} & \{1\} \\ * & \{1\} & \{1\} \end{bmatrix}.$$

Здесь полными являются только проекции: $[X_A]$ и $[X_B]$, так как содержат все значения атрибутов. Проекция $[X_AX_B]$ неполная, поскольку в ней нет пары (0, 0). Все остальные проекции – также неполные.

Сформируем для A некоторую совокупность неполных проекций. Тогда все многообразие вариантов формирования возможных следствий B_i может быть выражено тремя правилами.

Правила вывода следствий в АК

1) оставить в качестве B_i одну из неполных проекций;

2) выбрать в качестве B_i любую проекцию при условии, что в ее состав входит, по крайней мере, одна неполная проекция;

3) для выбранного по предыдущим правилам АК-объекта построить покрывающий его неполный АК-объект, добавив к нему элементарные кортежи или C-кортежи.

Если выбрать третье правило и использовать в качестве объекта его применения само A, то все многообразие следствий из A можно получить, добавляя в него по одному элементарному кортежу из множества $\overline{A}$. Ясно, что для практических целей этот способ не пригоден, однако использование первых двух правил позволяет относительно просто находить следствия с учетом заданных ограничений.

Для иллюстрации метода рассмотрим промежуточный результат из примера 4.3:

$$P[X_1X_2X_3X_4] = \begin{bmatrix} \{1\} & \{1\} & * & \{0\} \\ * & \{1\} & \{0\} & * \\ \{0\} & \{0\} & \{1\} & \{1\} \end{bmatrix}.$$

Пусть мы хотим получить следствие, в котором используются атрибуты X_2 и X_4. Для этого вычисляем проекцию

$$P[X_2X_4] = \begin{bmatrix} \{1\} & \{0\} \\ \{1\} & * \\ \{0\} & \{1\} \end{bmatrix} = \begin{bmatrix} \{1\} & * \\ \{0\} & \{1\} \end{bmatrix}.$$

Проекция оказалась неполной, она соответствует логической формуле

$B \vee (\neg B \wedge D) = B \vee D = \neg B \supset D = \neg D \supset B$.

Перевод этой формулы на естественный язык следующий:

1) Смит был убийцей или преступление имело место после полуночи;

2) если Смит не был убийцей, то преступление имело место после полуночи;

3) если преступление совершено до полуночи, то Смит был убийцей.

Нетрудно убедиться, что приведенных выше правил формирования возможных следствий достаточно для построения всех возможных следствий из заданного набора посылок.

4.3. Анализ модифицируемых рассуждений

4.3.1. Коллизии в рассуждениях

К модифицируемым рассуждениям относятся логические системы, у которых в ходе анализа могут изменяться исходные предпосылки. Это связано с предположительным характером многих наших знаний, которые могут уточняться. Процесс уточнения знаний не всегда однозначен, часто приходится выбирать подходящую гипотезу среди множества вариантов, поэтому целесообразно ввести определенные критерии корректности новых знаний. В классической логике таким критерием служит отсутствие противоречий в знаниях.

В математической логике противоречивость системы рассуждений (теории) определена лишь для случая, когда из посылок одновременно выводится некоторое следствие и его отрицание. В то же время и в повседневных, и в неформализованных научных рассуждениях один из бесспорных критериев несостоятельности системы знаний – вывод контрарных следствий (например, из посылок следует, что "всем A присуще B" и "всем A не присуще B"). Формально эти два суждения не являются отрицаниями друг друга. Отрицание формулы $\forall x(A(x) \supset B(x))$ в исчислении предикатов дается формулой $\exists x(A(x) \wedge \neg B(x))$ – "некоторым A не присуще B", но не формулой $\forall x(A(x) \supset \neg B(x))$, которая соответствует суждению "всем A не присуще B".

Чтобы устранить это и другие несоответствия между формальной логикой и естественными рассуждениями, в систему логического анализа предлагается ввести понятие ***коллизия***. Коллизии в основном возникают в модифицируемых рассуждениях при вводе новых знаний (гипотез) как нарушение некоторых формально выраженных правил или ограничений, регулирующих целостность

или смысловое содержание системы. В системах пересматриваемой аргументации коллизиям в какой-то степени соответствуют такие ситуации, как "опровержение", "подрыв аргумента", "атака" и т.д. [*Вагин*, 2004].

Термин "коллизия" был использован при анализе рассуждений типа силлогистики [*Кулик*, 2001]. Были определены два типа формальных коллизий:

коллизия парадокса, когда из посылок следует суждение типа "всем A не присуще A" ($A \supset \overline{A}$) и, значит, объем термина A пустой;

коллизия цикла, когда в системе множеств выводится соотношение $A \subseteq B \subseteq \ldots \subseteq A$, что означает эквивалентность терминов, входящих в данный цикл.

Эти коллизии распознаются как свойства самих моделей без сравнения с моделируемой ситуацией, поэтому они и названы формальными.

Третья коллизия не относится к формальным и характеризует ситуацию, в которой полученные следствия системы рассуждений не соответствуют бесспорным фактам или обоснованным утверждениям. Этот тип коллизий назван ***коллизией неадекватности***.

В отличие от логического противоречия, которое, по сути, означает безусловное вырождение посылок, коллизии в разных ситуациях могут иметь противоположный смысл. Другими словами, коллизия, в отличие от противоречия, зависит от семантики моделируемой системы. Например, в одной системе равенство $A = \varnothing$ означает отсутствие объекта, без которого существование моделируемой системы невозможно, в другой – уточнение статуса объекта A. В первом случае система посылок требует изменений, во втором – мы получаем новую полезную информацию.

Приведем примеры коллизий, которые могут иметь место в задачах анализа полисиллогизмов [*Кулик*, 1997; 2001].

Пример 4.4. Даны посылки:

1. Все мои друзья хвастуны и не скандалисты.

2. Все хвастуны не уверены в себе.

3. Все не скандалисты уверены в себе.

Из заданных посылок следует утверждение "Все мои друзья не мои друзья", показывающее, что множество моих друзей пусто (коллизия парадокса).

Для анализа коллизий при решении задач полисиллогистики была разработана методика, в которой используется разновидность частично упорядоченных множеств – *E*-структуры [*Кулик*, 2001]. Однако для этой цели методы АК также применимы, хотя и оказываются более сложными. Выразим посылки на языке исчисления предикатов. Допустим, A – предикат "x – мой друг", B – "x – хвастун", C – "x – скандалист", D – "x – уверенный в себе". Теперь условия задачи запишутся так:

1. $A \supset (B \wedge \overline{C}) = \overline{A} \vee (B \wedge \overline{C})$

2. $B \supset \overline{D} = \overline{B} \vee \overline{D}$.

3. $\overline{C} \supset D = \mathrm{C} \vee D$.

Им соответствуют АК-объекты заданные в универсуме $\{0, 1\}$ (для краткости компоненты $\{0\}$ и $\{1\}$ обозначены как 0 и 1):

$$P_1[ABC] = \begin{bmatrix} 0 & * & * \\ * & 1 & 0 \end{bmatrix}; P_2[BD] = \begin{bmatrix} 0 & * \\ * & 0 \end{bmatrix}; P_3[CD] = \begin{bmatrix} 1 & * \\ * & 1 \end{bmatrix}.$$

Пересечение этих структур с применением ортогонализации дает:

$$P[ABCD] = P_1[ABC] \cap_G P_2[BD] \cap_G P_3[CD] = \begin{bmatrix} 0 & * & * & * \\ * & 1 & 0 & * \end{bmatrix} \cap$$

$$\cap \begin{bmatrix} * & 0 & * & * \\ * & * & * & 0 \end{bmatrix} \cap \begin{bmatrix} * & * & 1 & * \\ * & * & * & 1 \end{bmatrix} = \begin{bmatrix} 0 & * & * & * \\ 1 & 1 & 0 & * \end{bmatrix} \cap \begin{bmatrix} * & 0 & * & * \\ * & 1 & * & 0 \end{bmatrix} \cap$$

$$\cap \begin{bmatrix} * & * & 1 & * \\ * & * & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & * & * \\ 0 & 1 & * & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \cap \begin{bmatrix} * & * & 1 & * \\ * & * & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & * \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

В результате получена *C*-система, где первый столбец содержит только одноэлементную компоненту $\{0\}$. Это и есть признак коллизии парадокса $A \supset \overline{A}$. Действительно, выражению $(A \supset \overline{A}) = \overline{A} \vee \overline{A} = \overline{A}$ в АК сопоставляется *C*-кортеж $S[ABCD] = [0 * * *]$, причем выполняется соотношение $P[ABCD] \subseteq_G S[ABCD]$, а значит, коллизия $A \supset \overline{A}$ есть следствие исходной системы посылок (см. п. 4.2).

Если вывод противоречит действительности (коллизия неадекватности), то требуется корректировка посылок. К успеху, в частности, приводит замена третьей посылки на обратную ей: "Все уверенные в себе не скандалисты". После такой замены коллизий не возникает.

Как пример противоположного смысла той же коллизии рассмотрим следующую задачу. Пусть задана некоторая система множеств целых чисел. Каждое множество этих чисел характеризуется делимостью на определенное целое число. Например, множество N_3 состоит из чисел, делящихся без остатка на 3. Дополнение $\overline{N_3}$ этого множества – все числа, не делящиеся на 3. Известно, что между множествами существуют следующие соотношения:

1) $N_2 \subseteq (N_3 \cap \overline{N_5})$;

2) $N_3 \subseteq \overline{N_7}$;

3) $\overline{N_7} \subseteq N_5$.

По структуре эта система полностью совпадает с предыдущей. Но возможность пересмотра посылок здесь исключена. Поэтому возникающая коллизия парадокса $N_2 \subseteq \overline{N_2}$ и соответствующий вывод $N_2 = \varnothing$ говорят лишь о том, что в данной системе посылок существование четных чисел невозможно.

Пример 4.5. Даны посылки:

1. Все, что существует, подтверждается экспериментом.

2. Все неизвестное не подтверждается экспериментом.

3. Все известное существует.

По аналогии с примером 4.4. проанализируем условия задачи методами АК с целью выявления в них возможных коллизий. Пусть A – предикат "x существует", B – "x экспериментально подтверждается", C – "x известен". Тогда посылки будут иметь вид:

1. $A \supset B = \overline{A} \vee B$.

2. $\overline{C} \supset \overline{B} = C \vee \overline{B}$.

3. $C \supset A = \overline{C} \vee A$.

Перепишем их в терминах АК:

$$P_1[AB] = \begin{bmatrix} 0 & * \\ * & 1 \end{bmatrix} = \begin{bmatrix} 0 & * \\ 1 & 1 \end{bmatrix}; P_2[BC] = \begin{bmatrix} 0 & * \\ * & 1 \end{bmatrix} = \begin{bmatrix} 0 & * \\ 1 & 1 \end{bmatrix}; P_3[AC] = \begin{bmatrix} 1 & * \\ * & 0 \end{bmatrix} = \begin{bmatrix} 1 & * \\ 0 & 0 \end{bmatrix},$$

$$P[ABC] = P_1[AB] \cap_G P_2[CB] \cap_G P_3[AC] = \begin{bmatrix} 0 & * & * \\ 1 & 1 & * \end{bmatrix} \cap \begin{bmatrix} * & 0 & * \\ * & 1 & 1 \end{bmatrix} \cap$$

$$\cap \begin{bmatrix} 1 & * & * \\ 0 & * & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & * \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \cap \begin{bmatrix} 1 & * & * \\ 0 & * & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

Таким образом, во всех кортежах различные атрибуты принимают одинаковые значения истинности, что свидетельствует о равносильности утверждений *A*, *B*, *C*.

Другими словами, при выводе следствий оказывается, что термины "известное", "существующее" и "подтверждающееся экспериментом" связаны циклом, что означает их эквивалентность. Но, по крайней мере, относительно "известного" и "существующего" это неверно (коллизия неадекватности), поэтому посылки требуют пересмотра.

В логических системах, выходящих за рамки силлогистики, коллизии могут моделировать следующие ситуации:

1) "вырождение" знаний – знание оказывается тождественно ложным при вводе новых знаний (в АК эта ситуация распознается как равенство знания пустому множеству, в логике данная ситуация соответствует правилу Дунса-Скотта "из лжи можно вывести все, что угодно");

2) "вырождение" атрибутов (при вводе новых знаний из некоторых атрибутов исчезают элементы, без которых существование моделируемой системы невозможно). Другими словами, при проверке оказывается, что некоторые значимые атрибуты или их значения равны пустому множеству;

3) при вводе новых знаний некоторые различные атрибуты становятся тождественными по составу элементов, что противоречит семантике системы;

4) несоответствие полученных результатов с трудноформализуемыми ограничениями, описанными в постановке задачи. Например, в моделируемой системе могут быть заданы ограничения в виде отношений, которые не должны быть в следствиях. Если эти ограничения вводить в начальные условия как дополнения запрещенных отношений, то система может существенно усложниться. Иногда проще производить отбраковку результатов с помощью их сопоставления с запрещенными отношениями.

5) иногда коллизией можно считать ситуации, которые используются для обоснования правомерности применения немонотонных логик. Например [*Тейз*, 1990], если известно, что 1) "все птицы летают" и 2) "страус Тити птица, но не летает", то для разрешения этой ситуации совсем не обязательно вводить неклассическую логику. Достаточно зафиксировать в этом рассуждении коллизию и произвести корректировку посылок, не нарушая принципов

классической логики.

Трудно предусмотреть заранее все возможные разновидности коллизий – в некоторых системах они могут быть уникальными. Предложим следующее краткое определение коллизий.

Коллизии – это ситуации, которые возникают в модифицируемых рассуждениях при вводе новых знаний (гипотез) и распознаются как нарушение некоторых формально выраженных правил или ограничений, регулирующих целостность или смысловое содержание системы.

Здесь будут рассматриваться две наиболее распространенные задачи анализа модифицируемых рассуждений:

1) формирование и проверка корректности гипотез;

2) поиск абдуктивных заключений.

4.3.2. Анализ гипотез

В терминах АК можно дать формальное определение гипотез. Пусть задана система посылок $A_1, \ldots, A_n$, представленных как АК-объекты, и вычислен АК-объект $A = A_1 \cap_G \ldots \cap_G A_n$.

Формула H называется ***гипотезой***, если неверно, что $A \subseteq_G H$. В противном случае в соответствии с (4.1) H есть следствие. Значит, H в первом приближении можно считать гипотезой, если $A \setminus_G H \neq \varnothing$.

Во втором приближении устанавливается корректность гипотезы. Гипотеза корректна, если объект $H \cap_G A$ не содержит коллизий (здесь предполагается, что гипотеза играет роль аксиомы или посылки). Рассмотрим пример.

Пример 4.6. В занимательной книге известного логика Раймонда Смаллиана [*Смаллиан*, 1986] есть серия задач об узнике, который должен был, используя определенные подсказки, определить, в какой из комнат находится принцесса, и открыть эту комнату. Сложность заключалась в том, что, по крайней мере, в одной из комнат находился тигр, встреча с которым для узника была явно нежелательна. В то же время встреча с принцессой сулила узнику не только освобождение, но и возможность жениться на принцессе. Задачи, которая приведена в этом примере, в книге Р. Смаллиана нет, но ситуация аналогичная.

Перед узником три комнаты. В одной из них находится тигр, в другой – принцесса, а третья комната пуста. Узнику даны две подсказки, причем одна из них истинная, а другая ложная, но какая именно – неизвестно.

Подсказка 1: во второй комнате нет тигра, а третья комната не пуста.

Подсказка 2: первая комната не пуста, а во второй нет тигра.

Выразим подсказки в виде *C*-кортежей. Для этого введем обозначения:

P – в комнате принцесса;

T – в комнате тигр;

E – комната пуста.

Тогда подсказки M_1 и M_2 можно выразить так:

$M_1 = [* \ \{P, E\} \ \{P, T\}]; \quad M_2 = [\{P, T\} \ \{P, E\} \ *].$

Для решения задачи достаточно исследовать две гипотезы:

Гипотеза 1: M_1 – истинно; M_2 – ложно;

Гипотеза 2: M_1 – ложно; M_2 – истинно.

Рассмотрим первую из них. Эта ситуация может быть описана с помощью следующего выражения: $M_1 \cap \overline{M_2}$. Найдем $\overline{M_2}$, используя соответствующие теоремы АК.

$$\overline{M_2} =]\{E\} \ \{T\} \ \varnothing[= \begin{bmatrix} \{E\} & * & * \\ * & (T) & * \end{bmatrix} = \begin{bmatrix} \{E\} & * & * \\ \{P,T\} & \{T\} & * \end{bmatrix}.$$

После этого:

$$M_1 \cap \overline{M_2} = [* \ \{P, E\} \ \{P, T\}] \cap \begin{bmatrix} \{E\} & * & * \\ \{P,T\} & \{T\} & * \end{bmatrix} = [\{E\} \ \{P, E\} \ \{P, T\}].$$

Положение несколько упростилась. Если вычислить декартово произведение $\{E\}\times\{P, E\}\times\{P, T\}$, то окажется, что из четырех полученных элементарных кортежей только один – (E, P, T) – удовлетворяет условию задачи (ситуацию, когда в разных комнатах находится один и тот же объект, можно считать коллизией). Следовательно, принцесса во второй комнате.

Теперь проверим вторую гипотезу.

$$\overline{M_1} =]\varnothing \ \{T\} \ \{E\}[= \begin{bmatrix} * & \{T\} & * \\ * & * & \{E\} \end{bmatrix} = \begin{bmatrix} * & \{T\} & * \\ * & \{P,E\} & \{E\} \end{bmatrix}.$$

$$\overline{M_1} \cap M_2 = \begin{bmatrix} * & \{T\} & * \\ * & \{P,E\} & \{E\} \end{bmatrix} \cap [\{P,\ T\}\ \{P,\ E\}\ *] = [\{P,\ T\}\ \{P,\ E\}\ \{E\}].$$

Здесь условиям задачи удовлетворяет элементарный кортеж (T, P, E), который хоть и отличается от полученного ранее, но оставляет неизменным местонахождение принцессы. Таким образом, обе гипотезы приводят к одному результату: принцесса находится во второй комнате.

Пример 4.7. Та же ситуация с узником и принцессой, но условия другие. Узнику предложен выбор из трех комнат, в одной из которых находилась принцесса, в другой – тигр, а третья – пуста, причем надпись на двери, где находилась принцесса, – истинна, на двери комнаты, где находился тигр – ложна, а надпись на двери пустой комнаты может быть истинной или ложной. Надписи таковы:

комната 1: Комната 3 пуста;

комната 2: Тигр сидит в комнате 1;

комната 3: Эта комната пуста.

Где находится принцесса?

Обозначим эти надписи M_1, M_2, M_3 и, используя уже введенные в предыдущей задаче обозначения, выразим их в виде АК-объектов:

$M_1 = [* * \{E\}]$; $M_2 = [\{T\} * *]$; $M_3 = [* * \{E\}]$.

Задачу можно решить разными способами. Здесь применим метод проверки гипотез. В качестве гипотез будем использовать оценки истинности надписей на дверях трех комнат в виде битовой строки из трех символов (например, строка 110 означает, что надписи на дверях первых двух комнатах истинные, а на двери третьей – ложная). По условиям задачи допустимы далеко не все сочетания оценок, поскольку:

1) значения надписей на дверях первой и третьей комнат совпадают, поэтому они должны иметь одну и ту же оценку истинности;

2) среди трех оценок должны быть, по крайней мере, одна истинная оценка и одна ложная, так как надписи на комнатах с принцессой и тигром имеют противоположные истинностные оценки.

С учетом этого гипотез может быть только две: 010 и 101. Тогда решение задачи можно получить, если вычислить и проверить два выражения:

1) $\overline{M_1} \cap_G M_2 \cap_G \overline{M_3}$; 2) $M_1 \cap_G \overline{M_2} \cap_G M_3$.

Первой коллизией будем считать ситуацию, когда в разных комнатах одно и то же содержимое. Вторая коллизия описывает несоответствие оценок истинности надписей на дверях комнат их содержимому: для комнаты с тигром – истинная надпись, для комнаты с принцессой – ложная.

Выполним вычисления для первой гипотезы:

$$\overline{M_1} \cap_G M_2 \cap_G \overline{M_3} = [* * \{P, T\}] \cap_G [\{T\} * *] \cap_G [* * \{P, T\}] =$$

$$= [\{T\} * \{P, T\}].$$

Проанализируем эту ситуацию. Поскольку в первой комнате тигр, то в третьей комнате тигра быть не должно, следовательно, там принцесса. Поскольку там принцесса, то надпись должна быть истинной. На самом деле это не так, гипотеза 010 отвергается.

Проверим вторую гипотезу.

$$M_1 \cap_G \overline{M_2} \cap_G M_3 = [* * \{E\}] \cap_G [\{P, E\} * *] \cap_G [* * \{E\}] = [\{P, E\} * \{E\}].$$

Раз третья комната пуста, то в первой может быть только принцесса и, следовательно, тигр находится во второй комнате. Проверка показывает, что на дверях комнаты с принцессой истинная надпись, а на дверях комнаты с тигром – ложная. Гипотеза 101 принимается, принцесса в первой комнате.

Для этой же задачи можно найти решение проверкой других гипотез, в частности, таких: принцесса находится в комнате с номером I ($I = 1, 2, 3$). Поиск решения в такой постановке предоставляем читателю.

Формулировка и проверка гипотез обычно применяется в совокупности с другими методами анализа модифицируемых рассуждений. Ниже мы опишем использование гипотез при поиске абдуктивных заключений.

4.3.3. Абдуктивные заключения

Абдукция – это процесс формирования объясняющей гипотезы, когда известны посылки и предполагаемое следствие, которое при формальной проверке не следует из посылок, но, тем не менее, подтверждается фактами или обоснованными аргументами. Прототип абдукции – задача диагностики.

Мы с детства приучены к тому, что Шерлок Холмс использовал дедуктивный метод. А так ли это? По определению дедукция – это когда имеются какие-то посылки или аксиомы и какие-то правила вывода, с помощью которых мы получаем следствия. Но мысль Холмса работала в другом направлении. Он знал некоторые факты, связанные с преступлением. Эти факты мало что говорили о мотивах преступления или о том, кто это преступление совершил. Если принять факты за аксиомы, то следствием этих фактов должно было стать некое утверждение, которое прямо указывает на того, кто преступник. Но по всем правилам логики из известных фактов такое утверждение получить было невозможно. Поэтому Холмс на первых этапах не пытался найти доказательства, а домысливал некоторые гипотезы или факты, которые позволяли бы построить строгую логическую цепочку между известными и пришедшими в голову фактами и самим преступлением. Таким образом, в данном случае дедукция была завершением мыслительной деятельности, а основная работа заключалась в том, чтобы найти не само следствие, а нечто, позволяющее его получить. Это нечто в логике называется не дедукцией, а абдукцией. Таким образом, метод Шерлока Холмса скорее абдуктивный, чем дедуктивный.

Классический пример абдукции – открытие нейтрино. Предполагалось, что одним из результатов эксперимента, связанного с изучением бета-распада, будет выполнение закона сохранения энергии. Расчеты показали, что при бета-распаде этот закон не соблюдается. Физик Вольфганг Паули в 1930 году предложил гипотезу о существовании некоторых "невидимых" частиц, которые образуются в ходе бета-распада и забирают часть энергии. В 1932 году Э. Ферми назвал эту частицу "нейтрино". Экспериментально существование нейтрино (точнее, его двойника – антинейтрино) было подтверждено лишь в 1953 году.

Дадим формальное определение абдукции. Пусть B – предполагаемое следствие из посылок $A_1, \ldots, A_n$, причем известно, что утверждение $A \subseteq_G B$, где по-прежнему $A = A_1 \cap_G \ldots \cap_G A_n$, неверно. Тогда формула H является допустимым абдуктивным заключением, если соблюдаются два условия:

1) H – корректная гипотеза (т.е. $H \cap_G A$ не содержит коллизий);

2) $(H \cap_G A) \subseteq_G B$ (т.е. при добавлении H в систему посылок

предполагаемое следствие B становится выводимым).

Рассмотрим интерпретацию абдукции с помощью диаграмм Эйлера. Когда B не является следствием A, то возможны два варианта: пересечение этих множеств пусто или не пусто (рис. 4.1). Обозначим закрашенную часть A, равную $A \setminus_G B$, как R.

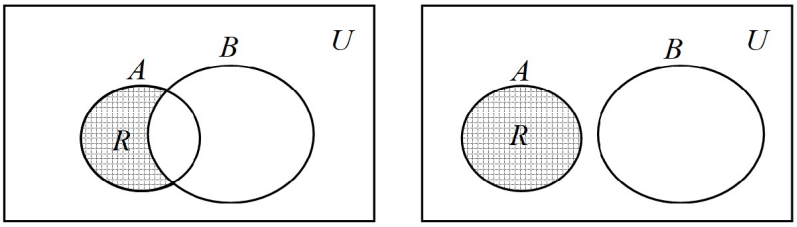

Рис. 4.1. Соотношения между посылками (A) и невыводимым следствием (B)

Вариант справа на рис. 4.1 – вырожденный. В таком случае никакое добавление посылок ни к чему не приведет. При непустом пересечении A и B (левая часть рисунка) получение абдуктивного заключения возможно, для этого область R должна быть полностью исключена из области допустимых гипотез, в противном случае $H \cap_G A$ не будет включено в B. Значит, область допустимых гипотез есть дополнение R, т.е. $H \subseteq_G \overline{R}$.

Пусть R_i – некоторое надмножество R (на рис. 4.2 оно ограничено прерывистой линией), тогда возможны два случая (рис. 4.2). $\overline{R_i}$ можно рассматривать как некоторую гипотезу H_i.

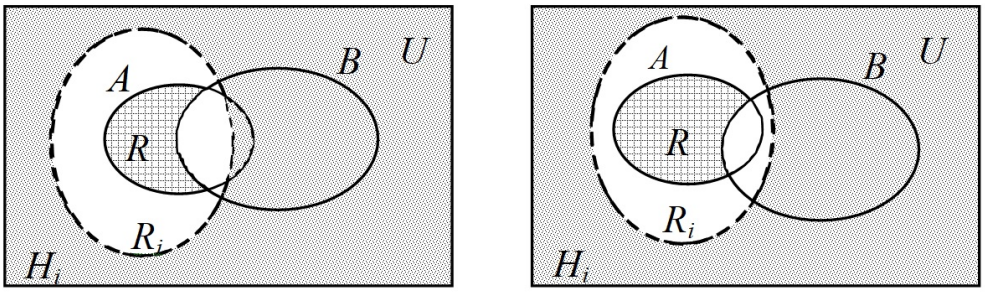

Рис. 4.2. Варианты ситуаций при выборе гипотез

В первом случае (на рисунке слева) R_i не охватывает полностью A, значит, при пересечении $\overline{R_i} \cap_G A$ получим непустое множество, включенное в B, и $\overline{R_i}$ будет формально корректным абдуктивным заключением. Вырожденный случай, когда $\overline{R_i} \cap_G A = \varnothing$, показан на рисунке справа. Здесь видно, что R_i

полностью покрывает не только R, но и A. К такой ситуации применимо правило Дунса-Скотта: из лжи можно вывести все, что угодно. Поэтому при формировании R_i надо не только обеспечивать $R \subseteq_G R_i$, но и следить за тем, чтобы не выполнялось $A \subseteq_G R_i$.

