

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Учебное пособие
по выполнению практических занятий
по дисциплине «МАТЕМАТИЧЕСКАЯ ЛОГИКА И ТЕОРИЯ
АЛГОРИТМОВ»
для студентов специальности 10.05.01 Компьютерная безопасность
специализация «Информационно-аналитическая и техническая экспертиза
компьютерных систем»

**Ставрополь
2026**

ВВЕДЕНИЕ

Главным содержанием лекционного курса дисциплины «Математическая логика и теория алгоритмов» является формирования у студентов фундаментальных знаний в области основ построения математических теорий, а также построения и анализа алгоритмов.

Основными задачами освоения дисциплины являются:

- формирование научного мировоззрения, понимания широты и универсальности методов математической логики, умения применять эти методы в решении прикладных задач;
- воспитание математической культуры, которая предполагает четкое осознание необходимости и важности математической подготовки для специалиста в области компьютерной безопасности;
- ознакомление с основными объектами математической логики, а также их приложениями для решения различных задач, требующих применения вычислительных средств;
- ознакомление с основами теории алгоритмов и их приложениями к задачам математической кибернетики;
- овладение современным математическим аппаратом для дальнейшего использования при решении теоретических и прикладных задач в области защиты информации.

Материал предлагаемого пособия ориентирован на студентов математических и IT-специальностей. Главная цель, которую преследовали авторы при написании пособия, состоит в ознакомлении студентов с концептуальными положениями теории алгоритмов и математической логики, выросшей из формальной логики, с приложениями формальных исчислений к построению содержательных теорий первого порядка и систем искусственного интеллекта.

1. ЭЛЕМЕНТАРНОЕ ВВЕДЕНИЕ В ТЕОРИЮ МНОЖЕСТВ

План занятия

1. Теоретико-множественная символика и терминология.
2. Операции над множествами и их основные свойства.
3. Декартово произведение множеств и отношения на множествах.
4. Элементы теории отображений.
5. Отношение эквивалентности и разбиение множества на классы.
6. Отношения порядка.
7. Базовые алгебраические системы.

Согласно канторовскому определению, множество M есть любая совокупность определенных и различных между собой объектов нашей интуиции или интеллекта, мыслимых как единое целое.

Понятие множества относится к категории наиболее общих, основополагающих понятий математики. Поэтому вместо строгого определения обычно принимается некоторое основное положение о множестве и его элементах.

Теория множеств как математическая дисциплина создана немецким математиком Г. Кантором (1845-1918) в послед. четверти XIX в. Исчерпывающее освещение предпосылок и проблем, связанных с ее возникновением и развитием требует довольно серьезных предварительных математических сведений, поэтому мы ограничимся кратким обзором этих вопросов.

Проводившиеся Кантором исследования тригонометрических рядов и числовых последовательностей привели его к задаче выяснения тех средств, которые необходимы для сравнения бесконечных множеств чисел по величине. Для решения этой проблем Кантор ввел понятие мощности множества, считая по определению, что два множества имеют одинаковую мощность, если члены любого их них можно сопоставить членам другого, образуя пары соответствующих членов. Поскольку между членами двух конечных множеств можно установить такое попарное соответствие в том и только в том случае, когда они имеют одинаковое число членов, мощность конечного множества можно отождествить с количественным числом. Таким образом, понятие мощности бесконечного множества представляет собой обобщение обычного понятия количественного числа.

В построении теории таких обобщений (или трансфинитных) чисел, включающих в себя их арифметику, и состояло создание Кантором теории множеств.

И как раз тогда, когда идеи Кантора стали завоевывать умы математиков и получать признания в 90-х гг. XIX в., в высших разделах его теории множеств были обнаружены противоречия.

Так, например, в 1902 г. был открыт парадокс Рассела, который связан с множеством всех множеств, которые не являются элементами самих себя. Обозначим это множество через M . Пусть множество M содержит само себя. Тогда, согласно определению множества M , оно не содержит самого себя. Но отсюда по определению M получаем: M содержит само себя. Получили противоречие, парадокс.

Рассел в 1919 г. предложил также следующий популярный вариант этого парадокса: парикмахер, живущий в некоторой деревне, бреет всех тех и только тех жителей этой деревни, которые не бреются сами. Вопрос: бреет ли он самого себя?

Приведем еще два парадокса.

Дилемма крокодила. Крокодил украл ребенка, но обещал вернуть его отцу, если тот отгадает, вернет ли ему крокодил ребенка. Неразрешимая дилемма встанет перед крокодилом, если отец скажет ему, что он не вернет ребенка.

Парадокс миссионера. Миссионер, очутившийся среди людоедов, обнаруживает, что он угодил как раз к обеду. Они разрешают ему произнести какое-нибудь высказывание с условием, что, если высказывание окажется истинным, его сварят, а если оно окажется ложным, его зажарят. Что надо сказать миссионеру?

Перечисленные выше и другие известные парадоксы демонстрируют трудности, неизбежно связанные с попытками построить теорию множеств на интуитивной основе, исходя из канторовской концепции множества. Попытки разрешения этих парадоксов привели к созданию аксиоматической теории множеств (Цермело, 1908 г.), хотя до сих пор в математическом мире полного согласия в вопросе о происхождении парадоксов и способах избавления от них нет.

Опыт парадоксов теории множеств «научил» математиков крайне осторожно обращаться с бесконечностью и по возможности даже о бесконечности рассуждать с помощью финитных методов. Существо финитного подхода заключается в том, что он допускает только конечные комплексы действий над конечным числом объектов. Выяснение того, какие объекты и действия над ними следует считать точно определенными, какими свойствами и возможностями обладают комбинации элементарных действий, что можно и чего нельзя сделать с их помощью все это стало предметом теории алгоритмов и формальных систем. Это направление первоначально возникло в рамках математики и стало важнейшей ее частью.

1. 1. Теоретико-множественная символика и терминология

Утверждение о том, что множество M состоит из различных элементов $m_1, m_2, \dots, m_n$ (и только из этих элементов) условно записывается $M = \{ m_1, m_2, \dots, m_n \}$. Принадлежность элемента множеству (отношение принадлежности) обозначается символом $\in$, т.е. $m_1 \in M, m_2 \in M, \dots, m_n \in M$.

Если a не является элементом множества M , то пишут $a \notin M$ или $a \in \bar{M}$.

Интуитивный принцип объемности. Два множества A и B называются равными (пишут $A = B$) в том и только в том случае, когда они состоят из одних и тех же элементов. Другими словами, $A = B$ тогда и только тогда, когда каждый элемент множества A является элементом множества B и наоборот, каждый элемент множества B является элементом множества A . В аксиоматической системе Цермело построения теории множеств этот принцип носит название аксиомы объемности.

Неравенство множеств A и B обозначается $A \neq B$. Это означает, что, по крайней мере, в одном из множеств A или B имеется элемент, которого нет в другом множестве (то есть или во множестве A есть такой элемент, которого нет во множестве B , или во множестве B есть такой элемент, которого нет во множестве A , или же имеет место то и другое).

Казалось бы, решение вопроса о том, равны два множества или нет, не должно представлять особых трудностей. Однако это не так. Приведем простой пример двух множеств A и B , очень просто определяемых, о которых до сих пор мы не можем решить, равны они или нет.

Пусть A множество всех четных чисел, больших 4, а B множество всех чисел, являющихся суммами двух простых нечетных чисел. Мы до сих пор не знаем, какое из соотношений справедливо: $A = B$ или $A \neq B$, и не знаем даже, как подойти к решению этого вопроса. В то же время можно легко доказать, что каждый элемент множества B является элементом множества A . Действительно, поскольку простым наименьшим нечетным числом является 3, сумма двух нечетных простых чисел всегда будет четным числом, неменьшим 6, следовательно, большим 4, откуда следует наше утверждение. В 1742 году Х. Гольдбах высказал предположение о том, что каждый элемент множества A является элементом множества B , а следовательно, что $A = B$. Однако предположение Гольдбаха до сих пор не доказано и не опровергнуто.

Множество, состоящее из одного элемента, называется синглетоном.

Множество, не содержащее ни одного элемента, называется пустым множеством и обозначается символом $\emptyset$. Роль пустого множества аналогична роли числа 0. Это понятие можно использовать для определения заведомо несуществующей совокупности элементов (например, множество действительных корней уравнения $x^2 + 1 = 0$ пусто). Более существенным мотивом введения пустого

множества является тот факт, что заранее не всегда известно (или неизвестно вовсе) существуют ли элементы, определяющие какое-то множество или нет. Например, мы не знаем, является пустым или нет множество всех решений в целых числах уравнения $x^3 + y^3 + z^3 = 30$. В то же время доказано, что множество всех решений этого уравнения в целых положительных числах является пустым.

Множество A , все элементы которого принадлежат множеству B , называется подмножеством (частью) множества B . Это отношение между множествами называют включением и обозначают символом $\subset$ (или $\subseteq$), то есть $A \subset B$ или $B \supset A$. Отношение $A \subset B$ допускает и равенство $A = B$, то есть любое множество можно рассматривать и как подмножество самого себя $A \subset A$. Полагают также, что подмножеством любого множества является пустое множество, то есть $\emptyset \subset A$.

Таким образом, любое непустое множество A имеет, по крайней мере, два различных подмножества: само множество A и пустое множество $\emptyset$. Эти подмножества называют несобственными, а все другие подмножества A называют собственными. При этом элементы множества сами могут являться некоторыми множествами. Множество, элементами которого являются все подмножества множества A , называется множеством подмножеств множества A и обозначают через $P(A)$.

Непустое множество может содержать любое число элементов конечное или бесконечное. Соответственно имеем конечные или бесконечные множества. Если число элементов множества может быть выражено с помощью целого положительного числа, является конечным. Например, множество всех решений в положительных числах уравнения $x^3 + y^3 + z^3 = 3$ конечно (то оно целых (существует только одно такое решение: $x = y = z = 1$)). В то же время мы не знаем, конечно или нет множество всех решений этого уравнения в целых числах, и не знаем никакого метода, который привел бы нас к решению этого вопроса. Известно, напротив, что множество всех решений в целых числах уравнения $x^3 + y^3 + z^3 = 1$ бесконечно (что сразу следует из тождества $(9n+4)^3 + (1-9n)^3 + (3n-9n+4)^3 = 1$ при $n = 1, 2, 3, \dots$). Существуют конечные множества, число элементов которых нам неизвестно. Таково, например, множество всех натуральных делителей числа $2^{217} + 1$. Но в этом случае мы можем указать число, большее, чем число элементов этого множества, например, число 2^{217} . Существуют множества, большие, чем число их членов. Таким является, например, множество T , определяемое следующим образом: если существует бесконечно много положительных четных чисел, не являющихся суммами двух простых чисел, то во множество T включаем только число 1; если же существует только конечное число таких чисел, то во множество T включаем все эти числа.

В случае конечного множества A , состоящего из n элементов, множество подмножеств $P(A)$ содержит 2^n элементов.

Обозначение множества с помощью фигурных скобок при явном его задании пригодно в случае, когда множество состоит из небольшого числа элементов. Однако этот способ задания множества становится слишком громоздким, если его использовать для задания множеств, имеющих, хотя и конечное, но большое число элементов, и вообще неприменимо для бесконечных множеств. В последнем случае множества задают определяющим (характеристическим) свойством $P(x)$ (формой от x), общим для всех элементов определяемого множества. Обычно $P(x)$ это высказывание, в котором что-то утверждается об объекте x , или некоторая функция от переменной x . Проще говоря, $P(x)$ это одно или несколько свойств, которыми обладают элементы x , входящие в данное множество.

Множество, заданное с помощью характеристического свойства $P(x)$ обозначается так: $X = \{x \mid P(x)\}$ или $X = \{x : P(x)\}$, например, $\{x \mid x^2 = 13\}$ множество действительных чисел, квадрат которых равен 13.

Надежным способом точно описать свойство элементов данного множества служит задание распознающей (или, как говорят в математике, разрешающей) процедуры, которая

для любого объекта устанавливает, обладает он данным свойством и, следовательно, является элементом данного множества или нет. Например, для множества со свойством «быть степенью числа a » разрешающей процедурой служит любой метод разложения целых чисел на простые множители.

При изучении множеств мы абстрагируемся или от природы и порядка их элементов, или же только от природы и обращаем внимание на их порядок. Если мы отвлекаемся от природы и порядка их элементов, то два множества могут отличаться своей численностью. Допустим, что у нас есть два коробка спичек: в одном белые спички, в другом красные. Вынем из каждого коробка по одной спичке и отложим их в сторону. Из оставшихся спичек вынем снова по одной спичке из каждого коробка и вновь отложим их в сторону. Вынимая последовательно по одной спичке из каждого коробка, мы либо исчерпаем оба коробка одновременно, либо один из коробков окажется пустым, когда в другом еще останутся спички. Первый случай, очевидно, будет иметь место тогда и только тогда, когда в обоих коробках было одно и то же число спичек, во втором случае коробок, оставшийся пустым, когда в другом еще были спички, содержал меньше спичек.

Итак, желая убедиться, равночисленны ли два данные множества, мы не должны обязательно пересчитывать элементы этих множеств; достаточно последовательно брать попарно по одному элементу из каждого из этих множеств. Два множества будут равночисленны тогда и только тогда, когда, выбирая попарно по одному элементу из каждого из этих множеств, мы исчерпаем их одновременно. Результат не будет зависеть от порядка извлечения элементов из каждого из двух данных конечных множеств. Можно сказать также, что два множества равночисленны, если их элементы можно соединить в пары таким образом, чтобы в каждой паре было по одному элементу из каждого из этих множеств (и чтобы каждый элемент каждого из этих множеств имел себе пару). В этом случае каждому элементу первого множества соответствует определенный и единственный элемент второго множества, а именно тот, что образует с ним пару. И наоборот, каждый элемент второго множества соответствует определенному и единственному элементу первого множества. В этом случае говорят, что между элементами, образующими наши множества, установлено взаимно однозначное соответствие. Для установления такого соответствия совсем не обязательно поочередно соединять в пары элементы, принадлежащие двум разным множествам; достаточно указать закон, определяющий это соединение в пары. Например, будет установлено взаимно однозначное соответствие между всеми нечетными положительными числами и всеми четными числами, большими ста, если каждому нечетному числу n мы поставим в соответствие четное число, большее его на 101 (то есть число $n+101$).

Если между элементами двух множеств можно установить взаимно однозначное соответствие, то говорят, что эти множества равномощны (или эквивалентны). Два конечных множества равномощны тогда и только тогда, когда они имеют одно и то же число элементов.

Вообще, мощностью конечного множества M называется число его элементов, которое обозначается символом $|M|$. Мощность бесконечного множества определяется через понятие соответствия. Мощность пустого множества считается равной 0.

К бесконечным множествам также применимо понятие равномощности множеств. Например, как мы видели, множество всех нечетных положительных чисел равномощно множеству всех четных чисел, больших ста. В этом случае, однако, могут иметь место факты, которые, на первый взгляд, кажутся парадоксальными и не поддающимися здравому смыслу и нашему воображению. Так, например, множество всех натуральных чисел равномощно множеству всех четных положительных чисел. Взаимно однозначное соответствие между элементами этих множеств можно установить, например, сопоставив каждому натуральному числу вдвое большее положительное число. Мы имеем здесь пример бесконечного множества, равномощного своей собственной части. В то же время не существует ни одного конечного множества, равномощного какой-либо своей

собственной части. Возникает вопрос: каждое ли бесконечное множество равномощно какой-либо своей собственной части? На этот вопрос нельзя ответить, не принимая специальных аксиом. Однако для многих бесконечных множеств, встречающихся в математике, можно доказать, что они все же равномощны некоторой своей собственной части.

Множества, равномощные множеству N всех натуральных чисел, называют счетными, остальные бесконечные множества называют несчетными. Следовательно, счетное множество – это такое бесконечное множество, все элементы которого можно перенумеровать с помощью натуральных чисел таким образом, чтобы каждому элементу множества соответствовал определенный номер, и чтобы каждый номер соответствовал элементу множества. Другими словами, элементы счетного множества можно расположить в виде бесконечной последовательности

$$u_1, u_2, u_3, \dots \quad (*)$$

по порядковым номерам (индексам). Обратно, множество всех членов последовательности (с различными членами) является счетным. Счетными, например, являются множества рациональных чисел, целых чисел, степеней числа 2 и др.

Очевидно, что часть счетного множества, если она не является конечным множеством, будет счетным множеством. Это следует из того, что элементы любой части множества (*) можно расположить в виде последовательности в порядке возрастания индексов.

Множество всех действительных чисел на отрезке $[0, 1]$ не является счетным (мы не доказываем этого утверждения это теорема Кантора), следовательно, множество всех действительных чисел несчетно. Несчетность множества всех действительных чисел можно доказать и не опираясь на теорему Кантора. Приведем одно из возможных доказательств этого факта.

Каждое действительное число имеет одно и только одно представление в виде бесконечной десятичной дроби, в которой бесконечно много цифр, отличных от 9. Доказательство несчетности множества всех действительных чисел можно вывести из общего утверждения о том, что для любой бесконечной последовательности действительных чисел вида (*) существует действительное число x , не совпадающее ни с одним из членов этой последовательности.

Пусть

$$u = k + 0, c^{(1)}c^{(1)}c^{(1)}\dots$$

$$_1 \quad 1 \quad 1 \quad 2 \quad 3$$

$$u = k + 0, c^{(2)}c^{(2)}c^{(2)}\dots$$

$$_2 \quad 2 \quad 1 \quad 2 \quad 3$$

$$u = k + 0, c^{(3)}c^{(3)}c^{(3)}\dots$$

$$_3 \quad 3 \quad 1 \quad 2 \quad 3$$

представления чисел $u_1, u_2, u_3, \dots$ в виде бесконечных десятичных дробей, каждая из которых содержит бесконечно много цифр, отличных от 9, где $k_1, k_2, k_3, \dots$ целые числа. Определим теперь бесконечную последовательность цифр c_n ($n = 1, 2, \dots$) следующим образом. Пусть для данного натурального числа n c_n будет наименьшей цифрой, отличной от 9. Очевидно, что имеется не меньше 8 цифр, отличных от $c_n^{(n)}$ и от 9. Таким образом, получим с $\Phi c(n)$ и с $\Phi 9$ при $n = 1, 2, \dots$. Бесконечная n p n

десятичная дробь $0, c_1 c_2 c_3 \dots$ будет, следовательно, представлять некоторое действительное число x такое, что $0 < x < 1$ и будет содержать бесконечно много цифр, отличных от 9. При этом $x \neq u_n$ при $n = 1, 2, \dots$, так как для любого натурального числа n дроби, представляющие числа x и u_n и содержащие бесконечно много цифр, отличных от 9, отличаются в n -ой цифре. Отсюда сразу вытекает, что множество всех действительных чисел не может быть счетным.

Ввиду наличия взаимно однозначного соответствия между множеством всех действительных чисел и точками прямой из факта несчетности множества всех

действительных чисел следует теорема о несчетности множества всех точек прямой.

Мощность множества всех действительных чисел называется мощностью континуума, а множества такой мощности называются континуальными. Мощность континуума обозначается символом $c = 2^{\aleph_0}$ (читается: два в степени алеф-нуль). Например, множество всех подмножеств счетного множества континуально.

Возникает вопрос: существуют ли на прямой несчетные множества точек, которые не были бы равномогущими? На этот вопрос мы не можем дать ответа. Предположение о том, что любое несчетное множество точек прямой имеет равную мощность с множеством всех точек прямой, носит название континуум-гипотезы. Эта гипотеза, впервые высказанная Г. Кантором, до сих пор не доказана, но К. Гёдель доказал, что она не противоречит общепринятым аксиомам теории множеств. В 1951 г. польский математик В. Серпинский доказал, что континуум-гипотеза равносильна следующей теореме, в формулировке которой не участвует понятие бесконечности: если из точки O трехмерного пространства провести три взаимно перпендикулярные прямые OX , OY , OZ , то множество всех точек пространства является объединением трех множеств, из которых первое конечно на любой прямой, параллельной прямой OX , второе конечно на любой прямой, параллельной прямой OY , а третье конечно на любой прямой, параллельной прямой OZ .

Рассмотрим теперь вопрос о том, каким образом сравнивать между собой мощности бесконечных множеств, которые имеют неодинаковую мощность. Если множество A имеет одинаковую мощность с какой-то частью множества B , а множество B не равномогуще никакой части множества A , то естественно считать мощность множества B большей, чем мощность множества A . Таким образом, мощность множества всех действительных чисел больше мощности множества всех натуральных чисел.

Мы знаем уже две разные мощности бесконечных множеств: мощность счетных множеств и мощность множества всех действительных чисел — мощность континуума. Существуют ли бесконечные множества с мощностью, большей мощности континуума? Ответ на этот вопрос положительный. Можно доказать, что таковым является, например, множество всех различных подмножеств множества действительных чисел. Г. Кантор доказал, что мощность множества всех различных подмножеств каждого непустого множества всегда больше мощности самого этого множества. Отсюда вытекает, что существует бесконечно много разных мощностей бесконечных множеств. Более того, не существует множества максимальной мощности. Для измерения мощности произвольных множеств вводятся так называемые кардинальные числа.

Приведем сводку основных понятий и определений, рассмотренных в данном пункте.

1. Множества и их элементы в рассуждениях обозначаются графическими символами, которые в основном являются буквами (различных алфавитов) или комбинациями букв и других знаков; отношения между элементами одного или нескольких множеств записывают, вставляя символы, обозначающие эти элементы, в характеристическую схему рассматриваемого отношения или свойства. Буква может обозначать как определенный элемент некоторого множества, так и произвольный его элемент (называемый также переменной, аргументом или общим элементом). Заменяя в отношении или свойстве произвольный элемент определенным элементом того же множества, говорят, что переменной придан в качестве значения этот определенный элемент. Отношение или свойство, в котором участвуют произвольные элементы, называют тождеством, если, какие бы значения не придавать этим элементам, оно становится истинным высказыванием.

2. Записывая два символа по обе стороны от знака « $=$ », получаем отношение, называемое отношением равенства, означающее, что эти два символа изображают один и тот же элемент.

3. Пусть дано множество E и некоторое свойство его общего элемента. Элементы из множества E , обладающие этим свойством, образуют новое множество, называемое

частью или подмножеством множества E . Таким образом, два эквивалентных свойства определяют одно и то же подмножество, и обратно.

4. Множество всех подмножеств данного множества A обозначают символом $P(A)$, при этом само множество $A \in P(A)$ и пустое множество $\emptyset \in P(A)$.

5. Для любого множества X считается, что $X \subset X$ и $\emptyset \subset X$ (несобственные подмножества).

6. Отношение $X \subset Y$ не исключает возможности $X = Y$.

7. Одновременное включение $X \subset Y$ и $Y \subset X$ эквивалентно отношению $X = Y$ (интуитивный принцип объемности, лежащий в основе доказательства теоретико-множественных тождеств).

8. Два множества называются равномошными, если их элементы можно поставить во взаимно однозначное соответствие.

9. Множество положительных целых чисел может рассматриваться как множество мощностей конечных подмножеств некоторого бесконечного множества; множество называется счетным, если оно равномошно некоторой части множества положительных целых чисел. Таким образом, всякое конечное множество счетно. Всякое счетное бесконечное множество равномошно множеству всех целых положительных чисел, в частности, бесконечное подмножество множества целых положительных чисел имеет ту же мощность, что и все множество целых положительных чисел.

10. Мощность множества всех действительных чисел несчетно. Его мощность называется мощностью континуума. До сих пор не доказана и не опровергнута континуум-гипотеза: любое несчетное множество точек прямой имеет равную мощность с множеством всех точек прямой.

11. Существуют мощности большие, чем мощность континуума и таких мощностей бесконечно много. Более того, не существует множества максимальной мощности.

1. 2. Операции над множествами и их основные свойства

Над множествами можно выполнять ряд операций, среди которых в этом пункте мы определим пять операций: объединение, пересечение, разность, дополнение и симметрическую разность.

Определение 1.1. Объединением множеств A и B (обозначение: $A \cup B$) называется множество, состоящее из тех и только тех элементов, которые принадлежат хотя бы одному из множеств A или B . Символически это можно записать так:

$$A \cup B = \{x | x \in A \text{ или } x \in B\}.$$

Аналогично определяется объединение произвольной (в том числе бесконечной) системы множеств. В случае их объединения обозначают одним из символов: $\bigcup_{i=1}^n A_i$ или $\bigcup_{A \in S} A$, где $S = \{A_1, A_2, \dots, A_n\}$.

$\bigcup_{i=1}^n$

$i=1$

и A_i , где буквой I обозначено некоторое множество индексов. $i \in I$

Определение 1.2. Пересечением множеств A и B (обозначение: $A \cap B$) называется множество, состоящее из тех и только тех элементов, которые принадлежат множествам A и B одновременно:

$$A \cap B = \{x | x \in A \text{ и } x \in B\}.$$

Аналогично определяется пересечение и обозначения произвольной (в том числе бесконечной) системы множеств.

Определение 1.3. Разностью множеств A и B (обозначение: $A \setminus B$) называется множество тех и только тех элементов из множества A , которые не содержатся во множестве B :

$$A \setminus B = \{x | x \in A \text{ и } x \notin B\}.$$

Эта операция вводится только для двух множеств.