Из вышесказанного следует

Алгоритм поиска абдуктивных заключений:

1. Вычислить «остаток» $R = A \setminus_G B$;

2. Построить промежуточный объект R_i такой, чтобы соблюдалось $R \subseteq_G R_i$;

3. Вычислить $H_i = \overline{R_i}$ (тогда R_i далее можно обозначить как $\overline{H_i}$);

4. Вычислить $H_i \cap_G A$ и выполнить проверку на наличие коллизий; если коллизии обнаружены, то возвратиться к шагу 2, иначе конец алгоритма.

Поиск вариантов R_i на шаге 2 относится к задачам вывода следствий, алгоритм решения приведен в предыдущем подразделе. В качестве иллюстрации рассмотрим задачу из примера 4.3.

Там предполагаемое следствие (Смит был убийцей) не выводимо. Чтобы подтвердить или опровергнуть правильность вывода, требуется уточнить некоторые обстоятельства. Поиск таких обстоятельств можно сформулировать как задачу поиска абдуктивного заключения.

Предположим, что вывод правильный. Тогда необходимо найти подходящие гипотезы, которые можно использовать в качестве посылок. Запишем в новых обозначениях промежуточные результаты примера 4.3:

$$A[X_1X_2X_3X_4] = \begin{bmatrix} \{1\} & \{1\} & * & \{0\} \\ * & \{1\} & \{0\} & * \\ \{0\} & \{0\} & \{1\} & \{1\} \end{bmatrix};$$

$B[X_1X_2X_3X_4] = [*\ \{1\}\ *\ *]$.

Далее используем алгоритм.

1. $R = A \setminus_G B = \begin{bmatrix} \{1\} & \{1\} & * & \{0\} \\ * & \{1\} & \{0\} & * \\ \{0\} & \{0\} & \{1\} & \{1\} \end{bmatrix} \cap_G [*\ \{0\}\ *\ *] = [\{0\}\ \{0\}\ \{1\}\ \{1\}]$.

2. Здесь можно выбрать в качестве R_i любую проекцию R. Пусть это будет $R[X_4]$. Тогда получим $R_i = [* * * \{1\}]$.

3. $H_i = \overline{R_i} = [* * * \{0\}]$.

4. Поскольку коллизии нам не заданы, проверим, вырождается ли общая предпосылка A при полученной гипотезе:

$$A \cap_G H_i = \begin{bmatrix} \{1\} & \{1\} & * & \{0\} \\ * & \{1\} & \{0\} & * \\ \{0\} & \{0\} & \{1\} & \{1\} \end{bmatrix} \cap_G [* * * \{0\}] = \begin{bmatrix} \{1\} & \{1\} & * & \{0\} \\ * & \{1\} & \{0\} & \{0\} \end{bmatrix}.$$

Проверка подтверждает корректность гипотезы. На естественном языке данная гипотеза означает, что убийство произошло до полуночи. Отсюда следует, что вывод будет правильным, если после уточнения времени убийства окажется, что оно соответствует гипотезе.

Рассмотрим более сложные примеры.

Пример 4.8. [*Страбыкин*, 2008]. База знаний диагностики автомобиля содержит следующие правила.

1) если ИЗ ВЫХЛОПНОЙ ТРУБЫ ЧЕРНЫЙ ДЫМ, то БОГАТАЯ СМЕСЬ или РАННЕЕ ЗАЖИГАНИЕ.

2) если СИНИЙ ДЫМ, то ПОВЫШЕННЫЙ РАСХОД МАСЛА.

3) если ВЫХЛОПНАЯ ТРУБА ЧЕРНАЯ, то БОГАТАЯ СМЕСЬ или РАННЕЕ ЗАЖИГАНИЕ или ПОВЫШЕННЫЙ РАСХОД МАСЛА.

4) если ПОВЫШЕННЫЙ РАСХОД МАСЛА, то ИЗНОС МАСЛОСЪЕМНЫХ КОЛПАЧКОВ или ИЗНОС ПОРШНЕВЫХ КОЛЕЦ или ИЗНОС ЦИЛИНДРОВ.

5) если НОРМАЛЬНАЯ КОМПРЕССИЯ, то нет ИЗНОСА ПОРШНЕВЫХ КОЛЕЦ и нет ИЗНОСА ЦИЛИНДРОВ.

Предполагается, что следствие должно быть таким:

Если НОРМАЛЬНАЯ КОМПРЕССИЯ и ВЫХЛОПНАЯ ТРУБА ЧЕРНАЯ и СИНИЙ ДЫМ, то ИЗНОС ПОРШНЕВЫХ КОЛЕЦ.

Требуется проверить правильность следствия, а в случае неудачи подобрать абдуктивное заключение.

Введем обозначения:

x_1 – выхлопная труба черная;

x_2 – богатая смесь;

x_3 – раннее зажигание;

x_4 – нормальная компрессия;

x_5 – повышенный расход масла;

x_6 – из выхлопной трубы черный дым;

x_7 – из выхлопной трубы синий дым;

x_8 – износ маслосъемных колпачков;

x_9 – износ поршневых колец;

x_{10} – износ цилиндров.

Выразим правила формулами исчисления высказываний.

1) $x_6 \supset (x_2 \vee x_3)$;
2) $x_7 \supset x_5$;
3) $x_1 \supset (x_2 \vee x_3 \vee x_5)$;
4) $x_5 \supset (x_8 \vee x_9 \vee x_{10})$;
5) $x_4 \supset (\neg x_9 \wedge \neg x_{10})$.

Предполагаемое следствие описывается формулой $(x_4 \wedge x_1 \wedge x_7) \supset x_9$.

Тогда база знаний, записанная в виде импликаций, есть множество из шести дизъюнктов, входящих в определенную КНФ (правило 5 преобразуется в конъюнкцию двух дизъюнктов):

$$(\neg x_6 \vee x_2 \vee x_3) \wedge (\neg x_7 \vee x_5) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee x_5) \wedge (\neg x_5 \vee x_8 \vee x_9 \vee x_{10}) \wedge$$

$$\wedge (\neg x_4 \vee \neg x_9) \wedge (\neg x_4 \vee \neg x_{10}).$$

Данную КНФ можно представить как D-систему

$$A[X_1X_2X_3X_4X_5X_6X_7X_8X_9X_{10}] = \begin{bmatrix} \varnothing & \{1\} & \{1\} & \varnothing & \varnothing & \{0\} & \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing & \{1\} & \varnothing & \{0\} & \varnothing & \varnothing & \varnothing \\ \{0\} & \{1\} & \{1\} & \varnothing & \{1\} & \varnothing & \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing & \{0\} & \varnothing & \varnothing & \{1\} & \{1\} & \{1\} \\ \varnothing & \varnothing & \varnothing & \{0\} & \varnothing & \varnothing & \varnothing & \varnothing & \{0\} & \varnothing \\ \varnothing & \varnothing & \varnothing & \{0\} & \varnothing & \varnothing & \varnothing & \varnothing & \varnothing & \{0\} \end{bmatrix}.$$

Заключение в данном примере дается формулой $(x_4 \wedge x_1 \wedge x_7) \supset x_9$, которая равна дизъюнкции $(\neg x_1 \vee \neg x_4 \vee \neg x_7 \vee x_9)$ и моделируется D-кортежем $B[X_1X_4X_7X_9] =]\{0\}\ \{0\}\ \{0\}\ \{1\}[$.

Теперь можно использовать приведенный выше алгоритм поиска абдуктивных заключений. Расчеты выполнялись с помощью разработанной

одним из авторов программы. После вычисления по формуле $R = A \setminus_G B$ получим C-систему:

$$R = \begin{array}{c} \begin{matrix} X_1 & X_2 & X_3 & X_4 & X_5 & X_6 & X_7 & X_8 & X_9 & X_{10} \end{matrix} \\ \begin{bmatrix} \{1\} & \{1\} & * & \{1\} & \{1\} & * & \{1\} & \{1\} & \{0\} & \{0\} \\ \{1\} & \{0\} & \{1\} & \{1\} & \{1\} & * & \{1\} & \{1\} & \{0\} & \{0\} \\ \{1\} & \{0\} & \{0\} & \{1\} & \{1\} & \{0\} & \{1\} & \{1\} & \{0\} & \{0\} \end{bmatrix} \end{array}.$$

В верхней строке для облегчения анализа записаны имена атрибутов. Рассмотрим, как применить полученную структуру для поиска вариантов R_i и абдуктивного заключения H_i (шаги 2 и 3 алгоритма). Для этого пригодны методы, представленные в подразделе 4.2.

Сформируем набор неполных проекций в R. В данном примере такими проекциями являются: $[X_1]$, $[X_4]$, $[X_5]$, $[X_7]$, $[X_8]$, $[X_9]$, $[X_{10}]$, $[X_2X_3X_6]$. Выбранные проекции позволяют легко находить АК-объекты R_i и соответствующие им логические формулы. Выберем проекцию $[X_1X_9]$. После сокращений она будет равна C-кортежу $R[X_1X_9] = [\{1\}\ \{0\}]$. Используя алгоритм вывода произвольных следствий из п.4.2, можно сформировать следующие формулы для R_i:

1) $x_1 \wedge \neg x_9$ (соответствует C-кортежу R_i);

2) $(x_1 \wedge \neg x_9) \vee (\neg x_1 \wedge x_9)$;

3) $x_1 \vee \neg x_9$ (формулы 2 и 3 соответствуют АК-объектам, которые покрывают C-кортеж R_i) и т.д.

Возьмем в качестве R_i проекцию $[X_2X_3X_6]$. Она является C-системой $\begin{bmatrix} \{1\} & * & * \\ \{0\} & \{1\} & * \\ \{0\} & \{0\} & \{0\} \end{bmatrix}$, которая равна D-кортежу $]\{1\}\ \{1\}\ \{0\}[$ и соответствует формуле $x_2 \vee x_3 \vee \neg x_6$. Отрицание этой формулы равно $\neg x_2 \wedge \neg x_3 \wedge \neg x_6$, оно подходит в качестве абдуктивного заключения.

Пример 4.9 (Волшебная фраза). Один путешественник был захвачен племенем, вождь которого дал пленнику возможность сказать только одну фразу. Если фраза оказывалась правдивой, то его сбрасывали с высокой скалы. Если она была лживой, то путешественника должны были растерзать львы. Но ни того, ни другого не случилось. Требуется найти "волшебную фразу", спасшую жизнь путешественнику. Приведенное решение ошибочно, извините!

Пусть A – некоторое искомое высказывание, B – "путешественник сброшен со скалы", C – "путешественник растерзан львами". Запишем посылки на языке исчисления высказываний:

1. $A \supset B = \overline{A} \vee B$.

2. $\overline{A} \supset C = A \vee C$.

Полученные высказывания можно представить как *D*-систему:

$$P[ABC] = \begin{bmatrix} \{0\} & \{1\} & \varnothing \\ \{1\} & \varnothing & \{1\} \end{bmatrix} = \begin{bmatrix} \{0\} & * & * \\ \{1\} & \{1\} & * \end{bmatrix} \cap \begin{bmatrix} \{1\} & * & * \\ \{0\} & * & \{1\} \end{bmatrix} = \begin{bmatrix} \{0\} & * & \{1\} \\ \{1\} & \{1\} & * \end{bmatrix}.$$

Для спасения путешественника из системы посылок должно быть выведено утверждение $\overline{B} \wedge \overline{C}$ или, на языке АК – *C*-кортеж $S[ABC] = [*\ \{0\}\ \{0\}]$.

Необходимо найти подходящую гипотезу (абдуктивное заключение), дающую решение задачи. Для этого воспользуемся тем же алгоритмом:

$$R[ABC] = P[ABC] \setminus S[ABC] = \begin{bmatrix} \{0\} & * & \{1\} \\ \{1\} & \{1\} & * \end{bmatrix} \cap \overline{[*\quad \{0\}\quad \{0\}]} =$$

$$= \begin{bmatrix} \{0\} & \{0\} & \{1\} \\ \{1\} & \{1\} & \{0\} \end{bmatrix}.$$

Выберем следующие проекции (содержащие атрибут *A*)

$$R_1[AB] = \begin{bmatrix} \{0\} & \{0\} \\ \{1\} & \{1\} \end{bmatrix} \text{ и } R_2[AC] = \begin{bmatrix} \{0\} & \{1\} \\ \{1\} & \{0\} \end{bmatrix}.$$

Им соответствуют гипотезы

$$H_1 = \overline{R_1}\,[AB] = \begin{bmatrix} \{1\} & \{1\} \\ \{0\} & \{0\} \end{bmatrix} = \begin{bmatrix} \{1\} & * \\ \{0\} & \{1\} \end{bmatrix} \cap \begin{bmatrix} \{0\} & * \\ \{1\} & \{0\} \end{bmatrix} = \begin{bmatrix} \{1\} & \{0\} \\ \{0\} & \{1\} \end{bmatrix},$$

$$H_2 = \overline{R_2}\,[AC] = \begin{bmatrix} \{0\} & \{1\} \\ \{1\} & \{0\} \end{bmatrix} = \begin{bmatrix} \{0\} & * \\ \{1\} & \{1\} \end{bmatrix} \cap \begin{bmatrix} \{1\} & * \\ \{0\} & \{0\} \end{bmatrix} = \begin{bmatrix} \{0\} & \{0\} \\ \{1\} & \{1\} \end{bmatrix}.$$

Первая из них означает эквивалентность высказываний *A* и $\overline{B}$, а вторая – *A* и *C*. Следовательно, путешественнику нужно сказать либо фразу: "Меня не сбросят со скалы", либо фразу "Меня растерзают львы".

Сформулированные выше алгоритм и правила формирования абдуктивных заключений применимы не только для АК-объектов, отображающих формулы исчисления высказываний, но и для общего случая, когда домены атрибутов имеют более двух значений. В конкретной системе знаний выбор переменных и их значений для абдуктивного заключения производится по критериям, связанным с содержанием системы. Предложенные алгоритмы позволяют более просто генерировать абдуктивные заключения с учетом ограничений, в частности, по составу и количеству переменных.

Глава 5. Управление данными и знаниями на базе алгебры кортежей

В каждой естественной науке заключено столько истины, сколько в ней есть математики.
Иммануил Кант.

Теоретические возможности алгебраического подхода, описанные в предшествующих главах, дают перспективные результаты в различных сферах обработки информации. Ниже представлены некоторые рассмотренные к настоящему времени приложения АК. Вначале освещены вопросы метризации операций с АК-объектами и их применение в вероятностных пространствах. Затем показаны АК-ориентированные способы управления данными и знаниями в современных программных системах: реляционных и дедуктивных СУБД, прикладных системах искусственного интеллекта, использующих семантические сети, аппарат формального анализа понятий, а также в поисковых системах. Изложения тем существенно отличаются друг от друга по степени завершенности, но авторы посчитали целесообразным привести их в книге для иллюстрации широты диапазона применения АК и приглашения исследователей, работающих в перечисленных и других предметных областях, к использованию алгебры кортежей для решения своих профессиональных проблем.

5.1. Метрические аспекты алгебры кортежей

5.1.1. Представление измеримых систем в алгебре кортежей

Понятие меры $\mu(A)$ множества A является естественным обобщением следующих понятий [*Колмогоров,* 1972]:

1) длины отрезка;
2) площади плоской фигуры;
3) объема пространственной фигуры;
4) приращения $f(b)-f(a)$ неубывающей функции $f(t)$ на полуинтервале $[a,b)$;
5) интеграла от неотрицательной функции по некоторой линейной, плоской или пространственной области.

Дополним этот список еще двумя разновидностями меры.

В качестве первой из них возьмем такую меру дискретных множеств, как количество элементов. Эту меру в теории множеств принято называть ***мощностью множеств***, но всеми свойствами меры (о них далее) обладают только мощности конечных множеств. Мощность множества A обозначается как $|A|$ (см. п.1.4).

Мера бесконечных множеств в теории множеств определена с помощью абстрактных понятий (мощность счетного множества, мощность континуума и т.д.), которые не позволяют различать мощности множеств в пределах одного класса, например, в классе счетных бесконечных множеств. Эту проблему можно решить, если ввести для бесконечных дискретных множеств или для дискретных множеств с потенциально бесконечным или просто чрезмерно большим числом элементов ***относительную мощность***, которая определяется как стремящееся к пределу отношение $\lim\limits_{n\to\infty}\dfrac{n(A)}{n(\boldsymbol{U})}$, где $n(\boldsymbol{U})$ – количество элементов возрастающего ряда универсума, $n(A)$ – количество элементов множества A, изменяющееся по мере возрастания $\boldsymbol{U}$. Например, если $\boldsymbol{U}$ – натуральный ряд, а A – множество чисел, кратных 3, то можно доказать, что относительная мощность A стремится к 1/3. Эта мера имеет много общих свойств с вероятностной мерой и легко может быть определена во многих случаях и для бесконечных подмножеств бесконечного множества. Например, относительная мощность подмножества четных чисел во множестве всех чисел натурального ряда равна 1/2, а относительная мощность всех чисел, кратных конечному целому числу k, равна $1/k$. Разумеется, с помощью относительной мощности различимы далеко не все подмножества счетных бесконечных множеств. Так, меру 0 имеют все конечные подмножества счетного множества и некоторые бесконечные множества, например, множество квадратов целых чисел. Для бесконечных подмножеств счетного множества, имеющих нулевую меру, можно использовать асимптотическую меру, т.е. аналитическую функцию, стремящуюся к нулю по мере возрастания ряда чисел, но имеющую конечное значение на конечном отрезке.

Возможны и другие типы мер. Но если работать в логической системе, изоморфной алгебре множеств, то должны соблюдаться следующие

Аксиомы меры μ ***множеств***:

1) мера μ есть действительное неотрицательное число;

2) если $A \cap B = \varnothing$, то $\mu(A \cap B) = \mu(A) + \mu(B)$;

3) для произвольных множеств A и B

$\mu(A \cup B) = \mu(A) + \mu(B) - \mu(A \cap B)$.

В качестве примера рассмотрим систему, универсум которой – бесконечное множество N положительных целых чисел. Ее подмножества есть числа, кратные каким-то целым конечным числам k. Обозначим такие подмножества N_k. Ясно, что относительная мощность универсума этой системы равна 1, а относительная мощность каждого из множеств N_k равна $1/k$. Предположим, нам необходимо определить относительную мощность подмножества, содержащего объединение всех чисел, кратных 2 (N_2), и чисел, кратных 3 (N_3). Из свойств меры множеств очевидно, что $\mu(N_2 \cup N_3) = \mu(N_2) + \mu(N_3) - \mu(N_2 \cap N_3)$.

Ясно, что $N_2 \cap N_3$ представляет собой множество чисел, кратных 6, поэтому $\mu(N_2 \cap N_3) = 1/6$. Отсюда

$$\mu(N_2 \cup N_3) = \frac{1}{2} + \frac{1}{3} - \frac{1}{6} = \frac{2}{3}.$$

В более общем случае:

$$\mu(N_m \cup N_k) = \frac{1}{m} + \frac{1}{k} - \frac{1}{R(m,k)},$$

где $R(m, k)$ – наименьшее общее кратное чисел m и k.

Опишем некоторые свойства меры в непрерывных подпространствах. Начнем с одномерного случая или с мер множеств на отдельных атрибутах такого подпространства.

Пусть на луче или отрезке расположена конечная система интервалов, каждый из которых имеет конечную меру. Интервалы заданы своими граничными точками (для некоторых разных интервалов какие-либо граничные точки могут совпадать). Будем считать, что все пространство состоит из двух типов ***элементарных интервалов***: открытых интервалов и точек, которые можно считать вырожденными интервалами. Нетрудно доказать, что в полученной системе любой входящий в нее изначально интервал можно представить как конечное множество элементарных интервалов. Примем в

качестве аксиомы, что мера каждого вырожденного элементарного интервала равна 0. Значит, при подсчете меры произвольного интервала достаточно учитывать только меры невырожденных элементарных интервалов.

В пространствах с мерой [*Халмош,* 1953] много трудностей связано с необходимостью соблюдения трех требований:

1) пересечение любых измеримых объектов должно принадлежать к классу измеримых объектов;

2) возможность представить любой объект как объединение непересекающихся объектов;

3) объединение всех измеримых объектов пространства должно быть в точности равно самому этому пространству, т.е. не должно быть при этом никаких пропусков, например, некоторых точек.

Для соблюдения первого требования традиционно [*Колмогоров,* 1972; *Халмош,* 1953] выбираются интервалы одного типа, например, только открытые или только полуоткрытые. Если взяты открытые интервалы, то не выполняется третье требование, поскольку точки при этом не учитываются. Если задать только замкнутые интервалы, то не удовлетворяется второе требование (сопряженные замкнутые интервалы имеют общую точку). Иногда применяют полуоткрытые интервалы [*Халмош,* 1953], но на практике это связано с определенными трудностями, и они используются редко. В то же время, если объединить в один класс открытые интервалы (пустой интервал тоже принадлежит этому классу) и добавить к ним ***вырожденные*** непустые интервалы с нулевой мерой (в одномерном пространстве – ***точки***), то можно легко построить систему, в которой все три требования выполняются.

Перейдем теперь к двумерному пространству. Пусть каждый домен атрибута системы отображен конечной системой интервалов. В таком случае система из двух атрибутов на плоскости будет представлена как объединение декартовых произведений соответствующих интервалов. Подобные декартовы произведения в двумерном и более пространстве называются гиперпрямоугольниками или ***брусами***. На плоскости вырожденными брусами являются не только точки, но и все открытые, замкнутые или полуоткрытые грани брусов и их объединения. Ясно, что все перечисленные вырожденные структуры на плоскости имеют нулевую меру. Но в соответствующей проекции их мера может отличаться от нуля. Рассмотрим с помощью АК отображение

описанных структур на плоскости (рис. 5.1).

Пусть в двумерном пространстве *XY* заданы интервалы, ограниченные точками *a, b, ..., h* на атрибуте *Y* и точками *i, j, ..., p* на атрибуте *X*. В качестве базовых примем следующие открытые интервалы:

на оси *Y*: (*a, c*); (*b*, *d*); (*e*, *g*); (*f*, *h*);

на оси *X*: (*i, k*); (*j*, *l*); (*m*, *o*); (*n*, *p*).

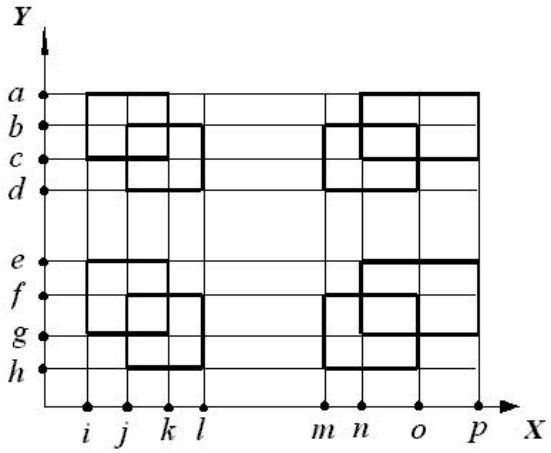

Рис. 5.1. Система брусов на плоскости

Систему *S* из 8-ми замкнутых брусов, изображенных на рис. 5.1, можно записать как объединение двух *C*-кортежей (или *C*-систему) в пространстве *XY*:

$$S[XY] = \begin{bmatrix} \{a,(a,c),c,e,(e,g),g\} & \{i,(i,k),k,n,(n,p),p\} \\ \{b,(b,d),d,f,(f,h),h\} & \{j,(j,l),l,m,(m,o),o\} \end{bmatrix}.$$

В системах с вырожденными интервалами соблюдаются все приведенные выше требования пространств с мерой, и в то же время имеется возможность преобразования их в систему с однородными элементами.

Каждая компонента системы *S* есть совокупность открытых интервалов и точек, представимая как совокупность замкнутых интервалов. Их декартово произведение дает уже систему замкнутых прямоугольников, изображенных на рис. 5.1. Извлекая из нее все точки, выделяем систему S_1 открытых брусов:

$$S_1 = \begin{bmatrix} \{(a,c),(e,g)\} & \{(i,k),(n,p)\} \\ \{(b,d),(f,h)\} & \{(j,l),(m.o)\} \end{bmatrix}, \text{ при этом очевидно, что } \mu(S) = \mu(S_1).$$

В системах *S* и S_1, где интервалы, формирующие компоненты, не пересекаются, сравнительно легко определяется мера каждого *C*-кортежа, исходя из свойств декартовых произведений (см. п. 1.4). Например, систему S_1 можно представить как объединение двух *C*-кортежей C_1 и C_2. Для *C*-кортежа

$C_1 = [\{(a, c), (e, g)\}\ \{(i, k), (n, p)\}]$

$\mu(C_1) = (\mu((a, c)) + \mu((e, g))) \times (\mu((i, k)) + \mu((n, p)))$.

Однако в более общем случае, когда компоненты *C*-кортежей состоят из пересекающихся интервалов, вычисление меры сильно затрудняется. Подобная ситуация возникает при необходимости вычислить меру *C*-системы на основе известных мер входящих в нее *C*-кортежей, имеющих непустые пересечения друг с другом. Тогда при вычислениях приходится вносить поправки, и система существенно усложняется. На рис. 5.1 пересечение двух *C*-кортежей, из которых состоит *C*-система S (или S_1), представлено в виде четырех малых прямоугольников, являющихся пересечением соответствующих пар больших прямоугольников. Вычисляя меру объединения этих же *C*-кортежей необходимо учитывать, что их пересечение непусто, и вводить соответствующие поправки. Для простых систем типа изображенной на рис. 5.1 можно при этом опираться на пространственное воображение. В более сложных системах с большим числом брусов в пространстве, имеющем большое число атрибутов, требуются более мощные формальные методы вычисления меры. Такие методы разработаны в алгебре кортежей, к ним, в частности, относится ортогонализация и метод квантования интервалов, к описанию которого мы и переходим.

Если в каждом атрибуте *C*-системы (*C*-кортежа) в компонентах содержатся произвольные интервалы числовой оси, то необходимо осуществить разбиение этой системы интервалов на элементарные интервалы, пересечение любой пары которых либо пусто, либо имеет единственную точку сочленения. Реализовать такое разбиение в случае, когда каждый интервал представлен парой концевых точек, можно с помощью простой процедуры, которую назовем ***методом квантования интервалов*** (МКИ).

Пусть область определения некоторого атрибута есть замкнутый интервал Ω на числовой оси и задано конечное множество $E = \{E_i\}$ замкнутых интервалов таких, что $\cup E_i \subseteq \Omega$. На числовой оси границы интервалов представлены множествами координат начальных и конечных точек. Если эти координаты расположить в порядке возрастания, то получим разбиение системы интервалов на ***кванты***, т.е. точки и открытые интервалы. Ясно, что при этом интервал Ω разбивается на некоторое комбинированное множество,

содержащее m открытых интервалов и $m + 1$ изолированных точек, из которых две являются концевыми точками интервала Ω.

Методологические трудности, связанные с тем, что множество состоит из разнородных объектов (точек и интервалов), можно обойти, если, как уже говорилось, определить точку как вырожденный интервал с нулевой мерой.

Пример процедуры квантования для четырех интервалов E_1, E_2, E_3, E_4, показан на рис. 5.2. Для наглядности интервалы E_i вынесены за пределы координатной оси.

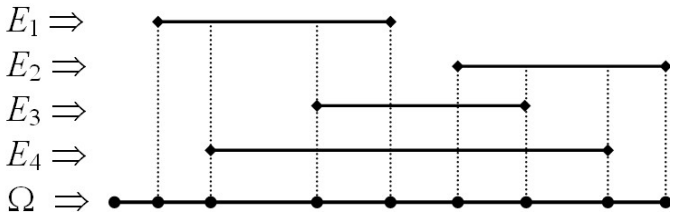

Рис. 5.2. Квантование системы интервалов

Здесь интервал Ω, ограниченный точками P и Q, содержит внутренние точки $a, b, ..., h$. Соответственно каждый интервал E_i может быть представлен множеством квантов, например, замкнутый интервал E_3 включает точки и открытые интервалы: $E_3 = \{c, d, e, f, (c, d), (d, e), (e, f)\}$. Если нас интересуют только метрические свойства описываемых объектов, то E_3 можно представить как множество $\{(c, d), (d, e), (e, f)\}$ открытых непересекающихся интервалов. После аналогичного квантования в каждом измеримом атрибуте метрическое пространство преобразуется в АК-объект, его компоненты – обычные множества, содержащие открытые интервалы и точки.

Дадим некоторые определения.

Несовместными называются интервалы при условии, что их пересечение пусто или характеризуется нулевой мерой.

Например, замкнутые интервалы $[c, d]$ и $[d, e]$ из приведенного выше примера имеют непустое пересечение (точка d), но представляют несовместную пару, поскольку их пересечение имеет меру 0.

Элементаризация (квантование) непрерывного или дискретно-непрерывного пространства есть изоморфное преобразование систем интервалов каждого непрерывного атрибута в системы попарно несовместных интервалов.

При преобразовании произвольной системы интервалов (Ω, E) отдельного атрибута в элементарную систему интервалов (Ω, F), где $E = \{E_i\}$ – множество

исходных интервалов, а $F = \{F_r\}$ – множество квантов, для любого E_i соблюдается соотношение

$$E_i = \bigcup_k F_k ,$$

где F_k – некоторые кванты из F. Причем мера μ для исходного интервала E_i вычисляется по формуле (в силу аддитивности меры):

$$\mu(E_i) = \sum_k \mu(F_k) . \qquad (5.1)$$

Из сказанного следует важный для дальнейшего изложения вывод: если каждая переменная логической системы допускает элементаризацию, то во всех промежуточных преобразованиях и сравнениях мер, заданных в непрерывных или дискретно-непрерывных пространствах с конечной системой интервалов в каждом непрерывном атрибуте, можно использовать изоморфные АК-объекты с атрибутами, представленными конечными множествами.

Теорема 5.1. Если каждая компонента *C*-кортежа имеет конечную меру, то мерой *C*-кортежа является произведение мер его компонент.

Доказательство. Для доказательства достаточно рассмотреть *C*-кортеж, состоящий из двух компонент. Случай, когда каждая из компонент состоит из единственного непрерывного интервала, тривиален, так как здесь утверждение теоремы непосредственно следует из метрических свойств декартового произведения. Рассмотрим случай, когда каждая компонента представляет собой объединение конечного числа интервалов. Тогда, применив МКИ, преобразуем системы E_1 и E_2 интервалов в каждом атрибуте в конечные системы F_1 и F_2 открытых элементарных интервалов. Таким образом, *C*-кортеж равен декартову произведению $F_1 \times F_2$. Ясно, что каждый элемент (F_{1i}, F_{2j}) этого декартова произведения представлен открытым прямоугольником G_{ij} с мерой $\mu(G_{ij}) = \mu(F_{1i}) * \mu(F_{2j})$. Поскольку все G_{ij} попарно не пересекаются, то мера *C*-кортежа будет равна сумме их мер $\sum\mu(G_{ij})$. Кроме того, из (5.1) следует, что $\mu(E_1) = \sum\mu(F_{1i})$ и $\mu(E_2) = \sum\mu(F_{2j})$. Значит, $\mu(E_1 \times E_2) = \sum\mu(G_{ij}) = \mu(E_1) * \mu(E_2)$. Случай, когда *C*-кортеж имеет большее число атрибутов, доказывается аналогично, при этом структура пространственных элементарных объектов (точки, линии, *k*-мерные брусы) определяется числом непрерывных атрибутов. *Конец доказательства*.

Из доказанного непосредственно следует

Теорема 5.2. Мера ортогональной *C*-системы равна сумме мер содержащихся в ней *C*-кортежей.

Из теоремы 5.2 вытекает, что мера произвольной *D*-системы может быть вычислена, если преобразовать ее дискретное представление в ортогональную *C*-систему. Для произвольной *C*-системы можно, используя соотношения АК, разработать алгоритм преобразования такой *C*-системы в ортогональную, но во многих случаях проще вычислить по теореме 2.8 дополнение этой *C*-системы, преобразовать полученную *D*-систему в ортогональную *C*-систему и вычислить меру исходной *C*-системы, используя соотношение

$$\mu(C) = \mu(\boldsymbol{U}) - \mu(\overline{C}), \qquad (5.2)$$

где $\mu(C)$ – мера исходной С-системы, $\mu(\boldsymbol{U})$ – мера универсума, $\mu(\overline{C})$ – мера АК-объекта, дополнительного к исходному. Мера универсума для множества атрибутов $\{X_1, X_2, ..., X_n\}$ вычисляется как произведение

$$\mu(X_1) * \mu(X_2) * ... * \mu(X_n).$$

Отметим следующие очевидные свойства АК-объектов, погруженных в вероятностное пространство или в любое другое нормированное пространство, в котором мера каждого атрибута равна 1:

a) мера полной компоненты (∗) в *C*-кортежах и *C*-системах равна 1;

b) мера любого частного универсума равна 1;

c) мера любого АК-объекта есть число в интервале [0, 1];

d) мера пустого АК-объекта равна 0;

e) для произвольного АК-объекта *A* мера его дополнения равна $1 - \mu(A)$;

f) для пары (*A*, *B*) АК-объектов $\mu(A \cup B) = \mu(A) + \mu(B) - \mu(A \cap B)$;

g) в любом АК-объекте после элементаризации мера каждой компоненты равна сумме мер соответствующих квантов, содержащихся в этой компоненте.

5.1.2. Логико-вероятностный анализ и алгебра кортежей

В настоящее время ***логико-вероятностный анализ*** (ЛВА) надежности и безопасности структурно сложных систем, разработанный И.А. Рябининым и его учениками, широко используется как в теоретических исследованиях, так и в практических приложениях [*Рябинин*, 1981; 2000; *Соложенцев*, 2004]. Однако в настоящее время ЛВА применяется в основном в моделях исчисления высказываний, и при переходе к более сложным моделям со многими состояниями приходится использовать методики, предназначенные для частных случаев. Имеются программные реализации логико-вероятностного анализа, но в них теряется прозрачность и доказательность используемых алгоритмов. Нередко сами алгоритмы, лежащие в основе программного обеспечения, приобретают статус "ноу-хау", в силу чего их суть в публикациях не раскрывается. Поэтому возникает необходимость в разработке методики, позволяющей с единых теоретических позиций подойти к созданию алгоритмов анализа структурно сложных систем. Применение АК позволяет решить перечисленные выше проблемы.

В ЛВА обычно в качестве исходных данных при вероятностных расчетах используются не логические формулы, а структурные схемы, отображающие причинно-следственные связи между множествами элементарных событий в системе. Если такие схемы могут быть представлены логическими формулами, то они легко преобразуются в вероятностные модели АК [*Кулик*, 2007 а]. В качестве примера рассмотрим двухполюсник (рис. 5.3) с мостиковой схемой.

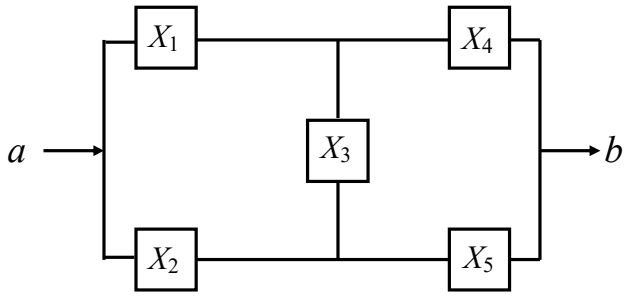

Рис. 5.3. Мостиковая схема

Такая схема, в частности, описывает систему энергоснабжения какого-либо объекта, в которой X_1 и X_2 – источники энергии, X_3 – распределительный щит, X_4 и X_5 – потребители. Здесь элемент X_3 выполняет роль переключателя, в силу чего между полюсами a и b допустимы только следующие пути: X_1X_4, X_2X_5, $X_1X_3X_5$, $X_2X_3X_4$. Это множество путей можно представить как ДНФ:

$$F = (X_1 \wedge X_4) \vee (X_2 \wedge X_5) \vee (X_2 \wedge X_3 \wedge X_4) \vee (X_1 \wedge X_3 \wedge X_5). \qquad (5.3)$$

Логическая функция F, связывающая состояние системы с состоянием элементов, в ЛВА называется ***функцией работоспособности системы*** (ФРС). Пусть каждый элемент системы имеет два состояния (работоспособен и неисправен) и известны вероятности p_i безотказной работы всех элементов. Требуется по ФРС определить вероятность безотказной работы всей системы.

В АК функцию F можно отобразить как C-систему в универсуме $X_1 \times X_2 \times X_3 \times X_4 \times X_5 = \{0, 1\}^5$:

$$R = \begin{bmatrix} 1 & * & * & 1 & * \\ * & 1 & * & * & 1 \\ * & 1 & 1 & 1 & * \\ 1 & * & 1 & * & 1 \end{bmatrix},$$

а ее отрицание – как D-систему

$$\overline{R} = \begin{bmatrix} 0 & \varnothing & \varnothing & 0 & \varnothing \\ \varnothing & 0 & \varnothing & \varnothing & 0 \\ \varnothing & 0 & 0 & 0 & \varnothing \\ 0 & \varnothing & 0 & \varnothing & 0 \end{bmatrix}.$$

Используя описанные в предыдущих разделах методы, преобразуем D-систему $\overline{R}$ в ортогональную C-систему (для упрощения компоненты {0} и {1} далее записываются как 0 и 1):

$$\overline{R} = \begin{bmatrix} 0 & * & * & * & * \\ 1 & * & * & 0 & * \end{bmatrix} \cap \begin{bmatrix} * & 0 & * & * & * \\ * & 1 & * & * & 0 \end{bmatrix} \cap \begin{bmatrix} * & 0 & * & * & * \\ * & 1 & * & 0 & * \\ * & 1 & 0 & 1 & * \end{bmatrix} \cap$$

$$\cap \begin{bmatrix} 0 & * & * & * & * \\ 1 & * & * & * & 0 \\ 1 & * & 0 & * & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & * & * & * \\ 0 & 1 & * & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & * & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & * & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & * & * & * \\ * & 1 & * & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & * & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

(в последней системе второй и шестой C-кортежи объединены).

Поскольку уже известно распределение вероятностей событий в атрибутах (p_i – вероятность безотказной работы элемента; $q_i = 1 - p_i$ – вероятность отказа), то можно сразу, используя ортогональную C-систему $\overline{R}$, написать формулу для

расчета вероятности безотказной работы системы:

$P(R) = 1 - P(\overline{R}) = 1 - (q_1q_2 + p_2q_4q_5 + q_1p_2q_3p_4q_5 + p_1q_2q_4q_5 + p_1q_2q_3q_4p_5)$.

Если речь идет о системе энергоснабжения, то из ФРС системы (5.3) следует, что система работоспособной даже тогда, когда работает хотя бы один потребитель. Но, допустим, по условию система работоспособна только если получают энергию и работают оба потребителя. В этом случае необходимо изменить ФРС:

$F_1 = F \wedge X_4 \wedge X_5$.

Тогда в структурах АК имеем:

$$R_1 = R \cap [* * * 1\ 1] = \begin{bmatrix} 1 & * & * & 1 & 1 \\ * & 1 & * & 1 & 1 \\ * & 1 & 1 & 1 & 1 \\ 1 & * & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & * & * & 1 & 1 \\ * & 1 & * & 1 & 1 \end{bmatrix}.$$

$$\text{Отсюда } \overline{R_1} = \begin{bmatrix} 0 & \varnothing & \varnothing & 0 & 0 \\ \varnothing & 0 & \varnothing & 0 & 0 \end{bmatrix} = \begin{bmatrix} * & * & * & 0 & * \\ * & * & * & 1 & 0 \\ 0 & * & * & 1 & 1 \end{bmatrix} \cap \begin{bmatrix} * & * & * & 0 & * \\ * & * & * & 1 & 0 \\ * & 0 & * & 1 & 1 \end{bmatrix}.$$

Выполнив вычисления, получим ортогональную *C*-систему:

$$\overline{R_1} = \begin{bmatrix} * & * & * & 0 & * \\ * & * & * & 1 & 0 \\ 0 & 0 & * & 1 & 1 \end{bmatrix}.$$

На основе описанного решения получаем расчетную формулу вероятности безотказной работы:

$F_1 = 1 - (q_4 + p_4q_5 + q_1q_2p_4p_5)$.

Для этой ситуации можно сформулировать и более жесткие ограничения. Например, мощность энергии источника X_2 недостаточна для работы двух потребителей. Значит, в систему необходимо еще ввести дополнительное ограничение: если не работает источник X_2, то потребителя X_5 необходимо отключить. Это условие можно выразить в виде формулы $\overline{X_2} \supset \overline{X_5} = X_2 \vee \overline{X_5}$, представимую в виде *D*-кортежа и ортогональной *C*-системы:

$$C =]\varnothing\ 1\ \varnothing\ \varnothing\ 0[= \begin{bmatrix} * & 1 & * & * & * \\ * & 0 & * & * & 0 \end{bmatrix}.$$

Введем это ограничение в систему R_1:

$$R_2 = R \cap [* * * 1\ 1] \cap \begin{bmatrix} * & 1 & * & * & * \\ * & 0 & * & * & 0 \end{bmatrix}.$$

Вычислим вначале пересечение $[* * * 1\ 1] \cap \begin{bmatrix} * & 1 & * & * & * \\ * & 0 & * & * & 0 \end{bmatrix} = [* 1 * 1\ 1]$.

После чего найдем:

$$R_2 = R \cap [* 1 * 1\ 1] = \begin{bmatrix} 1 & 1 & * & 1 & 1 \\ * & 1 & * & 1 & 1 \\ * & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Нетрудно убедиться, что в полученной С-системе второй *С*-кортеж включает в себя все остальные *С*-кортежи, т.е. $R_2 = [* 1 * 1\ 1]$, и вероятность безотказной работы системы при заданных ограничениях можно вычислить по формуле (здесь ортогонализации не требуется, так как система выражена единственным *С*-кортежем) $P(R_2) = p_2p_4p_5$.

В ЛВА пока не разработан единый подход к анализу и оценке вероятностных характеристик ***систем со многими состояниями***, в то время как в АК эта проблема в общем случае решена (см. п. 2.4).

Пример 5.1. Рассмотрим мостиковую схему из рис. 5.3, но теперь некоторые элементы имеют не 2, а 3 состояния. Система задана в универсуме

$$X_1 \times X_2 \times X_3 \times X_4 \times X_5 = \{a_1, a_2\} \times \{b_1, b_2, b_3\} \times \{c_1, c_2, c_3\} \times \{d_1, d_2, d_3\} \times \{e_1, e_2, e_3\}.$$

Модель такой системы в АК имеет вид:

$$Q = \begin{bmatrix} \{a_1\} & * & * & * & \{e_1, e_2\} \\ * & \{b_1, b_2\} & * & \{d_1, d_2\} & * \\ * & \{b_2, b_3\} & \{c_1, c_2\} & * & (e_2, e_3\} \\ \{a_2\} & * & \{c_2, c_3\} & \{d_3\} & * \end{bmatrix}.$$

Распределение вероятностей элементарных событий приведено в табл. 5.1:

Таблица 5.1.

X_1		X_2			X_3			X_4			X_5		
a_1	a_2	b_1	b_2	b_3	c_1	c_2	c_3	d_1	d_2	d_3	e_1	e_2	e_3
0.6	0.4	0.5	0.2	0.3	0.4	0.3	0.3	0.4	0.2	0.4	0.7	0.2	0.1

Необходимо выполнить расчет вероятностей событий Q и $\forall x_2(Q)$.

Для вычисления $P(Q)$ сначала найдем дополнение Q:

$$\overline{Q} = \begin{bmatrix} \{a_2\} & \varnothing & \varnothing & \varnothing & \{e_3\} \\ \varnothing & \{b_3\} & \varnothing & \{d_3\} & \varnothing \\ \varnothing & \{b_1\} & \{c_3\} & \varnothing & \{e_1\} \\ \{a_1\} & \varnothing & \{c_1\} & \{d_1, d_2\} & \varnothing \end{bmatrix}.$$

Используя методы п. 3.1, преобразуем $\overline{Q}$ в ортогональную C-систему:

$$\overline{Q} = \begin{bmatrix} \{a_2\} & \{b_1\} & \{c_1\} & \{d_3\} & * \\ \{a_1\} & \{b_1\} & * & \{d_3\} & \{e_3\} \\ \{a_2\} & \{b_2, b_3\} & \{c_1\} & \{d_3\} & \{e_1\} \\ \{a_2\} & \{b_3\} & * & \{d_1, d_2\} & \{e_1\} \\ \{a_1\} & \{b_2, b_3\} & \{c_3\} & \{d_3\} & \{e_3\} \\ \{a_2\} & \{b_3\} & \{c_3\} & \{d_1, d_2\} & \{e_2, e_3\} \\ \{a_1\} & \{b_3\} & \{c_3\} & \{d_1, d_2\} & \{e_3\} \end{bmatrix}.$$

Подставляя в эту матрицу значения вероятностей из таблицы, получим значение $P(\overline{Q}) = 0.13012$. Отсюда $P(Q) = 1 - P(\overline{Q}) = 0.86988$. Заметим, что вероятности сложных компонент, например, $\{b_2, b_3\}$, равны сумме вероятностей элементов этих компонент.

Чтобы вычислить $P(\forall x_2(Q))$, можно воспользоваться алгоритмом, приведенным в п. 4.5. Тогда:

$$\forall x_2(Q) = \begin{bmatrix} \{a_1\} & * & * & * & \{e_1, e_2\} \\ \{a_2\} & * & \{c_2, c_3\} & \{d_3\} & * \\ * & * & \{c_1, c_2\} & \{d_1, d_2\} & \{e_2, e_3\} \end{bmatrix}.$$

Полученная C-система не ортогональна. Можно воспользоваться уже известным методом и ортогонализовать дополнение этой C-системы. Но в данном случае можно поступить проще. Проанализируем множество $\{C_1, C_2, C_3\}$ C-кортежей полученной C-системы. Среди них только одна пара $\{C_1, C_3\}$ не ортогональна. Найдем пересечение этих C-кортежей:

$C_1 \cap C_3 = [\{a_1\} * \{c_1, c_2\}\ \{d_1, d_2\}\ \{e_2\}]$.

Следовательно, можно применить формулу:

$P(\forall x_2(Q)) = P(C_1) + P(C_2) + P(C_3) - P(C_1 \cap C_3)$.

После подстановки значений вероятностей элементарных событий в

формулу получим:

$P(\forall x_2(Q)) = 0.6864$.

Опишем систему, где события заданы пересекающимися интервалами.

Пример 5.2. Система задана в пространстве $X \times Y$, где атрибуты представлены в виде интервалов, при этом $X = [0, 7]$ и $Y = [0, 5]$. Заданы также плотности распределения вероятностей $f_1(x, d_1, e_1)$ и $f_2(y, d_2, e_2)$ на атрибутах, где d_1, e_1, d_2, e_2 – параметры распределений. В этой системе задано составное событие в виде АК-объекта

$$R[XY] = \begin{bmatrix} \{a_1\} & \{b_1\} \\ \{a_2, a_4\} & \{b_2\} \\ \{a_3\} & \{b_3\} \end{bmatrix},$$

где a_i, b_j – интервалы, заданные в табл. 5.2:

Таблица 5.2.

a_1	a_2	a_3	a_4	b_1	b_2	b_3
[0, 2.8]	[1.7, 3.4]	[3.4, 5.5]	[4.3, 6.4]	[0, 2.3]	[1.4, 3.2]	[2.3, 5.0]

Необходимо определить последовательность расчетов для вычисления вероятности событий R и $\forall y(\overline{R})$.

Чтобы решить задачу, достаточно использовать только открытые интервалы. Для элементаризации системы построим возрастающие ряды границ интервалов в атрибутах:

для X: 0; 1.7; 2.8; 4.4; 4.3; 5.5; 6.4; 7;

для Y: 0; 1.4; 2.3; 4.2; 5.

После этого получим следующие множества элементарных интервалов (таблицы 5.3 и 5.4):

для X:

Таблица 5.3.

r_1	r_2	r_3	r_4	r_5	r_6	r_7
(0, 1.7)	(1.7, 2.8)	(2.8, 3.4)	(3.4, 4.3)	(4.3, 5.5)	(5.5, 6.4)	(6.4, 7)

для Y:

Таблица 5.4.

q_1	q_2	q_3	q_4
(0, 1.4)	(1.4, 2.3)	(2.3, 3.2)	(3.2, 5)

Соответственно:

$a_1 = \{r_1, r_2\}$; $a_2 = \{r_2, r_3\}$; $a_3 = \{r_4, r_5\}$; $a_4 = \{r_5, r_6\}$;
$b_1 = \{q_1, q_2\}$; $b_2 = \{q_2, q_3\}$; $b_3 = \{q_3, q_4\}$.

Тогда после подстановки найдем:

$$R = \begin{bmatrix} \{r_1, r_2\} & \{q_1, q_2\} \\ \{r_2, r_3, r_5, r_6\} & \{q_2, q_3\} \\ \{r_4, r_5\} & \{q_3, q_4\} \end{bmatrix}.$$

Для каждого кванта r_i или q_j вычислим соответствующую вероятность. Например,

$$P(r_3) = \int_{2,8}^{3,4} f_1(x, d_1, e_1).$$

Теперь можно приступить к ортогонализации соответствующих сложных событий. Для события R вычислим $\overline{R}$ и, после преобразования $\overline{R}$ в ортогональную C-систему, найдем $P(R) = 1 - P(\overline{R})$.

$$\overline{R} = \begin{bmatrix} \{r_3, r_4, r_5, r_6, r_7\} & \{q_3, q_4\} \\ \{r_1, r_4, r_7\} & \{q_1, q_4\} \\ \{r_1, r_2, r_3, r_6, r_7\} & \{q_1, q_2\} \end{bmatrix} = \begin{bmatrix} * & \{q_3, q_4\} \\ \{r_3, r_4, r_5, r_6, r_7\} & \{q_1, q_2\} \end{bmatrix} \cap$$

$$\cap \begin{bmatrix} * & \{q_1, q_4\} \\ \{r_1, r_4, r_7\} & \{q_2, q_3\} \end{bmatrix} \cap \begin{bmatrix} * & \{q_1, q_2\} \\ \{r_1, r_2, r_3, r_6, r_7\} & \{q_3, q_4\} \end{bmatrix} = \begin{bmatrix} \{r_3, r_4, r_5, r_6, r_7\} & \{q_1\} \\ \{r_4, r_7\} & \{q_2\} \\ \{r_1, r_7\} & \{q_3\} \\ \{r_1, r_2, r_3, r_6, r_7\} & \{q_4\} \end{bmatrix}.$$

Подставляя вероятности квантов и используя теоремы 5.1 и 5.2, имеем:

$P(R) = 1 - P(\overline{R}) = 1 - ((P(r_3) + P(r_4) + P(r_5) + P(r_6) + P(r_7))P(q_1) + (P(r_4) +$
$+ P(r_7))P(q_2) + (P(r_1) + P(r_7))P(q_3) + (P(r_1) + P(r_2) + P(r_3) + P(r_6) + P(r_7))P(q_4))$.

Далее, вычисляем:

$$\forall y(\overline{R}) = -Y(\overline{R}[XY]) = \begin{bmatrix} \{r_3, r_4, r_5, r_6, r_7\} \\ \{r_1, r_4, r_7\} \\ \{r_1, r_2, r_3, r_6, r_7\} \end{bmatrix} = \{r_7\}.$$

Отсюда $P(\forall y(\overline{R})) = P(r_7)$.

В данном примере сравнительно легко непосредственно преобразовать C-систему R в ортогональную, используя то обстоятельство, что она представляет бинарное отношение и имеет в атрибуте Y сравнительно немного

компонент. Учитывая это, разложим *C*-систему *R* так, чтобы в атрибуте *Y* были только одноэлементные множества, а затем объединим *C*-кортежи, у которых компоненты в атрибуте *Y* одинаковы. Находим:

$$R = \begin{bmatrix} \{r_1, r_2\} & \{q_1, q_2\} \\ \{r_2, r_3, r_5, r_6\} & \{q_2, q_3\} \\ \{r_4, r_5\} & \{q_3, q_4\} \end{bmatrix} = \begin{bmatrix} \{r_1, r_2\} & q_1 \\ \{r_1, r_2\} & q_2 \\ \{r_2, r_3, r_5, r_6\} & q_2 \\ \{r_2, r_3, r_5, r_6\} & q_3 \\ \{r_4, r_5\} & q_3 \\ \{r_4, r_5\} & q_4 \end{bmatrix} = \begin{bmatrix} \{r_1, r_2\} & q_1 \\ \{r_1, r_2, r_3, r_5, r_6\} & q_2 \\ \{r_2, r_3, r_4, r_5, r_6\} & q_3 \\ \{r_4, r_5\} & q_4 \end{bmatrix}.$$

Тогда можно получить другую, несколько более простую, формулу вероятности события *R*:

$$P(R) = (P(r_1) + P(r_2))P(q_1) + (P(r_1) + P(r_2) + P(r_3) + P(r_5) + P(r_6))P(q_2) +$$
$$+ (P(r_2) + P(r_3) + P(r_4) + P(r_5) + P(r_6))P(q_3) + (P(r_4) + P(r_5))P(q_4).$$

5.1.3. Вероятностная логика на основе АК

Термин "вероятностная логика" получил широкое распространение в работах по искусственному интеллекту после опубликования статьи известного специалиста по искусственному интеллекту Н. Нильсона [*Nilsson,* 1986]. Его идея была продолжена другими исследователями. В публикациях по вероятностной логике ставилась следующая задача: заданы оценки вероятностей некоторого множества событий, представленных формулами исчисления высказываний, необходимо найти вероятностную оценку события, заданного логической формулой, отличающейся от исходных. Другой аспект совмещения вероятности и логики – логико-вероятностный анализ [*Рябинин*, 1981], где вероятность формул вычисляется по вероятностям значений логических переменных, – в этих работах не рассматривался. Кроме того, анализ работ по вероятностной логике показывает, что в них результатом соединения классических понятий "вероятность" и "логика" оказываются некоторые неклассические логики. Далее рассматривается концепция вероятностной логики в классическом варианте через призму алгебры кортежей.