Пример 1.1. Пусть $A = \{a, b, c, d, e, f\}$, $B = \{c, d, l, p, r, s\}$. Тогда

$$A \cap B = \{a, b, c, d, e, f, l, p, r, 5\}; A \cap B = \{c, d\};$$

$$A \setminus B = \{a, b, e, f\}.$$

Определение 1.4. Дополнением множества B до множества A , для которых $B \subset A$, называется разность $A \setminus B$ и обозначается B^c .

Если все рассматриваемые множества являются подмножествами некоторого универсального множества U (универсума), то для множества U справедливы следующие свойства:

$$10. A \cup A^c = U.$$

$$20. \emptyset = \emptyset.$$

$$30. A \cap U = A.$$

$$40. U = \emptyset.$$

$$50. \text{Если } A \cap B = \emptyset \text{ и } A \cap B = \emptyset, \text{ то } B = A^c.$$

$$60. A = U \setminus A^c$$

Кроме того, введенные операции объединения, пересечения, разности и дополнения обладают следующими свойствами:

Коммутативность'.

$$7a) A \cup B = B \cup A;$$

$$7b) A \cap B = B \cap A.$$

8°. Ассоциативность:

$$8a) A \cup (B \cap C) = (A \cup B) \cap C;$$

$$8b) A \cap (B \cup C) = (A \cap B) \cup C.$$

9°. Дистрибутивность'.

$$9a) A \cup (B \cap C) = (A \cup B) \cap (A \cup C);$$

$$9b) A \cap (B \cup C) = (A \cap B) \cup (A \cap C).$$

10°. Идемпотентность'.

$$10a) A \cup A = A;$$

$$10b) A \cap A = A.$$

11°. Законы поглощения'.

$$11a) A \cup (A \cap B) = A;$$

$$11b) A \cap (A \cup B) = A.$$

12°. Законы де Моргана'.

$$12a) A \cap B = (A^c \cup B^c)^c$$

$$12b) A \cup B = (A^c \cap B^c)^c$$

$$13°. A \cup \emptyset = A.$$

$$14°. A \cap A = A.$$

$$15°. A \cap \emptyset = \emptyset.$$

$$16°. A = A.$$

$$17°. A \cap B = A \cap B.$$

$$18°. A \subset B \text{ тогда и только тогда, когда либо } A \cap B = A, \text{ либо}$$

$$A \cap B = B, \text{ либо } A \cap B = \emptyset.$$

$$19°. A = B \text{ тогда и только тогда, когда } (A \cap B) \cup (A \cap B) = A.$$

Доказательство всех перечисленных свойств опирается либо на определение соответствующих операций над множествами, либо на определение равенства двух множеств.

В качестве примера докажем первый закон де Моргана. Пусть $x \in A \cap B$. Тогда $x \in A$ и $x \in B$, следовательно, $x \in A^c \cup B^c$.

$x \in B$, значит $x \in A$ и $x \in B$ поэтому $x \in A \cap B$.

a) Пусть $x \in A \cap B$, следовательно, $x \in A$ и $x \in B$, значит

$x \in A$ и $x \in B$, поэтому $x \in A$ и B , следовательно, $x \in A \cap B$.

Мы показали, что если x принадлежит левой части, то он принадлежит и правой части, и наоборот, то есть $A \cap B \subset A \cap B$ и $A \cap B \subset A \cap B$.

$A \cap B \subset A \cap B$. На этом основании по определению равенства двух множеств

получаем $A \cup B = A \cup B$, что и требовалось доказать.

Определение 1.5. Симметрической разностью (или дизъюнктивной суммой) множеств A и B (обозначение: $A \Delta B$ или $A + B$) называется множество $A \Delta B = (A \setminus B) \cup (B \setminus A)$.

Симметрическая разность обладает следующими свойствами:

200. $A \Delta B = (A \cap B) \cup (A \cap B)$.

21⁰. $A \Delta B = B \Delta A$.

22⁰. $(A \Delta B) \Delta C = A \Delta (B \Delta C)$.

230. $A \Delta 0 = 0 \Delta A = A$.

Для наглядной иллюстрации операций над множествами используются, так называемые, круги Эйлера или, по-другому, диаграммы Эйлера-Венна (рис.1.1).

Круги Эйлера можно использовать и для доказательства свойств операций над множествами и для графического выполнения тождественных преобразований.

Для графических методов алгебры множеств используют также диаграммы Венна. В литературе иногда различают круги Эйлера и диаграммы Венна, однако мы этого делать не будем, лишь заметим, что диаграммы Венна иногда используют для графического решения уравнений, содержащих в качестве неизвестных множества.

Если внимательно рассмотреть свойства операций над множествами, то можно заметить, что многие из этих свойств представлены парами дуальных (взаимно двойственных) соотношений, одно из которых получается заменой в другом символов: $\cap$ на $\cup$ и, наоборот, $\cup$ на $\cap$, а также 0 на U и U на 0 . Соответствующие пары символов $\{\cap, \cup\}$ и $\{0, U\}$ называют двойственными (или дуальными) друг другу.

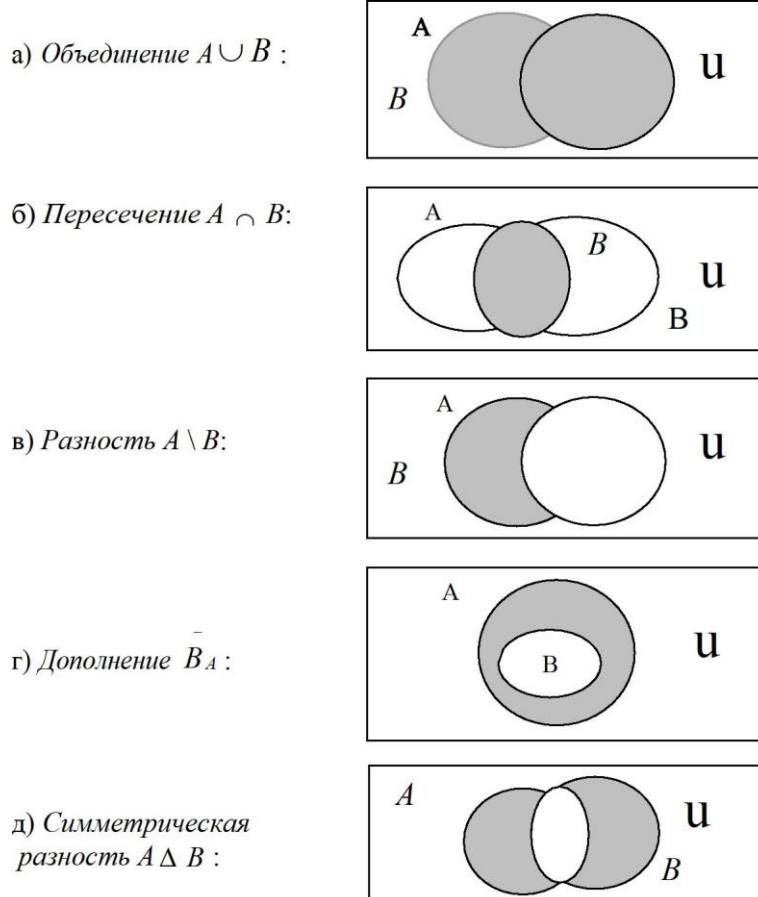


Рис. 1.1. Иллюстрация операций над множествами кругов Эйлера

При замене в любой теореме (теоретико-множественном тождестве) входящих в нее символов дуальными, мы получим новое утверждение, которое также является теоремой (тождеством). Это и есть принцип двойственности в теории множеств. Тождества 5⁰ и 6⁰ не изменяются при замене символов дуальными, поэтому их называют

самодвойственными. Этот принцип можно расширить, если считать дуальными друг другу символы $\cap$ и $\cup$.

Алгебра множеств представляет собой теоретико-множественный аналог обычной алгебры действительных чисел, исходящей из свойств операций сложения, умножения и отношения нестрого неравенства $<$, и их взаимосвязей. Эта алгебра содержит совокупность тождеств равенств, справедливых независимо от того, каково универсальное множество U и какие именно конкретные подмножества множества U обозначаются входящими в эти равенства буквами (отличными от U и 0).

1.3. Декартово произведение множеств и отношения на множествах

Многие задачи математики, техники, теоретического программирования, теории графов, теории систем, математической лингвистики и т. д. получают удобную интерпретацию на языке теории отношений или соответствий. В основе понятия соответствия между множествами A и B лежит понятие прямого (или декартова) произведения этих множеств.

Прямым (или декартовым) произведением множеств A и B (обозначение: $A \times B$) называется множество всех упорядоченных пар (a, b) таких, что $a \in A$, $b \in B$, то есть

$$A \times B = \{(a, b) \mid a \in A \text{ и } b \in B\}.$$

Например, если $A = \{a, b, Y\}$, $B = \{0, 1\}$, то

$A \times B = \{(a, 0), (a, 1), (b, 0), (b, 1), (Y, 0), (Y, 1)\}$. Множества A и B могут быть как конечными, так и бесконечными. Понятие декартова произведения двух множеств можно обобщить на случай нескольких множеств $A_1, A_2, \dots, A_n$ ($n \geq 2$). А именно:

Прямым произведением множеств $A_1, A_2, \dots, A_n$, некоторые из которых или даже все могут совпадать, называется множество A всех упорядоченных наборов (кортежей) $(a_1, a_2, \dots, a_n)$ таких, что

$$a_1 \in A_1, a_2 \in A_2, \dots, a_n \in A_n, \text{ то есть:}$$

$$A = \{(a_1, a_2, \dots, a_n) \mid a_1 \in A_1, a_2 \in A_2, \dots, a_n \in A_n\}.$$

Кортежи длиной n можно определить индуктивно в терминах упорядоченных пар, например, тройка (a_1, a_2, a_3) рассматривается

как упорядоченная пара $((a_1, a_2), a_3)$, где первая координата

(a_1, a_2) сама является упорядоченной парой, причем $(a_1, a_2) \in A_1 \times A_2$. Вообще, кортежи длины n (n -мерный вектор) определяется как упорядоченная пара через $((a_1, a_2, \dots, a_{n-1}), a_n)$, если определено $(a_1, a_2, \dots, a_{n-1})$. Соответствием (отношением) между множествами A и B называется любое подмножество r декартова произведения $A \times B$.

Как правило, отношение обозначают тройкой (A, B, r) и наряду с записью $(a, b) \in r$, пишут также $a \mathrel{r} b$ или $r(a, b)$. Поскольку соответствие r определено на двух множествах A и B , его также называют бинарным отношением (в широком смысле, не предполагая, что множества A и B совпадают).

Элемент a упорядоченной пары $(a, b) \in r$ называют первой координатой, элемент b второй координатой. Иногда различают понятия «соответствие» и «отношение». О соответствии обычно говорят, когда $A \neq B$, а об отношении в случае $A = B$. В последнем случае говорят также об отношении, заданном на множестве A : $r \subseteq A^2$.

Множество всех первых координат отношения r называется его областью определения и обозначается $D(r)$, а множество вторых координат множеством значений (областью значений) и обозначают через $E(r)$.

Соответствие (отношение) может быть определено и для n множеств (n -местное отношение) как множество кортежей $(a_1, a_2, \dots, a_n) \in A_1 \times A_2 \times \dots \times A_n$, где через $A_1 \times A_2 \times \dots \times A_n$ обозначено декартово произведение множеств $A_1, A_2, \dots, A_n$. Если $A_1 = A_2 = \dots = A_n = A$, то это прямое произведение обозначается степенью множества A^n .

Примерами трехместных (тернарных) отношений являются бинарные арифметические операции сложения, умножения, вычитания и деления над действительными числами; отношение между родителями и детьми (x сын y и z и т. д.). Пропорция $x : y = z : u$ иллюстрирует 4-местное отношение. Множество, задаваемое с

помощью общего свойства элементов $P(x)$ из некоторого универсума, можно рассматривать как одноместное (унарное) отношение в этом универсуме.

Рассмотрим теперь некоторые способы задания бинарных отношений.

Для конечных множеств широко используются матричное и графическое представление бинарных отношений.

а) Матричное представление

Пусть во множестве A имеется n элементов, а во множестве B m элементов и (A, B, p) некоторое отношение между множествами

A и B . Этому отношению сопоставляется матрица размером строки которой помечены элементами из множества A , столбцы элементами из множества B , и в которой на пересечении a -ой строки и b -го столбца стоит 1, если $(a, b) \in p$, и 0 в противном случае. Обратно, каждая $(m \times n)$ матрица, состоящая из нулей и единиц, описывает вполне определенное отношение между множествами A и B .

Пример 1.2. Пусть $A = \{a_1, a_2, a_3, a_4, a_5\}$, $B = \{b_1, b_2, b_3\}$ и $p = \{(a_1, b_1), (a_1, b_3), (a_2, b_1), (a_2, b_3), (a_2, b_2), (a_4, b_1), (a_4, b_2), (a_5, b_2), (a_5, b_3)\}$

Тогда матричное представление отношения p имеет вид таблицы 1.1.

Таблица 1.1

Табличное представление отношения

	b_1	b_2	b_3
a_1	1	0	1
a_2	1	1	1
a_3	0	0	0
a_4	1	1	0
a_5	0	1	1

б) Граф (график) отношения

Часто отношения (связи) между объектами удобно изображать с помощью совокупности точек плоскости и соединяющих их линий. Объекты изображают точками или кружочками, которые называются вершинами, а связи между ними линиями (произвольной конфигурации), называемыми ребрами. Множество вершин V , связи между которыми определены множеством ребер E , называют графом и обозначают $G = (V, E)$. Если в графе G учитывается определенная ориентация (направление) связи между вершинами графа, то соответствующее ребро снабжается стрелкой. Ориентированное таким образом ребро называют дугой, а граф с ориентированными ребрами ориентированным графом или, короче, орграфом.

При задании бинарных отношений орграфом вершины графа соответствуют элементам множеств A и B , а дуга, направленная из вершины a_i в вершину b_j означает, что $(a_i, b_j) \in p$. Граф отношения из примера 1 имеет вид рисунка 1.2.

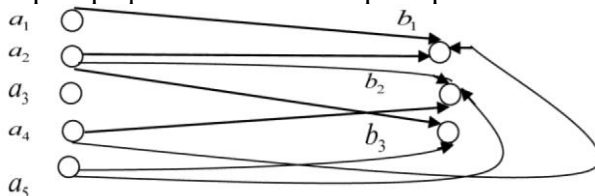


Рис. 1.2. Граф бинарного отношения из примера 1.1

Если бинарное отношение задано на числовых множествах, то его удобно задавать точками координатной плоскости. В этом случае говорят о графике отношения (соответствия).

Пусть заданы три множества A , B и C и два отношения $p \subset A \times B$ и $o \subset B \times C$.

Определение 1.6. Композицией отношений p и o (обозначение:

$o \circ p$) называется отношение T , состоящее из всех пар $(a, c) \in A \times C$, для которых

1.4. Элементы теории отображений

Изучение общих свойств различных видов отображений (соответствий) проводится в, так называемой, теории категорий. Возникновение теории категорий связано с наблюдением, что совокупности всех групп, или всех колец, или всех векторных пространств, рассматриваемые вместе с соответствующими отображениями, обладают рядом общих свойств. Язык теории категорий, в рамках которой формализуется то общее, что имеется в перечисленных выше теориях, очень емкий. Поэтому теория категорий изучает не отдельные алгебраические (и не только алгебраические) системы, а совокупность алгебраических систем.

Категория понятие, выделяющее ряд алгебраических свойств совокупностей морфизмов однотипных математических объектов (множеств, пространств, групп, колец и т.д.) друг в друга при условии, что эти совокупности содержат тождественные отображения и замкнуты относительно суперпозиции (последовательного выполнения) отображений. Дадим понятию категории более строгое определение. Говорят, что задана категория K , если:

1) Определен класс $Ob(K)$, называемый классом объектов категории K ; элементы этого класса обозначаются прописными буквами $A, B, C, \dots, U_1, \dots$ с индексами или без них.

2) Для каждой пары объектов (A, B) задано множество $Mor_K(A, B)$ такое, что для различных пар объектов $(A, B) \wedge (C, D) \wedge Mor_K(A, B) \wedge Mor_K(C, D) \Rightarrow Mor_K(A, C) \neq \emptyset$; элементы множества $Mor_K(A, B)$ называются морфизмами из объекта A в объект B и обозначаются строчными буквами $a, b, \gamma, \phi_1, \phi_2, \dots$

3) Для тройки объектов (A, B, C) и $a \in Mor_K(A, B)$, $b \in Mor_K(B, C)$ существует морфизм $\gamma \in Mor_K(A, C)$, который называется суперпозицией морфизмов a и b , и обозначается $\gamma = ba$; при этом:

а) выполняется закон ассоциативности $\gamma(ba) = (\gamma b)a$ для всех $a \in Mor_K(A, B)$, $b \in Mor_K(B, C)$, $\gamma \in Mor_K(C, D)$; для каждого объекта $A \in Ob(K)$ существует тождественный морфизм $e_A \in Mor_K(A, A)$ такой, что для всех $a \in Mor_K(A, B)$ $a e_A = a$ и $e_B a = a$.

Вместо $a \in Mor_K(A, B)$ пишут также $a : A \rightarrow B$ или $A \xrightarrow{a} B$, где символ $A \rightarrow B$ обозначает некоторый элемент из $Mor_K(A, B)$. Для простоты условимся вместо $A \in Ob(K)$ писать $A \in K$, а вместо $a \in Mor_K(A, B)$ писать $a \in K$.

Приведем определения частных видов морфизмов. Пусть K категория и $a : A \rightarrow B$ ее морфизм. Тогда

1. a — мономорфизм (или инъекция), если для любого объекта $C \in K$ и любых морфизмов $\gamma_1, \gamma_2 \in Mor(C, A)$ ($a \gamma_1 = a \gamma_2 \Rightarrow \gamma_1 = \gamma_2$). Таким образом, отображение $a : A \rightarrow B$ является мономорфизмом, если различные элементы объекта A отображаются в различные элементы объекта B .

2. a — эпиморфизм (или сюръекция), если для любого объекта $C \in K$ и любых морфизмов $\gamma_1, \gamma_2 \in Mor(B, C)$ ($\gamma_1 a = \gamma_2 a \Rightarrow \gamma_1 = \gamma_2$). Другими словами, отображение $a : A \rightarrow B$ является эпиморфизмом, если образом объекта A есть объект B (то есть каждый элемент объекта B служит образом, по крайней мере, одного элемента объекта A).

3. a — биекция (или взаимно однозначное соответствие), если a одновременно является мономорфизмом и эпиморфизмом.

4. Пусть даны два объекта $A, B \in K$, в каждом из которых определены какие-то соотношения между элементами, и a — биекция $A \rightarrow B$, сохраняющая эти соотношения. Тогда a называется изоморфизмом, а объекты A и B называются изоморфными относительно данных соотношений; пишут $A \cong B$. Таким образом, a — изоморфизм $A \rightarrow B$, если существует морфизм $b \in Mor(B, A)$ ($ba = e_A$ и $ab = e_B$).

5. Если объекты A и B совпадают, то изоморфизм называется автоморфизмом объекта A .

6. Если в пункте 3 a — эпиморфизм, то a называется гомоморфизмом из A в B ; если a — мономорфизм, то a в этом случае назы

вается гомоморфизмом из A в B .

7. Гомоморфное отображение объекта A в себя называется эндоморфизмом A .

1. Рассмотрим примеры категорий, каждая из которых определяется своей системой аксиом. P категория множеств: $Ob(P) =$ класс всех множеств; $Mor(A, B)$ множество всех отображений из множества A во множество B ; α и β суперпозиция (результат последовательного

выполнения сначала отображения α , а затем отображения β).

2. G категория групп: $Ob(G) =$ класс всех групп;

$Mor(A, B)$ множество всех групповых гомоморфизмов из A в B ; α и β суперпозиция гомоморфизмов.

3. A категория абелевых групп: $Ob(A) =$ класс всех абелевых групп; $Mor(A, B)$ множество всех групповых гомоморфизмов из A в B ; α и β суперпозиция гомоморфизмов.

4. R категория колец с единицей: $Ob(R) =$ класс всех колец с единицей; $Mor(R, S)$ множество всех кольцевых гомоморфизмов из R в S , которые единицу кольца R отображают в единицу кольца S (гомоморфизмы в этом случае называются унитарными); α и β суперпозиция гомоморфизмов.

5. M категория бинарных отношений: $Ob(M) =$ класс всевозможных непустых множеств; $Mor(A, B)$ бинарные отношения, заданные на множествах A и B , то есть все возможные подмножества декартова произведения $A \times B$; α и β суперпозиция бинарных отношений.

Замечание. Материал этого пункта при первом чтении можно опустить.

записывают это через $a R b$ в том и только в том случае, если пара (a, b) принадлежит подмножеству R , поэтому записи $a R b$ и $(a, b) \in R$ равносильны.

Изучение бинарных отношений на множестве M не отличается, следовательно, от изучения подмножеств M^2 . Можно говорить, в частности, о включении бинарного отношения R в бинарное отношение S , $R \subset S$, а также о пересечении и объединении бинарных отношений. Дополнением к бинарному отношению R является бинарное отношение R^c , определяемое подмножеством $R^c = M^2 \setminus R$; иными словами, $a R b$ тогда и только тогда, когда $(a, b) \notin R$.

Для всякого бинарного отношения R во множестве M существует обратное отношение R^{-1} , определяемое следующим образом: $a R^{-1} b \Leftrightarrow b R a$.

Пусть, например, R отношение делимости в кольце Z целых чисел, обозначаемое символом: $a R b$, то есть $a R b$ означает, что целое число a нацело делится на целое число $b \neq 0$ ($a : b$). Тогда отношением, обратным к R , будет отношение R^{-1} , обозначаемое символом $b | a$ (b делит a).

Бинарное отношение R , заданное на множестве M , называется рефлексивным, если $\forall a \in M \ a R a$; оно называется симметричным, если $a R b \Rightarrow b R a$, антисимметричным, если $a R b \ \& \ b R a \Rightarrow a = b$ и транзитивным, если $a R b \ \& \ b R c \Rightarrow a R c$.

Заметим, что если бинарное отношение R обладает любым из указанных четырех свойств, то обратное отношение R^{-1} обладает этим же свойством.

Важным типом бинарных отношений являются отношения эквивалентности, то есть бинарные отношения, обладающие свойствами рефлексивности, симметричности и транзитивности одновременно. Отношения эквивалентности, определенные на множестве M , в свою очередь, тесно связаны с разбиением множества M на попарно непересекающиеся классы. Под разбиением следует понимать такой выбор во множестве M системы непустых подмножеств (классов этого разбиения), что всякий элемент из этого множества принадлежит одному и только одному из этих подмножеств. Если обозначить через K_i классы разбиения, где i пробегает множество всех элементов множества M , то в компактной форме определение разбиения множества M на классы можно записать так:

$\forall i \in M \ K_i \neq \emptyset; M = \bigcup_i K_i; \forall i, j \in K \ i \neq j \Rightarrow K_i \cap K_j = \emptyset$.

i

Теорема 1.1. Всякое разбиение π множества M на попарно непересекающиеся

классы определяет в M некоторое отношение эквивалентности R и, наоборот, всякое отношение эквивалентности R , заданное во множестве M , определяет разбиение этого множества на попарно непересекающиеся классы.

Необходимость. Если $a, b \in M$, и если мы положим aRb в том и только в том случае, когда a и b принадлежат одному классу разбиения π , то получим во множестве M бинарное отношение, удовлетворяющее, очевидно, всем требованиям определения отношения эквивалентности.

Достаточность. Назовем классом элемента a и обозначим этот класс через K_a множество всех тех элементов $x \in M$, для которых aRx . Из свойства рефлексивности отношения R вытекает включение $a \in K_a$, то есть система классов K_a , $a \in M$, покрывает все множество M . Далее, симметричность отношения R показывает, что из $b \in K_a$, следует, что $a \in K_b$. Транзитивность отношения R приводит к тому, что если $b \in K_a$, то из $c \in K_b$ следует $c \in K_a$, то есть $K_b \subset K_a$, что в свою очередь приводит к тому, что если $b \in K_a$, то $K_b = K_a$, то есть класс определяется любым своим представителем. Если, наконец, K_a и K_b два произвольных класса с непустым пересечением, содержащим, например, элемент c , то $K_a = K_c$ и $K_b = K_c$, то есть классы K_a и K_b совпадают. Тем самым мы доказали, что система всех различных классов вида K_a является разбиением множества M .

Из доказанной теоремы следует, что между отношениями эквивалентности во множестве M и разбиением этого множества на попарно непересекающиеся классы существует биективное соответствие. Множество классов разбиения, соответствующее данному отношению эквивалентности R во множестве M обозначается через M/R и называется фактормножеством множества M по отношению эквивалентности R . Множество представителей всех классов, взятых по одному из каждого класса, называется полной системой представителей. Отображение $\varphi: M \rightarrow M/R$, сопоставляющие каждому элементу $x \in M$ тот класс K_x разбиения, соответствующего R , в котором находится элемент x , называется естественным отображением множества M на фактор-множество M/R .