Совмещение понятий "логика" и "вероятность" вызывает немало трудностей. На первый взгляд, здесь все просто, если взять за основу

аксиоматику вероятности, предложенную А.Н. Колмогоровым [*Колмогоров,* 1972], в которой алгебра событий, погруженных в вероятностную меру, соответствует алгебре множеств. Например, для событий, представленных множествами A и B, вероятностная мера их объединения вычисляется по формуле:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B).$$

Таким образом, для точного вычисления вероятности события $(A \cup B)$ помимо вероятностей $P(A)$ и $P(B)$ необходимо знать вероятность $P(A \cap B)$, в рамках некоторых ограничений (в частности, $P(A \cap B) \leq min(P(A), P(B))$) не зависящую от $P(A)$ и $P(B)$. Если при этом $A \cap B \neq \varnothing$, то события A и B зависимы. Но когда A и B являются не множествами, а разными логическими переменными исчисления высказываний, то вероятность дизъюнкции этих событий вычисляется по формуле:

$$P(A \vee B) = P(A) + P(B) - P(A)P(B),$$

для вычисления которой достаточно задать только вероятность событий A и B.

Возникает вопрос: почему для логических соотношений имеет место другая методика расчета вероятности, хотя многим представляется, что алгебра множеств и булева алгебра изоморфны? Ответ на него оказывается ключевым при совмещении понятий "логика" и "вероятность". Дело в том, что в классической логике элементарные события, соответствующие разным логическим переменным, несовместимы, потому что любая логическая формула, содержащая n свободных переменных, изоморфна некоторому n-местному отношению, и события, соответствующие различающимся переменным, принадлежат разным атрибутам. Другими словами, логические переменные могут быть зависимыми, но не изначально, а только потому, что они содержатся в некоторой логической формуле, которая и определяет зависимость между ними.

Абсурдность (с точки зрения математической логики) иного подхода видна из следующего примера, который иногда приводится в работах по вероятностной логике: для логических переменных (но не формул!) X и Y даются вероятности $P(X)$, $P(Y)$ и $P(X \wedge Y)$, причем последняя вероятность не обязательно равна произведению предыдущих.

Значит, при погружении логических систем в вероятностное пространство

необходимо учитывать, что в соответствии с аксиоматикой А.Н. Колмогорова они изоморфны алгебре множеств, причем сами множества по структуре есть множества кортежей, содержащихся в многоместных отношениях.

Именно это обстоятельство, явно или неявно учитываемое в АК и ЛВА, не учитывается в различных версиях "вероятностной логики". Предположение о том, что события, соответствующие разным логическим переменным, зависимы сами по себе, т.е. без учета связывающей их логической формулы, означает нарушение законов математической логики. Другое дело, когда речь идет о самих формулах, в которых устанавливается зависимость между различными переменными, или о разных логических формулах, которые могут быть зависимыми только при условии, что они содержат хотя бы одну общую для них свободную переменную.

Изложенный выше подход соответствует ***прямой задаче*** логико-вероятностного анализа, когда при заданных вероятностях элементарных событий выполняется расчет вероятности сложного события. В ***обратной задаче*** постановка иная – на основе данных о вероятностях некоторых сложных событий нужно найти вероятности элементарных событий, после чего можно рассчитать вероятности других сложных событий. К обратным относятся задачи, решаемые в вероятностной логике. Рассмотрим пример, приведенный в статье известного специалиста по вероятностной логике Н. Нильсона [*Nilsson,* 1986].

Пример 5.3. Дана совокупность событий, заданных формулами A и $A \supset B$ исчисления высказываний, при этом $P(A) = p_1$ и $P(A \supset B) = p_2$. Требуется оценить вероятность $P(B)$ события B.

В [*Nilsson,* 1986] для решения этой и аналогичных задач предложена теория, использующая геометрические построения и концепции возможных миров. Покажем, как данная задача решается с помощью АК.

В задаче имеются всего две логические переменные (A, B), которые можно считать элементарными событиями. Предположим, что вероятность этих событий равна соответственно $P(A)$ и $P(B)$. В условии задачи сказано, что $P(A) = p_1$. Выразим заданные формулы в структурах АК, используя универсум $A \times B = \{0, 1\}^2$:

$$A = [\{1\}\ *];\ B = [*\ \{1\}];\ A \supset B = \overline{A} \vee B =]\{0\}\ \{1\}[= \begin{bmatrix} \{0\} & * \\ \{1\} & \{1\} \end{bmatrix}$$

(здесь *D*-кортеж, соответствующий формуле $A \supset B$, преобразован в ортогональную *C*-систему).

На основании этого найдем формулы вероятностей событий A и $A \supset B$:

$P(A) = p_1$; $P(A \supset B) = (1 - P(A)) + P(A)P(B) = p_2$.

Получилась система из двух уравнений:

$P(A) = p_1$;

$(1 - P(A)) + P(A)P(B) = p_2$,

из которой несложно вывести:

$$P(B) = \frac{p_1 + p_2 - 1}{p_1}.$$

Это точный ответ, в то время как в [*Nilsson,* 1986] ответ получен как интервальная оценка:

$$p_2 + p_1 - 1 \le P(B) \le p_2.$$

Ответ, полученный методами АК, также позволяет по значениям заданных вероятностей событий оценить допустимость или недопустимость этих событий. Это можно сделать, используя неравенства:

$$p_1 + p_2 - 1 \ge 0 \text{ и } p_1 + p_2 - 1 \le p_1.$$

Из них следует, что события допустимы при условии $p_1 + p_2 \ge 1$. Второе неравенство после преобразования становится очевидным: $p_1 \le 1$. Таким образом, в ответе, полученном Н. Нильсоном, верхняя граница для $P(B)$ избыточна.

В общем случае ***алгоритм решения задач вероятностной логики*** следующий. Пусть заданы исходные логические формулы F_i с известными вероятностями $P(F_i)$ и формула G, вероятность которой $P(G)$ требуется вычислить. Тогда необходимо выполнить следующую последовательность операций:

1) формулы F_i и G преобразуются в ортогональные *C*-системы;

2) для каждой из этих систем составляется уравнение регрессии $E(F_i)$ и $E(G)$;

3) составляется и решается система уравнений $\{E(F_i)\}$;

4) если система уравнений $\{E(F_i)\}$ имеет единственное решение, то полученные значения переменных подставляются в формулу $E(G)$ и находится точный ответ.

Приведем более сложный пример.

Пример 5.4. Даны вероятности событий, описанных формулами исчисления высказываний: $P(A \supset \overline{C}) = p_1$; $P((A \wedge \overline{B}) \vee B) = p_2$; $P(A \vee \overline{B}) = p_3$. Требуется найти вероятность события $(A \vee C)$.

Выразим заданные события в терминах АК:

$$A \supset \overline{C} = \overline{A} \vee \overline{C} \Leftrightarrow]\{0\}\ \varnothing\ \{0\}[= \begin{bmatrix} \{0\} & * & * \\ \{1\} & * & \{0\} \end{bmatrix};$$

$$(A \wedge \overline{B}) \vee B \Leftrightarrow \begin{bmatrix} \{1\} & \{0\} & * \\ * & \{1\} & * \end{bmatrix};$$

$$A \vee \overline{B} \Leftrightarrow]\{1\}\ \{0\}\ \varnothing[= \begin{bmatrix} \{1\} & * & * \\ \{0\} & \{0\} & * \end{bmatrix};$$

$$A \vee C \Leftrightarrow]\{1\}\ \varnothing\ \{1\}[= \begin{bmatrix} \{1\} & * & * \\ \{0\} & * & \{1\} \end{bmatrix}.$$

Пусть $x = P(A)$; $y = P(B)$; $z = P(C)$. Тогда для первых трех событий получим следующую систему уравнений:

$(1 - x) + x(1 - z) = p_1$;

$x(1 - y) + y = p_2$;

$x + (1 - x)(1 - y) = p_3$.

Результатом решения этой системы будет:

$$x = p_2 + p_3 - 1;\ y = \frac{1 - p_3}{2 - p_2 - p_3};\ z = \frac{1 - p_1}{p_2 + p_3 - 1}.$$

Далее построим уравнение регрессии для события $A \vee C$: $x + (1 - x)z$. Подставляя в это уравнение значения вычисленных вероятностей, находим:

$$P(A \vee C) = \frac{1 - p_1(2 - p_2 - p_3)}{p_2 + p_3 - 1}.$$

В примерах 5.3 и 5.4 был получен точный ответ. Но такая ситуация возможна не всегда, в частности, когда число полученных уравнений меньше числа переменных. Но неопределенности возможны и тогда, когда количество уравнений и переменных одинаково. Рассмотрим пример.

Пример 5.5. Пусть вероятности событий заданы логическими формулами:

$P(A \vee B) = a$; $P(A \wedge B) = b$.

Требуется найти оценку $P(A)$ и $P(B)$. Выразим данные события в системе как ортогональные C-системы:

$A \vee B \Leftrightarrow]\{1\}\ \{1\}[= \begin{bmatrix} \{1\} & * \\ \{0\} & \{1\} \end{bmatrix}$;

$A \wedge B \Leftrightarrow [\{1\}\ \{1\}]$.

Составим систему уравнений:

$P(A) + (1 - P(A))\, P(B) = a$;

$P(A)P(B) = b$.

Решая данную систему уравнений, получим

$$P(A) = \frac{a + b \pm \sqrt{(a+b)^2 - 4b}}{2};\ P(B) = \frac{a + b \mp \sqrt{(a+b)^2 - 4b}}{2}.$$

Видно, что полученные решения не дают однозначного ответа в тех случаях, когда подкоренное выражение не равно 0 (это возможно для случая $P(A) \neq P(B)$).

Для приведенных примеров можно численно проверить выкладки, если построить соответствующие им вероятностные модели. Так, для примера 5.5 вероятностная модель такова: пусть одновременно бросаются две монеты, причем известна вероятность выпадения хотя бы одного герба (этому событию соответствует формула $A \vee B$) и вероятность выпадения герба в двух бросаниях (формула $A \wedge B$). Зная вероятность выпадения герба (для правильных монет – 0.5), нетрудно по законам теории вероятности подсчитать вероятности этих сложных событий: $P(A \vee B) = 0.75$; $P(A \wedge B) = 0.25$. Подстановка этих значений в приведенные выше формулы для $P(A)$ и $P(B)$ дает правильный ответ. Аналогичную проверку можно произвести и для «неправильных» монет, когда вероятности выпадения герба отличаются от 0.5.

При решении обратной задачи для систем со многими состояниями точное решение систем уравнений возможно не всегда, так как число переменных в уравнениях регрессии сопоставимо с числом всех квантов и может превышать число уравнений. Так, в примере 5.2 число квантов в атрибуте X равно 7, следовательно, число неизвестных параметров только для этого атрибута будет на единицу меньше, т.е. 6. Однако задача может быть решена приближенно, если ее представить как задачу аппроксимации. Допустим, атрибут X_i разделен на k_i квантов ($k_i > 2$). Тогда будем считать неизвестными не величины квантов, а типы и параметры непрерывных распределений вероятности для каждого атрибута. Число параметров распределений обычно не превышает двух – они и будут неизвестными величинами. Для их оценки можно использовать методы

оптимизации, где управляющими воздействиями будут типы и параметры маргинальных распределений, а целевой функцией – обобщенный параметр, например, среднее значение абсолютных отклонений между расчетными и фактическими значениями вероятностей исследуемых сложных событий.

Такой подход к решению обратной задачи оценки вероятностей для случая непрерывных распределений во многом аналогичен подходу, который в настоящее время интенсивно используется при моделировании и анализе систем методами нечеткой логики. В подобных методах аппроксимация осуществляется с помощью целенаправленного подбора функций принадлежности и их коэффициентов для нечетких переменных. Эта аналогия позволяет при компьютерном моделировании систем для оценки вероятностей событий использовать результаты программирования систем, основанных на нечеткой логике. В то же время системы, основанные на АК, имеют более ясную интерпретацию, так как в них используются не методы максимина, а логические соотношения. Законы распределения вероятностей, в отличие от функций принадлежности, нередко имеют физический смысл.

В случаях, когда некоторое состояние системы описывается сложной логической функцией F и необходимо часто оценивать условные вероятности разных потенциально возможных событий, целесообразно предварительно выполнить элементаризацию состояния F с помощью МКИ и преобразовать F в ортогональную C-систему F_o. Тогда все расчеты вероятностей F при изменении маргинальных распределений вероятностей или расчеты условных вероятностей $P(f_k \mid F)$, где f_k – сравнительно простое легко структурируемое событие, не потребуют больших затрат машинного времени, так как вычисление вероятности $P(F)$ осуществляется методом простой подстановки значений вероятностей элементарных событий в F_o. Если f_k преобразовано в ортогональную C-систему, то при расчете $P(f_k \mid F)$ в результате вычисления пересечения $f_k \cap F_o$ сразу получится ортогональная C-система (по теореме 3.2).

Каждая логическая формула – это совокупность всевозможных связей между значениями входящих в нее переменных. Поэтому представление формулы в вероятностном пространстве можно интерпретировать как точное решение уравнения регрессии. Это утверждение можно легко доказать, используя соотношения АК.

Теорема 5.3. Если для логической формулы F осуществима элементаризация, то ее представление $S(F)$ в виде структуры АК в вероятностном пространстве есть точное решение уравнения регрессии.

Доказательство. Математическое ожидание уравнения регрессии в многомерном пространстве определяется как уравнение

$$E(y \mid x_1, x_2, ..., x_n) = \eta(x_1, x_2, ..., x_n),$$

где $\eta(x_1, x_2, ..., x_n)$ – математическая модель уравнения регрессии в пространстве $X_1 \times X_2 \times \ldots \times X_n$ событий. Поскольку функция $y = S(F)$ – отображение, в котором множество y представлено одним значением ("истина" или заданное состояние системы), то справедливо

$$E(y \mid x_1, x_2, ..., x_n) = E(S(F) \mid x_1, x_2, ..., x_n).$$

Из определения условной вероятности следует

$$P(S(F) \mid x_1, x_2, ..., x_n) = \frac{P(S(F)) \cap (x_1, x_2, ..., x_n))}{P(x_1, x_2, ..., x_n)},$$

где $(x_1, x_2, ..., x_n)$ соответствует полной группе событий, равной универсуму в пространстве $X_1 \times X_2 \times \ldots \times X_n$ атрибутов. Поскольку $S(F) \cap (x_1, x_2, ..., x_n) = S(F)$ и $P(x_1, x_2, ..., x_n) = 1$, то $P(y \mid x_1, x_2, ..., x_n) = P(S(F))$ при любых значениях x_i и, следовательно, $E(y \mid x_1, x_2, ..., x_n) = S(F)$. *Конец доказательства.*

Рассмотрение вероятностных задач с помощью АК позволяет решать прямую и обратную задачу в многомерном пространстве, не ограничиваясь каким-либо одним классом распределений. В качестве маргинальных распределений при решении прямой и обратной задачи можно использовать не только нормальное распределение, но и любое другое.

Приведенные в этом и предыдущем разделах соотношения, алгоритмы и примеры показывают, что применение алгебры кортежей в логико-вероятностном анализе обеспечивает решение следующих задач:

1) унификация алгоритмов расчета вероятностей сложных событий не только для моделей, заданных формулами исчисления высказываний, но и исчисления предикатов, что позволяет разработать методы расчета вероятности для систем с многими состояниями;

2) логический анализ систем с учетом условий и любых ограничений, а также с возможностью вероятностного расчета получающихся вариантов;

3) обратная задача вероятностного анализа: по заданным оценкам вероятностей сложных событий восстанавливать функции распределения маргинальных вероятностей и вероятностей событий с неизвестными оценками.

5.2. Работа с данными в структурах АК

5.2.1. Реляционные СУБД

Основные объекты управления в реляционных БД – файлы, организованные в виде таблиц (отношений), которые состоят из множества элементарных кортежей. В терминах АК таблица представима как *C*-система, где все компоненты есть одноэлементные множества. Такое представление позволяет применять все средства управления БД, включая методы поиска по ключу на основе теории нормальных форм, но не учитывает специфические свойства структур АК.

Если реляционные СУБД входят в состав систем искусственного интеллекта, то возникает необходимость в ассоциативных методах поиска, а также в средствах логического анализа данных, не предусмотренных в "классических" реляционных СУБД. В интеллектуальных системах преобладают структуры (сети, правила логического вывода, предикаты и т.д.), моделируемые АК-объектами с многоэлементными компонентами. Посредством таких же АК-объектов удобно выражать некоторые проекции обычных таблиц БД. За счет представления разнородных структур данных и знаний в виде АК-объектов достигается унификация их обработки и ускорение связанных с ними вычислительных процедур.

Опишем, как в АК выражаются запросы к СУБД. Если запрос сформулирован в виде АК-объекта, то атрибуты в нем делятся на два типа. К первому типу относятся атрибуты с заданными значениями, а ко второму – проблемные атрибуты, значения которых надо установить или уточнить в процессе поиска. При построении запроса проблемные атрибуты вводятся как фиктивные (в этом случае область поиска значений проблемных атрибутов охватывает всю область определения атрибута) или же в виде нефиктивной компоненты (тогда область поиска сужается).

Для реализации запросов над файлами (базовыми таблицами) или представлениями (производными таблицами) БД используется ***реляционная***

алгебра (*RA*) [*Дейт*, 1999] со своим набором операций. Пять из них – основные: проекция, объединение, прямое произведение, разность и селекция. Остальные операции *RA* реализуются как комбинации основных операций. Покажем, что означают в АК основные операции *RA*.

Операция ***проекция*** в *RA* создает из отношения $Q[\boldsymbol{X}]$ отношение $Q_p[\boldsymbol{X}\backslash\boldsymbol{Y}]$, где для множеств атрибутов $\boldsymbol{X}$ и $\boldsymbol{Y}$ справедливо $\boldsymbol{Y} \subset \boldsymbol{X}$. Ясно, что в АК этой операции соответствует аналогичная операция.

Операция ***объединение*** в *RA* эквивалентна объединению *C*-систем с одинаковыми схемами отношений.

Операция ***прямое произведение*** (иногда эту операцию называют ***декартово произведение***) в *RA* реализуется в АК пересечением *C*-систем, у которых схемы отношений не содержат одинаковых атрибутов. Например, в *RA* прямым произведением двух отношений $P[XY] = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ и $Q[ZV] = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$ является отношение $R[XYZV] = \begin{bmatrix} A & B & E & F \\ A & B & E & H \\ C & D & E & F \\ C & D & G & H \end{bmatrix}$.

В АК осуществление этой операции намного упрощается за счет использования фиктивных атрибутов. Для выполнения такой операции в АК достаточно найти пересечение двух *C*-систем, приведенных к единой схеме отношения (обобщенное пересечение):

$$P[XYZV] = \begin{bmatrix} A & B & * & * \\ C & D & * & * \end{bmatrix} \text{ и } Q[XYZV] = \begin{bmatrix} * & * & E & F \\ * & * & G & H \end{bmatrix}$$

и получить тот же результат.

Операция ***разность*** в *RA* эквивалентна в АК одноименной операции с однотипными отношениями, при этом можно использовать известное соотношение алгебры множеств:

$$P[\boldsymbol{X}] \setminus Q[\boldsymbol{X}] = P[\boldsymbol{X}] \cap \overline{Q[\boldsymbol{X}]}.$$

Операция ***селекция*** в *RA* позволяет выбрать из некоторого отношения набор кортежей, удовлетворяющих заданным ограничениям. Допустимо задавать ограничения на один или несколько произвольных атрибутов данной

схемы отношения в двух вариантах (X и Y – имена атрибутов): 1) $X\,\Delta\,c$ и 2) $X\,\Delta\,Y$, где c – элемент или множество элементов сорта, соответствующего данному атрибуту, Δ – одно из множества отношений сравнения $\{<, \le, =, \ne, \ge, >\}$. Для обеспечения полного соответствия с реляционной алгеброй АК следует дополнить элементарными операциями, предназначенными для селекции данных согласно отношениям сравнения. В ряде случаев эти операции можно выразить через уже известные нам операции АК. Рассмотрим примеры селекции в АК.

Для случая, когда селектор задан в виде равенств $X_i = c_i$, ограничение можно выразить как C-кортеж, у которого выбранные атрибуты X_i представлены элементами или подмножествами соответствующих сортов, а остальные атрибуты данной схемы отношения – фиктивными компонентами. Например, для отношения $P[XYZV]$ C-кортеж $[*\ \{a\}\ *\ \{d, f\}]$ является селектором с ограничениями $Y = a$; $V = d$ или f. Селекция по данным ограничениям формируется при пересечении этого C-кортежа с отношением P. Для ограничения типа $X_i \ne c_i$ в соответствующей позиции C-кортежа-селектора применяется компонента $\{\overline{c_i}\}$. Для отношений типа $<$, $\le$, $\ge$ или $>$, которые применимы только в частично упорядоченных множествах, селекторы также можно формировать в виде C-кортежей, у которых компоненты заданы как интервалы.

Теперь рассмотрим случай, когда селектор содержит ограничения типа 2. Допустим, в селекторе – условие равенства двух атрибутов. Например, имеется отношение $P[XYZ] = \begin{bmatrix} \{a\} & \{a,b\} & \{b,c\} \\ \{a,c\} & \{b,c,d\} & \{a\} \\ \{b,c\} & \{a,c\} & \{a,b,c\} \end{bmatrix}$ и задано ограничение $Y = Z$. Тогда сначала надо сформировать C-систему, у которой компоненты в атрибутах Y и Z равны пересечению компонент этих атрибутов в каждом C-кортеже, и удалить пустые C-кортежи. В результате получится промежуточная C-система $\begin{bmatrix} \{a\} & \{b\} & \{b\} \\ \{b,c\} & \{a,c\} & \{a,c\} \end{bmatrix}$. Далее из декартова произведения $\{a, c\}\times\{a, c\}$ 2-й и 3-й компонент второго C-кортежа выбирается только диагональ, в результате чего будет получена C-система

$$\begin{bmatrix} \{a\} & \{b\} & \{b\} \\ \{b,c\} & \{a\} & \{a\} \\ \{b,c\} & \{c\} & \{c\} \end{bmatrix}.$$

Опишем некоторые производные операции *RA*.

Операция ***пересечение*** двух однотипных отношений R и S в RA реализуется как комбинация операций $R \setminus (R \setminus S)$. Нетрудно убедиться:

$$R \setminus (R \setminus S) = R \cap \overline{\overline{R} \cap \overline{S}} = R \cap (\overline{\overline{R}} \cup S) = (R \cap \overline{\overline{R}}) \cup (R \cap S) = R \cap S,$$

что этой операции соответствует обычное пересечение АК-объектов, причем в АК она применима не только к однотипным отношениям.

Операция ***соединения*** отношений в *RA* определяется так. Пусть $P[\boldsymbol{X}A]$ и $Q[B\boldsymbol{Y}]$ – отношения, где $\boldsymbol{X}$, $\boldsymbol{Y}$ – некоторые непересекающиеся множества атрибутов, A, B – разные имена одного атрибута или разные атрибуты, относящиеся к одному и тому же сорту. Тогда результатом операции соединения является отношение $R[\boldsymbol{X}(A\Delta B)\boldsymbol{Y}]$, где Δ – одно из множества отношений сравнения $\{<, \leq, =, \neq, \geq, >\}$. Если Δ – покомпонентное равенство, то соединение отношений представляет в АК обычное пересечение АК-объектов при условии, что атрибутам A и B присваивается одно имя. При других условиях сравнения выполняется та же операция пересечения АК-объектов, но перед этим все компоненты одного из сравниваемых атрибутов преобразуются с учетом семантики отношения Δ.

Приведем примеры. Пусть в БД имеется отношение, которому сопоставлен АК-объект $P[XYZ]$, и требуется найти возможные значения атрибутов X и Y при заданном диапазоне значений D атрибута Z. На языке *SQL* этот запрос выражается так:

SELECT X, Y FROM P WHERE $Z \subseteq D$.

В АК представленный запрос записывается как C-кортеж $Q_1[Z] = [D]$. Ответ на запрос можно получить, вычислив

$P[XYZ] \cap_G Q_1[Z] = P[XYZ] \cap Q_1[XYZ]$, где $Q_1[XYZ] = [* * D]$.

Приведем пример, где требуется соединение отношений. Предположим, что в БД помимо отношения $P[XYZ]$ имеется отношение $R[YVW]$ и требуется найти значения атрибутов X и V, если $Z = a$. На языке SQL запрос

формулируется так:

SELECT X, V FROM P, R WHERE $Z = a$ AND $P.Y = R.Y$.

Ясно, что в АК схема отношения запроса соответствует схеме отношения АК-объекта, полученного в результате соединения P и R. Тогда запрос выражается как C-кортеж $Q_2[Z] = [\{a\}]$ или $Q_2[XYZVW] = [* * \{a\} * *]$, а ответ на запрос будет получен в атрибутах X и V после вычисления по формуле

$$(P[XYZ] \oplus R[YVW]) \cap_G Q_2[Z] = (P[XYZ] \cap_G R[YVW]) \cap [* * \{a\} * *].$$

Если исходные отношения характеризуются большими объемами, то операции селекции и соединения могут оказаться трудоемкими. В таком случае целесообразно вначале выполнить селекцию по одному или нескольким исходным отношениям, за счет чего их объемы существенно сократятся, а затем вычислить соединение полученных отношений. Для данного примера соответствующий алгоритм будет таким:

$$(P[XYZ] \cap [* * \{a\}]) \oplus R[YVW] = (P[XYZ] \cap [* * \{a\}]) \cap_G R[YVW].$$

Здесь вначале элиминировались из Q_2 фиктивные атрибуты V и W.

Ту же схему вычислений можно использовать в запросе типа

SELECT X, V FROM P, R WHERE $Z = a$ AND $W = b$ AND $P.Y = R.Y$,

в котором дополнительно задано фиксированное значение для атрибута W.

Тогда запросом будет C-кортеж $Q_2[ZW] = [\{a\}\ \{b\}]$, а ответ на запрос получим в результате следующих вычислений

$$(P[XYZ] \cap_G Q_2[Z]) \oplus (R[YVW] \cap_G Q_2[W])$$

или $(P[XYZ] \cap_G Q_2[Z]) \cap_G (R[YVW] \cap_G Q_2[W])$.

Таким образом, АК поддерживает выполнение всех базовых и производных операций, определенных в реляционной алгебре. К тому же, в АК можно реализовать запросы, которые невыразимы в СУБД, например, запросы, где используются дополнения реляционных отношений.

Теперь опишем обработку запросов в структурах АК.