Заметим, что если дано отображение a множества M на некоторое множество S , то ему соответствует вполне определенное отношение эквивалентности R во множестве M , то есть разбиение множества M на классы. Действительно, для элементов положим aRb в том и только в том случае, если $a(a) = a(b)$. Сопоставляя каждому элементу $s \in S$ класс тех элементов из множества M , которые имеют S своим образом при отображении a : $M \rightarrow S$, мы получим биективное отображение $\varphi: S \rightarrow M/R$, причем суперпозиция $\varphi \circ a$ совпадает с естественным отображением множества M на фактормножество M/R .

Пример 1.4. Пусть N натуральный ряд чисел, а Z множество всех целых чисел. Назовем рациональным числом любой класс эквивалентности во множестве $Z \times N$ относительно следующего бинарного отношения: $(x, n) R(y, t) \Leftrightarrow (x \cdot t = y \cdot n)$. Иными словами, множеством рациональных чисел называется фактор-множество вида $(Z \times N)/R$. Элементы этого фактор-множества называются рациональными числами. Легко показать, что отношение R является отношением эквивалентности во множестве $Z \times N$.

Очевидно, что отношение равенства на любом множестве рефлексивно, симметрично и транзитивно, следовательно, оно является отношением эквивалентности. Рефлексивность нашего отношения R очевидна: $(x, n) R(x, n)$, так как $xn = nx$. Очевидна также и симметричность отношения R , так как $(xm = yn) \wedge (yn = xt)$.

Докажем транзитивность отношения R . Пусть $(x, n) R(y, t)$ и $(y, t) R(z, p)$. Тогда $xt = ny$ и $yp = tz$, $n, t, p \in N$; $x, y, z \in Z$. Умножая обе части равенства $xt = ny$ на $p > 0$, обе части равенства $yp = tz$ на $n > 0$, получим $xmp = nyp$ и $nyp = nmtz$. В силу транзитивности отношения « $=$ » можно сделать вывод о том, что $xmp = nmtz$, а так как $t > 0$, то $xp = nz$, откуда следует, что $(x, n) R(z, p)$, то есть отношение R транзитивно. Вместо (x, n) по традиции пишут — $\frac{x}{n}$. Таким образом, в рас

смотренной модели множество Q рациональных чисел каждое рациональное число представляет собой класс эквивалентных дробей, например, класс $\{..., (3, 7), (6, 14), ...\} =$

$\{..., 3, -, ..., \}$ отделяет $7/14$ несократимую дробь $1/2$

Пример 1.5. Разбиение множества Z всех целых чисел на три подмножества $Z = Z_0 \cup Z_+ \cup Z_-$ соответствует эквивалентности R , согласно которому два отличных от нуля целых числа x и y находятся в отношении R , если они имеют одинаковые знаки, а 0 находится в отношении R только самим с собой.

Пример 1.6. Для натурального числа $m > 0$ разбиение множества целых чисел Z на классы чисел, сравнимых по модулю m , есть разбиение, определяемое отношением эквивалентности $=$, при котором $x \sim y \Leftrightarrow x - y \equiv 0 \pmod{m}$, другими словами, если y целых чисел x и y при делении с остатком на m эти остатки будут одинаковыми. Соответствующие классы разбиения множества Z называются классами вычетов по модулю m . Любые два класса вычетов по модулю m либо совпадают, либо не пересекаются (в силу однозначности остатка). Объединение всех классов вычетов по модулю m совпадает с множеством Z всех целых чисел. Если A класс вычетов по модулю m и a любой элемент из A , то $A = a + mZ$, то есть $A = \{a + mk \mid k \in Z\}$. Каждый класс вычетов по модулю m однозначно определяется любым принадлежащим ему числом a .

Пусть, например, $m = 6$. Тогда имеем шесть классов разбиения множества всех целых чисел:

- $18, -12, -K_0 = \{ \dots, 6, 0, 6, 12, 18, \dots \}; r=0;$
- $17, -11, -K_1 = \{ \dots, 5, 1, 7, 13, 19, \dots \}; r=1;$
- $16, -10, -K_2 = \{ \dots, 4, 2, 8, 14, 20, \dots \}; r=2;$
- $15, -9, -K_3 = \{ \dots, 3, 3, 9, 15, 21, \dots \}; r=3;$
- $14, -8, -K_4 = \{ \dots, 2, 4, 10, 16, 22, \dots \}; r=4;$
- $13, -7, -K_5 = \{ \dots, 1, 5, 11, 17, 23, \dots \}; r=5,$ где через r обозначен остаток от

деления целого числа на 6 (напомним, что по определению, разделить число $a \in Z$ на число $b \in Z$,

с остатком, значит найти пару целых чисел q и r таких, что выполняются следующие условия: $a = bq + r$, $0 \leq r < |b|$. Фактор-множество $Z/ \sim = \{K_0, K_1, K_2, K_3, K_4, K_5\}$.

Понятие бинарного отношения допускает различные обобщения. Так, рассмотрим n -ю декартову степень M^n множества M , то есть множество всех кортежей $(a_1, a_2, \dots, a_n)$ из n элементов множества M . Тогда любое подмножество $R \subset M^n$ определяет во множестве M^n бинарное отношение. Множества, в которых задано некоторое число таких отношений, называются моделями и являются предметом изучения самостоятельной теории. 1. 6. Отношения порядка

Упорядочением множества X (или порядком на X) называется бинарное отношение $<$ на X , обладающее свойствами рефлексивности, антисимметричности и транзитивности. При $x < y$ и $y < x$ пишут $x < y$. В этом случае отношение $<$ антирефлексивно. Вместо $x < y$ используется также обозначение $y > x$. Пара элементов $x, y \in X$ может и не

находиться в отношении $<$. Если, однако, если $x < y$ или $y < x$ для каждой пары элементов из множества X , то X называется линейно упорядоченным множеством (или вполне упорядоченным множеством, цепью). В общем же случае говорят о частичном порядке на множестве X . Например, множество $X = P(S)$ всех подмножеств множества S с обычным отношением включения $\subset$ между подмножествами, а также множество Z целых чисел с отношением: являются частично упорядоченными множествами. Возможно использование и других символов для обозначения порядка. Разумеется, на одном и том же множестве можно зафиксировать различные порядки. Возникающие при этом частично упорядоченные множества считаются различными. Всякое непустое подмножество частично упорядоченного множества является частично упорядоченным множеством относительно того же порядка.

Пусть X произвольное частично упорядоченное множество и $x, y \in X$. Говорят, что y покрывает x , если $x < y$ и не существует z с условием $x < z < y$. В случае $|X| < \infty$ (то есть когда X конечное множество) $x < y$ тогда и только тогда, когда найдется цепочка

элементов $x = x_1, x_2, \dots, x_{n-1}, x_n = y$, в которой x_{i+1} покрывает x_i . Понятие накрытия удобно использовать при изображении конечного частично упорядоченного множества X плоской диаграммой. На этой диаграмме элементы множества X изображаются точками и, если y покрывает x , то y помещается выше x и x соединяется с y прямолинейным отрезком. Сравнимость y и x изображается «понижающейся» ломаной линией, соединяющей y с x , причем таких ломаных может быть несколько. Несравнимые элементы не соединяются. На приведенной ниже диаграмме (рис.1.5) изображено множество всех подмножеств множества $P = \{a, b, c\}$, то есть $P(\{a, b, c\})$. Порядок на 46 множестве P задается обычным отношением включения.

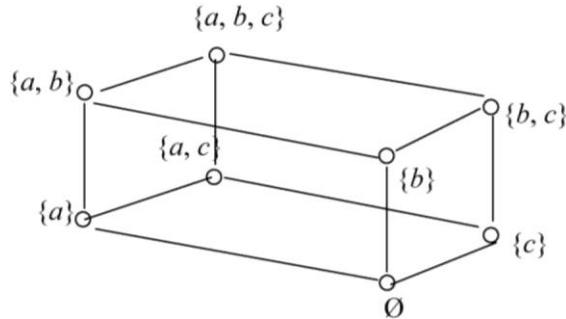


Рис. 1.5. Плоская диаграмма упорядоченного множества

Наибольшим элементом частично упорядоченного множества X называется элемент $p \in X$ такой, что $x < p$ для всех $x \in X$, а максимальным элемент $m \in X$, для которого из $m < x \in X$ следует равенство $x = m$. Наибольший элемент всегда максимален, но обратное утверждение не верно. Максимальных элементов может быть много, но наибольший элемент, если он существует, определен однозначно. Те же замечания относятся к наименьшему и минимальному элементам. На рис.1.5 наибольшим элементом является элемент $\{a, b, c\}$, а наименьшим $\emptyset$. На следующем рис.1.6 три максимальных элемента, один наименьший, но нет наибольшего элемента.

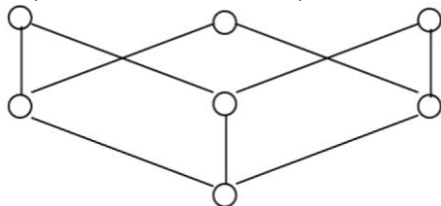


Рис.1.6. Здесь нет наибольшего элемента

Множество простых положительных целых чисел, отличных от единицы, упорядоченное по отношению делимости: содержит минимальные, но не наименьшие элементы; множество $\mathbb{Z}$ целых чисел, упорядоченное обычным отношением $<$, не имеет ни наибольшего, ни наименьшего элементов.

Если A непустое подмножество частично упорядоченного множества M , то верхним (нижним) конусом множества A называется множество всех таких элементов $x \in M$, что $a < x$ ($x < a$) для всех $a \in A$. Верхний и нижний конусы множества A обозначаются символами $A^\perp$ и $A^\vee$ соответственно. Наибольший (наименьший) элемент верхнего (нижнего) конуса множества A (если он существует) называется точной верхней (нижней) гранью множества A и обозначается символом $\sup A$ ($\inf A$). Поскольку $A^\perp$ ($A^\vee$) содержит не более одного наименьшего (наибольшего) элемента, $\sup A$ и $\inf A$ однозначно определенные элементы множества M . Например, пусть M частично упорядоченное множество неотрицательных целых чисел отношением делимости $:$. Тогда $\{18, 12\}^\perp = \{36n, n = 0, 1, \dots\}$, $\{18, 12\}^\vee = \{1, 2, 3, 6\}$, $\sup\{18, 12\} = 36$, $\inf\{18, 12\} = 6$, где $\{18, 12\}^\perp$ множество всех целых чисел из множества M , делящихся одновременно на 12 и 18, а $\{18, 12\}^\vee$ множество всех целых чисел из M , делящих 18 и 12 одновременно.

Принцип полной упорядоченности. Пусть k_0 — произвольное целое число. Тогда

всякое непустое множество целых чисел, больших или равных k_0 , имеет наименьший элемент.

Доказательство. Докажем по индукции, что всякое множество целых чисел, не меньших k_0 , не имеющее наименьшего элемента, будет пустым. Пусть S множество целых чисел, не меньших k_0 , которое не имеет наименьшего элемента, и пусть $p(k)$ утверждение «множество S не содержит целых чисел, больших k ». Если мы докажем, что утверждение $p(k)$ для всех $k < k_0$ обращается в истинное высказывание, то отсюда будет следовать, что $S = \emptyset$, так как если бы некоторый элемент $s \in S$, то утверждение $p(s)$ было бы ложным высказыванием. Очевидно, что $p(k_0)$ истинное высказывание, так как в противном случае множество S имело бы наименьший элемент, поскольку число k_0 содержалось бы во множестве S и вместе с ним все содержащиеся в S числа, меньшие k_0 . Предположим теперь, что высказывание $p(n)$ истинно для некоторого $n > k_0$. Покажем, что $p(n+1)$ также истинно и, тем самым, завершим методом полной математической индукции доказательство нашего утверждения.

Если $p(n+1)$ ложно, то некоторое число, большее $n+1$, содержится во множестве S . Но, поскольку $p(n)$ истинно, множество S не содержит чисел, больших n , откуда следует, что $n+1$ содержится в S и является его наименьшим элементом. Получили противоречие, следовательно, $p(n+1)$ истинное высказывание и $S = \emptyset$.

Имеет место и двойственный принцип: если S множество целых чисел, все элементы которого не больше k_0 , то S имеет наибольший элемент.

В общем случае имеет место принцип минимальности: всякое непустое подмножество M частично упорядоченного множества P обладает хотя бы одним минимальным во множестве M элементом.

Важность принципа минимальности состоит в том, что он эквивалентен принципу индукции, который формулируется следующим образом.

Если некоторым свойством p обладают все минимальные элементы частично упорядоченного множества M (в случае, когда они существуют) и если из справедливости свойства p для всех элементов, строго предшествующих некоторому элементу k , может быть выведена справедливость этого свойства для самого элемента k , то свойством p обладают все элементы множества M .

Многие математические исследования опираются на утверждение, которое М. Цорн в 1935 г. сформулировал и доказал в виде следующей леммы (принцип максимальности): если в частично упорядоченном множестве M всякое линейно упорядоченное подмножество A ограничено сверху, то M содержит максимальный элемент.

Как оказалось, лемма Цорна эквивалентна аксиоме выбора, гласящей: для всякого семейства F непустых множеств существует функция f такая, что для всякого множества $S \in F$ имеет место $f(S) \in S$ (при этом f называется функцией на F).

Аксиома выбора была явно сформулирована Э. Цермело в 1904 г. Впоследствии обнаружилось, что аксиома выбора эквивалентна не только лемме Цорна, но и другим содержательным утверждениям, например:

- на всяком множестве X существует отношение линейного порядка $R \subset X \times X$ такое, что любое непустое подмножество $U \subset X$ содержит наименьший в смысле отношения R элемент (принцип полной упорядоченности);
- всякая нетривиальная решетка с единицей имеет максимальный идеал;
- всякое множество X равномощно $X \times X$ и др.

Наиболее важным следствием аксиомы выбора является теорема Цермело о полном упорядочении: каждое множество может быть вполне упорядочено.

Отметим, что отношение, обратное к отношению частичной упорядоченности, само будет частично упорядоченным отношением. Минимальные элементы этой обратной упорядоченности являются максимальными элементами данного множества M в его исходной упорядоченности.

1.7. Базовые алгебраические системы

Определение 1.8. Пусть X произвольное множество. Бинарной алгебраической операцией (или законом композиции) на множестве X называется произвольное (но фиксированное) отображение $\phi : X^2 \rightarrow X$.

Таким образом, любой упорядоченной паре (x, y) элементов $x, y \in X$ ставится в соответствие однозначно определенный элемент $\phi(x, y) \in X$. Пишут также $X \phi Y$, но чаще бинарную операцию на множестве X обозначают каким-нибудь специальным символом: $*$, $\circ$, $\bullet$, $+$ и т.д. Понятие бинарной алгебраической операции, рассматриваемое в этом широком смысле, равносильно понятию заданного во множестве X тернарного отношения.

Действительно, если во множестве X задана бинарная операция, то в этом множестве можно ввести тернарное отношение R , полагая $R(x, y, z)$ тогда и только тогда, когда $x * y = z$, где символом $*$ обозначена некоторая бинарная операция. Обратно, если в X задано тернарное отношение R , то будет считать, что равенство $x * y = z$ имеет место тогда и только тогда, когда $R(x, y, z)$. В дальнейшем мы будем называть $x \bullet y$ (или $x \circ y$, или xy) произведением, а $x + y$ суммой элементов $x, y \in X$. Понятно, что в большинстве случаев эти названия условны. Если название бинарной операции безразлично, то мы будем обозначать ее символом $*$.

На множестве X может быть задано, вообще говоря, много разных операций (и не только бинарных унарных, тернарных и т.д.). Желая выделить одну из них, используют скобки $(X, *)$ и говорят, что операция $*$ определяет на множестве X алгебраическую структуру или, что $(X, *)$ алгебраическая система или алгебра. Так, например, на множестве Z целых чисел, помимо естественных операций сложения и умножения, можно указать производные операции: $n \circ m = n + m - nm$, $n * m = -n m$ и т.д. На этом пути мы приходим к различным алгебраическим системам $(Z, +)$, $(Z, \bullet)$, $(Z, \circ)$, $(Z, *)$ с одной алгебраической операцией.

Ниже мы иногда будем использовать символ $\forall$, который называется квантором всеобщности и читается «для всякого», «для любого». Строгое определение этому квантору будет дано в пункте 2.5.

Определение 1.9. Пусть на множестве X определена бинарная алгебраическая операция $*$. Элемент e называется левым нейтральным элементом относительно операции $*$, если $\forall x \in X e * x = x$, и элемент e' называется правым нейтральным элементом, если $\forall x \in X x * e' = x$.

Операция типа $+$ (сложения) называется аддитивной, а операция типа $\bullet$ (умножения) - мультипликативной. Для аддитивной операции нейтральный элемент обычно называется нулем (0), а для мультипликативной единицей (1 или e).

Определение 1.10. Операция $*$ на множестве X называется коммутативной, если $\forall x \in X \forall y \in X x * y = y * x$.

Операция $*$ называется ассоциативной, если $\forall x \in X \forall y \in X \forall z \in X x * (y * z) = (x * y) * z$.

Ясно, что если бинарная операция $*$ на множестве X коммутативна, то левый и правый нейтральные элементы относительно этой операции совпадают.

Методом полной математической индукции легко доказать, что если бинарная операция на множестве X ассоциативна, то результат ее последовательного применения к n элементам множества X не зависит от расстановки скобок.

Определение 1.11. Алгебра $(X, *)$ с одной бинарной алгебраической ассоциативной операцией называется полугруппой. Полугруппу с нейтральным элементом e относительно операции $*$ называют моноидом (или полугруппой с единицей).

Пример 1.7. Пусть $nZ = \{n^k \mid k \in Z\}$ множество целых чисел, делящихся на $n > 1$. Тогда $(nZ, +, 0)$ коммутативный моноид, а $(nZ, \bullet)$ коммутативная полугруппа без единицы.

В любом аддитивном коммутативном моноиде можно использовать знак суммирования

Полагая $x_1 \cdot x_2 \cdot \dots \cdot x_n \cdot x$, в моноиде $(M, +, 0)$ получим кратные px , а в моноиде $(M, \cdot, e)$ произведение, обозначаемое симметричным p_n и называемое n -ой степенью элемента x . При этом во всем $x^n \cdot x \cdot x \cdot \dots \cdot x = x^{n+1}$, $(x^m)^n = x^{m \cdot n}$, где $m, n \in \mathbb{N}$, и, по определению, $\forall x \in M, x \cdot 0 = 0$.

Определение 1.12. Пусть $(M, *, e)$ моноид. Элемент $b \in M$ называется симметричным элементу $a \in M$, если $a * b = e = b * a$, а элемент a в этом случае называется симметризуемым.

Если элемент $a \in M$ моноида $(M, *, e)$ симметризуем, то существует единственный элемент, симметричный элементу a . Действительно, пусть U, V элементы, симметричные элементу a относительно операции $*$, то есть $a * u = e = u * a$, $a * v = e = v * a$. Тогда в силу ассоциативности операции $*$ $u = u * e = u * (a * v) = (u * a) * v = e * v = v$, то есть $u = v$.

Элемент, симметричный элементу a при аддитивной форме записи обозначают через $(-a)$ и называют противоположным элементу a . При мультипликативной записи элемент, симметричный a , обозначают через a^{-1} и называют обратным элементу a , а сам элемент a называют обратимым.

Определение 1.13. Пусть на множестве M определены две бинарные алгебраические операции $*$ и $\circ$. Операция $*$ называется дистрибутивной относительно операции $\circ$, если $\forall x \forall y \forall z (x, y, z \in M) (x * (y \circ z) = (x * y) \circ (x * z))$ и $((y \circ z) * x = (y * x) \circ (z * x))$.

Пример 1.8.

1. Сложение и умножение рациональных чисел являются коммутативными и ассоциативными бинарными операциями.

2. Операция вычитания на множестве рациональных чисел не коммутативна и не ассоциативна.

3. Операции объединения и пересечения подмножеств данного множества коммутативны и ассоциативны.

4. Композиция функций ассоциативная операция, но не коммутативная.

5. На множестве всех подмножеств некоторого множества операции объединения и пересечения взаимно дистрибутивны друг относительно друга, как и операции конъюнкции и дизъюнкции на множестве высказываний.

6. Пусть $\mathbb{N}$ множество всех натуральных чисел и (x, y) наибольший общий делитель чисел x и y . Операция, сопоставляет $x, y \in \mathbb{N}$ наибольший общий делитель этих чисел, коммутативна и ассоциативна.

7. Умножение целых чисел дистрибутивно относительно сложения. Однако сложение целых чисел не дистрибутивно относительно умножения, так как в общем случае равенство $a + bc = (a + b)(a + c)$ не выполняется.

8. Пусть $*$ операция на множестве $\mathbb{N}$ всех целых положительных чисел возведения в степень, то есть $a * b = a^b$ для любых $a, b \in \mathbb{N}$. Операция $*$ не коммутативна и не ассоциативна.

9. Операция симметрической разности $X \Delta Y = (X \setminus Y) \cup (Y \setminus X)$ множеств X и Y коммутативная и ассоциативная операция на множестве $\mathcal{P}(M)$, а операция пересечения множеств дистрибутивна относительно операции Δ .

10. Операция $A * B = AB + BA$ на множестве квадратных матриц порядка n коммутативна, но не ассоциативна; нейтральным элементом относительно операции $*$ является матрица $\% E$, где E единичная матрица порядка n ; обратным элементом для матрицы A относительно операции $*$ является матрица $\% A^{-1}$, если $\det A \neq 0$, и обратной матрицы не существует, если $\det A = 0$.

Рассмотрим наиболее важные с точки зрения различных приложений алгебраические системы. Их условно можно разбить на три вида: с одной внутренней бинарной операцией (моноиды, полугруппы, группы), с двумя внутренними бинарными операциями (кольца, области целостности, тела, поля) и с внешними операциями над

полями (векторные пространства и линейные алгебры). Например, в векторном пространстве над бесконечным полем имеется одна бинарная операция (сложение векторов), бесконечное множество унарных внешних операций, а именно умножений векторов на элементы основного поля, и одна нуль-арная операция, фиксирующая нулевой вектор.

Определение 1.14. Моноид $G = (G, *, e)$ называется группой относительно операции $*$, если для каждого элемента $g \in G$ существует симметричный ему элемент. Группа называется коммутативной (или абелевой), если групповая операция, кроме того, коммутативна.

Пример 1.9.

1. Алгебры $(\mathbb{Z}, +, 0)$, $(\mathbb{Q}, +, 0)$, $(\mathbb{R}, +, 0)$, $(\mathbb{C}, +, 0)$ абелевы аддитивные группы соответственно целых, рациональных, действительных и комплексных чисел. $(\mathbb{Q} \setminus \{0\}, \cdot, 1)$, $(\mathbb{Q}^+, \cdot, 1)$, $(\mathbb{R} \setminus \{0\}, \cdot, 1)$, $(\mathbb{R}^+, \cdot, 1)$, $(\mathbb{C} \setminus \{0+0 \cdot i\}, \cdot, 1+0 \cdot i)$ это абелевы мультипликативные группы соответственно всех рациональных чисел без нуля, множества положительных рациональных чисел, всех действительных чисел без нуля, множества положительных действительных чисел, всех комплексных чисел без нуля.

3. Пусть $G = (G, \circ, e)$ множество преобразований плоскости, то есть множество биективных отображений плоскости на себя, с операцией композиции отображений и тождественным преобразованием e . Тогда алгебра G группа относительно композиции $\circ$.

Примерами ее подгрупп (то есть замкнутых подмножеств преобразований) являются группа движений, группа гомотетий с общим центром, группа подобия. В свою очередь в группе движений можно выделить подгруппу параллельных переносов, подгруппу поворотов с общим центром. С другой стороны, например, множество осевых симметрий группы не образует, так как композиция двух осевых симметрий осевой симметрией не является.

4. Множество M матриц одного и того же размера образует аддитивную коммутативную группу. Множество всех невырожденных квадратных матриц порядка n с элементами из некоторого числового поля образует мультипликативную не коммутативную группу.

Другим важным примером алгебраических систем с двумя бинарными алгебраическими операциями является кольцо.

Определение 1.15. Кольцом называется непустое множество K , в котором заданы две бинарные алгебраические операции сложение и умножение, причем по сложению это должна быть абелева группа аддитивная группа кольца K , а умножение должно быть связано со сложением левым и правым законами дистрибутивности: для любых элементов $a, b, c \in K$ $a(b + c) = ab + ac$, $(b + c)a = ba + ca$.

На умножение в общем случае не накладывается никаких ограничений, то есть кольцо K по умножению является лишь группоидом (множеством с мультипликативной бинарной алгебраической операцией). Если умножение в кольце ассоциативно, то кольцо называется ассоциативным. В этом случае можно говорить о мультипликативной полугруппе кольца K . Если умножение в кольце K ассоциативно и коммутативно, то это кольцо называется ассоциативно коммутативным. Если $(K, \cdot, e)$ моноид, то кольцо $(K, +, 0, \cdot, e)$ называется кольцом с единицей или унитарным кольцом.

Определение 1.16. Коммутативное кольцо K называется областью целостности, если в нем $0 \neq e$ и $\forall a \in K \forall b \in K (a \neq 0 \wedge b \neq 0 \Rightarrow ab \neq 0)$. Элементы a и b кольца K называются делителями нуля, если $(a \neq 0 \wedge b \neq 0) \wedge (ab = 0)$.

Отметим, что нетривиальное коммутативное кольцо K является областью целостности тогда и только тогда, когда $(a \neq 0 \wedge b \neq 0) \Rightarrow (ab \neq 0)$.

Определение 1.17. Подмножество L кольца K называется подкольцом, если $\forall x \forall y (x, y \in L \Rightarrow (x + y \in L \wedge xy \in L))$, то есть если L подгруппа аддитивной группы кольца K и подгруппа мультипликативной полугруппы кольца K .

Пример 1.10

1. Множество всех целых чисел относительно обычных операций сложения и умножения образуют ассоциативно-коммутативное кольцо с единицей.

2. Множество векторов евклидова трехмерного пространства с операциями сложения и векторного умножения векторов образует неассоциативное и некоммутативное кольцо. В этом кольце для любых векторов a, b, c выполняются следующие равенства: $[a, a] = 0$, где 0 нулевой вектор; $[[a, b], c] + [[b, c], a] + [[c, a], b] = 0$ тождество Якоби; $[a, b] = -[b, a]$ антикоммутативность.

3. Пусть $M_n(R)$ множество всех квадратных матриц (a_{ij}) порядка n с вещественными элементами a_{ij} . Тогда $M_n(R)$ кольцо с единицей E , где E единичная матрица порядка n . Это кольцо называется полным матричным кольцом над полем R действительных чисел. При $n > 1$ кольцо $M_n(R)$ не коммутативно. Оно содержит в качестве подколец кольца $M_n(Q)$ и $M_n(Z)$ квадратных матриц того же порядка над полем Q рациональных чисел и над кольцом Z целых чисел соответственно.

4. Алгебраическая система $(mZ, +, \cdot)$, где mZ множество всех целых чисел, делящихся на $m \neq 0$, будет в кольце всех целых чисел подкольцом без единицы, если $m > 1$. Множество всех многочленов от одной переменной, степень которых не превышает целого положительного числа n , с коэффициентами из некоторого кольца образует ассоциативно-коммутативное кольцо с единицей.

Определение 1.18. Полем P называется ассоциативно-коммутативное кольцо с единицей (не совпадающей с аддитивным нейтральным элементом), в котором каждый ненулевой элемент обратим (то есть для каждого элемента $a \neq 0$ существует обратный элемент, обозначаемый a^{-1}).

Из определения поля вытекает, что относительно умножения отличные от нуля элементы поля образуют коммутативную группу, которая называется мультипликативной группой поля P . Таким образом, поле представляет собой гибрид двух абелевых групп аддитивной и мультипликативной, связанных законом дистрибутивности (ввиду коммутативности умножения левый и правый законы дистрибутивности совпадают). Произведение ab^{-1} записывается обычно в виде дроби (или отношения, частного) a/b . Следовательно, дробь a/b , имеющая смысл только при $b \neq 0$, является единственным решением уравнения $b x = a$, при этом действия над дробями подчиняется обычным правилам, известным из школьного курса математики, и вытекающим из определения поля. Таким образом, поля — это класс колец с делением.

Пример 1.12. В примере 3 мы назвали рациональным числом любой элемент фактормножества $(Z \times N) \setminus R$, где R бинарное отношение, определяемое по правилу $(x, n) R (y, m) \Leftrightarrow (xm = yn)$, где $x, y \in Z, n, m \in N$. Положим по определению $(x, n) + (y, m) = (xm + yn, nm)$, $(x, n)(y, m) = (xy, nm)$, $(x, n) < (y, m) \Leftrightarrow (xm < yn)$ с обычными операциями сложения и умножения целых и натуральных чисел и обычным отношением порядка $<$. Определение введенных операций над упорядоченными парами корректны, так как эти операции не зависят от выбора представителей в классах эквивалентности. Тогда легко проверить, что алгебраическая система $Q = (Q, +, \cdot, ^{-1}, 0, 1, <)$, где символами $\cdot$ и $^{-1}$ обозначены соответственно операции умножения и нахождения обратного элемента, является полем. Мы привели пример построения одной из моделей поля рациональных чисел.

Подполем поля F называется подкольцо в F , само являющееся полем. Например, поле Q рациональных чисел является подполем поля R действительных чисел, которое, в свою очередь, является подполем поля C комплексных чисел. В случае $F \subset P$ говорят также, что поле P является расширением своего подполя F . Из определения подполя следует, что нуль и единица поля P будут содержаться также в F и служить для F нулем и единицей.

При проектировании кодирующих и декодирующих устройств, цифровых сигнальных процессоров и другой аппаратуры для цифровой обработки информации, а также при разработке соответствующего программного обеспечения находит широкое

применение теории конечных алгебраических систем (групп, колец, полей). Например, множества линейных кодов и цифровых сигналов образуют конечные абелевы группы относительно операции покомпонентного сложения векторов; множество классов вычетов по модулю m образует конечную абелеву группу относительно операции сложения классов, которая называется аддитивной группой классов вычетов по модулю m .

Выводы

На лекции введена теоретико-множественная терминология и символика, рассмотрены операции над множествами и их свойства, определены отношения и отображения на множествах, описаны базовые алгебраические системы, такие как: группы, кольца, поля.

Вопросы и упражнения для самопроверки

- 1.1. Объяснить, почему $2 \in \{1, 2, 3\}$, $0 \notin \{1, 2, 3\}$.
- 1.2. Верно ли, что $\{1, 2\} \in \{\{1, 2, 3\}, \{1, 3\}, 1, 2\}$?
- 1.3. Перечислить все элементы множества $P(A)$ и найти его мощность, если:
 - а) $A = \{0, \{0\}\}$; б) $A = \{\{0, \{0\}\}\}$; в) $A = \{7, 8, 9, \{7, 8\}, \{7, 8, 9\}\}$;
 - г) $A = \{0, \{0\}, 7, a, \{7, a\}, \{7, a, \{7, a\}\}\}$; д) $A = \{0, \{0\}, \{0, \{0\}\}\}$.
- 1.4. Описать словесно каждое из следующих множеств:
 - а) $\{x \mid x \in Z \text{ и } x^2 \text{ и } x^3\}$ (здесь Z означает множество целых чисел, а символ : делимость нацело);
 - б) $\{x \mid x \in A \text{ и } x \in B\}$;
 - в) $\{x^2 \mid x \text{ простое число}\}$;
 - г) $\{(x, y) \mid x \in R \text{ и } y \in R \text{ и } x^2 + y^2 = 1\}$ (R множество действительных чисел).
- 1.5. Сформулировать определение равенства двух множеств (интуитивный принцип объемности). Как выразить равенство двух множеств с помощью операции включения?
- 1.6. Показать, что $0 \cap \{0\}$.
- 1.7. В каком отношении находятся множества A и B , если $A \setminus B = B \setminus A = \emptyset$?
- 1.8. Дать определение операций объединения, пересечения, разности и дополнения множеств словесно и записать эти определения символически. Проиллюстрировать теоретико-множественные операции на кругах Эйлера.
- 1.9. Определить симметрическую разность двух множеств.
- 1.10. Перечислить наиболее важные свойства операций над множествами.
- 1.11. Сформулировать закон двойственности для множеств. Выписать попарно двойственные свойства операций над множествами.
- 1.12. Доказать справедливость тождеств и проиллюстрировать их с помощью диаграмм Эйлера-Венна:
 - 1.12.1. $A \cap B \cup B = A \cup B$;
 - 1.12.2. $A \cup (B \setminus A) = A \cup B$;
 - 1.12.3. $A \setminus (A \cap B) = A \setminus B$;
 - 1.12.4. $A \cap (B \setminus C) = (A \cap B) \setminus C$;
 - 1.12.5. $A \setminus (B \cup C) = (A \setminus B) \cap (A \setminus C)$;
 - 1.12.6. $A \setminus (B \cap C) = (A \setminus B) \cup (A \setminus C)$;
 - 1.12.7. $(A \cup B) \setminus C = (A \setminus C) \cup (B \setminus C)$;
 - 1.12.8. $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$;
 - 1.12.9. $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$;
 - 1.12.10. $A \cup B = A \cap B$.
- 1.13. Доказать, что
 - 1.13.1. $A \subset B \text{ и } C \subset A \Rightarrow B \subset C$;
 - 1.13.2. $A \subset B \text{ и } C \subset A \Rightarrow B \subset C$;
 - 1.13.3. $(A \setminus B) \cap B = A \cap B \setminus A$;
 - 1.13.4. $(A \cap B) \cap C = A \cap (B \cap C) \cap A$.
- 1.14. Упростить выражения:

$$1.14.1. (A \circ B \circ C) \circ (A \circ C) \circ (B \circ C);$$

$$1.14.2. (M \setminus N) \circ (N \setminus M).$$

1.15. Выразить: а) операции объединения и разности двух множеств через операции симметрической разности и пересечения; б) операции пересечения и разности двух множеств через операции симметрической разности и объединения; в) операции пересечения и объединения двух множеств через операции разности и симметрической разности.

Ответ: а) $A \circ B = A \Delta B \Delta A(A \circ B)$; $A \setminus B = A \Delta A(A \circ B)$.

б) $A \circ B = (A \circ B) \Delta A \Delta B$; $A \setminus B = (A \circ B) \Delta B$.

в) $A \circ B = A \setminus (A \setminus B)$; $A \circ B = (A \Delta B) \Delta A \Delta (A \setminus (A \setminus B))$.

1.16. Найти декартово произведение множеств A и B , если:

$$1.16.1. A = \{a, b, Y\}, B = \{l, m\};$$

$$1.16.2. A = B = \{a, b, 0, 1\};$$

$$1.16.3. A = [-1, 2], B = [-2, 1];$$

$$1.16.4. A = \{x \in \mathbb{R} \mid x > 0\}, B = \{y \in \mathbb{R} \mid y < 0\};$$

$$1.16.5. A = \{(x, y) \mid (x, y) \in \mathbb{R}^2 \text{ и } |x| + 2|y| = 1\};$$

$$1.16.6. A = \{(x, y) \mid (x, y) \in \mathbb{R}^2 \text{ и } y > 0 \text{ и } y < x \text{ и } x + y < 1\}.$$

1.17. Доказать, что

$$1.17.1. (A \times B) \circ (C \times D) \subset (A \circ C) \times (B \circ D);$$

$$1.17.2. (A \circ B) \times C = (A \times C) \circ (B \times C);$$

$$1.17.3. (A \circ B) \times (C \circ D) = (A \times C) \circ (B \times C) \circ (A \times D) \circ (B \times D);$$

$$1.17.4. (A \setminus B) \times C = (A \times C) \setminus (B \times C).$$

1.18. Найти область определения, область значений данного бинарного отношения; отношение, обратное данному отношению;

композицию отношения с самим собой и с обратным отношением (слева и справа):

$$1.18.1. p = \{(x, y) \mid x, y \in \mathbb{N} \text{ и } x|y\};$$

$$1.18.2. p = \{(x, y) \mid x, y \in \mathbb{N} \text{ и } y|x\};$$

$$1.18.3. p = \{(x, y) \mid x, y \in \mathbb{R} \text{ и } x + y < 0\};$$

$$1.18.4. p = \{(x, y) \mid x, y \in \mathbb{R} \text{ и } 2x > 3y\}.$$

Ответ к отношению 1.18.4:

$$E(p) = D(p) = \mathbb{R}; p^{-1} = \{(x, y) \mid x, y \in \mathbb{R} \text{ и } 2y > 3x\};$$

$$p^0 p = \{(x, y) \mid x, y \in \mathbb{R} \text{ и } 4x > 9y\}; p \circ p^{-1} = p \cap p = \mathbb{R}^2.$$

1.19. Пусть p_1 и p_2 — бинарные отношения. Доказать, что:

$$1.19.1. (p_1 \circ p_2)^{-1} = p_1^{-1} \circ p_2^{-1};$$

$$1.19.2. p \circ p = p \cap p = p;$$

$$1.19.3. (p^{-1})^{-1} = p;$$

$$1.19.4. (p \cap p)^{-1} = p^{-1} \cap p^{-1}.$$

1.20. Для перестановок $f = 1\ 2\ 3\ 4$ и $f^{-1} = 2\ 3\ 4\ 1$ найти перестановки $f \circ g$, $g \circ f$, f^{-1} и g^{-1} .

$$f = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 1 & 4 \end{pmatrix}, g = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix}$$

$$I = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix}$$

$$\text{новки } f \circ g, g \circ f, f^{-1} \text{ и } g^{-1}.$$

1.21. Найти все отображения множества $A = \{0, 1\}$ в себя и указать, какие из них инъективны.

1.22. Найти все отображения множества $A = \{0, 1, 2\}$ на множество $B = \{0, 1\}$.

1.23. Обозначим через $\text{Dom } f$ область определения функции (бинарного отношения, отображения) f . Доказать, что для каждой функции f и любого множества A $f(A) = \emptyset$ тогда и только тогда, когда $A \cap \text{Dom } f = \emptyset$.

1.24. Доказать, что если f функция и A, B множества, то $f(A \cap B) \subset f(A) \cap f(B)$. Показать на примерах, что случай равенства $f(A \cap B) = f(A) \cap f(B)$ может не иметь места.

1.25. Пусть f отображение множества A в B и g отображение множества B в C . Доказать, что:

- а) если отображение $g \circ f$ инъективно, то и f инъективно;
- 1.26. если $g \circ f$ сюръективное отображение A на C , то g сюръективное отображение B на C . Пусть A и B конечные множества, состоящие из m и n элементов соответственно.
- а) Сколько существует отображений множества A в B ?
- б) Сколько существует бинарных отношений между элементами множеств A и B ?
- 1.27. Какие из следующих бинарных отношений обладают свойствами рефлексивности, антирефлексивности, симметричности, антисимметричности и транзитивности:
- 1) $\{(x, y) | x, y \in \mathbb{Z} \wedge x \wedge y + 1\}$;
 - 2) $\{(x, y) | x, y \in \mathbb{Z} \wedge x^2 = y^2\}$;
 - 3) $\{(x, y) | x, y \in \mathbb{Z} \wedge x = |y|\}$;
 - 4) $\{(x, y) | x, y \in \mathbb{N} \wedge x | y\}$;
 - 5) $\{(x, y) | x, y \in \mathbb{Z} \wedge x^2 + y^2 = 1\}$;
 - 6) $\{(x, y) | x, y \in \mathbb{N} \wedge x \wedge y\}$, где $\mathbb{Z}$ множество всех целых чисел, $\mathbb{N}$ множество всех натуральных чисел, а символом $\wedge$ обозначен союз «и» (конъюнкция).
- 1.28. Привести примеры бинарных отношений:
- а) рефлексивных и транзитивных, но не антисимметричных;
 - б) транзитивных и симметричных, но не рефлексивных;
 - в) рефлексивных и транзитивных, но не симметричных;
 - г) рефлексивных и симметричных, но не транзитивных.
- 1.29. Рассматриваются следующие отношения:
- а) на множестве всех конечных подмножеств некоторого множества: подмножества A и B содержат одинаковое количество элементов;
 - б) на множестве всех подмножеств некоторого множества: $A \cap B \neq \emptyset$;
 - в) на множестве всех положительных целых чисел: все пары взаимно простых чисел;
 - г) на множестве всех положительных целых чисел: $x = y$ или $x = 2y$.
- Проверить, что отношение а) рефлексивно, симметрично и транзитивно; отношение б) рефлексивно, симметрично, но не транзитивно; отношение в) симметрично, антирефлексивно, но не транзитивно; отношение г) рефлексивно, антисимметрично, но не транзитивно.
- 1.30. Пусть $A = \{1, 2, 3, 4, 5, 6, 7\}$ и $R = \{(x, y) | x, y \in A \wedge (x - y) \in 2\mathbb{Z}\}$. Доказать, что R есть отношение порядка на множестве A и найти максимальные и минимальные элементы.
- 1.31. Частично упорядоченные множества A и A' называются изоморфными, если существует биективное отображение f множества A на множество A' такое, что для любых элементов $a, b \in A$ $a < b$ тогда и только тогда, когда $f(a) < f(b)$. Само отображение f в этом случае называется изотонным. Если f изоморфизм, а a наибольший (соответственно, наименьший, максимальный, минимальный) элемент множества A , то $f(a)$ также является наибольшим (соответственно, наименьшим, максимальным, минимальным) элементом множества A' . Построить примеры, показывающие, что это утверждение неверно для произвольного изотонного отображения.
- 1.32. Доказать, что в конечном частично упорядоченном множестве всегда существуют как максимальные, так и минимальные элементы.
- 1.33. Во множестве целых положительных чисел определим бинарное отношение $\sim$ следующим образом: $(a, b) \sim (c, d)$ тогда и только тогда, когда $a + d = b + c$.
- Доказать, что отношение $\sim$ является отношением эквивалентности на данном множестве. Описать классы эквивалентности по этому отношению.
- 1.34. Пусть R^2 вещественная плоскость с прямоугольной системой координат. Рассмотрим на множестве точек этой плоскости бинарное отношение: $A \sim B$ в том и только в том случае, когда точки A и B принадлежат одной горизонтальной прямой. Доказать, что отношение $\sim$ является отношением эквивалентности. Что является классами

эквивалентности при соответствующем разбиении? Доказать, что фактор-множество $R^2/\sim$ и любая прямая, пересекающая ось ОХ, находятся в биективном соответствии.

1.35. Являются ли приведенные ниже отображения бинарными операциями на заданных множествах? Для операций найти нейтральные и симметричные элементы, проверить, обладают ли операции свойствами коммутативности и ассоциативности.

1.1. $(x, y) \wedge ax + by$ на множестве рациональных чисел, где a, b фиксированные, а x и y произвольные рациональные числа;

1.2. $(x, y) \wedge \max(x, y)$ на множестве натуральных чисел.

1.36. Доказать, что симметрическая разность $X \wedge Y = (X \setminus Y) \vee (Y \setminus X)$

на множестве всех подмножеств данного непустого множества является коммутативной и ассоциативной операцией. Показать также, что операция пересечения множеств дистрибутивна относительно операции симметрической разности.

1.37. Доказать, что множество всех квадратных матриц данного порядка n с вещественными элементами со следом (то есть суммой всех элементов главной диагонали), равным нулю, образует полугруппу относительно обычной операции умножения матриц. Является ли эта полугруппа моноидом?

1.38. Доказать, что квадратные матрицы порядка n , в каждой строке и в каждом столбце которых имеется не более чем один элемент, равный 1, а остальные нули, образуют полугруппу. Доказать, что в этой полугруппе для каждого элемента a существует единственный элемент a^+ такой, что $a a^+ = a$, $a^+ a = a^+$ (полугруппы с таким свойством называются инверсными).

1.39. Выяснить, являются ли следующие множества рациональных чисел замкнутыми относительно главных операций мультипликативной группы рациональных чисел: 1) множество $\{1, -1\}$; 2) множество всех отличных от нуля чисел с четными (нечетными) знаменателями; 3) множество целочисленных степеней простого числа.

2. АЛГЕБРА ВЫСКАЗЫВАНИЙ И ЛОГИКА ПРЕДИКАТОВ

План занятия

1. Логические операции над высказываниями и их свойства.
2. Формулы и функции алгебры логики. Закон двойственности.
3. Совершенные дизъюнктивные и конъюнктивные нормальные формы булевых функций.
4. Полнота и замкнутость системы булевых функций. Представление о классах Поста.
5. Логика предикатов.

2.1. Логические операции над высказываниями и их свойства

Под высказыванием мы будем понимать грамматически правильно построенное повествовательное предложение, про которое можно сказать, что оно либо истинно, либо ложно. В логике принят закон исключенного третьего: высказывание может быть истинным, либо ложным третьего не дано. Кроме того, высказывание не может быть одновременно истинным и ложным (закон противоречия). Простое повествовательное предложение называется простым или элементарным высказыванием. Высказывания мы будем обозначать строчными или прописными латинскими буквами с индексами или без них, например, $a, b, c, \dots, x, y, z, a_1, a_2, a_3, A, B, C, \dots, X, Y, Z$ и т.д., отвлекаясь от их содержания. Из простых высказываний с помощью частицы «не», союзов «и», «или», слов «если, то», «тогда и только тогда» можно составлять составные или сложные высказывания, к которым снова можно применять эти связки. Эти связки называются сентенциональными связками. Сложные высказывания также могут быть истинными или ложными в зависимости от того, истинны или ложны исходные высказывания, и какие при этом использовались связки. В алгебре высказываний сентенциональным связкам соответствует пять логических операций. Приведем соответствующие определения этих операций.

Определение 2.1. Отрицанием высказывания x называется новое высказывание, обозначаемое символом $\neg x$ (или $\neg x$), (читается: не x ; не верно, что x), которое будет истинным в том и только в том случае, когда высказывание x ложно.

Определение 2.2. Конъюнкцией (или логическим умножением) двух высказываний x и y называется новое высказывание, обозначаемое $x \& y$ (или $x \wedge y$, или $x \blacksquare y$, или xy), (читается x и y), которое будет истинным в одном и только одном случае, когда оба высказывания x и y истинны.

Логический союз «и» не обязательно должен представляться через грамматический союз «и». С союзом «и» часто совпадают по смыслу союзы «а» и «но». Однако языковая ситуация может стать такой, что союз «и» перестает играть роль конъюнкции. Пусть, например, x = «Ему стало страшно», y = «он убил человека». Составим два сложных высказывания: «Ему стало страшно, и он убил человека», «Он убил человека, и ему стало страшно». Здесь одно простое предложение обуславливает другое, поэтому мы имеем здесь скрытое условное предложение, а не конъюнкцию.

Определение 2.3. Дизъюнкцией (или логическим сложением) двух высказываний x и y называется новое высказывание, обозначаемое $x \vee y$ (или $x + y$), (читается: x или y), которое будет ложным в том и только в том случае, когда оба высказывания x и y ложны.

Союз «или» в операции дизъюнкции используется в неисключающем смысле, т.е. этот союз может включать в себя и союз «и». Иначе говоря, выражение $x \vee y$ следует понимать как: либо x , либо

y (по отдельности), либо x и y (одновременно), что не всегда согласуется с употреблением союза «или» в естественном языке, в котором большую роль играет смысловое значение высказываний.

Определение 2.4. Импликацией «если x , то y » высказываний x и y называется новое высказывание, обозначаемое через $x \supset y$ (или

$x \supset y$), которое будет ложным лишь в одном случае: когда первое высказывание (в нашем случае x) истинно, а второе (в нашем случае y) ложно.

В импликации x выступает в роли причины, а y в роли следствия.

Определение 2.5. Эквиваленцией двух высказываний x и y называется новое высказывание, обозначаемое $x \equiv y$ (или $x \sim y$) (читается: x тогда и только тогда, когда y), которое будет истинным в том и только в том случае, когда высказывания x и y одновременно истинны, либо одновременно ложны.

Условимся для дальнейшего значение «истина» обозначать цифрой 1, а значение «ложь» цифрой 0 (в литературе используются также обозначения T (true) и F (false) и др.) Тогда определения 1 5 74 логических операций над высказываниями можно свести в одну таблицу (таблица 2.1), которая называется таблицей истинности:

Таблица 2.1

Сводная таблица истинности логических операций

x	y	x	$x \& y$	$x \vee y$	$x \supset y$	$x \equiv y$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Подобные таблицы истинности часто используют для доказательства логических тождеств и равносильностей, законов логики, в различных приложениях булевой алгебры.

В алгебре логики помимо указанных операций используются и другие (производные) операции, которые получаются различными способами из основных операций. Среди них особую роль играют следующие четыре операции.

Сложение по модулю 2, обозначаемое через $x \oplus y$ и определяемое таблицей истинности (таблица 2.2):

Таблица 2.2
Сложение по модулю 2

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

Как видно из таблицы 2.2, сложение по модулю 2 есть ни что иное, как отрицание эквиваленции, то есть $x \oplus y = x \circ y$.

Отрицание конъюнкции (и-не), называемое штрихом Шеффера и обозначаемое через $x \mid y$. Таким образом, по определению, $x \mid y = x \& y$, откуда легко получить соответствующую таблицу истинности.

Отрицание дизъюнкции (или-не), называемое стрелкой Пирса или штрихом Лукасевича и обозначаемое через $x \Phi y$. Итак, по определению, $x \Phi y = x \vee y$.

Исключающее «или» или альтернатива (XOR) то же, что и сложение по модулю 2.

Определение 2.6. Всякое сложное высказывание, которое может быть получено из элементарных высказываний посредством применения логических операций и круглых скобок, регулирующих порядок их выполнения, называется формулой алгебры логики. При этом сами элементарные высказывания, а также константы 0 и 1 также являются простейшими формулами.

Все возможные логические значения формулы в зависимости от значений входящих в нее элементарных высказываний могут быть описаны полностью с помощью таблицы истинности.

Пример 2.1. Для формулы $F = (x \vee y) \wedge (x \& y)$ таблица истинности имеет вид:

Таблица 2.3

Таблица истинности для формулы примера 1

x	y	x	y	$x \vee y$	$x \& y$	F
0	0	1	1	1	0	0
0	1	1	0	1	0	0
1	0	0	1	0	1	1
1	1	0	0	1	0	0

Если формула содержит n элементарных высказываний, то она принимает 2^n значений, состоящих из нулей и единиц, то есть таблица истинности содержит 2^n строк. Это становится очевидным, если применить формулу числа размещений с повторениями.