5.2.2. Анализ незапланированных запросов (реляционные СУБД)

К главным достоинствам реляционных СУБД, помимо поддержки ссылочной целостности данных и многопользовательского режима доступа на основе клиент-серверной технологии, относится также возможность реализовывать незапланированные запросы (ad hoc запросы [*Дейт*, 1999]), т.е. запросы, формируемые во время исполнения клиентского приложения с учетом оперативных изменений схемы и содержимого базы данных. В реальных задачах часто возникает необходимость априорно (без привлечения данных из таблиц, кроме сведений о доменах атрибутов) проверять согласованность частей запроса или корректность незапланированных запросов с точки зрения ограничений моделируемой предметной области [*Зуенко*, 2008 с]. Такой анализ позволяет существенно снизить нагрузку на сервер БД, однако реляционные СУБД не обладают адекватными средствами для выполнения подобного рода проверок. Сами SQL-запросы (их селекторы) и большинство упомянутых ограничений представимы в виде логических формул над элементарными одно- и двуместными предикатами [*Зуенко*, 2008 а; 2010 а]. Такие формулы моделируются с помощью многоместных отношений, а значит, для их анализа могут быть применены методы АК.

До настоящего момента мы рассматривали самый простой случай, когда в качестве компонент АК-объектов выступают обычные множества (или интервалы), соответствующие одноместным предикатам. Чтобы значениями атрибутов АК-объектов могли быть многоместные отношения, моделирующие *n*-арные предикаты, требуется дополнить базовые определения и операции АК.

Далее описывается одна из возможных модификаций АК – алгебра условных кортежей (АУК), позволяющая использовать бинарные отношения при задании компонент и служащая для моделирования логических формул с элементарными двуместными предикатами [*Зуенко,* 2009; 2010 с].

Пусть $\boldsymbol{A} = \{A_1, A_2, \ldots, A_n\}$ – простые атрибуты, чьими доменами являются обычные множества, а $\boldsymbol{C} = \{A_1, A_2, \ldots, A_n, A_1 \times A_2, A_1 \times A_3, \ldots\}$ – множество, состоящее из ***простых атрибутов*** и их всевозможных попарных декартовых произведений (***сложных атрибутов***).

Алгебра условных кортежей задается следующим образом:

$\Lambda = < \{R_{Ki}[\boldsymbol{Q}_i]\}, \cup, \cap, +Atr, -Atr, Pr, \leftrightarrow Atr, \$ >$, где: $R_{Ki}[\boldsymbol{Q}_i]$ – некоторый ***АУК-объект*** (условный *C*-кортеж или условная *C*-система); $\boldsymbol{Q}_i$ – схема

отношения, представляющая собой некоторое множество простых и сложных атрибутов; *$* – операция наполнения АУК-объектов, остальные операции (+*Atr* – добавление атрибутов, -*Atr* – элиминация атрибутов, *Pr* – взятие проекции, ↔*Atr* – перестановка атрибутов) аналогичны операциям АК.

Условным С-кортежем называется заданный в определенной схеме отношения кортеж множеств (***компонент***), каждое из них представляет собой либо подмножество одного из доменов простых атрибутов, либо подмножество бинарного отношения, заданного на соответствующем сложном атрибуте. В качестве основных бинарных отношений (двуместных предикатов) выступают: «<» (меньше), «>» (больше), «=» (равно), «≠» (не равно). Компоненты простых атрибутов формируются из открытых интервалов вида (a, b) и дискретных значений. В описании компонент сложных атрибутов используются лишь имена бинарных отношений, а конкретные множества значений (элементарных кортежей) формируются в результате операции наполнения.

Условный *С*-кортеж будем именовать ***элементарным***, если каждая его компонента либо является полной (равна некоторому домену), либо содержит только одно из возможных значений. Элементарные условные *С*-кортежи соответствуют логическим формулам, содержащим только связки «и» между элементарными предикатами.

Условной С-системой называется множество однотипных условных *С*-кортежей, которые записываются в виде матрицы, ограниченной прямыми скобками.

Чтобы отличать АУК-объекты от структур АК, в их именах явно указывается множество сложных атрибутов. Например, $R_{\boldsymbol{K}}[\boldsymbol{M}]$ означает, что $\boldsymbol{K}$ – множество сложных атрибутов. АУК-объекты – это расширение *С*-систем и *С*-кортежей, в частности, *С*-система $R[\boldsymbol{M}]$ есть условная *С*-система $R_{\boldsymbol{K}}[\boldsymbol{M}]$, где $\boldsymbol{K} = \varnothing$.

Остановимся подробнее на том, как осуществляется пересечение двух условных *С*-кортежей. Пересечение компонент простых атрибутов сводится к пересечению между собой интервалов и дискретных значений. Пересечение компонент сложных атрибутов регламентируется, исходя из свойств отношений «<», «>», «=», «≠», представленных в табл. 5.5.

Проверки, производимые при пересечении, касаются в отдельности простых и в отдельности сложных атрибутов, они не учитывают того, как

соотносятся между собой разные виды атрибутов. Кроме того, не анализируется сама возможность одновременно сформировать все бинарные отношения, описанные в элементарном кортеже, на основе значений компонент простых атрибутов. Для анализа таких соотношений предназначена операция ***наполнения*** АУК-объектов значениями.

Таблица 5.5.

Предикаты		Наличие совместимости	Результат
Первый	Второй		
$x = y$	$x = y$	Да	$x = y$
	$x \neq y$	Нет	-
	$x < y$	Нет	-
	$x > y$	Нет	-
$x < y$	$x < y$	Да	$x < y$
	$x \neq y$	Да	$x < y$
	$x > y$	Нет	-
$x > y$	$x > y$	Да	$x > y$
	$x \neq y$	Да	$x > y$

Введем на *C*-кортежах вида $S[X,Y]=[A, B]$ следующие операторы: операторы взятия диагонали (Δ), взятия элементов, лежащих над ($\overline{\Delta}$) и под диагональю ($\underline{\Delta}$). Оператор $\Delta([A, B])$ возвращает $\Delta([A \cap B, A \cap B])$ и соответствует отношению «равно», т.е. Δ находит диагональ множества $A \cap B$ (подмножество диагонали универсума $X \times Y$). Операторы $\overline{\Delta}$ и $\underline{\Delta}$ (сопоставляются отношениям «меньше» и «больше») определяют кортежи, которые принадлежат прямоугольнику $A \times B$ и лежат соответственно над и под диагональю универсума $X \times Y$. Отношение «не равно» может быть выражено либо как дополнение отношения «равно», либо как объединение отношений «меньше» и «больше». Операторы возвращают пустое множество, если их область определения не содержит необходимых элементов.

Операция ***наполнения значениями*** произвольного АУК-объекта (конкретизации всех бинарных отношений внутри АУК-объекта) сводится к последовательности аналогичных операций для условных *C*-кортежей.

Результат операции наполнения АУК-объекта $R_K[M]$ обозначается как $\$R_K[M]$.

***Алгоритм 5.1. (наполнение условного С-кортежа значениями)*:**

Не снижая общности, будем полагать, что задан условный *С*-кортеж $R_K[F]$, где компоненты сложных атрибутов принимают только одно из возможных значений: «=», «<», «>», «≠». Если в компонентах встречаются сразу несколько значений, такой *С*-кортеж всегда можно разложить в набор кортежей указанной структуры.

1. Разбиваем множество простых атрибутов, формирующих сложные атрибуты из K, на непересекающиеся подмножества L_i следующим образом: простые атрибуты X и Y включаются в одно множество L_i, если в моделируемой логической формуле соответствующие переменные x и y связаны предикатами «=», «<», «>», «≠» либо непосредственно, либо через другие переменные.

2. Для каждого множества L_i из компонент атрибутов L_i формируем систему интервалов, после чего производим элементаризацию этой системы (см. п. 5.1).

3. Полученный после шагов 1 и 2 условный *С*-кортеж раскладываем в элементарные. Результат шага – некоторая условная *С*-система.

4. Элементы «=», «<», «>», «≠» в описании сложных компонент XY выражаем через операторы $\Delta([A, B])$, $\overline{\Delta}([A, B])$, и $\underline{\Delta}([A, B])$, где $A \subseteq X$, $B \subseteq Y$, а затем вычисляем эти операторы.

5. Конец алгоритма.

Поясним алгоритм на примере. Пусть имеется логическая формула:

$(x > 3) \wedge (x < 4) \wedge (y > 2) \wedge (y < 7) \wedge (z > 5) \wedge (z < 6) \wedge (y < x) \wedge (y > z)$,

где областью определения всех переменных служит множество вещественных чисел. Требуется установить ее выполнимость.

В АУК этой формуле соответствует условный *С*-кортеж

$P_{XY,YZ}[X,Y,Z,YX,YZ] = \left[\{(3,4)\},\{(2,7)\},\{(5,6)\},\{<\},\{>\}\right]$,

а решение задачи выполнимости заключается в наполнении заданного кортежа значениями.

На первом шаге алгоритма 5.1 можно сформировать только одно множество простых атрибутов – $\{X, Y, Z\}$, из которых образуются сложные атрибуты YX, YZ. Далее следуют шаги 2 и 3. Для кортежа $P_{XY,YZ}[X,Y,Z,YX,YZ]$ и множества атрибутов $\{X, Y, Z\}$ множество квантов будет выглядеть так:

(2, 3), (3, 4), (4, 5), (5, 6), (6, 7).

Тогда $P_{XY,YZ}[X,Y,Z,YX,YZ]$ запишется в виде:

$$\left[\{(3,4)\},\{(2,3),(3,4),(4,5),(5,6),(6,7)\},\{(5,6)\},\{<\},\{>\}\right].$$

Этот результат раскладываем в элементарные кортежи.

На шаге 4 выполняем конкретизацию бинарных отношений условной C-системы $P_{XY,YZ}[X,Y,Z,YX,YZ]$, т.е. вычисляем соответствующие операторы $\Delta([A, B])$, $\overline{\Delta}([A, B])$, и $\underline{\Delta}([A, B])$. В итоге имеем:

$$\$P_{XY,YZ} = \begin{bmatrix} \{(3,4)\},\{(2,3)\},\{(5,6)\},[\{(2,3)\},\{(3,4)\}],\varnothing \\ \{(3,4)\},\{(3,4)\},\{(5,6)\},\overline{\Delta}[\{(3,4)\},\{(3,4)\}],\varnothing \\ \{(3,4)\},\{(4,5)\},\{(5,6)\},\varnothing,\varnothing \\ \{(3,4)\},\{(5,6)\},\{(5,6)\},\varnothing,\underline{\Delta}[\{(5,6)\},\{(5,6)\}] \\ \{(3,4)\},\{(6,7)\},\{(5,6)\},\varnothing,\{[(6,7)\},\{(5,6)\}] \end{bmatrix} = \varnothing.$$

Значит, исходная формула невыполнима.

Поставленную задачу также можно решить графически (см. рис. 5.4). Из рисунка видно, что после подстановки значений логических переменных в предикаты ($x > y$) и ($y > z$) получим области (множества точек), выделенных темно-серым цветом, которые не имеют пересечения в проекции на ось Y.

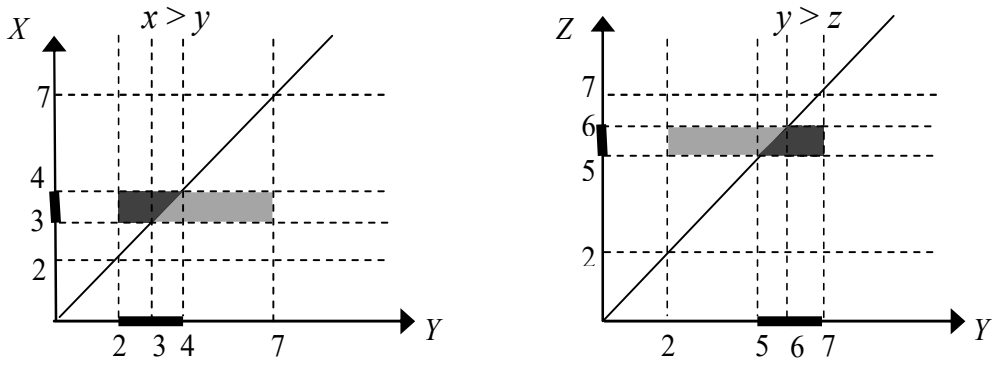

Рис. 5.4. Декартовы произведения интервалов.

Таким образом, с помощью АУК появляется возможность автоматизировать анализ составляющих логической формулы [*Зуенко*, 2008 b], содержащей элементарные двуместные предикаты, и делать выводы относительно их совместимости, а в случаях, когда заданы области определения переменных (как в приведенном примере) – о выполнимости или невыполнимости формулы.

Если логическая формула соответствует некоторому SQL-запросу, то отрицательный результат ее проверки на основе алгоритма 5.1 означает, что запрос всегда возвращает пустой набор данных, а положительный итог

свидетельствует о возможности получения релевантных данных, при условии их наличия в базе данных.

В следующем подразделе рассмотрим другой аспект применения АК при реализации запросов в СУБД, а именно рекурсивные запросы, которые в структурах АК осуществляются построением транзитивных замыканий отношений с последующей селекцией и элиминацией атрибутов.

5.2.3. Дедуктивные СУБД

В дедуктивных СУБД активно применяется ТФС и доказательно-теоретический подход. Другими словами, дедуктивная СУБД есть результат интеграции СУБД с системами логического вывода. Выполнение запроса здесь представляет собой доказательство некоторой теоремы с помощью специализированных дедуктивных аксиом и правил вывода. Основные аксиомы, соответствующие элементам домена и кортежам базовых отношений, называются экстенсиональной базой данных (extensional database – EDB). Дополнительные дедуктивные аксиомы вместе с ограничениями целостности образуют интенсиональную базу данных (intensional database – IDB).

Разработке единого языка для таких комбинированных систем посвящено большое число работ. Среди них можно выделить два направления [*Чери,* 1992]:

1) работы, в которых предусматривается интеграция систем управления реляционными БД с Пролог-системами, т.е. *CPR*-системы (аббревиатура от Coupling Prolog to Relational databases);

2) работы, в которых принят единый язык управления БД и системой логического вывода – *Дейталог*. Описание постановки задачи на этом языке называется *Дейталог-программой*, а часть Дейталог-программы, не содержащей факты, – *Дейталог-правилами.*

Особенность дедуктивных СУБД состоит в поддержке рекурсивных запросов. Далее показана реализация рекурсивных запросов не посредством логического вывода, а с помощью методов АК и теории графов, поэтому вначале приводятся некоторые сведения из теории графов.

Графы. Как правило, граф определяется в математике "геометрическим" способом: на множестве ***вершин*** *V* задается множество *E* их упорядоченных пар, которые называются ***дугами*** графа и изображаются в виде линий со

стрелками, направленными от одних вершин к другим. Если в графе какая-либо пара вершин соединена прямой и обратной дугой, то такое "двунаправленное" соединение между вершинами называется ***ребром*** графа. Ребра изображаются с помощью линий без стрелок. Граф, у которого все связи представлены только ребрами, называется ***неориентированным*** графом (или просто графом), граф, не содержащий ребер, – ***ориентированным*** графом, а граф, включающий и ребра, и дуги – ***смешанным***.

Путь в графе есть последовательность его вершин, в которой пары соседних вершин соединены дугой, для которой одна из вершин – начальная, а другая – конечная. Вершина y графа G называется ***достижимой*** из вершины x, если в G существует путь из вершины x в вершину y. Если в графе имеется путь с совпадающими начальной и конечной вершиной, такой путь именуется ***циклом***.

Транзитивное замыкание графа G, содержащего n вершин, – это граф G^+, где каждая вершина соединена дугой со всеми достижимыми вершинами.

Обычно графы представлены в компьютерах как списковые структуры. В системах искусственного интеллекта логический вывод на графах реализуется как алгоритм поиска достижимых вершин (построения транзитивного замыкания графа). Подобные алгоритмы недостаточно эффективны и плохо поддаются распараллеливанию. Опишем, как выражаются графы в структурах АК. В качестве примера используем граф, изображенный на рис. 5.5.

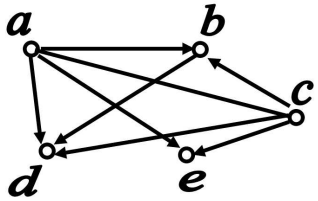

Рис. 5.5. Пример графа.

Его можно записать как C-систему $G[XY] = \begin{bmatrix} \{a\} & \{b,c,d,e\} \\ \{b\} & \{d\} \\ \{c\} & \{a,b,d,e\} \end{bmatrix}$, изоморфную матрице смежности графа.

На графах часто используется композиция $G \circ G$, т.е. композиция графа с самим собой, сокращенно обозначаемая как G^2. Применяется и более высокая "степень" композиции, например, $G^3 = G \circ G \circ G$ и т.д.

Чтобы определить для каждой вершины графа G, имеющего n вершин, множество всех достижимых из нее вершин, необходимо вычислить транзитивное замыкание графа. Это можно сделать с помощью следующей последовательности операций:

$G^+ = G \cup G^2 \cup G^3 \cup \ldots \cup G^k$, где $k \le n$.

Практически во всех случаях преобразование конечного графа G в граф G^+ заканчивается прежде, чем будет получена последняя "степень" G^k. Критерием более раннего завершения этой операции служит отсутствие в очередной "степени" графа не встречавшихся ранее дуг.

В качестве примера продемонстрируем, как в АК строится вторая "степень" графа, то есть его композиция с самим собой: $G^2[XY] = G[XY] \circ G[YZ] = -Y(G[XY] \cap_G G[YZ])$, где атрибуты X, Y первого операнда переименованы во втором операнде в атрибуты Y, Z. Сначала производим пересечение:

$$G[XY] \cap_G G[YZ] = \begin{bmatrix} \{a\} & \{b,c,d,e\} & * \\ \{b\} & \{d\} & * \\ \{c\} & \{a,b,d,e\} & * \end{bmatrix} \cap \begin{bmatrix} * & \{a\} & \{b,c,d,e\} \\ * & \{b\} & \{d\} \\ * & \{c\} & \{a,b,d,e\} \end{bmatrix} =$$

$$= \begin{bmatrix} \{a\} & \{b\} & \{d\} \\ \{a\} & \{c\} & \{a,b,d,e\} \\ \{c\} & \{a\} & \{b,c,d,e\} \\ \{c\} & \{b\} & \{d\} \end{bmatrix}.$$

Затем элиминируем атрибут Y:

$$G^2 = -Y(G[XY] \cap_G G[YZ]) = \begin{bmatrix} \{a\} & \{a,b,d,e\} \\ \{c\} & \{b,c,d,e\} \end{bmatrix}.$$

Рекурсивные запросы. Для иллюстрации возможностей применения АК при решении задач, типичных для дедуктивных СУБД, рассмотрим пример из [*Чери*, 1992]. В базе данных содержится упрощенная коммерческая система, состоящая из 11 фирм. Каждая фирма занимает определенную долю рынка в процентах для каждого вида товаров. Эти сведения содержатся в таблице "Фирмы". В таблице "Отношения собственности" показаны пары фирм и проценты всех акций Фирмы2, являющиеся собственностью Фирмы1. В таблице "Трестовский лимит" находятся наименования товаров и квота рынка в

процентах для каждого товара (см. таблицы 5.5-5.7).

В [*Чери*, 1992] решается "Антитрестовская задача", состоящая в следующем. В законодательстве некоторых стран существуют законы, не позволяющие фирмам непосредственно или косвенно (через другие фирмы) контролировать долю рынка более заданной. Известно также, что если фирма владеет контрольным пакетом акций другой фирмы (51% и выше), то она имеет возможность контролировать деятельность дочерней фирмы. Таким образом, возможна ситуация, когда доля рынка товаров самой фирмы не выходит за рамки квоты, но при учете других участников рынка, контролируемых этой фирмой, суммарная доля может превысить квоту для определенного товара. В данном случае необходимо выявить фирмы, торгующие товаром T_1 и нарушающие антитрестовский закон.

В [*Чери*, 1992] для постановки задачи используется язык логического программирования Пролог. Решим задачу в терминах АК и теории графов. Селектором для данной задачи являются фирмы, торгующие товаром T_1 и имеющие контрольный пакет акций каких-либо других фирм.

Таблица 5.5. **Фирмы**

Фирма	Товар	Доля Рынка
A	T_1	8
B	T_1	7
C	T_2	33
D	T_1	13
E	T_1	0
F	T_1	12
G	T_2	30
H	T_1	8
I	T_1	12
J	T_1	2
K	T_1	4
L	T_1	19
M	T_1	15
N	T_2	35

Таблица 5.6. **Отношения собственности**

Фирма1	Фирма2	Процент акций
A	*B*	51
A	*D*	20
A	*E*	30
A	*J*	51
B	*H*	30
B	*I*	70
E	*D*	51
N	*C*	2
E	*F*	60
J	*K*	59
F	*L*	20
F	*M*	25
M	*L*	60

Таблица 5.7. **Трестовский лимит**

Товары	Предельная квота
T_1	20
T_2	38
T_3	40

Обозначим таблицу "Фирмы" как $R_1[X_1X_2X_3]$, таблицу "Отношения

собственности" – как $R_2[Y_1Y_2Y_3]$, а таблицу "Трестовский лимит" – как $R_3[Z_1Z_2]$. Тогда вычисления можно разбить на следующие шаги:

Шаг 1. Найти фирмы, имеющие контрольный пакет акций в других фирмах. Для этого применим селектор [* * {>50}] к таблице "Отношение собственности":

$P[Y_1Y_2Y_3] = R_2[Y_1Y_2Y_3] \cap_G [* * \{>50\}]$.

В проекции $P[Y_1]$ содержатся искомые на данном шаге фирмы, в проекции $P[Y_2]$ – дочерние фирмы, а проекция $P[Y_1Y_2]$ выражается как *C*-система:

$$P[Y_1Y_2] = P[\textbf{Фирма1 Фирма2}] = \begin{bmatrix} \{A\} & \{B,J\} \\ \{B\} & \{I\} \\ \{E\} & \{D,F\} \\ \{J\} & \{K\} \\ \{M\} & \{L\} \end{bmatrix}.$$

Отношение $P[Y_1Y_2]$ можно изобразить в виде графа, в котором дуги – направленные связи между фирмами и контролируемыми ими другими участниками рынка.

Теперь необходимо откорректировать отношение $P[Y_1Y_2Y_3]$, оставив в нем только головные фирмы, работающие с товаром T_1. На шагах 2, 3 выполняется эта корректировка.

Шаг 2. Выделить участников рынка, производящих товар T_1, для чего применим селектор [* T_1 *] к отношению "Фирмы":

$E[X_1X_2X_3] = R_1[X_1X_2X_3] \cap_G [* T_1 *]$.

Шаг 3. Определить головные фирмы из отношения $P[Y_1Y_2Y_3]$, которые производят товар T_1. Это можно осуществить разными способами, например, если вычислить соединение $P[Y_1Y_2Y_3] \oplus E[X_1X_2X_3] = P[Y_1Y_2Y_3] \cap_G E[X_1X_2X_3]$, полагая эквивалентными первые атрибуты этих отношений. Здесь же предлагается другой способ – сравнивать проекции $P[Y_1]$ и $E[X_1]$. Можно убедиться в том, что они в примере идентичны, поэтому для *C*-системы $P[Y_1Y_2Y_3]$ корректировка не нужна.

Шаг 4. Выявить множества фирм на рынке товара T_1, которые входят в тресты. На данном этапе вычислений необходимо найти транзитивное замыкание графа, соответствующего *C*-системе $P[Y_1Y_2]$. В результате получим такой АК-объект:

$$P^{+}[Y_1Y_2] = \begin{bmatrix} \{A\} & \{B,I,J,K\} \\ \{B\} & \{I\} \\ \{E\} & \{D,F\} \\ \{J\} & \{K\} \\ \{M\} & \{L\} \end{bmatrix},$$

где каждая строка соответствует определенному тресту. В левом столбце содержатся фирмы, контролирующие тресты, а в правом – остальные фирмы, которые входят в эти тресты.

Шаг 5. Вычислить доли рынка товара T_1, контролируемого соответствующей фирмой (включая долю самой фирмы). В итоге:

для фирмы A – {8, 7, 12, 2, 4}; Σ = 33;

для фирмы B – {7, 12}; Σ = 19;

для фирмы E – {0, 13, 12}; Σ = 25;

для фирмы J – {2, 4}; Σ = 6;

для фирмы M – {19, 15}; Σ = 34.

Шаг 6. Теперь, используя данные отношения "Трестовский лимит", можно определить, что антитрестовский закон нарушают фирмы A, E и M.

Таким образом, задачу, для решения которой традиционно использовались методы и средства логического программирования, можно решать с применением методов АК.

5.3. Системы искусственного интеллекта

5.3.1. Представление знаний в АК

В интеллектуальных системах наиболее распространены три типа моделей представления знаний: продукции (правила), семантические сети и фреймы.

Правила. Чаще всего используются *продукционные правила*, то есть импликации типа:

Если A_1 и A_2 и ... и A_k, *то* B_1 или B_2 или ... или B_n,

где A_i и B_j – некоторые предикаты.

Предикаты, по сути, являются отношениями. Например, набор предикатов экспертной системы (ЭС)

isa(a_1, S_1);

$\text{isa}(a_2, S_1)$;
$\text{isa}(a_3, S_1)$;
$\text{isa}(a_4, S_2)$; (5.4)
$\text{prop}(S_1, c_1, d_1)$;
$\text{prop}(S_1, c_2, d_2)$;
$\text{prop}(S_2, c_1, d_3)$.

отображает два отношения: бинарное *isa* и трехместное *prop*. В структурах АК предикаты можно выразить как *C*-системы. Тогда модель (5.4) примет вид:

$$isa[XY] = \begin{bmatrix} \{a_1, a_2, a_3\} & \{S_1\} \\ \{a_4\} & \{S_2\} \end{bmatrix}; \; prop[YZV] = \begin{bmatrix} \{S_1\} & \{c_1\} & \{d_1\} \\ \{S_1\} & \{c_2\} & \{d_2\} \\ \{S_2\} & \{c_1\} & \{d_3\} \end{bmatrix}. \quad (5.5)$$

Если правило выражено в виде импликации $X \supset Y$, то логическое выражение X называется *телом правила*, а логическое выражение Y – *головой* (или *целью*) *правила*. База данных ЭС хранит факты, содержащие значения некоторых предикатов. Ввод фактов в базу данных инициирует процедуру поиска новых фактов с учетом правил, содержащихся в базе знаний. Она продолжается до тех пор, пока не будет установлено, что больше никаких новых фактов из этой базы знаний извлечь невозможно.

Экспертные системы во многом сходны с СУБД, а процедуры вывода на основе поступивших фактов аналогичны реализациям запросов в БД. Основное отличие состоит в том, что в процедурах вывода ЭС используется больше аналитических средств, чем при реализации запросов СУБД. Это как раз те средства логического анализа, которые отсутствуют в современных СУБД, но разработаны в АК.

Рассмотрим пример, используя совокупность предикатов (5.4) и их представление в АК (5.5). Пусть задано следующее правило вывода:

ЕСЛИ *isa*[*XY*] И *prop*[*YZV*] ТО *property*[*XZV*].

В терминологии АК это типичное для БЗ правило предусматривает формирование нового отношения *property* с помощью композиции отношений *isa* и *prop*. Тогда ввод в систему новых фактов, например, $X = a_1$ и $V = d_2$, эквивалентен запросу $Q[XV] = [\{a_1\} \; \{d_2\}]$. Его можно реализовать (в терминологии ЭС – для поиска новых следствий из этих фактов в базе знаний) с помощью нижеприведенного алгоритма, если предварительно выполнить

операции с отношениями-предикатами, предусмотренные в теле правила:

$$(isa[XY] \circ prop[YZV]) \cap [\{a_1\} * \{d_2\}],$$

где в запрос $Q[XV]$ добавлен фиктивный атрибут Z. Популярный пример правила, реализация которого требует композиции отношений, – предложение: «Если X – отец Y и Y – родитель Z, то X – дед Z».

Один из критических узлов любой экспертной системы – это *интерпретатор* – программно-математическая система преобразования данных в соответствии с правилами. Принципы работы интерпретаторов основаны на символьных преобразованиях математической логики, их программная реализация весьма сложна и, как правило, не показывается даже в самой подробной технической документации. В них обычно используются списковые структуры данных, которые при машинной реализации больших систем характеризуются невысоким быстродействием и поэтому часто не могут использоваться в системах реального времени.

Рассмотрим пример, иллюстрирующий работу интерпретатора в структурах АК. В статье [*Индейцев,* 1995] описана экспертная система, предназначенная для принятия решений на уровне командира корабля в случае возникновения нештатных ситуаций на корабле или в окружающей обстановке, в частности, в боевых условиях. Исходная информация для принятия решений – обобщенные данные по подсистемам корабля и окружающей среды, т.е. данные типа "утрата непотопляемости возрастает медленно (или быстро)", "возник пожар в энергоотсеках в одном эшелоне", "состояние моря 6 баллов" и т.д.

Опишем пример перевода правил ЭС на язык АК для комплекта правил по анализу электрообеспечения корабля, где используются 4 фактора. Данные по ним и соответствующие обозначения приведены в табл. 5.8.

На предварительном этапе реализации системы был составлен следующий комплект правил для системы электрообеспечения.

***Инициализация*: W** = неизвестно.

***Цель*: W**.

***Правило 1*; приоритет 30:**

IF ($X = c$ AND $y = b$) OR $Z = b$

THEN $W = c$

***Правило 2*; приоритет 20:**

IF $X = b$ OR $Z = b$

THEN $W = b$

Таблица 5.8.

Наименования факторов и их значения	**Обозначения факторов**	**Обозначения значений факторов**
СНАБЖЕНИЕ ЭЛЕКТРОЭНЕРГИЕЙ	*W*	
обеспечено полностью		*a*
утрачено частично		*b*
утрачено полностью		*c*
ОСНОВНЫЕ ИСТОЧНИКИ	*X*	
в строю		*a*
вышли из строя в одном эшелоне		*b*
вышли из строя в обоих эшелонах		*c*
АВАРИЙНЫЕ ИСТОЧНИКИ	*Y*	
в строю		*a*
вышли из строя		*b*
КАНАЛИЗАЦИЯ ЭЛЕКТРОЭНЕРГИИ	*Z*	
не нарушена		*a*
нарушена частично		*b*
нарушена полностью		*c*

Приоритет в данном случае устанавливает порядок применения правил: чем больше приоритет, тем раньше должно быть применено данное правило; переход к следующему правилу происходит в случае безуспешной проверки всех правил с более высокими приоритетами (это идеология нормальных алгоритмов Маркова). При успешной проверке происходит выход из комплекта правил с присвоением значения целевому фактору (в данном случае фактору *W*). Если в обоих правилах проверка безуспешна, то фактору *W* присваивается значение *a*.

Для решения задачи средствами АК введем четыре атрибута:

$W = \{a, b, c\}$; $X = \{a, b, c\}$; $Y = \{a, b\}$; $Z = \{a, b, c\}$.

Преобразуем условия правил 1 и 2 в АК-объекты, приведя их к единой схеме отношения [*XYZ*]. Тогда получим:

Условие правила 1: $R_1 = \begin{bmatrix} \{c\} & \{b\} & * \\ * & * & \{c\} \end{bmatrix}$,

Условие правила 2: $R_2 = \begin{bmatrix} \{b\} & * & * \\ * & * & \{b\} \end{bmatrix}$.

Покажем, как используются эти правила в АК. Пусть поступила информация: "Основные источники вышли из строя в одном эшелоне, аварийные источники в строю и канализация энергии нарушена частично". Эта информация преобразуется в *C*-кортеж $T = [\{b\}\ \{a\}\ \{b\}]$. Применим правило 1, для чего найдем пересечение $T \cap R_1 = \varnothing$. Следовательно, ситуация не соответствует этому правилу. Применим правило 2:

$T \cap R_2 = [\{b\}\ \{a\}\ \{b\}]$.

Поскольку пересечение не пусто, то $W = b$.

Проанализируем корректность указанных правил средствами АК. Во-первых, проверим неоднозначности, то есть определим, содержат ли разные условия какие-либо общие ситуации. Для этого вычислим пересечение $R_1 \cap R_2 = \begin{bmatrix} \{c\} & \{b\} & \{b\} \\ \{b\} & * & \{c\} \end{bmatrix}$. Анализ показывает, что все общие для данных условий ситуации соответствуют состоянию $W = c$. Но в самой ЭС неоднозначности не возникают, так как приоритет правила 1, в котором предусмотрено это состояние, более высок, и эта ситуация распознается правильно. Возможен и другой вид некорректности, когда некоторые критические состояния не включены ни в одно из условий для аварийных ситуаций. Для анализа вычислим множество таких состояний с помощью формулы $\overline{R_1 \cup R_2}$.

$$R_1 \cup R_2 = \begin{bmatrix} \{c\} & \{b\} & * \\ * & * & \{c\} \\ \{b\} & * & * \\ * & * & \{b\} \end{bmatrix} = \begin{bmatrix} \{c\} & \{b\} & * \\ * & * & \{b,c\} \\ \{b\} & * & * \end{bmatrix},$$

$$\overline{R_1 \cup R_2} = \begin{bmatrix} \{a,b\} & \{a\} & \varnothing \\ \varnothing & \varnothing & \{a\} \\ \{a,c\} & \varnothing & \varnothing \end{bmatrix}.$$

Получилась *D*-система. Преобразуем ее в ортогональную *C*-систему:

$$\begin{bmatrix} \{a,b\} & \{a\} & \varnothing \\ \varnothing & \varnothing & \{a\} \\ \{a,c\} & \varnothing & \varnothing \end{bmatrix} = \begin{bmatrix} \{a\} & * & \{a\} \\ \{c\} & \{a\} & \{a\} \end{bmatrix}.$$

Для специалиста понятно, что из 3-х возможных ситуаций одна (а именно [{*c*} {*a*} {*a*}]) является аварийной и соответствует состоянию $W = c$. Но она не включена ни в одно из условий, поэтому правила нужно откорректировать. В данном случае корректировка заключается в добавлении *C*-кортежа [{*c*} {*a*} {*a*}] в условие R_1. В результате получим:

$$R_1 = \begin{bmatrix} \{c\} & \{b\} & * \\ * & * & \{c\} \\ \{c\} & \{a\} & \{a\} \end{bmatrix}.$$

Используя средства АК, можно также заменить комплект правил одним АК-объектом. Для этого объединим созданные в системе АК условия в один АК-объект и добавим атрибут, характеризующий состояние всей системы, в который запишем соответствующие состояния. Тогда получим АК-объект, позволяющий распознать все возможные ситуации. В частности, для рассмотренного примера таким АК-объектом будет *C*-система

$$R[XYZW] = \begin{bmatrix} \{c\} & \{b\} & * & \{c\} \\ * & * & \{c\} & \{c\} \\ \{c\} & \{a\} & \{a\} & \{c\} \\ \{b\} & * & * & \{b\} \\ * & * & \{b\} & \{b\} \\ \{a\} & * & \{a\} & \{a\} \end{bmatrix}$$

Допустим, на вход поступила информация $X = b$, $Z = c$. Она преобразуется в *C*-кортеж $T[XYZW] = [\{b\} * \{c\} *]$, и находится пересечение $T \cap R$. В данном случае после выполнения операции получим *C*-систему

$$\begin{bmatrix} \{b\} & * & \{c\} & \{c\} \\ \{b\} & * & \{c\} & \{b\} \end{bmatrix} = [\{b\} * \{c\} \ \{c, b\}]$$

с неоднозначным результатом в атрибуте *W*. *О*днако эту неоднозначность можно устранить, если использовать правило приоритета. Приоритет здесь определяется порядком расположения *C*-кортежей в *R*: первый полученный непустой результат пересечения является окончательным. С учетом этого для заданного входа получается результат $W = c$.

Из приведенного примера видно, что на основе АК можно создавать "прозрачные" и сравнительно простые в реализации интерпретаторы экспертных систем и к тому же проверять корректность правил на этапе проектирования системы. В последние годы по мере усложнения прикладных ЭС стало ясно, что для их разработки явно недостаточно средств, которые используются в "интеллектуальных" языках программирования (Пролог, Лисп и др.). Поэтому разработчики вынуждены переходить на программирование ЭС с помощью языков высокого уровня, таких как C^{++}, Object Pascal, Perl и т.д., где используются и средства управления базами данных, и методы объектно-ориентированного программирования. Вызвано это необходимостью совмещения структур данных и знаний. Использование АК позволяет более просто решить эту проблему, так как в ней данные и знания органично выражаются в одних и тех же структурах и обеспечены соответствующими алгоритмами.

Семантические сети. Покажем, как в АК осуществляется вывод на семантических сетях [*Kulik*, 2010]. Каждую семантическую сеть можно представить как совокупность бинарных отношений. Правила вывода в семантической сети записываются в виде продукций, в левой части которых содержится какой-либо фрагмент семантической сети (соединения или композиции некоторых из исходных отношений), а в правой части – фрагмент, заменяющий левую часть или добавляемый в семантическую сеть как новое отношение при срабатывании правила. Например, отношение "быть внуком" появляется в результате композиции отношения "быть сыном" с самим собой.

При реализации продукционного правила средствами АК в общем случае необходимо выполнить следующую последовательность шагов:

1) получить соединение или композицию отношений, стоящих в левой части правила;

2) исключить из получившегося отношения *P* все кортежи, не удовлетворяющие заданным ограничениям;

3) получить соединение (композицию) отношений *T*, стоящих в правой части продукции, при необходимости с учетом множества кортежей, получившихся при вычислении левой части правила;

4) если в правиле предусматривается замена левой части на правую, то из исходной базы знаний удалить все связи, содержащиеся в *P*;

5) добавить в базу знаний все связи из отношения *T*.

Рассмотрим пример из книги Д.А. Поспелова [*Поспелов*, 1989].

Исходная семантическая сеть изображена на рис. 5.6, правило вывода на этой сети на рис. 5.7, а результат применения правила – на рис. 5.8.

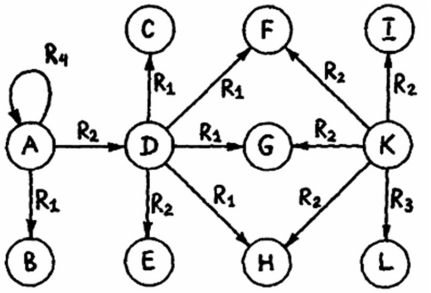

Рис. 5.6. Исходная семантическая сеть

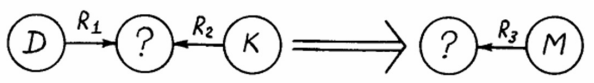

Рис. 5.7. Продукция

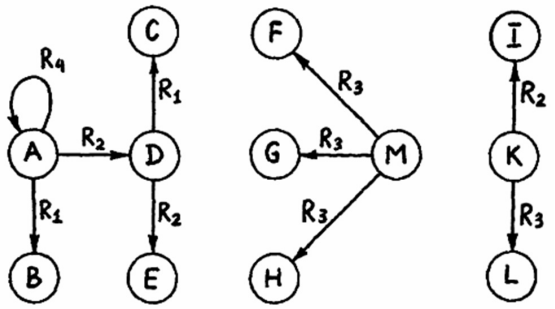

Рис. 5.8. Результат применения продукции

Сначала выразим в структурах АК исходную сеть. Ее на языке АК можно записать в виде совокупности *C*-систем и *C*-кортежей:

$$R_1[XY] = \begin{bmatrix} \{A\} & \{B\} \\ \{D\} & \{C,F,G,H\} \end{bmatrix}; \; R_2[ZY] = \begin{bmatrix} \{A\} & \{D\} \\ \{D\} & \{E\} \\ \{K\} & \{F,G,H,I\} \end{bmatrix};$$

$$R_3[VY] = [\{K\} \; \{L\}]; \; R_4[ZW] = [\{A\} \; \{A\}].$$

В левой части продукции предусматривается соединение отношений R_1 и

R_2 при условии, что значением атрибута X в R_1 является D, а в R_2 – K. Процедура выполнения продукции осуществляется в соответствии с уже описанным алгоритмом, за исключением того, что фильтрация отношений производится перед их соединением (иногда это более эффективно).

Шаг 1. Сначала отношения R_1 и R_2 приводятся в соответствие с ограничениями.

$P_1[XY] = R_1[XY] \cap [\{D\}\ *] = [\{D\}\ \{C, F, G, H\}]$;

$P_2[ZY] = R_2[ZY] \cap [\{K\}\ *] = [\{K\}\ \{F, G, H, I\}]$.

Шаг 2. Далее выполним перестановку атрибутов в отношении $P_2[ZY]$ и вычислим соединение полученных отношений:

$P[XYZ] = P_1[XY] \oplus P_2[YZ] = [\{D\}\ \{C, F, G, H\}\ *] \cap [*\ \{F, G, H, I\}\ \{K\}] =$

$= [\{D\}\ \{F, G, H\}\ \{K\}]$.

Шаг 3. Отсутствует.

Шаг 4. Теперь, когда определено значение атрибута Y, можно вычислить правую часть продукции: $P_3[VY] = [\{M\}\ \{F, G, H\}]$.

Шаги 5 и 6. Остается откорректировать базу знаний. Для этого отношение R_3 объединим с отношением P_3 и откорректируем отношения R_1 и R_2 с учетом проекций отношения P: $P_4[XY] = P[XY] = [\{D\}\ \{F, G, H\}]$ и $P_5[ZY] = P[ZY] = [\{K\}\ \{F, G, H\}]$, соответственно. Для этого с помощью алгоритмов АК вычислим разности $R_1[XY] \setminus P_4[XY] = R_1[XY] \cap \overline{P_4}[XY]$ и $R_2[ZY] \setminus P_5[ZY] = R_2[ZY] \cap \overline{P_5}[ZY]$. Теперь добавим к $R_3[VY]$ кортежи отношения $P_3[VY]$ и получим сеть, изображенную на рис. 5.8:

$$R_1[XY] = \begin{bmatrix} \{A\} & \{B\} \\ \{D\} & \{C\} \end{bmatrix};\ R_2[ZY] = \begin{bmatrix} \{A\} & \{D\} \\ \{D\} & \{E\} \\ \{K\} & \{I\} \end{bmatrix};$$

$$R_3[VY] = \begin{bmatrix} \{K\} & \{L\} \\ \{M\} & \{F, G, H\} \end{bmatrix};\ R_4[ZW] = [\{A\}\ \{A\}].$$

Фреймы. Каждый фрейм состоит из набора *слотов*. Имя фрейма соответствует имени некоторого отношения, а имя слота внутри фрейма – имени какого-либо атрибута этого отношения. Таким образом, в простейшем случае фрейм есть своеобразное представление обычных файлов базы данных (таблиц или АК-объектов). Фрейм может иметь более разветвленную структуру, когда значениями некоторых слотов являются не элементы

отношения, а другие слоты. С точки зрения АК атрибуты, представленные такими усложненными слотами, имеют в качестве доменов не множества с простыми элементами, а многоместные отношения, элементами которых являются кортежи. Такие разветвленные фреймы также представимы в структурах АК. Иногда в качестве слотов используются правила – ранее было показано, что их также можно выразить на языке АК.

Выше было показано, как с помощью АК-объектов могут быть отображены основные модели представления знаний, применяемые в системах искусственного интеллекта. Далее в главе демонстрируются возможные применения АК для задач искусственного интеллекта [*Зуенко*, 2010 b].

5.3.2. АК и неоднородные семантические сети

Понятие неоднородной семантической сети введено Г. С. Осиповым для описания плохо структурированных предметных областей, где знания об индивидах и их взаимосвязях не имеют завершенного вида [*Осипов*, 2009].

Неоднородной интенсиональной семантической сетью называют алгебраическую систему

$H = <\boldsymbol{D}, \boldsymbol{N}, \boldsymbol{R}, \boldsymbol{F}>$,

где $\boldsymbol{D} = \{D_1, D_2, \ldots, D_n\}$ – семейство непустых множеств; $\boldsymbol{N} \subseteq N$ – выделенное подмножество слов конечной длины над заданным алфавитом; $\boldsymbol{R} = \{R_1, R_2, \ldots, R_q\}$ – семейство бинарных отношений на N^2, $\boldsymbol{F} = \{f_1, f_2, \ldots, f_m\}$ – семейство типизированных функций. Функция f_i имеет тип $<\tau, \omega>$, где $\tau = <k1, k2, \ldots, km>$, если она определена на декартовом произведении $D_{k1}\times D_{k2}\times\ldots\times D_{km}$, а областью ее значений является множество D_ω, т. е. каждому кортежу $\delta \in D_{k1}\times D_{k2}\times\ldots\times D_{km}$ функция f_i ставит в соответствие некоторый элемент $f_i(\delta)$ из D_ω.

Если каждому $n \in \boldsymbol{N}$ приписать тип τ, а также $\boldsymbol{e}$ – некоторое подмножество декартова произведения $D_{k1}\times D_{k2}\times\ldots\times D_{km}$ (экстенсионал, множество примеров), а затем определить на множестве экстенсионалов $\boldsymbol{E} = \{e_i\}$ отношения R_i^* вместо отношений R_i на N^2, то полученная конструкция:

$\boldsymbol{H}^* = <\boldsymbol{D}, \boldsymbol{E}, \boldsymbol{R}^*, \boldsymbol{F}>$

называется ***экстенсиональной неоднородной семантической сетью*** (ЭНС).

ЭНС применяются в некоторых методах приобретения знаний, а также при организации правдоподобных рассуждений, в частности, в задачах аргументации [*Осипов*, 2009]. Процедуры вывода основаны на том, что элементы в парах из R_i^* (элементы множества $\boldsymbol{E}$) сами имеют некоторую внутреннюю структуру, то есть, по существу, тоже являются отношениями, определенными на множествах из $\boldsymbol{D}$. Это позволяет выразить отношения R_i^* через внутреннюю структуру элементов из $\boldsymbol{E}$, а также выделить следующие классы отношений: a) отношения структурного сходства, b) ассоциативные и каузальные отношения. Далее приведем определения отношений структурного сходства, а затем перепишем условия этих определений на языке АК.

Пусть e_1, e_2 – произвольная пара отношений из $\boldsymbol{E}$, δ, γ – примеры (в терминах АК – это элементарные кортежи) отношений e_1, e_2 соответственно. Результатом пересечения $\delta \cap \gamma$ будет общий фрагмент слов δ и γ, а под записью $\delta \subset \gamma$ понимается, что слово δ включено в слово γ (здесь кортежи рассматриваются как слова некоторого языка, в АК отношение включения для двух кортежей трактуется иначе (см. п. 2.3)).

Определение 5.1. Отношением R_1 на $\boldsymbol{E}$ называется такое $X \subseteq \boldsymbol{E}^2$, что для всякой пары $(e_1, e_2) \in X$ имеет место $\forall \delta \in e_1, \exists \gamma \in e_2$, что $\gamma \subseteq \delta$.

Определение 5.2. Отношением R_2 на $\boldsymbol{E}$ называется такое $X \subseteq \boldsymbol{E}^2$, что для всякой пары $(e_1, e_2) \in X$ имеет место $\forall \delta \in e_1, \exists \gamma \in e_2$ что $\delta \cap \gamma \neq \emptyset$ и $\gamma \not\subset \delta$ и $\delta \not\subset \gamma$.

Определение 5.3. Отношением R_3 на $\boldsymbol{E}$ называется такое $X \subseteq \boldsymbol{E}^2$, что для всякой пары $(e_1, e_2) \in X$ имеет место $\exists \delta \in e_1, \exists \gamma \in e_2$, что $\gamma = \delta$.

Определение 5.4. Отношением R_4 на $\boldsymbol{E}$ называется такое $X \subseteq \boldsymbol{E}^2$, что для всякой пары $(e_1, e_2) \in X$ имеет место $\exists \delta \in e_1, \exists \gamma \in e_2$, что $\delta \cap \gamma \neq \emptyset$ и $(\delta \cup \gamma) \setminus (\delta \cap \gamma) \neq \emptyset$.

Определение 5.5. Отношением R_5 на $\boldsymbol{E}$ называется такое $X \subseteq \boldsymbol{E}^2$, что для всякой пары $(e_1, e_2) \in X$ имеет место $\forall \delta \in e_1, \exists \gamma \in e_2$, что $\delta \subset \gamma$.

Определение 5.6. Отношением R_6 на $\boldsymbol{E}$ называется такое $X \subseteq \boldsymbol{E}^2$, что для всякой пары $(e_1, e_2) \in X$ имеет место $\exists \delta \in e_1, \exists \gamma \in e_2$, что $\gamma \subset \delta$.

В [*Осипов*, 2009] дано естественно-языковое описание этих отношений. Так, например, принадлежность пары событий отношению R_1 означает, что событие e_1 всегда сопровождается событием e_2; а R_3 – событие e_1 иногда

сопровождается e_1. Там же вводятся специальные диаграммы для наглядного представления таких отношений: каждое отношение e_i представляется в виде прямоугольника, по горизонтальной стороне которого откладываются свойства отношения, а по вертикали – множества значений по каждому из свойств или объемы. В частности:

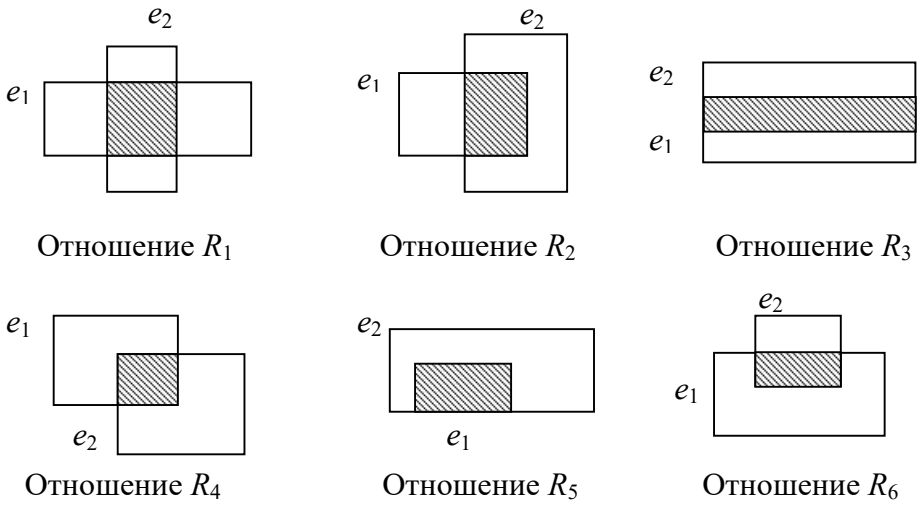

Рис. 5.9. Диаграммы отношений структурного сходства

На языке АК условия, соответствующие введенным определениям и рис. 5.9, примут следующий вид (жирным шрифтом выделены множества нефиктивных атрибутов):

1. Отношения e_1 и e_2 должны быть представимы в виде C-систем $E1[\boldsymbol{XYZ}]$ (как атрибуты $\boldsymbol{X}$, так и атрибуты $\boldsymbol{Z}$ могут отсутствовать) и $E2[\boldsymbol{Y}]$, а также удовлетворять соотношению $E1[\boldsymbol{XYZ}] \subseteq_G E2[\boldsymbol{Y}]$.

2. Отношения e_1 и e_2 должны выражаться как C-системы $E1[\boldsymbol{XY}]$ и $E2[\boldsymbol{YZ}]$, (атрибуты $\boldsymbol{X}$ и атрибуты $\boldsymbol{Z}$ либо одновременно присутствуют в схемах отношений, либо одновременно отсутствуют), и при этом требуется, чтобы $E1[\boldsymbol{Y}] \subseteq E2[\boldsymbol{Y}]$, где $E1[\boldsymbol{Y}]$, $E2[\boldsymbol{Y}]$ здесь и далее проекции соответствующих отношений на атрибут $\boldsymbol{Y}$.

3. Отношения e_1 и e_2 должны записываться в виде C-систем $E1[\boldsymbol{X}]$ и $E2[\boldsymbol{X}]$, причем $E1[\boldsymbol{X}] \cap E2[\boldsymbol{X}] \neq \varnothing$;

4. Отношения e_1 и e_2 должны соответствовать C-системам $E1[\boldsymbol{XY}]$ и $E2[\boldsymbol{YZ}]$ (либо атрибуты $\boldsymbol{X}$, либо атрибуты $\boldsymbol{Z}$ могут отсутствовать), для которых выполняется $E1[\boldsymbol{Y}] \cap E2[\boldsymbol{Y}] \neq \varnothing$.

5. Отношения e_1 и e_2 должны допускать представление в виде ***C***-систем $E1[\boldsymbol{Y}]$ и $E2[\boldsymbol{XYZ}]$ (либо атрибуты ***X***, либо атрибуты ***Z*** могут отсутствовать), и удовлетворять требованию $E1[\boldsymbol{Y}] \subseteq E2[\boldsymbol{Y}]$.

6. Отношения e_1 и e_2 должны выражаться как *C*-системы $E1[\boldsymbol{XYZ}]$ (либо атрибуты ***X***, либо атрибуты ***Z*** могут отсутствовать) и $E2[\boldsymbol{Y}]$, а также должно соблюдаться $E1[\boldsymbol{Y}] \cap E2[\boldsymbol{Y}] \neq \emptyset$.

При проверке условий 1-6 представление отношений e_1 и e_2 в виде множеств *C*-кортежей и применение методов АК, описанных в главах 2 и 3, позволяет существенно снизить трудоемкость операций, производимых над этими отношениями, а также более экономно расходовать память для их хранения.

5.3.3. АК и формальный анализ понятий

В отличие от статистических методов, формальный анализ понятий (ФАП) дает возможность описывать отношения между объектами изучаемых предметных областей в виде решеток формальных понятий [*Kuznetsov*, 2002].

Формальный контекст – это тройка $K = (G, M, I)$, где G – множество объектов, M – множество признаков, $I \subseteq G \times M$ – отношение обладания признаком. Пара $\langle g_i, m_j \rangle \in I$ показывает, что объект g_i обладает признаком m_j. Часто формальный контекст представляют в виде бинарной матрицы, например, такой, как в табл. 5.9.

Определение формального понятия содержит операторы Галуа A' и B'. Для $A \subseteq G$ и $B \subseteq M$: $A' = \{m \in M \mid \forall g \in A: gIm\}$, $B' = \{g \in G \mid \forall m \in B: gIm\}$. Другими словами, A' – множество признаков, которыми обладают все объекты из множества A, B' – множество объектов, которые обладают всеми признаками из множества B.