Формула называется тождественно истинной или тавтологией, если она принимает значение 1 на всех наборах входящих в нее элементарных высказываний. Если формула на всех наборах входящих в нее элементарных высказываний принимает значение 0, то она называется тождественно ложной. Формула называется выполнимой, если она не является тавтологией и принимает значение 1 хотя бы на одном наборе входящих в нее элементарных высказываний.

Задача установления принадлежности данной формулы к одному из трех перечисленных выше видов называется проблемой разрешимости. Существует несколько критериев установления тождественной истинности данной формулы и, следовательно, определения, к какому виду она относится.

Две формулы F_1 и F_2 называются равносильными (обозначение: $F_1 = F_2$), если они принимают одинаковые значения на одних и тех же наборах входящих в них элементарных высказываний.

Равносильные формулы выражают обычно законы логики. Если вместо знака « $=$ » поставить знак эквиваленции между формулами, т.е. $F \equiv F$, то новая формула $F \equiv F$ окажется тавтологией.

Доказательство законов логики осуществляется, как правило, с помощью таблиц истинности, или с использованием определений логических операций над высказываниями.

В качестве примеров приведем ряд наиболее важных законов алгебры логики.

$$1. \quad x \& x = x$$

$\wedge$ законы идемпотентности

$$2. \quad x \vee x = x$$

$$3. \quad x \& 1 = x$$

$$4. \quad x \vee 1 = 1$$

- законы для 1 и 0

$$5. \quad x \& 0 = 0$$

$$x \vee 0 = x$$

$$6. \quad x \& (y \vee x) = x$$

$\wedge$ законы поглощения

$$7. \quad x \vee (y \& x) = x$$

$x \vee x = 1$ закон исключения третьего

$x \& x = 0$ закон противоречия

$x = \neg \neg x$ закон двойного отрицания

$x \& y = \neg (x \vee \neg y)$ первый закон де Моргана

$x \vee y = \neg (x \& \neg y)$ второй закон де Моргана

$x \wedge y = \neg (x \vee \neg y)$ выражение эквиваленции через импликацию и конъюнкцию

$x \wedge y = x \vee y$ выражение импликации через дизъюнкцию и отрицание

$x \wedge y = y \wedge x$ закон контрапозиции

$$x \& y = y \& x$$

коммутативные законы

$$1. \quad x \vee \neg x = 1$$

$$19. \quad x \& (y \& z) = (x \& y) \& z$$

!> ассоциативные законы

$$20. \quad x \vee (y \vee z) = (x \vee y) \vee z$$

$$21. \quad x \& (y \vee z) = (x \& y) \vee (x \& z)$$

$\vee$ дистрибутивные законы.

$$22. \quad x \vee (y \& z) = (x \vee y) \& (x \vee z)$$

Пусть задано произвольное непустое множество $M = \{a, b, c, \dots, x, y, z, \dots\}$ и на нем определены две бинарные алгебраические операции, условно называемые умножением и сложением, и обладающие свойствами, аналогичными свойствам 1, 2, 11, 7, 8, 12, 13, 17-22. Множество M в этом случае называется алгеброй Буля. Эта алгебра абстрактная. Если рассматривать конкретные множества и конкретные операции на них, то можно получать различные интерпретации алгебры Буля. Таким образом, алгебра логики является одной из интерпретаций алгебры Буля. Другими примерами интерпретаций алгебры Буля могут служить алгебра множеств, алгебра логических схем в цифровой схемотехнике и т. д.

Таким образом, мы ознакомились с одной из интерпретаций алгебры Буля алгеброй логики, являющейся одним из важных с прикладной точки зрения разделов математической логики.

2.2. Формулы и функции алгебры логики. Закон двойственности

Всякую формулу алгебры логики можно рассматривать как функцию входящих в нее элементарных высказываний. С содержательной точки зрения булева функция от n переменных это функция вида $y = f(x_1, x_2, \dots, x_n)$, которая, как и каждая ее переменная $x_1, x_2, \dots, x_n$, принимает значения из множества $\{0, 1\}$.

Строго понятие булевой функции определяется так:

Определение 2.7. Булева функция от n переменных это произвольное отображение

вида $f: \{0,1\}^n \rightarrow \{0,1\}$, где $\{0,1\}^n$ n -я декартова степень множества $\{0,1\}$.

Заметим, что $n > 0$. При $n = 0$ мы получаем две булевы константы: 0 и 1 (рассматриваемые как булевы функции от 0 переменных).

Будем обозначать через P_2 множество всех булевых функций (для всех возможных значений n числа переменных), а через $P_2(n)$ множество всех булевых функций от n переменных (для фиксированного n). Из определения понятно, что $P_2 = \bigcup_{n \geq 0} P_2(n)$.

Таким образом, областью определения любой булевой функции от n переменных является множество $\{0,1\}^n$ всех n -компонентных кортежей на множестве $\{0,1\}$. Это множество состоит из наборов нулей и единиц по n штук в каждом. Эти наборы еще называются n -мерными булевыми векторами или n -мерным единичным кубом. Число всех таких наборов (то есть вершин булева куба) составляет 2^n .

На множестве $P_2(n)$ можно определить структуру булевой алгебры следующим образом.

Для произвольных булевых функций $f, g \in P_2(n)$ определим новые функции так, что для любого вектора $x = (x_1, x_2, \dots, x_n) \in \{0,1\}^n$

$$(f \vee g)(x) = f(x) \vee g(x),$$

$$(f \& g)(x) = f(x) \& g(x),$$

$$f(x) = f(x).$$

Нулем булевой алгебры всех булевых функций из множества $P_2(n)$ будет функция, принимающая на любом наборе $x \in \{0,1\}^n$ значение 0, а единицей этой алгебры функция, принимающая на любом наборе $x \in \{0,1\}^n$ значение 1.

Мощность множества всех булевых функций для заданного n равна 2^{2^n} .

Действительно, поскольку каждая булева функция отображает множество из 2^n элементов в 2-элементное множество, а мощность множества всех отображений из n -элементного множества в m -элементное равна m^n , то мы и получим указанное число 2^{2^n} как мощность множества $P_2(n)$.

Две булевы функции $f, g \in P_2(n)$ называются равными, если для любых значений переменных $x_1, x_2, \dots, x_n$ $f(x_1, x_2, \dots, x_n) = g(x_1, x_2, \dots, x_n)$, при этом будем писать $f(x_1, x_2, \dots, x_n) = g(x_1, x_2, \dots, x_n)$, или просто $f = g$,

подчеркивая употреблением символа « $=$ » тот факт, что для всех наборов значений переменных выполняется равенство значений указанных функций. В то же время могут быть построены такие две, не равные (в только что определенном смысле) функции f и g , что для некоторого набора $a \in \{0,1\}^n$ имеет место равенство $f(a) = g(a)$.

Рассмотрим примеры булевых функций. При $n=0$ имеем две булевы функции булевы, или логические, константы 0 и 1. При $n=1$ получаем 4 булевы функции (таблица 2.4).

Таблица 2.4
Булевы функции от одной переменной

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
0	0	0	1	1
1	1	0	1	0

Очевидно, что $f_1(x) = x$, $f_4(x) = x$, $f_2(x) = 0$, $f_3(x) = 1$.

В любом случае, булева функция от n переменных может быть задана таблицей истинности, состоящей из $n+1$ столбцов и 2^n строк. В первых n столбцах перечисляются все наборы из множества $\{0,1\}^n$ в лексикографическом (словарном) порядке, а в последнем $(n+1)$ -ом столбце значения функций на этих наборах. При $n=2$ мы имеем $2^{2^2} = 16$ функций, при $n=3$ имеем $2^{2^3} = 256$ функций.

При больших значениях n количество функций становится практически необозримым и составление таблиц истинности вручную для больших значений n становится неудобным.

В таблице 2.5 приведены наиболее важные функции двух переменных (из 16

функций в таблице представлено 7 функций):

Таблица 2.5

Булевы функции от двух переменных

x_1	x_2	$f_1(x_1, x_2)$	$f_2(x_1, x_2)$	$f_3(x_1, x_2)$	$f_4(x_1, x_2)$	$f_5(x_1, x_2)$	$f_6(x_1, x_2)$	$f_7(x_1, x_2)$
0	0	0	0	0	1	1	1	1
0	1	1	0	1	1	0	1	0
1	0	1	0	1	0	0	1	0
1	1	1	1	0	1	1	0	0

Очевидно, что $f_1(x_1, x_2) \equiv x_1 \vee x_2$; $f_2(x_1, x_2) \equiv x_1 \& x_2$; $f_3(x_1, x_2) \equiv x_1 \oplus x_2$; $f_4(x_1, x_2) \equiv x_1 \wedge x_2$; $f_5(x_1, x_2) \equiv x_1 \vee \neg x_2$; $f_6(x_1, x_2) \equiv x_1 \wedge \neg x_2$; $f_7(x_1, x_2) \equiv \neg x_1 \wedge \neg x_2$.

Приведем еще в качестве примера таблицу истинности булевой функции трех переменных (одну из возможных 256 функций таблица 6). Эта функция называется мажоритарной функцией или функцией голосования. Заметим, что в первом столбце приведенной таблицы проставлены номера наборов из множества $\{0,1\}^3$ при их лексикографическом упорядочении.

Заметим, что, записывая таблицу истинности любой булевой функции от n переменных, нет необходимости каждый раз перечислять все наборы, достаточно записать вектор значений функции, понимая, что i -ая компонента данного вектора есть значение функции на i -ом наборе (двоичном коде числа i). Тогда, например, мажоритарная функция f может быть задана так: $f = (0,0,0,1,0,1,1,1)$. Можно также перечислить номера тех наборов, на которых функция принимает значение 1: $f = \{3,5,6,7\}$, понимая при этом, что на остальных наборах функция принимает значения, равные 0.

Таблица 2.6

Таблица истинности мажоритарной функции

Номер набора	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Определение 2.8. Переменную x_i , булевой функции $f(x_1, x_2, \dots, x_n)$ называют фиктивной переменной, если для любых значений переменных $x_1, x_2, \dots, x_{i-1}, x_{i+1}, \dots, x_n$ $f(x_1, x_2, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n) = f(x_1, x_2, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n)$.

Таким образом, значение фиктивной переменной не влияет на значение функции, и ее тем самым, можно рассматривать как функцию из множества $P_2(n-1)$.

С другой стороны, если дана функция $f(x_1, x_2, \dots, x_{n-1}) \in P_2(n-1)$, то можно ввести новую фиктивную переменную x_n таким образом, что $f^1(x_1, x_2, \dots, x_{n-1}, x_n) = f(x_1, x_2, \dots, x_{n-1}) \& (x_n \vee \neg x_n)$ функция f существенно зависит от переменной x . Рассмотрим два примера, представленных таблицами 2.6 и 2.7.

Таблица 2.7

Таблица истинности мажоритарной функции с фиктивной переменной

Номера наборов	x_1	x_2	x_3	x_4	$f(x_1, x_2, x_3, x_4)$
0	0	0	0	0	0
1	0	0	0	1	0

2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	1

Мажоритарная функция (таблица 2.6) не имеет фиктивных переменных, так как $f(0,0,1)=0$, а $f(1,0,1)=1$, то есть переменная x_1 существенная. Аналогично, $f(1,0,0)=0$, но $f(1,1,0)=0$, что означает существенность переменной x_2 ; для x_3 имеем $f(1,0,0)=0$, но $f(1,0,1)=1$, что означает существенность и этой переменной. В то же время легко проверить, что функция, заданная таблицей 2.7, является мажоритарной, зависящей существенно от x_1, x_3, x_4 , а переменная x_2 в ней является фиктивной.

Понятие фиктивной переменной позволяет обобщить понятие равенства булевых функций независимо от числа переменных этих функций.

Определение 2.9. Две булевы функции f и g называются равными, если они существенно зависят от одного и того же числа переменных, и для любого набора a значений существенных переменных $f(a) = g(a)$.

Кроме таблиц истинности всякую булеву функцию можно представить в виде формулы с помощью некоторого набора логических операций. Например, с помощью набора операций $\{ \neg, \wedge, \vee \}$ произвольную функцию $f(x_1, x_2, \dots, x_n)$ можно представить в виде следующей формулы:

$$F = (f(1,1,\dots,1) \wedge X_1 \wedge X_2 \wedge \dots \wedge X_n) \vee (f(1,1,\dots,1,0) \wedge x_1 \wedge \dots \wedge X_{n-1} \wedge X_n) \vee (f(1,1,\dots,1,0,1) \wedge x_1 \wedge \dots \wedge x_{n-2} \wedge x_{n-1} \wedge x_n) \vee \dots \vee (f(0,0,\dots,0) \wedge x_1 \wedge x_2 \wedge \dots \wedge x_n).$$

Мы не приводим доказательства этого разложения.

Легко проверить, что значения функции $f(X_1, X_2, \dots, X_n)$ и приведенной формулы совпадают на всех наборах значений переменных $X_1, X_2, \dots, X_n$. Соответствие между формулами над фиксированным множеством логических операций и представляемыми ими функциями не является взаимно однозначным, поскольку формулу можно с помощью законов логики преобразовать в равносильную ей формулу. Однако равным функциям соответствуют равносильные (эквивалентные) формулы.

Определение 2.10. Функция $f(x_1, x_2, \dots, x_n)$, равная $f(x_1 \sim x_2, \dots, x_n)$,

называется двойственной функцией к функции $f(x_1, x_2, \dots, x_n)$.

Поскольку всякую булеву функцию можно представить некоторой формулой, то определение понятия двойственности можно перенести и на формулы.

Так, в списке законов алгебры логики, рассмотренных в п.1, двойственными являются законы 1 и 2, 3 и 6, 4 и 5, 7 и 8, 9 и 10, 12 и 13, 17 и 18, 19 и 20, 21 и 22.

Таблица истинности для двойственной функции (формулы) при выбранном порядке наборов значений переменных получается из таблицы истинности для исходной функции инвертированием (то есть заменой 0 на 1 и 1 на 0) столбца функции и его транспонированием, например, для функции от трех переменных эта таблица может иметь такой вид:

Таблица 2.8

Таблица истинности для двойственной функции

Номер	X1	X2	X3	$f(X_1, X_2, X_3)$	$f(X_1, X_2, X_3)$
0	0	0	0	1	0
1	0	0	1	1	1

2	0	1	0	0	0
3	0	1	1	0	1
4	1	0	0	0	1
5	1	0	1	1	1
6	1	1	0	0	0
7	1	1	1	1	0

2.3. Совершенные дизъюнктивные и конъюнктивные нормальные формы булевых функций

В результате эквивалентных преобразований с использованием законов логики любую логическую функцию можно представить лишь с помощью операций отрицания, конъюнкции и дизъюнкции, что приводит к дизъюнктивным или конъюнктивным нормальным формам данной функции (ДНФ и КНФ). Поскольку булева функция представляется в виде ДНФ или КНФ неоднозначно, вводятся понятия совершенной дизъюнктивной или конъюнктивной нормальных форм (СДНФ или СКНФ). Они получаются из ДНФ или КНФ наложением на них некоторых условий. СДНФ и СКНФ обеспечивают единственность представления любой булевой функции, кроме функций, тождественно равных 0 или 1.

Определение 2.11. Элементарной конъюнкцией n переменных называется конъюнкция переменных или их отрицаний:

$$x_1^{g_1} \& x_2^{g_2} \& \dots \& x_n^{g_n},$$

где g_k — 0 или 1, если $g_k = 1$,

$$x_k^{g_k} = \begin{cases} x_k & \text{если } g_k = 1 \\ \neg x_k & \text{если } g_k = 0 \end{cases}$$

или ее отрицания

Определение 2.12. Дизъюнктивной нормальной формой формулы $F(x_1, x_2, \dots, x_n)$ называется равносильная ей формула, представляющая собой дизъюнкцию элементарных конъюнкций.

На основе законов логики 14 и 15 импликацию и эквиваленцию можно выразить через отрицание, конъюнкцию и дизъюнкцию. Следовательно, методом тождественных преобразований с использованием законов логики для любой формулы алгебры логики можно получить ее ДНФ, причем не единственную. Среди всех ДНФ данной формулы существует единственная ДНФ, которая обладает следующими свойствами:

- Каждое логическое слагаемое формулы содержит все переменные или их отрицания.
- Ни одно логическое слагаемое формулы не содержит одну и ту же переменную дважды.
- Ни одно логическое слагаемое формулы не содержит одновременно переменную и ее отрицание.
- Все логические слагаемые формулы различны.

Тогда каждой не тождественно ложной формуле соответствует единственная формула, удовлетворяющая перечисленным условиям. Доказательство этого факта основано на разложении функции по всем переменным. Такое разложение функции

$$f(x_1, x_2, \dots, x_n) = \bigvee_{i=1}^n x_1^{g_1} \& x_2^{g_2} \& \dots \& x_n^{g_n} \quad (2.1)$$

называется ее совершенной дизъюнктивной нормальной формой. В формуле (2.1) дизъюнкция осуществляется по всем наборам $(x_1, x_2, \dots, x_n)$ значений переменных, для которых заданная функция принимает значение, равное 1.

Теорема 2.1. Каждая функция алгебры логики может быть выражена в виде формулы через отрицание, конъюнкцию и дизъюнкцию.

Доказательство. 1) Пусть $f(x_1, x_2, \dots, x_n) = 0$. Тогда, очевидно, $f(x_1, x_2, \dots, x_n) = \neg x_1$

$\& x_1$.

2) Пусть $f(x_1, x_2, \dots, x_n) \neq 0$. Тогда представим ее в СДНФ (2.1).

Теорема доказана.

Данная теорема носит конструктивный характер, так как она позволяет для каждой булевой функции фактически построить формулу, ее реализующую (в виде СДНФ), а именно: в таблице истинности для функции $f(x_1, x_2, \dots, x_n)$ ($f \neq 0$) отмечаем все строки

$(5_1, 5_2, \dots, 5_n)$, для которых $f(5_1, 5_2, \dots, 5_n) = 1$; для каждой такой строки образуем логическое произведение

$$x_1^{5_1} \& \dots \& x_n^{5_n}, \quad (2.2)$$

1 n

и затем все полученные элементарные конъюнкции соединим знаком дизъюнкции. Каждое логическое произведение вида (2.2) называется конституентой единицы или минтермом.

Пример 2.2. Пусть функция $f(x_1, x_2, x_3)$ задана таблицей истинности 2.9.

Таблица 2.9

Таблица истинности к примеру 2

Номер набора	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

2.4. Полнота и замкнутость системы булевых функций. Представление о классах Поста

На множестве булевых функций можно определить операцию суперпозиции, которая сводится к переименованию переменных или к подстановке некоторой функции вместо какой-либо переменной в другую функцию.

Такие суперпозиции образуют некоторые классы функций. Среди этих классов большое значение имеют замкнутые классы, то есть такие классы, которые вместе с функциями данного класса содержат и все их суперпозиции. Другим важным свойством классов булевых функций является свойство полноты. Это свойство заключается в том, что любая булева функция может быть выражена с помощью операции суперпозиции через функции полного класса.

Американский математик Эмиль Пост (1897-1954) дал полное описание всех функционально замкнутых классов. Он установил, что функционально замкнутых классов булевых функций счетное множество. Теперь эти классы называются классами Поста.

Среди всех замкнутых классов Пост особо выделил 5 классов: класс функций, сохраняющих 0; класс функций, сохраняющих 1; класс линейных функций; класс самодвойственных функций и класс монотонных функций. Как оказалось, каждый из этих классов неполон. Кроме того, Пост доказал критерий полноты для данного множества булевых функций, а именно: для того, чтобы данное множество булевых функций было полным, необходимо и достаточно, чтобы оно не являлось подмножеством ни одного из перечисленных выше пяти замкнутых классов.

Пусть $K^0 = \{f(x_1, x_2, \dots, x_k), f(x_1, x_2, \dots, x_k), \dots, f(x_1, x_2, \dots, x_k)\}$

1 1 2 k1 2 1 2 k2 m 1 2 km

конечная система булевых функций из множества P_2 . Функция $f \in P_2$ называется

суперпозицией функций $f_1, f_2, \dots, f_m$ ранга 1, если

она может быть получена одним из следующих двух способов:

- переименованием некоторой переменной какой-нибудь функции f_i , то есть:

$$f = f_i(x_1, x_2, \dots, x_{j-1}, y, x_{j+1}, \dots, x_k),$$

где y может совпадать с любой переменной;

подстановкой некоторой функции f_i вместо какой-

либо переменной x_j любой из функций $f \in K_0$, то есть

f_i

Суперпозиции ранга 1 образуют класс функций K^1 . Из функций класса K^1 можно получить класс функций K^2 с помощью суперпозиций ранга 2 и так далее. Класс функций, получающийся из функций класса K^{r-1} суперпозицией ранга $r-1$ с помощью суперпозиций, называется классом функций K^r суперпозиций ранга r . Суперпозициями функций из класса K^0 называются функции, входящие в какой-либо из классов K^r .

Определение 2.13. Пусть $M \subseteq P_2$. Замыканием множества M называется множество $[M]$, которое состоит из функций множества M и функций, которые могут быть получены из функций множества M путем конечного числа суперпозиций.

Определение 2.14. Множество M называется замкнутым (или функционально замкнутым), если оно совпадает со своим замыканием, то есть если $[M] = M$.

Проще говоря, множество M булевых функций называется функционально замкнутым, если вместе с функциями этого множества (класса) оно содержит и все их суперпозиции.

Очевидно, что для доказательства замкнутости того или иного класса булевых функций достаточно показать, что элементарные суперпозиции не выводят из этого класса.

Определение 2.15. Класс M булевых функций называется полным, если $[M] = P_2$, то есть, если любая булева функция из P_2 может быть выражена через функции класса M с помощью суперпозиций.

Пример 2.4. Класс $M = \{0, 1\}$ замкнут и неполон; P_2 замкнут и полон, $\{ \vee, \& \}$ незамкнут и полон.

Определение 2.16. Пусть $f(x_1, x_2, \dots, x_n) \in P_2(n)$. Говорят, что

функция $f(x_1, \dots, x_n)$ сохраняет 0, если $f(0, 0, \dots, 0) = 0$, и что функция $f(x_1, x_2, \dots, x_n)$ сохраняет 1, если $f(1, 1, \dots, 1) = 1$.

Эти классы обозначают соответственно через $P_0(n)$ и $P_1(n)$. Через P_0 обозначают множество всех булевых функций, сохраняющих 0, то есть $P_0 = \bigcup_{n=0}^{\infty} P_0(n)$

, а через множество P_1 — о всех функций из множеств P_1

P_2 , сохраняющих 1, то есть $P_1 = \{x \vee y \mid x, y \in P_0; x \& y \in P_0\}$;

Пример 2.5. $0 \in P_0$; $1 \in P_0$; $0 \in P_1$; $0 \in P_1$;

$x_1 \wedge x_2 \in P_0$; $x \vee y \in P_1$; $x_1 \odot x_2 \in P_1$.

Определение 2.17. Булева функция $f(x_1, x_2, \dots, x_n)$ называется линейной, если ее можно представить в виде

$f(x_1, x_2, \dots, x_n) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \dots \oplus a_n x_n$, где выражение в правой части называется линейным полиномом Жегалкина, коэффициенты $a_0, a_1, \dots, a_n \in \{0, 1\}$, а произведения $a_i x_i$ означают конъюнкцию a_i и x_i .

Обозначим через $L(n)$ класс линейных функций от n переменных, а

а через L класс всех линейных функций, то есть $L = \bigcup_{n=0}^{\infty} L(n)$.

$n=0$

Пример 2.6. $1 \in L$; $0 \in L$; $x = x \oplus 1 \in L$; $x_1 \& x_2 \in L$;

$x_1 \vee x_2 = x_1 \& x_2 \oplus x_1 \oplus x_2 \in L$; $x_1 \odot x_2 = x_1 \vee x_2 \oplus (x_1 \oplus 1) \oplus (x_2 \oplus 1) \oplus 1 =$

$= x_1 \& x_2 \oplus x_1 \oplus x_2 \in L$.

Определение 2.18. Функция $f(x_1, x_2, \dots, x_n)$ называется самодвойственной, если

$$f(x, x, \dots, x) = f^*(x, x, \dots, x) = f^*(x^1, x^1, \dots, x^0).$$

Обозначим через $S(n)$ класс самодвойственных функций от n переменных

менных, а через $S = jS(n)$ класс всех самодвойственных функций.

$n = 0$

Пример 2.7. $0 \in S; i \in S; x \in S, "x \in S, x \vee x \in S; x \& x \in S. i^2 \ i^2$

Определение 2.19. Отношением частичного порядка на множестве наборов $\langle a_1, a_2, \dots, a_n \rangle$ переменных $x_1, x_2, \dots, x_n$ называется отношение, обозначаемое символом $\square$, и

удовлетворяющее отношению: $\langle a_1, a_2, \dots, a_n \rangle \square \langle b_1, b_2, \dots, b_n \rangle \Leftrightarrow a_i \leq b_i$

для всех $i = 1, n$,

где

де

Определение 2.20. Функция $f \in P_2(n)$ называется монотонной, если для любых наборов $\langle a_1, a_2, \dots, a_n \rangle$ и $\langle b_1, b_2, \dots, b_n \rangle$ значений переменных $x_1, x_2, \dots, x_n$ таких, что $\langle a_1, a_2, \dots, a_n \rangle \square \langle b_1, b_2, \dots, b_n \rangle$ выполняется

отношение

$f(a_1, a_2, \dots, a_n) \leq f(b_1, b_2, \dots, b_n)$. Обозначим через $M(n)$ множество всех монотонных

функций от n

переменных, а через $M = \bigwedge M(n)$ — множество всех монотонных $n = 0$

функций.