Таблица 5.9.

	m1	**m2**	**m3**	**m4**
g1			**X**	**X**
g2		**X**	**X**	
g3	**X**			**X**
g4	**X**	**X**	**X**	

Определение 5.7. Формальное понятие (A, B) состоит из множества объектов $A \subseteq G$ и множества признаков $B \subseteq M$, таких что $B' = A$ и $A' = B$. A называется объемом, а B – содержанием понятия. В матрице контекста формальное понятие представляет собой подматрицу, состоящую из единиц. Например, в табл. 5.9 есть формальное понятие – ([g2, g4], [m2, m3]).

Множество понятий контекста K образуют решетку формальных понятий $\beta(K)$, так как создают частичный порядок по вложению объемов понятий и всегда имеют наименьшее и наибольшее по вложению понятия. Находятся такие формальные понятия алгоритмом ″замыкай по одному″. Функция начинает работать с самого общего формального понятия, которое содержит все объекты и чаще всего ни одного признака. Затем находятся все остальные понятия рекурсивным добавлением признаков.

При помощи АК также можно генерировать понятия, удовлетворяющие определению 5.7 [*Зуенко*, 2010 b]. Для этого необходимо выполнить следующие шаги:

1) выписать в виде C-системы дополнение исходного отношения, заданного бинарной матрицей;

2) обратить эту C-систему (получить соответствующую D-систему);

3) преобразовать полученную D-систему в эквивалентную ей C-систему;

4) исключить дублирование C-кортежей в C-системе. Опишем процесс генерации понятий на основе табл. 5.9.

После выполнения первого шага получим следующую C-систему $K[XY]$ (X – множество объектов, Y – множество признаков этих объектов):

$$K[XY]=\begin{bmatrix} \{g_1\} & \{m_1, m_2\} \\ \{g_2\} & \{m_1, m_4\} \\ \{g_3\} & \{m_2, m_3\} \\ \{g_4\} & \{m_4\} \end{bmatrix}.$$ Тогда $$\overline{K}[XY] = \begin{bmatrix} \{g_2, g_3, g_4\} & \{m_3, m_4\} \\ \{g_1, g_3, g_4\} & \{m_2, m_3\} \\ \{g_1, g_2, g_4\} & \{m_1, m_4\} \\ \{g_1, g_2, g_3\} & \{m_1, m_2, m_3\} \end{bmatrix},$$ где

каждая компонента есть дополнение соответствующей компоненты C-системы $K[XY]$. Третий шаг алгоритма, реализуется в соответствии с методами, подробно описанными в главе 2:

$$\overline{K}\,[XY]=\begin{bmatrix}\{g_2,g_3,g_4\} & * \\ * & \{m_3,m_4\}\end{bmatrix}\cap\begin{bmatrix}\{g_1,g_3,g_4\} & * \\ * & \{m_2,m_3\}\end{bmatrix}\cap$$

$$\cap\begin{bmatrix}\{g_1,g_2,g_4\} & * \\ * & \{m_1,m_4\}\end{bmatrix}\cap\begin{bmatrix}\{g_1,g_2,g_3\} & * \\ * & \{m_1,m_2,m_3\}\end{bmatrix}.$$

После исключения дублирования в результирующей *C*-системе остаются такие *C*-кортежи:

[{g_1, g_2, g_4}, {m_3}], [{g_1, g_3}, {m_4}], [{g_3, g_4}, {m_1}], [{g_2, g_4}, {m_2, m_3}], [{g_1}, {m_3, m_4}], [{g_3}, {m_1, m_4}], [{g_4}, {m_1, m_2, m_3}].

Каждый них соотносится с определенным формальным понятием, в чем можно убедиться, используя табл. 5.9. Таким образом, генерацию понятий ФАП можно свести к задаче преобразования *D*-системы в *C*-систему. Следовательно, АК может успешно применяться при построении решеток понятий и прикладных систем искусственного интеллекта, использующих этот аппарат.

5.3.4. Поисковые системы: математическая модель понятия "вопрос"

В общем случае вопрос содержит три компоненты [*Гетманова*, 1995]: 1) искомое знание; 2) то, что известно; 3) требование или намерение получить искомое знание (в естественном языке это выражается вопросительными местоимениями – кто, где, когда и т.д. – и знаком вопроса или интонацией в устной речи, а в инструментальных средствах вопросно-ответных систем в виде определенных команд). Другие необходимые компоненты упоминаются реже. К ним относятся: 4) предполагаемый источник искомых знаний и 5) точное указание на категории (множества, атрибуты), к которым относятся искомое знание и то, что известно. В вопросах на естественном языке эта компонента может отсутствовать, так как предполагается, что собеседник сам может ее без труда восстановить.

Различают несколько видов вопросов. В общем случае их можно разделить на две категории [*Гетманова*, 1995]: 1) ***уточняющие вопросы*** или ли-вопросы (Верно ли, что жирафы живут в Африке?) и 2) ***восполняющие вопросы***, т.е. вопросы, начинающиеся с вопросительных местоимений "кто", "где", "когда" и т.д. Д.А. Поспелов дополняет эту классификацию еще тремя типами вопросов: 3) ***ПОЧЕМУ-вопросы***; 4) ***ЗАЧЕМ-вопросы*** и 5) ***КАК-вопросы*** [*Поспелов*, 1989]. Кроме того, вопросы подразделяются на ***простые*** и ***сложные***. В сложных

уточняющих вопросах то, что неизвестно, имеет вид условных предложений (например, "Правда ли, что если *A* то *B*?"). Для восполняющих вопросов то, что известно, может быть выражено в виде сложных предложений – такие вопросы можно разложить на множество простых вопросов.

Сложности начинаются при переходе от общения между людьми (здесь тоже есть свои трудности, например, найти "знатока" или правильно сформулировать вопрос, но на этих проблемах мы останавливаться не будем) к инструментальным вопросно-ответным системам (или подсистемам). Здесь основных трудности две:

1) разработать язык запросов, с помощью которого можно формулировать вопросы так, чтобы легко было перевести его с естественного языка на язык машинных команд;

2) структурировать многообразные источники информации таким образом, чтобы их можно было обрабатывать, используя язык запросов. В этом направлении создано немало оригинальных разработок (реляционная алгебра, языки баз данных, Semantic Web, различные поисковые системы, реляционно-ситуационный анализ текстов **[***Осипов*, 2009**]**).

Существуют разнообразные поисковые или вопросно-ответные системы, основанные на следующих основных принципах:

1) информация, к которой поступают запросы, преобразуется таким образом, чтобы она была представлена в виде определенных структур. Речь идет, в основном, о таких структурах, как отношения (в частности, таблицы) или элементы определенных отношений, графы, сети (семантические, по сочетаемости и т.д.), частично упорядоченные множества (в частности, решетки), правила и т.д. Например, предложения естественного языка часто преобразуются в элементы определенных отношений. Так предложение "Студент Петров сдал экзамен по математике на отлично" может быть преобразовано в элемент отношения РЕЗУЛЬТАТЫ ЭКЗАМЕНОВ (Фамилия студента, Предмет, Оценка) и представлено в виде кортежа (Петров, математика, отлично).

2) язык запросов должен быть не только сопоставим с формулировками произвольных вопросов на естественном языке, но и инициировать процедуру поиска с учетом многообразия структур подготовленной к обработке информации.

Что касается процедуры поиска ответа на запрос, то она заключается в том, чтобы в соответствии со структурой запроса осуществлять поиск определенных фрагментов базы знаний, выполнять операции с отношениями (соединение, композиция, проекция и т.д.), если требуется, и искать кортежи, которые совпадают с тем, что известно в вопросе. Такая процедура достаточно сложна и трудоемка, и ее суть рассматривается в основном с точки зрения программистов. Точной и однозначной математической формулировки этой процедуры пока что не найдено. В настоящем подразделе сделана попытка найти решение описанной проблемы. В качестве инструмента использована алгебра кортежей.

Рассмотрим сначала ***восполняющие вопросы***. Пусть задан произвольный АК-объект $Q[XYZ]$ = [{a} {b} *]. В данном случае Q есть C-кортеж, в котором значения атрибутов X и Y определены, а атрибут Z может принимать любое значение в пределах своего домена. Этот АК-объект нетрудно сформулировать как вопрос: "Каково значение атрибута Z, если атрибуты X и Y заданы?". Такой вопрос имеет смысл, если в качестве исходной информации определено отношение $R[\ldots XYZ\ldots]$. Тогда ответ на вопрос можно найти, выполнив операцию $R[\ldots XYZ\ldots] \cap_G Q[XYZ]$. Может оказаться, что будет получен результат, в котором атрибут Z имеет более одного значения и, тем не менее, ответ на вопрос может быть получен. Усложнять восполняющие вопросы допустимо в следующих двух направлениях.

1) Вопрос может иметь более сложную форму. Например, требуется определить произвольное значение Z при заданных значениях X и Y или значение того же атрибута в диапазоне D при других значениях атрибутов X и Y. Пример такого вопроса можно представить как C-систему

$$Q_1[XYZ] = \begin{bmatrix} \{a\} & \{c\} & * \\ \{a,b\} & \{e\} & D \end{bmatrix}.$$

Ответ на вопрос также находится с помощью обобщенного пересечения: $R[\ldots XYZ\ldots] \cap_G Q_1[XYZ]$.

2) В исходной информации отсутствует отношение, в котором содержатся все атрибуты вопроса. В этом случае необходимо найти объекты, схемы отношений которых содержат подмножества атрибутов, содержащихся в схеме отношения вопроса, и вычислить их соединение. Приведем пример. Пусть в системе содержатся два отношения $P[XYZ]$ и $R[YVW]$, и требуется ответить на

вопрос, каковы значения атрибутов X и V, если $Z = a$. В заданных условиях сам вопрос на языке АК выражается как С-кортеж $Q_2[XVZ] = [* * \{a\}]$, а отношение, содержащее все атрибуты вопроса, можно найти соединением отношений $P[XYZ]$ и $R[YVW]$. В итоге для получения ответа на вопрос требуется выполнить следующие операции:

$(P[XYZ] \oplus R[YVW]) \cap_G Q_2[XVZ]$.

Поскольку соединение отношений равносильно обобщенному пересечению, то вышеприведенную формулу можно записать так:

$P[XYZ] \cap_G R[YVW] \cap_G Q_2[XVZ]$.

Учитывая ассоциативность и коммутативность операции $\cap_G$, можно выбрать порядок вычислений, обеспечивающий существенное сокращение трудоемкости.

Таким образом, восполняющие вопросы могут быть формализованы как АК-объекты, в которых искомое знание представлено определенным диапазоном значений, а процедура поиска ответа заключается в вычислении обобщенного пересечения соответствующих АК-объектов.

Рассмотрим ***уточняющие вопросы***. В некоторых случаях они сводятся к восполняющим вопросам, например, когда речь идет о существовании какого-то определенного факта. Однако в большинстве случаев уточняющие вопросы имеют более сложную структуру. Пусть в результате поиска по атрибутам, содержащимся в вопросе, получено некоторое отношение $\boldsymbol{R}$, которое является подходящей для данного вопроса базой знаний, а $\boldsymbol{Q}$ – искомое знание в уточняющем вопросе типа "Верно ли, что $\boldsymbol{Q}$?". Проанализируем два случая.

1) Уточняющий вопрос о существовании, содержащий в искомом знании более одного факта. В этом случае $\boldsymbol{Q}$ представлено АК-объектом, содержащим некоторое множество элементарных кортежей. Тогда ответ может быть получен после проверки обобщенного включения $\boldsymbol{Q} \subseteq_G \boldsymbol{R}$.

2) В качестве искомого знания в уточняющем вопросе содержится определенная закономерность, выразимая в виде АК-объекта $\boldsymbol{Q}$. Тогда она должна соблюдаться для всех элементарных кортежей отношения $\boldsymbol{R}$. Значит, положительный ответ на вопрос может быть получен, если при проверке подтвердится соотношение $\boldsymbol{R} \subseteq_G \boldsymbol{Q}$.

Заключение

Приятны завершенные труды.
Гомер.

В книге разработан новый математический аппарат – алгебра кортежей, которая относится к классу булевых алгебр и позволяет реализовывать алгебраический подход к логическому анализу в системах искусственного интеллекта. Алгебра кортежей использует в качестве основной структуры не элементарный кортеж, а декартово произведение множеств, и реализует общую теорию многоместных отношений. В отличие от формальных систем, где основа – символьные конструкции, в качестве базового понятия АК выбрано многоместное отношение и предложены обобщения операций алгебры множеств для работы с отношениями, заданными в разных схемах. Это позволило расширить возможности существующих систем обработки данных и знаний, основанных на бинарных и реляционных отношениях. В алгебре кортежей, помимо известных методов логических исчислений, реализованы новые алгебраические методы проверки корректности следствия и поиска следствий из заданной системы аксиом.

Предложенные в книге методы сопровождаются четкими и обоснованными алгоритмами решения многих общих задач логического анализа, к тому же структуры АК, характеризующиеся матрицеподобной формой, позволяют эффективно распараллеливать операции. Кроме того, разработаны дополнительные методы уменьшения трудоемкости, а в некоторых случаях – и вычислительной сложности интеллектуальных процедур. В АК выявлены новые структурные и статистические классы конъюнктивных нормальных форм с полиномиально распознаваемым свойством выполнимости. Таким образом, многие алгоритмы, имеющие в теории высокую оценку сложности, в АК исполняются в среднем за полиномиальное время.

В ходе вывода учитывается внутренняя структура обрабатываемых знаний, что ускоряет решение стандартных задач логического анализа. Алгебра кортежей дает возможность унифицировать представление и анализ как данных, так и знаний, и, следовательно, решить проблему сопряжения баз данных и баз знаний в рамках одной программной системы.

Помимо логического вывода, АК служит для формализации и решения

широкого круга логических задач (абдуктивные и модифицируемые заключения, моделирование графов и семантических сетей, экспертных правил и т.д.).

С точки зрения интеллектуализации баз данных АК представляет собой расширение реляционной алгебры на задачу обработки знаний. По мнению авторов, АК может служить основой программных средств для совместной обработки данных и знаний в интеллектуальных системах.

Предложенный в рамках АК подход к логико-семантическому анализу позволяет решать следующие задачи:

выразить в единых структурах данные и знания;

использовать в семантических исследованиях универсальные методы анализа модифицируемых рассуждений, не нарушая при этом законов классической логики;

за многообразием терминов и инструментальных средств увидеть единую математическую основу семантического анализа.

Будущие исследования предполагается вести в следующих основных направлениях:

алгебраический анализ и синтез динамических систем, в том числе динамических интеллектуальных систем [*Виноградов*, 2002], в рамках ситуационного подхода [*Фридман*, 2010];

моделирование вопросно-ответных и обучающих систем [*Сулейманов*, 2010; *Зуенко* 2010 b];

контекстно-ориентированные системы управления базами данных и знаний [*Зуенко*, 2008 d];

исследование дополнительных возможностей погружения структур АК в измеримые пространства.

Список литературы

Арнольд, В. И. Математическая дуэль вокруг Бурбаки. / В. И. Арнольд // Вестник РАН. 2002. – Т. 72, – № 3. – С. 245-250.

Башмаков, А. И. Интеллектуальные информационные технологии: Учебн. пособие. / А. И. Башмаков, И. А. Башмаков – М., Изд-во МГТУ им. Н.Э. Баумана, 2005. – 304 с.

Бурбаки, Н. Теория множеств. / Бурбаки Н. – М., Мир, 1965. – 465 с.

Вагин, В. Н. Достоверный и правдоподобный вывод в интеллектуальных системах. / В. Н. Вагин, Е. Ю. Головина, А. А. Загорянская, М. В. Фомина / Под ред. В. Н. Вагина, Д. А. Поспелова. – М., ФИЗМАТЛИТ, 2004. – 704 с.

Виноградов, А. Н. Динамические интеллектуальные системы. – Ч.1. Представление знаний и основные алгоритмы. / А. Н. Виноградов, Л. Ю. Жилякова, Г. С. Осипов // Известия РАН. Теория и системы управления. – М., Наука, 2002. – С. 87-94.

Гетманова, А. Д. Учебник по логике. / А. Д. Гетманова – М., "Владос", 1995. – 303 с.

Генцен, Г. Исследование логических выводов. / Г. Генцен – В кн.: Математическая теория логического вывода. М., Наука, 1967. – С. 9 – 74.

Гильберт, Д. Основы теоретической логики. / Д. Гильберт, В. Аккреман – М., ИЛ, 1947. – 302 с.

Гладкий, А. В. Введение в современную логику. / А. В. Гладкий – М., МЦНМО, 2001. – 200 с.

Глушков, В .М. Алгебра. Языки. Программирование. / В. М. Глушков, Г. Е. Цейтлин, Е. Л. Ющенко – Киев, Наукова думка, 1989. – 376 с.

Гэри, М. Вычислительные машины и труднорешаемые задачи. / М. Гэри, Л. Джонсон – М., Мир, 1982. – 416 с.

Данцин, Е. А. Алгоритмика задачи выполнимости / Е. А. Данцин // Вопросы кибернетики. Проблемы сокращения перебора. М., 1987. – С. 7-29.

Дейт, К. Дж. Введение в системы баз данных. / К. Дж Дейт – 6-е издание. – Киев, М., СПб., Издательский дом «Вильямс», 1999. – 848 с.

Закревский, А. Д. Представление знаний и логический вывод в пространстве многозначных признаков. / А. Д. Закревский – В кн. Логика и компьютер. – Вып. 2: Логические языки, содержащие рассуждения и методы поиска

доказательств. – М., Наука, 1995. – С. 3-16.

Зуенко, А. А. Анализ контекстов при моделировании слабо формализованных предметных областей / А. А. Зуенко, Б. А. Кулик, А. Я. Фридман // Двенадцатая национальная конференция по искусственному интеллекту с международным участием КИИ-2010 (20 -24 сентября 2010 г., г. Тверь, Россия): Труды конференции. Т.2 – М.: Физматлит, 2010. – С.164-172.

Зуенко, А. А. Контекстный подход в системах сопровождения открытых моделей предметной области. / А. А. Зуенко, А. Я. Фридман // Искусственный интеллект и принятие решений. 2008. – №3. – С.41-51.

Зуенко, А. А. Логический вывод при семантическом анализе нерегламентированных путевых запросов. / А. А. Зуенко, А. Я. Фридман // Одиннадцатая национальная конференция по искусственному интеллекту с международным участием КИИ-2008 (28 сентября – 3 октября 2008 г., Дубна, Россия): труды конференции. – М., ЛЕНАНД, 2008. – Т. 1. – С.298-304.

Зуенко, А. А. Примеры применения алгебры кортежей в интеллектуальном анализе данных / А. А. Зуенко, Б. А. Кулик, А. Я. Фридман // Двенадцатая национальная конференция по искусственному интеллекту с международным участием КИИ-2010 (20-24 сентября 2010 г., г.Тверь, Россия): Труды конференции. Т.3 – М.: Физматлит, 2010. – С.279-287.

Зуенко, А. А. Метод семантического анализа нерегламентированных запросов в реляционной базе данных с иерархической структурой. / А. А. Зуенко, А. Я. Фридман // Труды ИСА РАН. Прикладные проблемы управления макросистемами / Под ред. Ю.С. Попкова, В.А. Путилова. – Т. 39. – М., Книжный дом "ЛИБРОКОМ", 2008. – С. 141-159.

Зуенко, А. А. Развитие алгебры кортежей для логического анализа баз данных с использованием двуместных предикатов. / А. А. Зуенко, А. Я. Фридман // Известия РАН. Теория и системы управления. 2009. – №2. – С. 95-103.

Зуенко, А. А. Унификация обработки данных и знаний на основе общей теории многоместных отношений. / А. А. Зуенко, Б. А. Кулик, А. Я. Фридман // Искусственный интеллект и принятие решений, 2010. – Вып. 3. – С. 52-62.

Зуенко, А. А. Управление контекстом при организации интеллектуализированного интерфейса БД в системах моделирования на основе концептуального подхода. / А. А. Зуенко, А. Я. Фридман // Труды

ИСА РАН. Прикладные проблемы управления макросистемами / Под ред. Ю.С. Попкова, В.А. Путилова. Т. 39. – М., Книжный дом «ЛИБРОКОМ», 2008. – С.128-141.

Игошин, В. И. Математическая логика и теория алгоритмов. / В. И. Игошин – М., Изд. центр "Академия", 2008. – 448 с.

Индейцев, А. И. Система интеллектуальной поддержки борьбы за живучесть надводного корабля. / А. И. Индейцев, А. Г. Сергеев // Методы и средства информационной поддержки борьбы за живучесть надводных кораблей. – СПб., ИПМаш РАН. 1995. – С. 15-35.

Искусственный интеллект. – В 3-х книгах. Кн. 2. Модели и методы: Справочник/ Под ред. Д. А. Поспелова – М., Радио и связь. 1990. – 304 с.

Клини, С. Математическая логика./ С. Клини, – М., Мир. 1973. – 480 с.

Колмогоров, А. Н. Элементы теории функций и функционального анализа. / А. Н. Колмогоров, С. В. Фомин – М., Наука, 1972. – 496 с.

Кофман, А. Введение в теорию нечетких множеств. / А. Кофман – М., Радио и связь, 1982. – 432 с.

Кристофидес, Н. Теория графов. Алгоритмический подход. / Н. Кристофидес – М., Мир, 1978. – 432 с.

Кузичев, *А. С.* Диаграммы Венна. / *А. С.* Кузичев – М.: Наука, 1968. – 252 с.

Кулик, Б. А. Вероятностная логика на основе алгебры кортежей. / Б. А. Кулик // Известия РАН. Теория и системы управления. 2007. – № 1. – С. 118-127.

Кулик, Б. А. Логика естественных рассуждений. / Б. А. Кулик – СПб., Невский диалект 2001. – 128 с.

Кулик, Б. А. Логические основы здравого смысла. / Б. А. Кулик / Под ред. Д. А. Поспелова. – СПб., Политехника. 1997. – 131 с.

Кулик, Б. А. Математическое отношение как основная структура логики. / Б. А. Кулик // Труды междунар. научной школы "Моделирование и анализ безопасности и риска в сложных системах – 2008" СПб., ГУАП, 2008. – С. 190-192.

Кулик, Б. А. Моделирование рассуждений на основе законов алгебры множеств. / Б. А. Кулик // Труды Пятой национальной конференции по искусственному интеллекту, Казань, 7-11 октября 1996 г. – Т.1. – С. 58-61.

Кулик, Б. А. Новые классы КНФ с полиномиально распознаваемым свойством выполнимости. / Б. А. Кулик // Автоматика и телемеханика. 1995. – № 2. –

С. 111-124.

Кулик, Б. А. Обобщенный подход к моделированию и анализу интеллектуальных систем на основе алгебры кортежей. / Б. А. Кулик // Труды VI Международной конференции «Идентификация систем и задачи управления» SICPRO’07. – М., ИПУ РАН, 2007. – С. 679-715.

Кулик, Б. А. Система логического программирования на основе алгебры кортежей. / Б. А. Кулик // Изв. РАН. Техн. кибернетика. 1993. – № 3. – С. 226-239.

Кулик, Б. А. Управление логико-семантическим анализом на основе теории отношений. / Б. А. Кулик, А. А. Зуенко, А. Я. Фридман // VIII Всероссийская школа-семинар «Прикладные проблемы управления макросистемами» (Апатиты, 29 марта – 2 апреля 2010 г.). Материалы докладов. – Апатиты, изд-во КНЦ РАН, 2008. – С. 23-24.

Курант, Р. Что такое математика. Элементарный очерк идей и методов. / Р. Курант, Г. Роббинс – М-Л., ОГИЗ, 1947. – 664 с.

Левин, Л. А. Универсальные задачи перебора. / Л. А. Левин // Проблемы передачи информации. 1973. – Вып.3. – С.115-116.

Липский, В. Комбинаторика для программистов. / В. Липский – М., "Мир", 1988. – 213 с.

Лукасевич, Я. Аристотелевская силлогистика с точки зрения современной формальной логики. / Я. Лукасевич – М., 1959. – 312 с.

Мальцев, А. И. Алгебраические системы. / А. И. Мальцев – М., Наука. 1970. – 392 с.

Маслов, С. Ю. Теория дедуктивных систем и ее применения. / С. Ю. Маслов – М., Радио и связь. 1986. – 136 с.

Мелихов, А. Н. Ориентированные графы и конечные автоматы. / А. Н. Мелихов – М., Наука, 1971. – 416 с.

Мендельсон, Э. Введение в математическую логику. / Э. Мендельсон – М., Наука. 1984. – 320 с.

Непейвода, Н. Н. Прикладная логика: учебное пособие. / Н. Н. Непейвода – Ижевск,. Издательство Удмуртского университета, 1997. – 385 с.

Новиков, П. С. Элементы математической логики. / П. С. Новиков – М., Наука.

1973. – 400 с.

Новиков, Ф. А. Дискретная математика для программистов. / Ф. А. Новиков – СПб., Питер. 2002. – 304 с.

Осипов, Г. С. Лекции по искусственному интеллекту. / Г. С. Осипов – М., КРАСАНД, 2009. – 272 с.

Порецкий, П. С. Решение общей задачи теории вероятностей при помощи математической логики. / П. С. Порецкий // Собрание протоколов заседаний секции физико-математических наук общества естествоиспытателей при Казанском университете. – Казань, 1887. – Т. 5. – С. 83-116.

Поспелов, Д. А. Моделирование рассуждений. Опыт анализа мыслительных актов. / Д. А. Поспелов – М., Радио и связь, 1989. – 184 с.

Представление и использование знаний: пер. с япон. / под ред. Х. Уэно, М. Исидаука – М., Мир, 1989. – 220 с.

Рассел, С. Искусственный интеллект: современный подход, 2-е изд.: пер. с англ. / С. Рассел, П. Норвиг – М., Издательский дом "Вильямс", 2006. – 1408 с.

Рейнгольд, Э. Комбинаторные алгоритмы. Теория и практика. / Э. Рейнгольд, Ю. Нивергельт, Н. Део – М., "Мир", 1980. – 476 с.

Рябинин, И .А. Надежность и безопасность структурно-сложных систем. / И .А. Рябинин – СПб., Политехника, 2000. – 248 с.

Рябинин, И. А. Логико-вероятностные методы исследования надежности структурно-сложных систем. / И. А. Рябинин, Г. Н.Черкесов – М,. Радио и связь, 1981. – 264 с.

Сикорский, Р. Булевы алгебры. / Р. Сикорский – М., Мир, 1969. – 375 с.

Скорняков, Л. А. Элементы теории структур. / Л. А. Скорняков – М., Наука, Гл. ред. физ.-мат. лит. 1982. – 160 с.

Смаллиан, Р. М. Принцесса или тигр? / Р М. Смаллиан – М., Мир, 1986. – 221 с.

Смирнов В. А. Логические методы анализа научного знания. / В. А. Смирнов – М., Наука, 1987. – 256 с.

Смолин, Д. В. Введение в искусственный интеллект. / Д. В. Смолин – М., ФИЗМАТЛИТ, 2004. – 208 с.

Соложенцев, Е. Д. Сценарное логико-вероятностное управление риском в

бизнесе и технике. / Е. Д. Соложенцев – СПб., Издательский дом "Бизнес-пресса". 2004. – 432 с.

Стокмейер, Л. Классификация вычислительной сложности проблем / Л. Стокмейер // Кибернетический сборник, новая серия. – М., Мир. 1989. – Вып. 26. – С. 20-83.

Столл, Р. Р. Множества. Логика. Аксиоматические теории. / Р. Р. Столл – М., "Просвещение", 1968. – 232 с.

Страбыкин, Д. А. Метод логического вывода модифицируемых рассуждений. / Д. А. Страбыкин, М. Н. Томчук // Известия РАН. Теория и системы управления. 2008. – № 2. – С. 89–95.

Стяжкин, Н. И. Формирование математической логики. / Н. И. Стяжкин – М., Наука, 1967. – 508 с.

Сулейманов, Д. Ш. Управление процессом обучения с использованием интеллектуальной подсистемы контроля ответов обучаемого. / Д. Ш. Сулейманов // VIII Всероссийская школа-семинар "Прикладные проблемы управления макросистемами" (Апатиты, 29 марта – 2 апреля 2010 г.): Тезисы докладов. – Апатиты, КНЦ РАН, 2010. – С. 72, 73.

Такеути, Г. Теория доказательств. / Г. Такеути – М., Мир, 1978. – 412 с.

Тейз, А. Логический подход к искусственному интеллекту: от классической логики к логическому программированию. / А. Тейз, П. Грибомон, Ж. Луи и др. – М., Мир, 1990. – 432 с.

Ульман, Д. Д. Введение в системы баз данных. / Д. Д. Ульман, Д. Уидом – М., Издательство «Лори», 2000. – 374 с.

Фридман, А. Я. Ситуационное моделирование природно-технических комплексов. / А.Я. Фридман, О.В. Фридман, А.А. Зуенко. – СПб., Издательство Политехнического ун-та, 2010. – 436 с.

Халмош, П. М. Теория меры. / П. М. Халмош – ИЛ, 1953. – 232 с.

Чень, Ч. Математическая логика и автоматическое доказательство теорем. / Ч. Чень, Р. Ли – М., Наука. 1983. – 360 с.

Чери, С. Логическое программирование и базы данных. / С. Чери, Г. Готлоб, Л. Танка – М., Мир, 1992. – 352 с.

Яблонский, С. В. Функции алгебры логики и классы Поста. / С. В. Яблонский,

Г. П. Гаврилов, В. Б. Кудрявцев – М., Наука, 1966. – 120 с.

Яновская, С. А. Математическая логика и основания математики. В кн.: Математика в СССР за сорок лет. / С. А. Яновская - Физматгиз. 1959. – Т. 1 – С. 13-120.

Codd, E. F. A relational model of data for large shared data banks. / E. F. Codd // Comm. ACM, 1970, – v. 13, – № 6. – pp. 377-387.

Codd, E. F. Normalized Data Base Structure./ E. F. Codd // A Brief Tutorial Proc. 1971 ACM SIGFIDET Workshop on Data Description, Access and Control. - pp. 1-17.

Codd, E. F. Relational completeness of data base sublanguages. / E. F. Codd // Data Base Systems: Proc of Courant Computer Science Symposia 6, New York City, May 24-25. 1971. – Prentice Hill, 1972. – pp. 33-64.

Cook, S. A. The complexity of theorem proving procedures. / S. A. Cook // Proceedings of the 3rd ACM Symposium on Theory of Computing, 1971. – pp. 151-158. [Рус. перевод: С.А.Кук. Сложность процедур вывода теорем // Кибернетический сборник, новая серия. – М., Мир. 1972. – Вып. 12. – С. 5-15].

Davis, M. A computing procedure for quantification theory. / M. Davis, H. Putnam // J. ACM, 1960. – v. 1. – pp. 201-215.

Fagin, R. Normal Forms and Relational Database Operations. / R. Fagin // Proc. 1971 ACM SIGMOD Intern. Conf. On Management of Data. – pp. 153-160.

Hartmanis, J. On the computational complexity of algorithms. / J. Hartmanis, R.E. Stearns // Transactions of the American Mathematical Society, 1965. v. 117. – pp. 285-306. [Русский перевод: Хартманис Дж., Стирнз Р.Е. О вычислительной сложности алгоритмов // Кибернетический сборник, новая серия. – М., Мир. 1967. – Вып. 4. – С. 57-86].

Karp, R. M. Reducibility among computational problems. / R. M. Karp // Complexity of computer computations, Plenum press, New York, 1972. – pp. 85-104. [Русский перевод: Р.М.Карп. Сводимость комбинаторных проблем // Кибернетический сборник, новая серия. – М., Мир. 1972. – Вып. 12. – С. 16-38].

Krom, M. R. The decision problem for a class of first-order formulas in which all disjunctions are binary. / M. R. Krom // Z. math. Logic Grundl. Math., 1967, 13,

- pp. 15-20.

Kulik, B. Algebraic Approach to Logical Inference Implementation. / B. Kulik, A. Fridman, A. Zuenko // Computing and Informatics (CAI), Slovakia (в печати).

Kulik, B. Algebraic Method of Intelligent Data and Knowledge Processing. / B. Kulik, A. Fridman, A. Zuenko // Proceedings of First Russia and Pacific Conference on Computer Technology and Applications (Vladivostok, 6-9 September, 2010) – pp.130-135.

Kulik, B. Logical Analysis of Intelligence Systems by Algebraic Method. / B. Kulik, A. Fridman, A. Zuenko // Cybernetics and Systems 2010: Proceedings of Twentieth European Meeting on Cybernetics and Systems Research (EMCSR 2010) Vienna, Austria. 2010. – pp.198-203.

Kuznetsov, S. Comparing Performance of Algorithms for Generating Concept Lattices / S. Kuznetsov, S. Obiedkov // Journal of Experimental and Theoretical Artificial Intelligence, 2002. – v. 14.

Nilsson, N. J. Probabilistic Logic. / *N. J. Nilsson* // Artificial Intelligence. 1986. – №28. – pp. 71-87.

Rabin, M. O. Degree of difficulty of computing a function and a partial ordering of recursive sets. / M. O. Rabin – Technical Report 2. Hebrew University. Jerusalem. 1960.

Russel, S. Artificial Intelligence: A Modern Approach. Second edition. / S. Russel, P. Norvig – Prentice Hall, 2003. – 1081 p.

Перечень условных обозначений и сокращений

АК - алгебра кортежей

АУК - алгебра условных кортежей

БД - база данных

БЗ - база знаний

В∀ - введение общности

В∃ - введение существования

ВД - введение дизъюнкции

ВИ - введение импликации

ВК – введение конъюнкции

ВО – введение отрицания

ДНФ – дизъюнктивная нормальная форма

ДС – правило Дунса-Скотта

ИТ – закон исключенного третьего

КНФ – конъюнктивная нормальная форма

ЛВА – логико-вероятностный анализ

МКИ – метод квантования интервалов

СДНФ – совершенная ДНФ

СУБД – система управления базами данных

ТВ – правило тривиальной выводимости

ТФС – теория формальных систем

ФРС – функция работоспособности системы

У∀ – удаление общности

У∃ – удаление существования

УД – удаление дизъюнкции

УИ – удаление импликации

УК – удаление конъюнкции

УО – удаление отрицания

ЭНС – экстенсиональная неоднородная семантическая сеть

SAT – задача проверки выполнимости заданной КНФ

SQL – Structured Query Language – язык структурированных запросов

Приложение 1. Сводка теорем алгебры кортежей

Формулировка теоремы	*№ теоремы в тексте, № страницы*
Теорема П1.1. Алгебра кортежей для однотипных АК-объектов изоморфна алгебре множеств.	***2.1.,*** 75
Теорема П1.2. $P \cap Q = [P_1 \cap Q_1 \;\; P_2 \cap Q_2 \;\; \ldots \;\; P_n \cap Q_n]$.	***2.2.,*** 76
Теорема П1.3. $P \subseteq Q$, если и только если $P_i \subseteq Q_i$ для всех $i = 1, 2, \ldots, n$.	***2.3.,*** 76
Теорема П1.4. $P \cup Q \subseteq [P_1 \cup Q_1 \;\; P_2 \cup Q_2 \ldots P_n \cup Q_n]$, причем равенство возможно лишь в двух случаях: (iii) $P \subseteq Q$ или $Q \subseteq P$; (iv) $P_i = Q_i$ для всех соответствующих пар компонент, за исключением одной пары.	***2.4.,*** 76
Теорема П1.5. Пересечение двух однотипных *C*-систем равно *C*-системе, содержащей все непустые пересечения каждого *C*-кортежа первой *C*-системы с каждым *C*-кортежем второй *C*-системы.	***2.5.,*** 76
Теорема П1.6. Объединение двух однотипных *C*-систем равно *C*-системе, содержащей все *C*-кортежи операндов.	***2.6.,*** 77
Теорема П1.7. Для произвольного *C*-кортежа $P = [P_1 \, P_2 \ldots P_n]$ его дополнение равно *D*-кортежу $\overline{P} =]\overline{P_1} \;\; \overline{P_2} \;\; \ldots \;\; \overline{P_n}[$.	***2.7.,*** 78
Теорема П1.8. Дополнение *C*-системы есть *D*-система той же размерности, в которой каждая компонента равна дополнению соответствующей компоненты в исходной *C*-системе.	***2.8.,*** 79
Теорема П1.9. Для произвольного *D*-кортежа $P =]P_1 \, P_2 \ldots P_n[$ его дополнение равно *C*-кортежу $\overline{P} = [\overline{P_1} \;\; \overline{P_2} \;\; \ldots \;\; \overline{P_n}]$.	***2.9.,*** 80

Формулировка теоремы	*№ теоремы в тексте, № страницы*
Теорема П1.10. Дополнение *D*-системы есть *C*-система той же размерности, в которой каждая компонента равна дополнению соответствующей компоненты в исходной *D*-системе.	***2.10.,*** 80
Теорема П1.11. $]P_1 \cap Q_1 \quad P_2 \cap Q_2 \ldots P_n \cap Q_n[\subseteq P \cap Q$, причем равенство возможно лишь в двух случаях:	
(iii) $P \subseteq Q$ или $Q \subseteq P$;	***2.11.,*** 80
(iv) $P_i = Q_i$ для всех соответствующих пар компонент, за исключением одной пары.	
Теорема П1.12. $P \subseteq Q$, если и только если $P_i \subseteq Q_i$ для всех $i = 1, 2, \ldots, n$.	***2.12.,*** 81
Теорема П1.13. $P \cup Q =]P_1 \cup Q_1 \;\; P_2 \cup Q_2 \ldots P_n \cup Q_n[$.	***2.13.,*** 81
Теорема П1.14. Пересечение двух однотипных *D*-систем равно *D*-системе, содержащей все *D*-кортежи операндов.	***2.14.,*** 81
Теорема П1.15. Объединение *D*-систем *P* и *Q* эквивалентно *D*-системе, состоящей из всех не равных универсуму *D*-кортежей, образующихся при выполнении операций объединения каждого *D*-кортежа из *P* с каждым *D*-кортежем из *Q*.	***2.15.,*** 81
Теорема П1.16. Для *C*-кортежа $P = [P_1 P_2 \ldots P_n]$ и *D*-кортежа $Q =]Q_1 Q_2 \ldots Q_n[$ справедливо $P \subseteq Q$, если и только если по крайней мере для одного *i* соблюдается $P_i \subseteq Q_i$.	***2.16.,*** 82
Теорема П1.17. Для *C*-кортежа (или *D*-кортежа) *P* и *D*-системы *Q* справедливо $P \subseteq Q$, если и только если для каждого *D*-кортежа Q_j из *Q* выполняется $P \subseteq Q_j$.	***2.17.,*** 82
Теорема П1.18. Для *C*-системы *P* и *D*-системы *Q* справедливо $P \subseteq Q$, если и только если каждый *C*-кортеж из *P* включен в каждый *D*-кортеж из *Q*.	***2.18.,*** 82

Формулировка теоремы	*№ теоремы в тексте, № страницы*
Теорема П1.19. Каждый *C*-кортеж (*D*-кортеж) *P* преобразуется в эквивалентную ему диагональную *D*-систему (*C*-систему).	***2.19.,*** 83
Теорема П1.20. *D*-система *P,* содержащая *m* *D*-кортежей, эквивалентна *C*-системе, которая равна пересечению *m* *C*-систем, полученных с помощью преобразования каждого *D*-кортежа из *P* в *C*-систему.	***2.20.,*** 83
Теорема П1.21. *C*-система *P*, содержащая *m* *C*-кортежей, эквивалентна *D*-системе, которая равна объединению *m* *D*-систем, полученных с помощью преобразования каждого *C*-кортежа из *P* в *D*-систему.	***2.21.,*** 84
Теорема П1.22. Добавление нового фиктивного атрибута в *D*-кортеж или в *D*-систему соответствует правилу обобщения.	***2.22.,*** 97
Теорема П1.23. Пусть $R[\ldots X\ldots]$ – *C*-система, у которой отсутствуют *C*-кортежи с пустыми компонентами в атрибуте *X.* Тогда для соответствующего этой *C*-системе предиката $P(\ldots, x, \ldots)$ результат операции $-X(R)$ моделирует формулу $\exists x(P)$.	***2.23.,*** 97
Теорема П1.24. Пусть $R[\ldots X\ldots]$ – *D*-система, у которой отсутствуют *D*-кортежи с компонентами "*" в атрибуте *X*. Тогда для соответствующего этой *D*-системе предиката $P(\ldots, x, \ldots)$ результат операции $-X(R)$ моделирует формулу $\forall x(P)$.	***2.24.,*** 98
Теорема П1.25. *D*-кортеж вида $]Q_1\ Q_2\ \ldots\ Q_{m-1}\ Q_m[$, где Q_i – произвольные компоненты, преобразуется в эквивалентную ему ортогональную *C*-систему*:* $\begin{bmatrix} Q_1 & * & \ldots & * & * \\ \overline{Q_1} & Q_2 & \ldots & * & * \\ & & \ldots & & \\ \overline{Q_1} & \overline{Q_2} & \ldots & Q_{m-1} & * \\ \overline{Q_1} & \overline{Q_2} & \ldots & \overline{Q_{m-1}} & Q_m \end{bmatrix}$.	***3.1.,*** 102

Формулировка теоремы	*№ теоремы в тексте, № страницы*
Теорема П1.26. Если P и Q – ортогональные C-системы, то пересечение этих C-систем, вычисленное в соответствии с теоремой 2.5, либо пусто, либо состоит из одного C-кортежа, либо представляет собой ортогональную C-систему.	***3.2.,*** 103
Теорема П1.27. При разбиении матрицы D-системы T размерности $M \times N$ на R вертикальных блоков ($R < N$) справедливо равенство $$\boldsymbol{T} = \bigcup_{k \in S(R,M)} \left(\bigcap_{i=1}^{M} D_{ij}\right)$$ где D_{ij} – D-кортеж, являющийся подстрокой i-ой строки, взятой из j-го блока матрицы T, а индекс j в подформуле $\bigcap_{i=1}^{M} D_{ij}$ пробегает все значения соответствующего элемента $k \in S(R, M)$.	***3.3.,*** 108
Теорема П1.28. Если D-система Q содержит монотонный блок, то она непуста и $C_{int} \subseteq Q$.	***3.4.,*** 110
Теорема П1.29. Если в D-системе Q имеется бесконфликтный блок, то Q непуста, если и только если при разложении ее в блочную матрицу T непуста D-система, представленная блоком T_{22}.	***3.5.,*** 111
Теорема П1.30. Если в C-системе Q существует атрибут X, такой что пересечение всех C-кортежей в проекции $–X(Q)$ непусто и равно C-кортежу C, то при добавлении в C атрибута X с компонентой, равной объединению всех компонент этого атрибута в Q, образуется C-кортеж C_1, такой что $C_1 \subseteq Q$.	***3.6.,*** 122
Теорема П1.31. Если для логических формул A и B имеется интерпретация в виде множеств S_A и S_B, то общезначимость импликации $A \supset B$ равносильна выполнению отношения $S_A \subseteq S_B$.	***4.1.,*** 130

Формулировка теоремы	***№ теоремы в тексте, № страницы***
Теорема П1.32. Пусть даны АК-объекты $F_1, ..., F_n$ и G. Тогда G есть следствие $F_1, ..., F_n$ тогда и только тогда, когда $(F_1 \cap_G ... \cap_G F_n) \neq \varnothing$ и $(F_1 \cap_G ... \cap_G F_n) \subseteq G$.	***4.2.,*** 132
Теорема П1.33. Пусть даны АК-объекты $F_1, ..., F_n$ и G. Тогда G есть следствие $F_1, ..., F_n$ тогда и только тогда, когда $(F_1 \cap_G ... \cap_G F_n) \neq \varnothing$ и $F_1 \cap_G ... \cap_G F_n \cap_G \overline{G} = \varnothing$.	***4.3.,*** 132
Теорема П1.34. Если каждая компонента C-кортежа имеет конечную меру, то мерой C-кортежа является произведение мер его компонент.	***5.1.,*** 164
Теорема П1.35. Мера ортогональной C-системы равна сумме мер содержащихся в ней C-кортежей.	***5.2***., 165
Теорема П1.36. Если для логической формулы F осуществима элементаризация, то ее представление $S(F)$ в виде структуры АК в вероятностном пространстве есть точное решение уравнения регрессии.	***5.3***., 180

Приложение 2. АК и логические исчисления

Алгебра кортежей	**Исчисление высказываний и предикатов**
Элементарный кортеж, принадлежащий АК-объекту	Выполняющая подстановка соответствующей формулы
C-кортеж	Конъюнкция одноместных предикатов или конъюнкция в исчислении высказываний
C-система	Дизъюнктивная нормальная форма (ДНФ)
D-кортеж	Дизъюнкция одноместных предикатов или дизъюнкция в исчислении высказываний
D-система	Конъюнктивная нормальная форма (КНФ)
Непустой АК-объект	Выполнимая формула
АК-объект, равный частному универсуму	Тождественно истинная формула
АК-объект, равный ∅	Тождественно ложная формула
Операция $+Atr$ (если добавляемый атрибут отсутствует в схеме отношения АК-объекта)	Правило обобщения или равносильное по смыслу преобразование формулы
Операция $-Atr$ для *C*-кортежей и *C*-систем	Навешивание квантора $\exists_{Atr}$
Операция $-Atr$ для *D*-кортежей и *D*-систем	Навешивание квантора $\forall_{Atr}$
Включение одного АК-объекта в другой	Логический вывод, отношение выводимости

Предметный указатель

Абдуктивное заключение 149
Абдукция 149
АК-объект 72
– однотипный 73
Аксиомы
– исчисления высказываний 45
– исчисления предикатов 48
– меры множеств 159
Алгебра
– булева 12
– кортежей 6,14
– множеств 6, 15
– реляционная 6
– условных кортежей 186
Алгебраическая система 9
– многосортная 9
Алгоритм
– выполнения кванторных операций 121
– поиска абдуктивных заключений 152
– проверки включения АК-объектов 113
– решения задач вероятностной логики 176
– решения задачи выполнимость КНФ 116
Альтернативные классы 81
Атрибут 72
Блок
– бесконфликтный 110
– конфликтный 110
Брус 160
Вершина графа 191
Вывод формулы 36
Выполнимость КНФ 54
Вычислительная сложность
– алгоритма 55
Гипотеза 146
Грань 11
– верхняя 11
– нижняя 11
– точная верхняя 11
Графы 191
Дедуктивная система 38
Декартово произведение 61
Декларативный подход 4
Диаграммы Эйлера 17
Дизъюнктивная нормальная форма (ДНФ) 50
Дизъюнкция 27
– строгая 30
Доказательство 36
– комбинаторным способом 22
– на основе принципа резолюции 52
– табличным способом 33
Домен 72

– бесконфликтный 110
– конфликтный 110
– монотонный 110
АУК-объект 187
– атрибуты 186
– – простые 186
– – сложные 186
Базы данных 5
– дедуктивные 191
Базы знаний 5
– Де Моргана 13
– дистрибутивности 21
– инволюции 21
– исключенного третьего 21
– контрапозиции 13
– коммутативности 12
– непротиворечия 21
– поглощения 12
– сохранения 23
– транзитивности 22
Замкнутость 12
Идемпотентности свойство 12
Импликация 27
Интервалы
– вырожденные 160
– несовместные 163
– элементарные 159
Интерпретация 38
– логических исчислений 90
– логического вывода 130
Истинностные таблицы 29
Исчисление 38
Дополнение
– в алгебре кортежей 78
– в алгебре множеств 18
– в решетке 12
Дуга графа 192
Задача
– NP-полная 56
– #P-полная 56
Законы
– ассоциативности 12
Мера множеств 159
Метод квантования интервалов 162
Минимальное покрытие 123
Множество 10, 16
– частично упорядоченное 10
Модель 10
Модифицируемые рассуждения 141
Модус силлогизма 41
Монотонность логического вывода 23
Мощность множеств 63
– относительная 158
Носитель 10
Обобщенные операции и отношения 89
Объединение
– в алгебре кортежей 76
– в алгебре множеств 19
– в решетке 209
– декартовых произведений 65

– высказываний 31, 45
натуральное 46
– предикатов многосортное 50
Квантор 37
Коллизия 141
– неадекватности 142
– парадокса 142
– цикла 142
Композиция отношений 88
Компоненты АК-объектов 73
– фиктивные: 73
– – полная компонента 73
– – пустое множество 73
Конъюнктивная нормальная форма (КНФ) 51
Конъюнкция 27
Кортеж 59, 72
Логика
– вероятностная 173
– немонотонная 15, 24
– в алгебре множеств 18
– декартовых произведений 63
Подстановка формулы
– выполняющая 32
Поисковые системы 210
Полисиллогизм 43
Правила вывода 36
– исчисления предикатов 47
– модус поненс 45
– натурального исчисления высказываний 45
– обобщения 96
Операция наполнения значениями АУК-объектов 188
Ортогонализация 101
Ортогональная
– ДНФ 101
– *C*-система 102
Отношение 6
– антисимметричности 11
– бинарное 6, 68
– включения 16
– выводимости 36, 89
– принадлежности 16
– рефлексивности 11
– транзитивности 11
– частичного порядка 10
Отображение 67
Отрицание 27
Пересечение
– в решетке 209
– в алгебре кортежей 76
– *D*-кортеж 78
– *D*-система 79
– диагональная *C*-система 77
– элементарный кортеж 73
Схема отношения 50, 73
Теорема
– дедукции 45
– Стоуна 13
Теория
– бинарных отношений 6
– сложности вычислений 53
– формальных систем 3

– подстановки 45
– продукционные 196
– семантические 133
– следствий в АК 133
Предикат 39, 60
Принцип резолюции 52
Проекция 139
Процедурный подход 4
Пустое множество 17
Разность
– декартовых произведений 66
– множеств 19
Ребро графа 192
Реляционная алгебра 182
Решетка 11
– Булева 12
– ограниченная 12
Семантические сети 69
– неоднородные 205
Силлогизм 39
Следствие
– в алгебре кортежей 133
Слот 69
Соединение отношений 87
Соответствие 67
Сорт 43, 72
Структуры алгебры кортежей
– *C*-кортеж 74
– *C*-система 75
Транзитивное замыкание графа 192
Универсум 18
Условный *C*-кортеж 187
Условная *C*-система 187
Фигура силлогизма 41
Формальная система 35
Формальная теория 36
Формальный анализ понятий 208
Формула
– выполнимая 31
– общезначимая 31
– тождественно истинная 31
– тождественно ложная 31
Фрейм 69
Функционально полный базис 30
Функция работоспособности системы 167
Частный универсум 73
Эквивалентность 30
Экспоненциальная катастрофа 5
Элемент 16
Язык
– исчисления предикатов 37

О Г Л А В Л Е Н И Е

Предисловие …..…….. 3

Введение …… 9

Глава 1. Основные математические структуры …… 15

1.1. Алгебра множеств ……. 15

1.2. Алгебра логики ……..… 27

1.3. Булевы алгебры и системы логического вывода …… 34

1.3.1. Формальные системы ……..…. 34

1.3.2. Силлогистика Аристотеля и алгебра множеств … 39

1.3.3. Логические исчисления и их интерпретация …… 45

1.3.4. Сложность алгоритмов логического вывода (задача "выполнимость КНФ") ……... 53

1.4. Отношения в математике и информационных системах .. 57

1.4.1. Понятие "отношение" и декартово произведение множеств ……... 59

1.4.2. Бинарные отношения ……...… 67

1.4.3. Отношения в реляционной алгебре …… 68

1.4.4. Отношения в логике и искусственном интеллекте 69

Глава 2. Теоретические основы алгебры кортежей …..… 71

2.1. Основные термины и структуры ……..…… 71

2.2. Преобразования АК-объектов в альтернативные классы .. 83

2.3. Операции с атрибутами, операции соединения и композиции, обобщенные операции ……..… 85

2.4. Логические исчисления и АК ……...…... 90

2.4.1. АК как интерпретация логических исчислений … 90

2.4.2. Соответствие между АК и исчислением предикатов …….……... 96

Глава 3. Методы снижения трудоемкости в АК ……. 100

3.1. Ортогонализация ……... 101

3.2. Матричные свойства АК-объектов …….…. 107

3.3. Алгоритм проверки включения *C*-системы в *C*-систему ... 113

3.4. Алгоритм решения задачи "Выполнимость КНФ" …… 116
3.5. Алгоритмы выполнения кванторных операций ……. 121
3.6. Оценка вычислительной сложности алгоритмов …..…… 126
Глава 4. Логический вывод и анализ модифицируемых рассуждений в АК …… 130
4.1. Интерпретация логического вывода ……..… 130
4.2. Формулировки задач и алгоритмы логического вывода… 133
4.2.1. Типы задач логического вывода …… 133
4.2.2. Алгоритмы решения задачи проверки правильности следствия ……..... 134
4.2.3. Алгоритмы решения задачи вывода произвольных следствий ….…… 139
4.3. Анализ модифицируемых рассуждений ……..…. 141
4.3.1. Коллизии в рассуждениях ……..…….. 141
4.3.2. Анализ гипотез ……. 146
4.3.3. Абдуктивные заключения ……..….. 149
Глава 5. Управление данными и знаниями на базе алгебры кортежей……. 157
5.1. Метрические аспекты алгебры кортежей ……..…. 157
5.1.1. Представление измеримых систем в алгебре кортежей …… 157
5.1.2. Логико-вероятностный анализ и алгебра кортежей …… 166
5.1.3. Вероятностная логика на основе АК ……..… 173
5.2. Работа с данными в структурах АК ……...……..… 181
5.2.1. Реляционные СУБД …… 181
5.2.2. Анализ незапланированных запросов (реляционные СУБД)…… 186
5.2.3. Дедуктивные СУБД ……..… 191
5.3. Системы искусственного интеллекта …….. 196
5.3.1. Представление знаний в АК ……..…….. 196
5.3.2. АК и неоднородные семантические сети ……..…. 205

5.3.3. АК и формальный анализ понятий …… 208
5.3.4. Поисковые системы: математическая модель понятия "вопрос" …….. 210
Заключение ……..... 214
Список использованных источников ……..…… 216
Перечень условных обозначений и сокращений …….. 224
Приложение 1. Сводка теорем алгебры кортежей …..……. 225
Приложение 2. АК и логические исчисления ……..……..… 230
Предметный указатель……... 231
Оглавление……... 235