Пример 2.8. $0 \in M; x_1 \& x_2 \in M; x \in M; x_1 \vee x_2 \in M$.

Оказывается, что каждый из определенных выше пяти классов P_0, P_1, L, S и M является функционально замкнутым и неполным

Доказательство замкнутости каждого класса сводится к проверке того факта, что суперпозиция любых функций из рассматриваемого класса не выводит за пределы этого класса. Доказательство неполноты каждого из пяти классов Поста сводится к установлению факта существования булевой функции, не принадлежащей к рассматриваемому классу (см. определения 14 и 15 замкнутого и полного классов).

Для проверки полноты заданной системы булевых функций достаточно ограничиться классами P_0, P_1, L, S и M , хотя замкнутых классов Поста, как отмечалось выше, существует счетное множество. Имеет место, так называемый, критерий полноты.

Теорема 2.2 (теорема Поста). Для того чтобы некоторое множество M булевых функций было функционально полным, необходимо и достаточно, чтобы оно не являлось подмножеством ни одного из классов P_0, P_1, L, S и M .

Определение 2.21. Система функций $\{f_1, f_2, \dots, f_t, \dots\}$ из замкнутого класса M называется его базисом, если она полна в M , но всякая ее собственная подсистема не

Пример 2.9. Базисами в множестве P_2 могут служить системы функций

является полной в M .

$\{x, x \& x_2\}, \{x, x \vee x_2\}, \{x_1 | x\}, \{x_1 \Phi x\}, \{x_1, x_2, 0, 1, x \odot x_2 \odot x_3\}$.

Пост также доказал, что каждый замкнутый класс из множества P_2 имеет конечный базис.

Хотя каждый из рассмотренных выше классов Поста не является полным, каждый из них является предполным, то есть таким, что добавление к нему любой функции, не принадлежащей этому классу, приводит к образованию полного класса. При этом кроме пяти рассмотренных нами классов Поста, других предполных классов во множестве P_2 нет.

На основании теорем 2.1, 2.2, определений 2.14, 2.15 и примера

2.4 можно сделать вывод о том, что каждый из классов дизъюнктивных и конъюнктивных нормальных форм является незамкнутым и функционально полным.

2. 5. Логика предикатов

Мы видели, что можно дать два различных толкования логики высказываний: содержательное описание и в виде аксиоматической системы.

Переходя к рассмотрению другой логической системы, которую мы будем

называть логикой предикатов, опять сначала изложим ее содержательно, а затем в виде формальной аксиоматической системы исчисления предикатов.

Для того чтобы изложить содержательно логику предикатов, нам придется привлечь понятие актуальной бесконечности и принять без обоснования способы проведения рассуждений, употребляемые в теории множеств. Теорию множеств мы принимаем за основу нашего изложения. Содержательная трактовка логики предикатов обладает тем достоинством, что она облегчает изучение, как исчисления предикатов, так и других абстрактных логических систем.

В отличие от алгебры высказываний, логика предикатов имеет явно неконструктивный характер. Все понятия логики предикатов определяются для произвольной области или произвольного множества объектов. Ввиду этого при содержательном изложении логики предикатов приходится опираться на неконструктивные принципы теории множеств.

Язык логики предикатов является расширением языка логики высказываний. Мы должны овладеть достаточно богатым запасом тождеств логики предикатов и уметь их обосновывать при помощи содержательных теоретико-множественных соображений.

Язык логики предикатов удобен для записи математических предложений. Он дает возможность выражать логические связи между понятиями, записывать определения, теоремы, доказательства. Кроме того, логика предикатов позволяет путем равносильных преобразований формулы придать ей хорошо обозримый вид.

Понятие предиката

Логика предикатов представляет собой развитие алгебры высказываний. Она содержит в себе всю алгебру высказываний, то есть элементарные высказывания, рассматриваемые как величины, которые принимают одно из двух значений истина или ложь, все операции алгебры высказываний и, следовательно, все её формулы. Но, помимо этого, логика предикатов вводит в рассмотрение высказывания, отнесенные к предметам. В ней уже имеется расчленение высказывания на субъект и предикат, в то время как в алгебре высказываний высказывания рассматриваются как нераздельные целые объекты и только с точки зрения их истинности или ложности. В алгебре высказываний ни структура высказываний, ни тем более, их содержание, не затрагиваются.

Субъект (буквально подлежащее, хотя оно и может играть роль дополнения) это то, о чем что-то утверждается в высказывании.

Предикат (буквально сказуемое, хотя оно может играть и роль определения) это то, что утверждается о субъекте.

Например, в высказывании «47 простое число», «47» субъект, «простое число» предикат. Это высказывание утверждает, что субъект «47» обладает свойством «быть простым числом». Если в рассмотренном примере заменить число 47 переменной x из множества натуральных чисел, то получим, так называемую, высказывательную форму (или неопределенное высказывание) « x простое число». При одних значениях x (например, $x = 19$, $x = 23$) это форма обращается в истинное высказывание, а при других (например, $x = 27$, $x = 33$) в ложное. Ясно, что эта высказывательная форма определяет функцию от одной переменной x , определенной на множестве N натуральных чисел, и принимающую значения из множества $\{0,1\}$. Здесь предикат становится функцией субъекта x и выражает свойство этого субъекта.

Определение 2.22. Одноместным предикатом $P(x)$ называется произвольная функция от одной переменной x , определенная на некотором множестве M и принимающая значения из множества $\{0,1\}$. Множество M называется областью определения предиката $P(x)$. Множество всех элементов $x \in M$, при которых предикат

$P(x)$ обращается в истинное высказывание, называется областью истинности предиката $P(x)$.

Например, предикат $P(x)$ « $\sin x = 0$ » определен на множестве всех действительных чисел, а его областью истинности является множество $I_P(x) = \{k\pi, k \in Z\}$.

Определение 2.23. Предикат $P(x)$, определенный на множестве M , называется тождественно истинным (тождественно ложным), если его область истинности $I_P(x) = M$ (соответственно $I_P(x) = \emptyset$).

Например, предикат $P(x) = \langle x^2 + 1 = 0 \rangle$ тождественно ложный на множестве всех действительных чисел, а предикат $Q(x) = \langle x^2 + 1 > 0 \rangle$ тождественно истинный на этом же множестве. Естественным обобщением понятия одноместного предиката является понятие n -местного предиката, с помощью которого выражаются отношения между объектами данного множества. Так, высказывательные формы о двух и более объектах

$Q(x, y)$, $H(x, y, z)$ и т.д. представляют собой функции от 2, 3 и т.д. переменных.

Определение 2.24. n -местным предикатом $P(X_1, X_2, \dots, X_n)$ называется функция от n переменных $X_1, X_2, \dots, X_n$, определенная на множестве $M = M_1 \times M_2 \times \dots \times M_n$ со значениями из множества $\{0, 1\}$. Например, отношения $Q(x, y) = \langle x = y \rangle$, $H(x, y, z) = \langle x + y < z \rangle$, заданные на множествах R^2 и R^3 , являются соответственно

2-местным и 3-местным предикатами. По аналогии с определением области определения и области истинности одноместного предиката вводятся такие же понятия для n -местных предикатов. Точно так же можно определить и понятия тождественно истинного и тождественно ложного n -местного предиката.

Логические операции над предикатами

Поскольку каждый из предикатов может принимать только одно из двух значений истина или ложь (1 или 0), то к предикатам применимы все операции логики высказываний.

Рассмотрим применение операций логики высказываний к предикатам на примерах одноместных предикатов, хотя эти операции можно распространить и на n -местные предикаты.

Пусть на некотором множестве M определены два одноместных предиката $P(x)$ и $Q(x)$

Определение 2.25. Конъюнкцией предикатов $P(x)$ и $Q(x)$ называется новый предикат $P(x) \& Q(x)$, который обращается в истинное высказывание для тех и только тех значений $x \in M$, для которых каждый из предикатов $P(x)$ и $Q(x)$ обращается в истинное высказывание.

Очевидно, что областью истинности предиката $P(x) \& Q(x)$ является множество $I_{P \& Q}(x)$ и

$$I_{P \& Q} = I_P \cap I_Q$$
 области истинности предикатов

Определение 2.26. Дизъюнкцией предикатов $P(x)$ и $Q(x)$ называется предикат $P(x) \vee Q(x)$, который обращается в истинное высказывание для тех и только тех значений переменной $x \in M$, для которых хотя бы один из предикатов $P(x)$ или $Q(x)$ обращается в истинное высказывание.

Таким образом, областью истинности предиката $P(x) \vee Q(x)$ является множество $I_P \cup I_Q$

Определение 2.27. Импликацией предикатов $P(x)$ и $Q(x)$ называется новый предикат $P(x) \rightarrow Q(x)$, который обращается в ложное высказывание тогда и только тогда, когда предикат $P(x)$ обращается в истинное высказывание, а предикат $Q(x)$ в ложное.

Из определения 2.27 и равносильности $P(x) \rightarrow Q(x) \equiv \neg P(x) \vee Q(x)$ следует, что областью истинности предиката $P(x) \rightarrow Q(x)$ является мно-

жество $I_P \cup I_Q$, где I_P дополнение множества I_P до множества M .

Определение 2.28. Отрицанием предиката $P(x)$ называется предикат $\neg P(x)$, который обращается в истинное высказывание для тех и только тех значений $x \in M$, для которых предикат $P(x)$ обращается в ложное высказывание.

Областью истинности предиката $P(x)$ является множество $I_P = M \setminus I_P$.

Логические законы, справедливые для алгебры высказываний, остаются справедливыми и для предикатов, так как значениями этих предикатов являются те же высказывания. Этим обстоятельством мы и воспользовались при нахождении области истинности импликации одноместных предикатов. В силу соответствия между предикатами и множествами законам логики предикатов соответствуют известные законы для теоретико-множественных операций. Например, двум законам дистрибутивности для логических операций соответствуют законы дистрибутивности для теоретико-множественного объединения и пересечения. Первому закону дистрибутивности

$P(x) \& (Q(x) \vee R(x)) = (P(x) \& Q(x)) \vee (P(x) \& R(x))$ соответствует теоретико-множественный закон

$$I_P \cap (I_Q \cup I_R) = (I_P \cap I_Q) \cup (I_P \cap I_R),$$

где I_P, I_Q, I_R области истинности предикатов $P(x), Q(x)$ и $R(x)$. На самом деле, говоря о соответствии между предикатами и множествами, можно сказать, что первому закону дистрибутивности предикатов отвечает аналогичный закон дистрибутивности для произвольных множеств. Кванторы и кванторные операции

Кроме операций алгебры высказываний, мы будем использовать ещё две новые операции, которых не было в алгебре высказываний, так как они связаны с особенностями логики предикатов. Эти операции называются операциями связывания переменных кванторами всеобщности и существования.

Определение 2.29 (определение квантора всеобщности). Пусть $P(x)$ предикат, определенный на множестве M . Тогда под выражением $\forall x P(x)$ понимают высказывание, истинное, когда предикат

$P(x)$ обращается в истинное высказывание для каждого элемента $x \in M$, и в ложное в противном случае. Символ $\forall$ называют квантором всеобщности (или общности).

Переменную x в предикате $P(x)$ называют свободной переменной, а в высказывании $\forall x P(x)$ эту же переменную называют связанной квантором $\forall$.

Определение 2.30 (определение квантора существования). Пусть $P(x)$ одноместный предикат, определенный на множестве M . Тогда под выражением $\exists x P(x)$ понимают высказывание, которое

является истинным в том и только в том случае, когда существует элемент $x \in M$, для которого предикат $P(x)$ обращается в истинное высказывание. Символ $\exists$ называют квантором существования. Про переменную x в высказывании $\exists x P(x)$ говорят, что она связана квантором существования.

Кванторы $\forall$ и $\exists$ называются двойственными друг другу. Эта двойственность особо проявляется при использовании операции отрицания.

Рассмотрим выражение $\forall x P(x)$. Высказывание « $\forall x P(x)$ — ложно» равносильно высказыванию «существует элемент u , для которого высказывание $P(u)$ ложно», или, что то же, «существует элемент u , для которого высказывание $P(u)$ истинно». Следовательно, вы-

ражение $\forall x P(x)$ равносильно выражению $\exists u \neg P(u)$. Аналогично, выражение $\exists x P(x)$ равносильно выражению $\forall y \neg P(y)$.

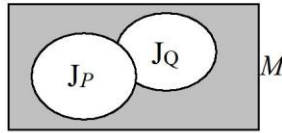
Таким образом, мы получили следующее правило: знак отрицания можно ввести под знак квантора, заменив квантор двойственным ему квантором. Рассмотрим несколько примеров на логические и кванторные операции над предикатами.

Пример 2.10. Изобразить на диаграммах Эйлера-Венна области истинности для следующих предикатов:

а) $P(x) \& Q(x)$; б) $(P(x) \wedge Q(x)) \vee (R(x) \& Q(x))$.

Решение: а) $J_P \& Q = J_P \cap J_Q = J_P \cap J_Q$ закрашен-

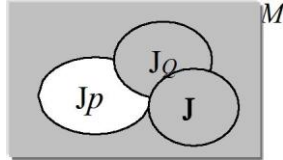
ная часть множества M .



$$б) J_{(P \rightarrow Q) \vee (R \& Q)} = J_{P \rightarrow Q} \cup J_{R \& Q} =$$

$$= (J_P \cup J_Q) \cup (J_R \cup J_Q) = (J_P \cup J_Q \cup J_R) \cup (J_Q) =$$

$$= (J_P \cup J_Q \cup J_R) \cup M = J_P \cup J_Q \cup J_R.$$



Пример 2.11. Какие из следующих выражений (слов) являются формулами логики предикатов? В каждой формуле выделить свободные и связанные переменные.

- а) $\exists x \forall z (P(x, y) \wedge P(y, z));$
- б) $\forall x (P(x) \wedge Q(x)) \wedge (\exists x P(x) \wedge xR(x, y));$
- в) $(P(x) \wedge Q(x)) \vee \exists y (VyR(y));$
- г) $\exists x, y P(x, y).$

Решение. Формулами являются выражения а) и б). Остальные выражения формулами не являются. В формуле а) у свободная переменная, x и z связанные переменные; в формуле б) у свободная переменная. Выражение в) не является формулой, так как квантор существования по переменной у навешен на формулу $\forall y R(y)$, в которой переменная у связана квантором всеобщности, что противоречит определению формулы. Наконец, выражение г) не является формулой, так как переменная у не связана квантором.

Пример 2.12. Вычислить значение формулы $\forall x \exists y P(x, y) \wedge$

$\wedge \exists x \forall y P(x, y)$, где $P(x, y)$ = «число x меньше числа y» и предикат $P(x, y)$ задан на множестве N^2 .

Решение. Высказывание $\forall x \exists y P(x, y)$ истинно, а высказывание $\exists x \forall y P(x, y)$ ложно на множестве натуральных чисел (и не только на этом множестве), следовательно, вся формула принимает значение «ложь» по определению импликации высказываний.

Пример 2.13. Доказать равносильность предикатов $c \& \forall x A(x) = x(c \& A(x))$.

Решение. Рассмотрим два случая.

а) Пусть высказывание c ложно. Тогда для любого x ложными будут высказывания $c \& \forall x A(x)$ и $\forall x (c \& A(x))$. Следовательно, в этом случае обе исходные формулы тождественно ложны.

б) Пусть теперь высказывание c истинно. Тогда, очевидно, значения исходных формул будут целиком зависеть от значений предиката $A(x)$. Если $A(x)$ тождественно истинный предикат, то будет тождественно истинным и предикат $c \& A(x)$ и, следовательно, будут тождественно истинными высказывания $\forall x A(x)$, $c \& \forall x A(x)$, $\forall x (c \& A(x))$, то есть тождественно истинны исходные формулы. Если же предикат $A(x)$ не тождественно истинный, то будет не тождественно истинным предикат $c \& A(x)$, а высказывания $\forall x A(x)$,

$c \& \forall x A(x)$, $\forall x (c \& A(x))$ будут ложными, то есть ложные значения принимают обе исходные формулы, что в итоге доказывает их равносильность.

Пример 2.14. Найти отрицания следующих формул:

а) $\forall x (A(x) \vee \exists y B(y));$ б) $\forall x (A(x) \wedge \forall y B(y)).$ Решение.

$$а) \quad \forall x (A(x) \vee \exists y B(y)) = \forall x A(x) \& \exists y B(y) = \exists x A(x) \& yB(y).$$

$$б) \quad \forall x (A(x) \wedge \forall y B(y)) = \exists x (A(x) \wedge \forall y B(y)) = \exists x (A(x) \& \forall y B(y)) =$$

$$= \exists x (A(x) \& \neg yB(y)) = \neg xA(x) \& \neg yB(y).$$

Мы воспользовались правилом введения отрицания под знак квантора.

Пример 2.15. Записать на языке логики предикатов определение линейно упорядоченного множества (напомним, что множество называется линейно упорядоченным, если для любых элементов этого множества x и y либо $x = y$, либо $x < y$, либо $x > y$).

Решение. $\forall (x \in M) \forall (y \in M) ((x = y) \vee (x > y) \vee (x < y))$.

Выводы

По лекции можно сделать следующие выводы:

- введено понятие высказывания, рассмотрены операции над высказываниями и их свойства;
- введено понятие функции алгебры логики и рассмотрены способы представления логических функций при помощи формул;
- особо выделен совершенный вид дизъюнктивных и конъюнктивных нормальных форм булевых функций;
- на основе свойств полноты и замкнутости системы булевых функций рассмотрено понятие базиса и основных классов логических функций;
- понятие высказывания расширено понятием предиката, и кроме операций над высказываниями, определенных и над предикатами, введены кванторные операции.

Вопросы и упражнения для самопроверки

Составить таблицы истинности для следующих формул и установить, какие из них являются тавтологиями:

$$(x \vee y) \wedge ((x \& y \vee x) \wedge y);$$

$$(x \& y) \wedge ((y \vee x) \wedge z);$$

$$(x \vee z) \& (y \wedge (z \wedge x));$$

$$x \vee y \wedge x \& y.$$

Используя основные законы логики, с помощью равносильных преобразований доказать следующие соотношения:

$$x \vee y = x \& y;$$

$$(x \& y) \vee (x \& y) = x;$$

$$(x \& y) \vee (x \& y) \vee (x \& y) = x \wedge y;$$

$$(x \wedge y) = (y \wedge x).$$

Докажем, например, соотношение 2.2.3:

$$(x \& y) \vee (x \& y) \vee (x \& y) = (x \& y) \vee (x \& (y \vee y)) =$$

$$= (x \& y) \vee x = (x \vee x) \& (y \vee x) = y \vee x = x \vee y.$$

Доказать тождественную истинность или тождественную ложность формул, применяя равносильные преобразования:

$$(x \vee (x \& y)) \wedge (y \vee x);$$

2.3.2. $(x > y) \vee (x > y);$ В качестве примера докажем тождественную ложность формулы

$$2.3.3. (x \vee y) \& x > y;$$

$$2.3.4. x \& (x > y) \& (x > y) = \bar{x} \& (x \vee y) \& (x \vee y) = x \& (x \vee (y \& y)) = x \& x \vee 0 = x \& x = 0.$$

$$2.3.5. y);$$

2.3.6. $(x > y) \& y > x;$ Следующие формулы преобразовать так, чтобы они содержали только операции конъюнкции и отрицания.

$$x \vee y; \quad 2.4.1.$$

$$x > y; \quad 2.4.2.$$

$$x \wedge y; \quad \text{ч} \quad 2.4.3.$$

$$x \vee (x \circ y); \quad 2.4.4.$$

$$(x \& y) > (y > x); \quad 2.4.5.$$

$$(x \vee y) > (x > z). \quad 2.4.6.$$

Рассмотрим, например, 2.4.4:

$$x \vee (x \circ y) = x \vee ((x > y) \& (y > x)) = x \vee (x \vee y) \& (y \vee x) =$$

$$= x \vee ((x \& y) \& (y \& x)) = x \& ((x \& y) \& (y \& x))$$

Найти двойственные формулы к формулам:

$$x \& y \vee z;$$

$$(x \& y) \vee (x \& z);$$

$$(x \vee y) \& (x \vee (y \sim \& z));$$

$$((x \& y) \vee (y \& z) \vee (z \& x)) \& (x \vee y \vee z); ((x \vee y) \& (x \vee z) \vee (x \& y)) \vee ((x \vee y) \& z \vee x);$$

$$(x \& y \& z) \vee (x \& y \& z) \vee (x \& y \& z) \vee (x \& y \& z).$$

Рассмотрим в качестве примера формулу 2.5.3. Сначала получим формулу f^* , двойственную формуле $f(x, y, z) = (x \vee y) \& (x \vee (y \& z))$, исходя из определения двойственной формулы:

$f^*(x, y, z) = f(x, y, z) = (x \vee y) \& (x \vee (y \& z)) = (x \& y) \& (x \vee (y \& z)) = (x \& y) \vee (x(y \& z)) = (x \& y) \vee (x \& (y \& z)) = (x \& y) \vee (x \& (y \vee z))$, что совпадает с формулой, которая получается из формулы 2.5.3 после замены операции $\&$ на $\vee$ и $\vee$ на $\&$.

Основываясь на законе контрапозиции $(x \wedge y) = (y \wedge x)$, для следующих теорем сформулировать обратное, противоположное и обратное противоположному предположения и установить истинность каждого из них:

Теорема Пифагора.

Точка пересечения диагоналей параллелограмма является его центром симметрии.

Всякая монотонная и ограниченная последовательность имеет предел.

Решение 2.6.1. Теорему Пифагора сформулируем на языке импликации: если треугольник прямоугольный, то в нем сумма квадратов катетов равна квадрату гипотенузы истинное высказывание $(x \wedge y)$.

Обратное высказывание имеет вид $y \wedge x$: если в некотором треугольнике сумма квадратов длин каких-либо сторон равна квадрату длины третьей стороны, то этот треугольник прямоугольный. Это утверждение также истинно.

Противоположное высказывание: если треугольник не прямоугольный, то в нем сумма квадратов длин никаких двух сторон не равна квадрату длины третьей $(x \wedge y)$. Применяя закон контрапозиции в виде $(y \wedge x) = x \wedge y$, убеждаемся в истинности и этого утверждения.

Наконец, выражая словесно формулу $y \wedge x$, получаем предложение, обратное противоположному теореме Пифагора: если в треугольнике сумма квадратов длин никаких двух сторон не равна квадрату длины третьей стороны, то этот треугольник не прямоугольный. В силу закона контрапозиции, записанного в начале задачи, получаем, что и это высказывание истинно.

Заметим, что закон контрапозиции лежит в основе доказательства утверждений методом от противного.

Составить таблицу истинности штриха Шеффера и выразить его через операции $\&$, $\vee$, $\sim$ и наоборот, операции $\&$, $\vee$, выразить через штрих Шеффера.

Ответ: $x | y = x \& y = x \vee y; \quad x = xX; \quad x \& y = (Xy) | (xIy);$

$x \vee y = (xI x) | (yI y).$

Представить в универсальном базисе «штрих Шеффера» выражения $x_1 \vee x_2 \& x_3$ и $(x_1 \vee x_2) \& x_3$.

Составить таблицу истинности стрелки Пирса и выразить ее через $\&$, $\vee$, и обратно, операции $\&$, $\vee$, выразить через стрелку Пирса.

Ответ: $x \Phi y = x \vee y = x \& y; \quad x = x \Phi x; \quad x \vee y = (x \Phi y) \Phi (x \Phi y);$

$x \& y = (x \Phi x) \Phi (y \Phi y).$

Найти x , если $(x \vee a) \vee (x \vee a) = b$.

Ответ: $x = b$.

Для следующих функций найти СДНФ каждой двумя способами (путем равносильных преобразований и используя таблицы истинности):

$f(x, y) = (xy \rightarrow x) \& | xy \rightarrow y \wedge |;$

$f(x, y, z) = (x \vee y \rightarrow x \& z) \rightarrow f x \rightarrow x \vee y \& z.$

I)

Что является отличительной особенностью логики предикатов от логики высказываний?

Дать определение одноместного предиката, его области определения и множества истинности.

Какие предикаты называются тождественно истинными (тождественно ложными)?

Каким образом понятие одноместного предиката обобщается на понятие многоместного предиката? Дать определение логических операций над предикатами. Как связаны результаты логических операций над предикатами с множествами истинности исходных предикатов?

Дать определение кванторных операций.

Сформулировать правило о введении знака отрицания под знак квантора.

Дать определение формулы и подформулы логики предикатов.

Дать определение равносильных формул логики предикатов.

На подмножестве $M = \{14, 15, 16, \dots, 38, 39\}$ натуральных чисел заданы предикаты:

$A(x) = \langle x \text{ не делится на } 4 \rangle$;

$B(x) = \langle x \text{ четное число} \rangle$;

$C(x) = \langle x \text{ простое число} \rangle$;

$D(x) = \langle x \text{ кратно } 7 \rangle$.

Данные ниже предикаты сформулировать словесно на русском языке и найти их области истинности. Изобразить области истинности графически (на кругах Эйлера). Преобразовать данные предикаты в высказывание двумя способами:

связав переменную x квантором общности;

связав переменную x квантором существования.

Построить отрицания полученных формул с кванторами и установить истинность этих формул.

$A(x) \& B(x)$;

$B(x) \& C(x)$;

$B(x) \vee D(x)$;

$A(x) \vee B(x) \vee D(x)$;

$C(x) \rightarrow A(x)$;

$D(x) \wedge C(x)$;

$(A(x) \& C(x)) \wedge D(x)$;

$(A(x) \& D(x)) \wedge C(x)$.

Выписать все подформулы данных формул:

1. $(p_1 \& p_2) \wedge (p_1 \vee p_2)$;

2. $(p_1 \rightarrow p_2 \vee p_3) \wedge p_3$;

$(x \rightarrow (y \rightarrow x) \& y) \vee z$;

$(x \rightarrow y) \rightarrow ((x \rightarrow y) \rightarrow x)$;

$(x \rightarrow z) \rightarrow ((y \rightarrow z) \rightarrow ((x \vee y) \rightarrow z))$;

$a \vee b \rightarrow c \& a$;

$a \& c \vee b \rightarrow (b \rightarrow c)$;

$((p \rightarrow q) \& (p \rightarrow r)) \rightarrow (p \rightarrow (q \& r))$;

$(x \rightarrow y \& z) \vee (x \& z)$;

$((x \vee y) \& z) \rightarrow z$;

$x \rightarrow y \vee (x \& y)$;

$((x \rightarrow y) \& (y \rightarrow z)) \rightarrow (x \vee z)$;

$(x \rightarrow y) \rightarrow ((x \rightarrow y) \rightarrow y)$;

$x \rightarrow y \& (x \vee y)$;

$((p_1 \& p_2) \rightarrow (p_1 \vee p_2)) \rightarrow p_2$;

$((x \vee y) \vee (x \& y)) \rightarrow z$.

Ввести предикат на соответствующей области и записать на языке логики предикатов с использованием кванторов данное высказывание. Определить значение истинности данного высказывания. Записать отрицание высказывания, полученного в символической форме, и сформулировать его на русском языке.

Между любыми двумя различными точками на прямой лежит, по крайней мере, одна точка, с ними не совпадающая.

Через две различные точки проходит единственная прямая.

Каждый курсант выполнил, по крайней мере, одну лабораторную работу.

Если произведение натуральных чисел делится на простое число, то на него делится, по крайней мере, один из сомножителей.

Через три точки, не лежащие на одной прямой, проходит единственная плоскость.

Любое действительное число либо положительное, либо отрицательное, либо равно нулю.

Существует действительное число, которое больше своего квадрата.

Модуль любого комплексного или действительного числа положителен.

Всякое рациональное число представимо в виде обыкновенной дроби, в которой числитель целое число, а знаменатель натуральное число.

Не существует наибольшего простого числа.

Функция $y = 2^x$ положительна при любом действительном значении аргумента и принимает любое действительное положительное значение.

Из перпендикулярности любых двух векторов следует равенство нулю их скалярного произведения.

Не всякое уравнение имеет действительный корень.

Не все люди умеют играть в шахматы или в шашки, зато все умеют играть в «дурака».

Не всякая река впадает в море, но все реки берут начало с гор.

Ни один человек не бессмертен.

Изобразить на действительной координатной плоскости области истинности заданных предикатов.

$\exists x (x < 2, \dots + y);$
 $\exists x (x < 1, \dots)$
 $\exists x (x < 1, \dots)$
 $(x^2 + y^2 = 1) \vee (x > 0);$
 $(x^2 + y^2 = 4) \vee (xy = 0);$
 $(x > 5) \wedge (x > 1);$
 $(|x| < 2) \wedge (x < 3);$
 $(x^2 + y^2 < 1) \wedge (xy > 0);$
 $(x^2 + y^2 > 0) \wedge (xy > 0);$
 $(x^2 + y^2 < 0) \wedge (x = y);$
 $(x > 0) \vee (x + y = 3);$
 $(x > 1) \wedge (x < 5);$
 $(|x| > 3) \wedge (|x| < 4).$

Изобразить на диаграммах Эйлера-Венна области истинности для следующих предикатов:

$P(x) \& Q(x);$
 $(P(x) \wedge Q(x)) \vee (R(x) \& Q(x));$
 $P(x) \wedge (Q(x) \vee R(x));$
 $P(x) \& Q(x) \wedge R(x).$

Какие из следующих выражений (слов) являются формулами логики предикатов? В каждой формуле выделить свободные и связанные переменные.

$\exists x \forall z (P(x, y) \wedge P(y, z));$
 $\forall x (P(x) \wedge Q(x)) \wedge (\exists x P(x) \wedge xR(x, y));$
 $(p \wedge q) \& (r \vee p)$
 $P(x) \& \forall x Q(x);$
 $(P(x) \wedge Q(x)) \vee \exists y (\forall y R(y));$
 $\exists x \vee z (P(x, y) \wedge P(y, z));$
 $p \wedge \forall x P(x, y);$
 $\exists x, y P(x, y).$

Ответ. Формулами являются 2.1.1, 2.1.2, 2.1.3, 2.1.6 и 2.1.7. Остальные выражения формулами не являются.

Вычислить значения формул:

$\forall x (P(x) \& Q(x) \wedge R(x));$

²²⁷2. $\forall x \exists y P(x, y) > \exists x \forall y P(x, y),$

где $P(x) = \langle x \text{ делится на } 3 \rangle$, $Q(x) = \langle x \text{ делится на } 4 \rangle$, $R(x) = \langle x \text{ делится на } 2 \rangle$ и предикаты определены на множестве натуральных чисел; $P(x, y) = \langle \text{число } x \text{ меньше числа } y \rangle$ и предикат $P(x, y)$ задан на множестве $\mathbb{N}^2$.

Доказать следующие равносильности:

$\forall x A(x) = \exists x A(x);$
 $\exists x A(x) = \forall x A(x);$
 $c \& \forall x A(x) = c \& A(x);$
 $c \vee cA(x) = \forall x(c \vee A(x));$
 $\exists x (A(x) \vee B(x)) = \exists x A(x) \vee \exists x B(x);$
 $\exists x(c \vee A(x)) = c \vee \exists x A(x).$

В качестве примера докажем равносильность предикатов в 2.28.3. Рассмотрим два случая.

Пусть высказывание c ложно. Тогда для любого x ложными будут высказывания $c \& \forall x A(x)$ и $\forall x(c \& A(x))$. Следовательно,

в этом случае обе исходные формулы тождественно ложны.

Пусть теперь высказывание c истинно. Тогда, очевидно, значения исходных формул будут целиком зависеть от значений предиката $A(x)$. Если $A(x)$ тождественно истинный предикат, то будет тождественно истинным и предикат $c \& A(x)$, и, следовательно, будут тождественно истинными высказывания $\forall x A(x)$, $c \& \forall x A(x)$, $\forall x(c \& A(x))$, то есть тождественно истинны исходные формулы. Если же предикат $A(x)$ не тождественно истинный, то будет не тождественно истинным и предикат $c \& A(x)$, а высказывания $\forall x A(x)$, $c \& \forall x A(x)$, $\forall x(c \& A(x))$ будут ложными, то есть ложные значения принимают обе исходные формулы, что в итоге доказывает их равносильность.

Найти отрицания следующих формул:

$\exists x (A(x) \& B(x) \& C(x));$
 $\forall x (A(x) \vee \exists y B(y)); \forall x (A(x) \wedge \forall y B(y));$
 $\forall x (A(x) \wedge B(x)) \& \exists x (S(x) \& R(x));$
 $\exists x (R(x) \wedge R(x)).$

Решим, например, 2.29.2 и 2.29.3.

$\forall x (A(x) \vee \exists y B(y)) = \forall x A(x) \& \exists y B(y) = \exists x A(x) \& y B(y).$

$\forall x (A(x) \wedge \forall y B(y)) \wedge \exists x (A(x) \wedge \forall y B(y)) \wedge \exists x (A(x) \& \forall y B(y)) \wedge$
 $\wedge \exists x (A(x) \& \exists y B(y)) \wedge \exists x A(x) \& \exists y B(y).$

Пусть $A(x)$ и $B(x)$ два одноместных предиката, определенных на множестве M таких, что высказывание $\exists x (A(x) \wedge (A(x) \vee (B(x) \wedge A(x))))$ истинно. Доказать, что высказывание $\forall x A(x)$ ложно. Доказательство проводится методом от противного.

Записать на языке логики предикатов определения:

Линейно упорядоченного множества (определение линейно упорядоченного

множества дано в п.1.6).

Ограниченной функции (напомним, что функция $f(x)$ называется ограниченной на множестве M , если существует такое неотрицательное число l , что для всех $x \in M$ справедливо неравенство $f(x) \leq l$).

Ответ: 2.31.1. $\forall (x \in M) \forall (y \in M) ((x = y) \vee (x > y) \vee (x < y))$.

$\exists (l \in \mathbb{R}) \forall (x \in M) (|f(x)| < l)$.

Записать на языке логики предикатов следующие высказывания:

Некоторые действительные числа являются рациональными.

Ни одно простое число не является точным квадратом.

Некоторые четные числа не делятся на 8.

Всякое число, кратное 6, делится на 3.

Числа 5 и 12 не имеют общих делителей, отличных от +1 и -1.

Натуральное число, делящееся на 6, делится на 2 и на 3.

Для любого целого числа x существует такое целое число y , что $x = 2y$ или $x = 2y+1$.

Для любого натурального числа существует другое натуральное число, которое больше первого.

Существует наименьшее натуральное число.

Не существует такого рационального числа x , что $x^2 = 2$.

Для всяких целых чисел x и z существует целое число y такое, что $x + y = z$.

Для любых двух рациональных чисел x и y существует рациональное число z такое, что $x < z$ и $z < y$.

Пусть $P(x)$ обозначает « x простое число», $Q(x)$ « x четное число», $R(x)$ « x целое число», $D(x, y)$ « x делит y ». Сформулировать словами следующие высказывания, записанные на языке логики предикатов. Отметить, какие из них истинные и какие ложные:

$\forall x P(x) \wedge Q(x)$;

$\forall x (P(x) \wedge \forall y P(y) \wedge D(z, y))$;

$\forall x (Q(x) \supset \exists y (D(x, y) \wedge Q(y)))$;

$\forall x \exists y (R(x) \wedge R(y) \wedge D(x, y))$;

$\forall y \forall x (R(x) \wedge R(y) \wedge D(x, y))$;

$x \vee y (R(x) \wedge R(y) \wedge D(x, y))$.

Замечание. В результате изучения и исследования цикла задач на логику предикатов, мы фактически совершаем переход на но-вый уровень абстракции такого типа, какой был совершен в шко-ле от арифметики вещественных чисел к алгебре числовых функций. Новым здесь по сравнению с алгеброй высказываний является введение кванторных операций. Эти операции широко используются для записи определений и компактной записи доказательства различных теорем.

3. ИСЧИСЛЕНИЕ ВЫСКАЗЫВАНИЙ

План занятия

1. Алфавит, формулы и подформулы исчисления высказываний.
2. Аксиомы исчисления высказываний.
3. Основные правила вывода.
4. Определение доказуемой (выводимой) формулы.
5. Производные правила вывода. Теорема дедукции.
6. Связь алгебры высказываний с исчислением высказываний.

Исторически понятие формальной теории было разработано в период интенсивных исследований в области оснований математики для формализации собственно логики и

теории доказательства. Сейчас этот аппарат широко используется при создании специальных исчислений для решения конкретных прикладных задач.

Таблицы истинности в алгебре высказываний позволяют ответить на многие вопросы, касающиеся логики высказываний. Однако более сложные вопросы логики высказываний уже не могут быть решены с помощью таблиц истинности. Давая описание алгебры высказываний, мы пользовались логическими значениями высказываний (истина, ложь). Но понятия истинности и ложности не являются математическими. Эти понятия во многих случаях субъективны и скорее относятся к методологии познания. В связи с этим желательно построить теорию, не пользуясь понятием истинности и ложности, не применяя законов логики. Такой теорией, в частности, является исчисление высказываний, которое строится по общим правилам построения формальных аксиоматических (дедуктивных) систем (или исчислений). Именно построение такой формальной системы или теории является одним из способов моделирования наших рассуждений во многих практически важных приложениях.

Формальная теория строится следующим образом.

1. Определяется множество формул, или правильно построенных выражений, образующее язык теории. Это множество задается конструктивными средствами (как правило, индуктивным определением) и, следовательно, перечислимо, а в большинстве случаев и разрешимо.

2. Выделяется подмножество формул, называемых аксиомами теории, которое должно быть разрешимым.

3. Задаются правила вывода теории, то есть вычислимое отношение на множестве формул.

Процесс построения теорем в таких теориях называется формальным выводом или доказательством в формальной теории.

Одной из таких формальных теорий является исчисление высказываний. Исчисление высказываний – это аксиоматическая логическая система, интерпретацией которой является алгебра высказываний.

Описание всякого исчисления включает в себя описание символов этого исчисления (алфавита), формул, являющихся конечными конфигурациями символов, и определение выводимых формул.

3. 1. Алфавит, формулы и подформулы исчисления высказываний

Алфавит исчисления высказываний состоит из символов трех категорий:

1. Символы, которые обозначают переменные высказывания: $x, y, z, \dots, x_1, x_2, \dots$ строчные буквы латинского алфавита (с индексами или без них).

2. Логические связи: $\&, \vee, \wedge, \neg$.

3. Круглые скобки $(,)$.

Других символов алфавит исчисления высказываний не имеет.

Формулы исчисления высказываний представляют собой конечные последовательности символов алфавита исчисления высказываний. Для обозначения формул будем пользоваться прописными буквами латинского алфавита. Эти буквы не являются символами алфавита исчисления высказываний, а представляют собой только условные обозначения формул, то есть это символы так называемой метатеории, с помощью которой описывается исчисление высказываний.

Множество формул обычно задается индуктивным определением с помощью формальной грамматики. Как правило, это множество бесконечно.

1. Всякое переменное высказывание $x, y, z, \dots$ является формулой.

2. Если A и B формулы, то формулами является и следующие комбинации символов: $(A \& B), (A \vee B), (A \wedge B), (A \neg B)$.

3. Никакая другая комбинация символов формулой не является.

Переменные высказывания будем называть элементарными формулами.

Примеры формул:

$x, y, (x \& y), (x \vee y), (y \wedge z), x, (x \& y), ((x \vee z) \& (y \wedge z)), ((x \& y) \wedge (y \wedge z))$ ■

Очевидно, не являются формулами комбинации $xu, \& x, (x \& y, x \vee y$. В последнем примере нет скобок. Одновременно с понятием формулы вводится понятие подформулы или части формулы.

1. Подформулой элементарной формулы является только она сама.
2. Если формула имеет вид A , то ее подформулами являются: она сама, формула A и все подформулы формулы A .
3. Если формула имеет вид $(A * B)$, где под символом $*$ понимается любая из логических связок $\vee, \&, \wedge$, то ее подформулами являются:

она сама, формулы A и B и все подформулы формул A и B .

Например, для формулы $(x \& y) \wedge z \vee y$ ее подформулами будут:

- $((x \& y) \wedge (z \vee y))$ подформула нулевой глубины!;
- $(x \& y), (z \vee y)$ подформулы! первой глубины!;
- $x, (y), (z \vee y)$ подформулы второй глубины;
- y, z подформулы третьей глубины;
- z подформула четвертой глубины.

Очевидно, что на самой большой глубине находятся лишь элементарные формулы. Однако элементарные формулы могут быть и на других глубинах.

Введем в запись формул некоторые упрощения, а именно: будем опускать в записи формул скобки по тем же правилам, что и в алгебре высказываний.

Например, вместо формул $((A \& B) \vee C), (A \& B), ((A \vee B) \wedge (C \& D))$ будем писать соответственно $A \& B \vee C, A \& B, A \vee B \wedge C \& D$.

Таким образом, мы будем опускать внешние скобки и некоторые внутренние, если последовательность выполнения логических операций не вызывает сомнений ввиду их приоритета: $\vee, \&, \vee, \wedge$.

3. 2. Аксиомы исчисления высказываний

Следующим шагом в описании исчисления высказываний будет выделение некоторого класса формул, которые мы будем называть исходными доказуемыми (выводимыми) формулами или аксиомами исчисления высказываний.

Одна из возможных систем аксиом исчисления высказываний состоит из 11 аксиом, которые делятся на четыре группы.

Первая группа аксиом:

11 $x \wedge (y \wedge x).$

$(x \wedge (y \wedge z)) \wedge ((x \wedge y) \wedge (x \wedge z)).$ Вторая группа аксиом:

111 $x \& y \wedge x.$

112 $x \& y \wedge y.$

113 $(z \wedge x) \wedge ((z \wedge y) \wedge (z \wedge x \& y)).$

Третья группа аксиом:

III1. $x \wedge x \vee y.$

III2. $y \wedge x \vee y.$

III3. $(x \wedge z) \wedge ((y \wedge z) \wedge (x \vee y \wedge z)).$

Четвертая группа аксиом:

I V1. $(x \wedge y) \wedge (y \wedge x).$

I V2.

I V3. $x \wedge x.$

Все аксиомы первой группы из логических связок содержат только импликацию. Эта связка входит и во все остальные группы, но во второй группе к импликации присоединяется конъюнкция, в третьей группе дизъюнкция, а в четвертой отрицание.

Примеры других систем аксиом исчисления высказываний будут приведены в конце этого раздела.

3. 3. Основные правила вывода

Выявление того факта, что из множества высказываний (формул исчисления

высказываний) логически следует некоторое другое высказывание (формула) и является, по существу, одной из основных задач исчисления высказываний.

Определение доказуемых формул имеет тот же характер, что и определение формулы. После введения исходных доказуемых формул (аксиом) определяются основные правила вывода, которые позволяют из имеющихся доказуемых формул получать новые доказуемые формулы. Образование доказуемой формулы из исходных доказуемых формул путем применения правил вывода, называется логическим выводом данной формулы, доказательством или логическим следствием.

Правило заключения. Если формулы A и $A \wedge B$ доказуемы в исчислении высказываний, то формула B также доказуема. Схематически это правило записывают так:

$$\frac{A, A \wedge B}{B} \quad (3.1)$$

В литературе доказуемость формулы F чаще всего обозначают символом $\vdash F$ или символом $\provdash F$ (мы будем использовать обозначение $\vdash F$). Тогда правило заключения можно записать в виде:

Запятая над чертой означает союз «и».

Правило заключения принято называть правилом *modus ponens*. В этом правиле мы принимаем процедуру перехода от двух формул вида A и $A \wedge B$ к одной формуле B . В выводе по этому правилу (то есть в применении этого правила) формулы A и $A \wedge B$ являются посылками, а формула B заключением.

В некоторых источниках символ доказуемости обозначается символом логического следствия « $\Rightarrow$ » (запись $P \Rightarrow Q$ означает, что формула Q логически следует из формулы P , то есть что импликация $P \Rightarrow Q$ является тавтологией). Концепция логического следствия обычно используется в логическом программировании и в искусственном интеллекте при описании методов логического вывода. Мы коснемся этих вопросов во второй главе. Заметим, что импликацию иногда обозначают символом « $\supset$ ».

В теории доказательств в большинстве случаев правило *modus ponens* (или правило отделения, как его еще называют) принимается в качестве единственного основного правила вывода. Однако часто в качестве основного правила вывода вводится еще правило подстановки.

Правило подстановки. Если формула A доказуема в исчислении высказываний, x переменная, B произвольная формула исчисления высказываний, то формула, полученная в результате замены в формуле A переменной x всюду, где она входит, формулой B , является также доказуемой формулой.

Символически это правило можно записать в виде

$$\frac{A(x) \vdash A(B)}{A(B)} \quad (3.2)$$

Формулы, используемые в правилах (3.1) и (3.2), называются метаформулами или схемами формул. Смысл метаформул состоит в том, что они обозначают множество всех формул исчисления, которые получаются, если ее метаварiable заменить формулами исчисления.

Использование схем формул можно распространить и на аксиомы. Например, если в первой группе аксиом переменные высказывания x, y, z заменив метаварiableными X, Y, Z , то получается две схемы аксиом, задающие три бесконечных множества аксиом. В результате возникает другой способ построения исчисления высказываний с бесконечным множеством аксиом (задаваемым конечным числом схем), но без правила подстановки, поскольку оно неявно содержится в истолковании схем аксиом. Первый способ, то есть способ, которым воспользовались мы, более конструктивен: все его средства явно зафиксированы и конечны. С другой стороны, второй способ больше соответствует математической традиции истолкования формул: например, алгебраическое тождество $(x + y)^2 = x^2 + 2xy + y^2$ или тождества булевой алгебры истолковываются именно как схемы тождеств, а не конкретные тождества, верные лишь для конкретных букв.

Теперь несколько уточним правило подстановки.

а) Если формула A есть переменная x , то подстановка (3.2) дает формулу B .

б) Если формула A есть переменная $y \wedge x$, то подстановка (3.2) дает формулу A .
 в) Если A формула, для которой подстановка уже определена, то подстановка V вместо x в отрицание формулы A есть отрицание подстановки.

г) Если A_1 и A_2 формулы, для которых подстановки уже определены, то $\vdash_{\mathcal{L}} \neg L(B) * L_2(5)$,
 где под символом $*$ понимается любая из логических связок $\vee, \&, \wedge$.

3. 4. Определение доказуемой (выводимой) формулы

1. Каждая аксиома является доказуемой формулой.
2. Формула B , полученная из доказуемых формул A и $A \wedge B$ путем применения правила заключения, есть доказуемая формула.
3. Формула, полученная из доказуемой формулы путем применения правила подстановки вместо переменной x произвольной формулы B есть доказуемая формула.
4. Никакая другая формула исчисления высказываний не считается доказуемой.

Процесс получения доказуемой формулы называется доказательством.

Пример 3.1. Докажем рефлексивность импликации, то есть $p \wedge A \wedge A$. Доказательство. Аксиома I_2 доказуема, поэтому $p \wedge (x \wedge (y \wedge z)) \wedge ((x \wedge y) \wedge (x \wedge y))$. Применим правило подстановки: вместо z в I_2 подставим x . В результате получим:

$$p \wedge (x \wedge (y \wedge x)) \wedge ((x \wedge y) \wedge (x \wedge y)). \quad (3.3)$$

Так как в формуле (3.3) $(x \wedge (y \wedge x))$ есть аксиома I_1 , то $p \wedge (x \wedge (y \wedge x))$. Тогда, применяя к формуле (3.3) правило *modus ponens*, получим $p \wedge (x \wedge (y \wedge x)) \wedge ((x \wedge y) \wedge (x \wedge x))$

$$p \wedge ((x \wedge y) \wedge (x \wedge x)).$$

Таким образом, имеем

$$p \wedge (x \wedge y) \wedge (x \wedge x).$$

В последнюю формулу вместо y подставим x . В результате получим доказуемую формулу

$$p \wedge (x \wedge x) \wedge (x \wedge x). \quad (3.4)$$

По аксиоме IV_2 $x \wedge x$ и пункту 1 определения доказуемой формулы получаем, что

$$p \wedge (x \wedge x). \quad (3.5)$$

Применяя правило *modus ponens* последовательно к формулам (3.5) и (3.4), получим:

$$p \wedge (x \wedge x), p \wedge ((x \wedge x) \wedge (x \wedge x)) \vdash p \wedge (x \wedge x),$$

то есть получим доказуемость формулы $x \wedge x$. По правилу подстановки вместо x формулы A , получим

$$pA \wedge A. \quad _$$

Пример 3.2. Доказать, что $p \wedge x \vee y \wedge x \& y$ (второй закон де Моргана).

Доказательство. В аксиоме II_3 выполним последовательно две подстановки, заменяя сначала x на x , а затем y на y . В результате получим доказуемую формулу

$$p \wedge (z \wedge x) \wedge ((z \wedge y) \wedge (z \wedge x \& y)). \quad (3.6)$$

В формуле (3.6) выполним подстановку $x \vee y$ вместо z :

$$p \wedge (x \vee y \wedge x) \wedge ((x \vee y \wedge y) \wedge (x \vee y \wedge x \& y)). \quad (3.7)$$

Теперь докажем, что формулы

$$x \vee y \wedge x \quad (3.8)$$

и

$$x \vee y \wedge y \quad (3.9)$$

доказуемы. Для этого в аксиоме IV_1 выполним подстановку $x \vee y$ вместо y :

$$p \wedge (x \wedge x \vee y) \wedge (x \vee y \wedge x). \quad (3.10)$$

Аксиома III_1 является доказуемой формулой, поэтому:

$$p \wedge (x \wedge x \vee y). \quad (3.11)$$

Применяя правило *modus ponens* к формулам (3.10) и (3.11), получим $p \wedge (x \vee y \wedge x)$, то есть доказуемость формулы (3.8). Аналогично устанавливается доказуемость формулы

(3.9).

Применяя правило modus ponens к формулам (3.8) и (3.7), получим доказуемую формулу

$$p(x \vee y) \wedge (x \vee y \wedge x \wedge y). \quad (3.12)$$

Применяя правило modus ponens к формулам (3.9) и (3.12), получим $p(x \vee y \wedge x \wedge y)$, что и требовалось доказать.

Предлагаем доказать самостоятельно выводимость формулы $(x \wedge y) \wedge \neg(x \wedge x)$.

3. 5. Производные правила вывода. Теорема дедукции

Следует оговориться по поводу терминов «доказательство» и «вывод» (иногда вместо термина «вывод» используют термин «выводимость»). В дальнейшем мы будем различать эти термины. Обозначим через T формальную теорию исчисления высказываний. Тогда под доказательством формулы B в теории T мы будем понимать такой вывод, в котором в качестве исходных формул используются только аксиомы. Формула B , для которой существует доказательство, называется формулой, доказуемой в теории T или теоремой теории T , а сам факт доказуемости формулы B мы, как и ранее, будем обозначать символом pB .

Однако вывод формулы B может осуществляться не только из аксиом, но и из других формул, то есть из совокупности некоторого числа формул, не обязательно являющихся только аксиомами. Таким образом, понятие доказательства является частным случаем понятия выводимости. На этом пути при дальнейшем развитии теории исчисления высказываний мы можем получить ряд новых правил вывода, отличных от правил заключения и подстановки, которые в свою очередь могут использоваться для доказательства большинства законов логики и установления связи между алгеброй высказываний и исчислением высказываний. Эти правила (то есть правила, отличные от аксиом) и называются производными правилами вывода. Рассмотрим наиболее важные производные правила вывода.

Правило одновременной подстановки: пусть A доказуемая формула; $x_1, x_2, \dots, x_n$ переменные высказывания, а $B_1, B_2, \dots, B_n$ -

любые формулы теории T . Тогда результат одновременной подстановки в формулу A вместо переменных $x_1, x_2, \dots, x_n$ соответственно

формул $B_1, B_2, \dots, B_n$ является доказуемой формулой теории T .

Правило сложного заключения:

$$\frac{A_1, A_2, \dots, A_n, A_1 \wedge (A_2 \wedge (A_3 \wedge (\dots A_n) \dots))}{A_1 \wedge (A_2 \wedge (A_3 \wedge (\dots A_n) \dots))} Z.$$

Правила одновременной подстановки и сложного заключения являются естественным обобщением соответственно правил подстановки, и заключения на случай n формул.

Правило силлогизма:

$$\frac{A \wedge B, \neg B \wedge C}{A \wedge C}.$$

Доказательство. В аксиоме I1 вместо переменных высказываний x, y подставим формулы $B \wedge C$ и A , а в аксиоме I2 вместо x, y, z подставим соответственно формулы A, B и C . Получим доказуемые формулы

Правило контрапозиции:

$$p(A \wedge B) \rightarrow p(B \wedge A).$$

Правило снятия двойного отрицания:

$$\frac{A \wedge \neg \neg B}{A \wedge B}, \quad \frac{\neg \neg A}{A}.$$

Докажем в качестве примера правило контрапозиции, которое лежит в основе мощного метода доказательства теорем методом от противного. Для доказательства в аксиоме IV1 $((x \wedge y) \wedge (y \wedge x))$ сделаем одновременную подстановку: вместо x подставим формулу A , а вместо y формулу B . Получим доказуемую формулу $p(A \wedge B) \wedge (B \wedge A)$. Из посылки правила контрапозиции формула $A \wedge B$ доказуема, поэтому, применяя правило

modus ponens, получим заключение правила контрапозиции. Предлагаем доказать самостоятельно правила сложного заключения и снятия двойного отрицания.

Указание. При доказательстве правила сложного заключения следует последовательно применить правило modus ponens, а при доказательстве, например, первого из правил снятия двойного отрицания применить к аксиоме IV2 подстановку: вместо x подставить формулу A и далее воспользоваться правилом силлогизма.

Рассмотрим конечную совокупность формул $\Gamma = \{A_1, A_2, \dots, A_n\}$.

Определение 3.1. Выводом формулы B из формул множества Γ называется последовательность формул $F_1, F_2, \dots, F_m$ такая, что $F_m = B$, а любая из формул $F_1, F_2, \dots, F_m$ либо аксиома, либо одна из исходных формул $A_1, A_2, \dots, A_n$, либо непосредственно выводима из формул, предшествующих ей, по одному из правил вывода.

Если существует вывод формулы B из совокупности формул Γ , то говорят, что формула B выводима из совокупности формул Γ и записывают $\Gamma \vdash B$ или $A_1, A_2, \dots, A_n \vdash B$. Формулы $A_1, A_2, \dots, A_n$ называются гипотезами или посылками вывода. Переход в выводе F_{i+1} к F_i называется i -ым шагом вывода. Очевидно, что присоединение формул к гипотезам не нарушает выводимости. Поэтому, если

B , то $A \vdash B$, и если $A_1, A_2, \dots, A_n \vdash B$, то $A_1, A_2, \dots, A_n, A_{n+1} \vdash B$.

Анализируя определение 3.1 выводимой формулы, отметим, что:

- 1) Всякая формула из множества Γ является выводимой из формул того же множества.
- 2) Всякая доказуемая формула выводима из множества Γ .
- 3) Если формулы C и $C \vdash B$ выводимы из совокупности формул Γ , то формула B также выводима из множества Γ .

Нетрудно видеть, что класс формул, выводимых из совокупности Γ , совпадает с классом доказуемых формул в случае, когда совокупность Γ содержит только доказуемые формулы и в случае, когда $\Gamma = \emptyset$. Если же совокупность Γ содержит хотя бы одну недоказуемую формулу, то класс формул, выводимых из множества Γ , шире класса доказуемых формул.

Из определения 3.1 следуют очевидные свойства вывода:

1⁰. Всякий начальный отрезок последовательности формул $F_1, F_2, \dots, F_m$ из совокупности формул $\Gamma = \{A_1, A_2, \dots, A_n\}$ есть вывод из множества формул Γ .

2⁰. Если между двумя соседними членами последовательности формул $F_1, F_2, \dots, F_m$, или в ее начале, или в ее конце вставить некоторый вывод из множества формул Γ , то полученная последовательность формул также будет выводом из множества Γ .

3⁰. Всякая формула F_i ($i = 1, m$) является формулой, выводимой из множества Γ .

4⁰. Если $\Gamma \vdash D$, то всякий вывод из множества формул Γ является выводом из множества формул D .

На интуитивном уровне применительно к некоторой конкретной и содержательной теории T вывод есть ничто иное, как получение в процессе доказательства некоторого утверждения (теоремы, леммы, формулы и т.д.) промежуточных утверждений, основанных либо на аксиомах, либо на полученных ранее утверждениях по имеющимся правилам вывода (умозаключениям).

Теорема дедукции. Пусть Γ множество формул, A и B формулы. Тогда

$\Gamma, A \vdash B$

$\Gamma \vdash A \rightarrow B$.

Доказательство. Предварительно методом полной математической индукции докажем, что для любого вывода $F_1, F_2, \dots, F_m$ из совокупности формул Γ и A следует утверждение $\Gamma, A \vdash F_m$.

1. При $m = 1$ утверждение верно. Действительно, если F_1 есть вывод из совокупности формул Γ и A , то возможны три случая:

- а) $F_1 \in \Gamma$,
- б) F_1 доказуема формула,

в) $F1 = A$.

В случаях а) и б) можно записать вывод из множества Γ : $F1, F1 \wedge (A \wedge F1), A \wedge F1$ и, следовательно, $\Gamma \vdash A \wedge F1$. В случае в) фактически нужно доказать, что $\Gamma \vdash A \wedge A$. Но формула $A \wedge A$ доказуема и поэтому выводима из любой совокупности формул.

2. Предположим теперь, что утверждение справедливо для любого вывода длины $k < m$ и докажем, что оно справедливо для вывода длины m .

Пусть $F1, F2, \dots, Fm$ вывод из Γ и A . Ясно, что здесь $m > 1$, и поэтому относительно формулы Fm возможны четыре случая:

а) $Fm \in \Gamma$;

б) Fm доказуема формула;

в) $Fm = A$;

Fm получается по правилу *modus ponens* из двух формул, предшествующих ей в выводе. Для первых трех случаев доказательство утверждения $\Gamma \vdash A \rightarrow Fm$ полностью совпадает с доказательством, приведенным для $m = 1$.

Рассмотрим четвертый случай. Так как здесь формула Fm получается из двух формул Fk и Fj при $k < m$ и $j < m$, то формула Fj долж-

на иметь вид $Fk \rightarrow Fm$, при этом

$\Gamma \vdash A \rightarrow Fk, \quad (3.13)$

$\Gamma \vdash A \rightarrow (Fk \rightarrow Fm). \quad (3.14)$

В аксиоме $I2$ ($x \rightarrow (y \rightarrow z) \rightarrow ((x \rightarrow y) \rightarrow (x \rightarrow z))$) выполним подстановку: вместо переменных x, y, z подставим соответственно формулы A, Fk и Fm . В результате получим

$\vdash (A \rightarrow (Fk \rightarrow Fm)) \rightarrow ((A \rightarrow Fk) \rightarrow (A \rightarrow Fm)). \quad (3.15)$

Каждая из формул (3.13), (3.14) и (3.15) выводима из множества Γ . Применяя к ним правило сложного заключения, получим $\Gamma \vdash A \rightarrow Fm$.

Вернемся теперь к общему случаю. Пусть $\Gamma, \Delta \vdash B$. Тогда существует вывод $F1, F2, \dots, Fm$ из совокупности формул Γ и A . По доказанным выше частным случаям теоремы дедукции справедливо утверждение $\Gamma \vdash A \rightarrow B$. Теорема дедукции доказана.

На основе понятия вывода из совокупности формул можно доказать ряд новых производных правил вывода и законов логики. Рассмотрим несколько наиболее важных в исчислении высказываний таких правил и законов.

1. Пусть Γ и Δ две совокупности формул исчисления высказываний, тогда, как следует из определения 3.1,

$\Gamma \vdash \Gamma$

$\Gamma, \Delta \vdash A$, где A произвольная формула алгебры высказываний.

2. $\Gamma, \Gamma \vdash A, \Gamma \vdash G$,

$\Gamma \vdash A$

где Γ совокупность формул, а A и G произвольные формулы исчисления высказываний.

Действительно, так как по условию из совокупности формул Γ и формулы G выводима формула A , то существует вывод из Γ и G , последней формулой которого является формула A :

$F1, F2, \dots, Fk, A. \quad (3.16)$

Так как по условию из совокупности формул Γ выводима формула G , то существует вывод из совокупности формул Γ , последней формулой которого является G : $G1, G2, \dots, Gm, G. \quad (3.17)$

Если в выводе (3.16) отсутствует формула G , то он является выводом только из совокупности Γ и, значит, из совокупности формул Γ выводима формула A . Если в выводе (3.16) одна из формул есть формула G (например, формула $F1$), то вставим между формулами Fi и $Fi+1$ вывод из множества Γ (3.16). В результате получим вывод только из формул множества Γ :

$F1, F2, \dots, Fi-1, Gi, G2, \dots, Gm, G, Fi+1, \dots, Fk-1, A$, следовательно, формула A

выводима из множества формул Γ .

Правило введения конъюнкции:

$$\frac{\Gamma \vdash A, \Gamma \vdash B}{\Gamma \vdash A \& B}$$

Доказательство. По условию

$$\Gamma \vdash A, \quad (3.18)$$

$$\Gamma \vdash B. \quad (3.19)$$

Докажем, что из совокупности формул $\{A, B\}$ выводима конъюнкция $A \& B$, то есть

$$\{A, B\} \vdash A \& B. \quad (3.20)$$

Действительно, так как в результате подстановки в аксиоме ПЗ вместо x, y, z формул A, B, A соответственно, а в аксиоме П1 вместо x, y формул B и A , мы получаем доказуемые формулы, которые выводимы из множества формул $\{A, B\}$ по определению, то

$$\{A, B\} \vdash (A \wedge A) \wedge ((A \wedge B) \wedge (A \wedge A \& B)), \quad (3.21)$$

$$\{A, B\} \vdash B \wedge (A \wedge B). \quad (3.22)$$

Так как формула $A \wedge A$ доказуема, то

$$\{A, B\} \vdash A \wedge A. \quad (3.23)$$

Из формул (3.18), (3.16), $\{A, B\} \vdash A$, $\{A, B\} \vdash B$, (3.22), (3.20) и (3.19) по правилу заключения последовательно получим следующие три формулы:

$$\{A, B\} \vdash (A \wedge B) \wedge (A \wedge A \& B). \quad (3.24)$$

$$\{A, B\} \vdash A \wedge B. \quad (3.25)$$

$$\{A, B\} \vdash A \wedge A \& B. \quad (3.26)$$

Из формул $\{A, B\} \vdash A$ и (3.26) следует истинность утверждения (3.20). Поскольку присоединение формул к гипотезам не нарушает выводимости (производное правило вывода 1 из совокупности формул), получим

$$\Gamma, A, B \vdash A \& B, \quad (3.27) \quad \Gamma, A \vdash B. \quad (3.28)$$

По правилу сложного заключения из формул (3.27) и (3.28) вытекает формула

$$\Gamma, A \vdash A \& B. \quad (3.29)$$

Наконец, по правилу сложного заключения из формул (3.13) и (3.29) следует истинность доказываемого правила введения конъюнкции.

Правило введения дизъюнкции:

$$\frac{\Gamma \vdash A, \Gamma \vdash B}{\Gamma \vdash A \vee B} \text{ ПС}$$

Доказательство. Из условия $\Gamma, A \vdash C, \Gamma, B \vdash C$ по теореме дедукции имеем

$$\Gamma \vdash A \rightarrow C, \Gamma \vdash B \rightarrow C.$$

Аксиома ПЗ выводима из совокупности формул Γ как доказуемая формула, то есть $\Gamma \vdash (A \wedge C) \wedge ((B \wedge C) \wedge (A \vee B \wedge C))$.

Применяя к последним трем формулам правило сложного заключения, получим

$$\Gamma \vdash A \vee B \wedge C, \text{ откуда } \Gamma, A \vee B \vdash C.$$

Опираясь на теорему дедукции и на правила подстановки, заключения, силлогизма, контрапозиции и снятия двойного отрицания, можно доказать большинство законов логики, например:

Закон перестановки посылок:

$$P(x \wedge (y \wedge z)) \wedge (y \wedge (x \wedge z)) \bullet \quad (3.30 \text{ а)})$$

Закон соединения посылок:

$$P(x \wedge (y \wedge z)) \wedge (x \& y \wedge z). \quad (3.30 \text{ б)})$$

Закон разъединения посылок:

$$P(x \& y \wedge z) \wedge (x \wedge (y \wedge z)). \quad (3.30 \text{ в)})$$

3.6. Связь алгебры высказываний с исчислением высказываний

Формула F исчисления высказываний содержательно интерпретируется как составное высказывание, истинность которого зависит от истинности входящих в него элементарных высказываний. Эта зависимость в точности соответствует зависимости

значения логической функции, представляемой формулой F , от значений переменных этой функции. Иначе говоря, если задана формула $F(x_1, x_2, \dots, x_n)$ и распределение истинностей входящих в нее элементарных высказываний $x_1, x_2, \dots, x_n$, то для выяснения значения истинности этой формулы ее значение можно вычислить методами, приведенными в алгебре высказываний, то есть как значение логической функции на двоичном наборе $\langle a_1, a_2, \dots, a_n \rangle$, где $a_i = 1$, если X_i истинно, и $a_i = 0$, если X_i ложно. Если $F(a_1, a_2, \dots, a_n) = 1$, то высказывание F истинно при данном распределении истинностей $X_1, X_2, \dots, X_n$; если $F(a_1, a_2, \dots, a_n) = 0$, то F ложно.

Возникает вопрос: как связано такое содержательное, «истинностное» истолкование формул с их выводимостью в исчислении высказываний? Ответ на этот вопрос дают три теоремы, которые устанавливают связь между основными фактами алгебры высказываний и исчисления высказываний.

Предварительно введем следующее обозначение. Как известно, двоичный набор $\langle a_1, a_2, \dots, a_n \rangle$ имеет 2^n значений. Определим значение формулы F на одном таком наборе значений переменных $X_1, X_2, \dots, X_n$, обозначая его через $\langle a_1, a_2, \dots, a_n \rangle^{(F)}$:

1. Если для формулы F ее подформулой самой большей глубины является X_i , то

$$\langle a_1, a_2, \dots, a_n \rangle^{(F)} = a_i.$$

2. Если определены значения всех подформул глубины $k+1$, то подформулы глубины k , полученные в результате операций

$F_i \& F_j, F_i \vee F_j, F_i \wedge F_j, F_i$ будут иметь значения:

$$\langle a_1, a_2, \dots, a_n \rangle^{(F_i \& F_j)} = \langle a_1, a_2, \dots, a_n \rangle^{(F_i)} \& \langle a_1, a_2, \dots, a_n \rangle^{(F_j)},$$

$$\langle a_1, a_2, \dots, a_n \rangle^{(F_i \vee F_j)} = \langle a_1, a_2, \dots, a_n \rangle^{(F_i)} \vee \langle a_1, a_2, \dots, a_n \rangle^{(F_j)},$$

$$\langle a_1, a_2, \dots, a_n \rangle^{(F_i \wedge F_j)} = \langle a_1, a_2, \dots, a_n \rangle^{(F_i)} \wedge \langle a_1, a_2, \dots, a_n \rangle^{(F_j)},$$

$$\langle a_1, a_2, \dots, a_n \rangle^{(F_i)} = \langle a_1, a_2, \dots, a_n \rangle^{(F_i)}.$$

$$\langle a_1, a_2, \dots, a_n \rangle^{(F_i)} = \langle a_1, a_2, \dots, a_n \rangle^{(F_i)}.$$

Например, формула $x_1 \vee x_4 \wedge x_2 \& x_3$ на наборе значений $\langle 0, 0, 1, 1 \rangle$

переменных x_1, x_2, x_3, x_4 имеет значение $\langle 0, 0, 1, 1 \rangle^{(x_1 \vee x_4 \wedge x_2 \& x_3)} = 1$, в чем

$$\langle 0, 0, 1, 1 \rangle^{(x_1 \vee x_4 \wedge x_2 \& x_3)} = 1.$$

легко убедиться, рассматривая подформулы первой, второй, третьей и четвертой глубины.

Теперь докажем вспомогательное предложение. Лемма 3.1. Пусть $x_1, x_2, \dots, x_n, x$ список переменных, которые входят в формулы F и G , и пусть $\langle a_1, a_2, \dots, a_n, a \rangle$ фиксированный двоичный набор. Тогда, если $N_{a_1 a_2 \dots a_n} (G) = v$, то в результате подста-

$$\langle a_1, a_2, \dots, a_n, a \rangle^{(F)}$$

новки в формулу F вместо переменной x формулы G получим

$$\langle a_1, a_2, \dots, a_n, a \rangle^{(F)}$$

$$\langle a_1, a_2, \dots, a_n, a \rangle^{(F)}$$

Доказательство. Доказательство леммы проведем индукцией по построению формулы F .

а) Пусть F есть переменная x_i . Тогда $N_{a_1 a_2 \dots a_n} (x_i) = a_i$; ре-

зультат подстановки вместо x формулы G дает x_i ; результат подстановки в формулу F вместо переменной x формулы G дает a_i , и поэтому $N_{a_1 a_2 \dots a_n} (F) = a_i$, то есть в этом случае утверждение леммы 1 2 n

справедливо.

б) Пусть $F = G$. Тогда подстановка в формулу F вместо переменной x формулы G дает $G (F=G)$, а $N_{a_1 a_2 \dots a_n} (G) = v$, то есть и в этом

$$\langle a_1, a_2, \dots, a_n \rangle^{(F)}$$

случае лемма верна.

в) Пусть теперь $F = F_1 * F_2$ (* одна из логических операций конъюнкции, дизъюнкции или импликации) и для формул F_1 и F_2 утверждение леммы справедливо. Подставляя в формулу F вместо x формулу G и учитывая, что $F = F_1 * F_2$, получим

$$H_{1 \dots n} (F) \supset H_{1 \dots n} (F_1 * F_2) = H_{1 \dots n} (F_1) * H_{1 \dots n} (F_2).$$

Аналогично лемма доказывается для отрицания формулы F .

Методом полной математической индукции по построению формулы докажем справедливость следующей леммы.

Лемма 3.2 (о выводимости). Пусть A некоторая формула исчисления высказываний; $x_1, x_2, \dots, x_n$ полный набор переменных, входящих в формулу A ; $\langle a_1, a_2, \dots, a_n \rangle$ — произвольный набор значений пе

ременных $x_1, x_2, \dots, x_n$; $\{x_{a_1}, x_{a_2}, \dots, x_{a_n}\}$ — конечная совокупность

$$x_i = 1, \text{ если } a_i = 1, \\ x_i = 0, \text{ если } a_i = 0.$$

формулы, где

$x_i = 1$, если $a_i = 1$,

$x_i = 0$, если $a_i = 0$.

Тогда: 1) если $S \vdash A$, то $\Gamma \vdash A$; 2) если $S \vdash A$, то

если $S \vdash A$, то $\Gamma \vdash A$; 2) если $S \vdash A$, то

Гр A . Доказательство. 1. Базис индукции по построению формул. Пусть формула A совпадает с одной из переменных x_i ($i = 1, n$). Тогда, если $\Gamma \vdash x_i = a = 1$, то, очевидно, $\Gamma \vdash x_i$, или $\Gamma \vdash x_i$, то

есть $x_i \vdash A$, и тем более $\Gamma \vdash A$. Если же $\Gamma \vdash x_i = a_i = 0$, то $x_i \vdash \neg A$, или $x_i \vdash \neg A$, то есть $x_i \vdash \neg A$ и $\Gamma \vdash \neg A$.

2. Индуктивное предположение. Предположим, что для формул B_1 и B_2 теорема справедлива, а формула A имеет один из следующих четырех видов:

I. $A = B_1 \& B_2$, II. $A = B_1 \vee B_2$, III. $A = B_1 \supset B_2$, IV. $A = B_1$.

3. Проведем доказательство для каждого из четырех видов формулы A .

1. Пусть $A = B_1 \& B_2$. Возможны два случая:

а) $\Gamma \vdash B_1 \& B_2 = 1$, б) $\Gamma \vdash B_1 \& B_2 = 0$.

В случае а) $\Gamma \vdash B_1 = 1$ и $\Gamma \vdash B_2 = 1$. По индуктивно-

му предположению $\Gamma \vdash B_1$ и $\Gamma \vdash B_2$. Отсюда по правилу введения конъюнкции $\Gamma \vdash B_1 \& B_2$, то есть $\Gamma \vdash A$.

В случае б) либо $\Gamma \vdash B_1 = 0$, либо $\Gamma \vdash B_2 = 0$. Не

нарушая общности, можно считать, что $\Gamma \vdash B_1 = 0$. Тогда по индуктивному

предположению $\Gamma \vdash B_1$. (3.31)

С другой стороны, применяя к аксиоме Π_1 правило контрапозиции, получим

(3.32)

Из формул (3.31) и (3.32) по правилу modus ponens получаем $\Gamma \vdash B_1 \& B_2$, то есть $\Gamma \vdash A$.

II. Пусть $A = B_1 \vee B_2$. Здесь также возможны два случая:

а) $\Gamma \vdash B_1 \vee B_2 = 1$, б) $\Gamma \vdash B_1 \vee B_2 = 0$.

В случае а) выполняется хотя бы одно из равенств $\Gamma \vdash B_1 = 1$ или $\Gamma \vdash B_2 = 1$. Пусть, например, выполняется первое из них. Тогда по индуктивному предположению $\Gamma \vdash B_1$. По аксиоме Π_1 $\Gamma \vdash B_1 \vee B_2$. Применяя к последним двум формулам правило modus ponens , получим $\Gamma \vdash B_1 \vee B_2$, то есть $\Gamma \vdash A$.

В случае б) одновременно $\neg \text{Ч}_{12n} \text{ а } \dots \text{а} (B_1) = 0$ и $\text{Я}_{12n} \text{ а } \dots \text{а} (B_2) = 0$. От-

сюда следует, что ΓB_1 и ΓB_2 . Но тогда по правилу введения конъюнкции $\Gamma B_1 \& B_2$. (3.33)

Воспользуемся доказуемой формулой

$\neg B_1 \& B_2 \vee B_1$. (3.34)

Из формул (3.33) и (3.34) по правилу *modus ponens* получим $\Gamma \neg B_1 \vee B_2$, то ΓA .

III. Пусть $A = B_1 \rightarrow B_2$. Тогда возможны два случая:

а) $\neg \text{а } \dots \text{а} (B_1 \vee B_2) = 1$, б) $\text{ч}_{12n} \text{ а } \dots \text{а} (B_1 \vee B_2) = 0$.

В случае а) либо $\text{Ч}_{12n} \text{ а } \dots \text{а} (B_1) = 0$, либо $\text{Ч}_{12n} \text{ а } \dots \text{а} (B_2) = 1$.

Если $\text{ч}_{12n} \text{ а } \dots \text{а} (B_1) = 0$, ΓB_1 . (3.35)

Воспользуемся доказуемой формулой $\neg B \rightarrow (B \rightarrow B)$. Из этой формулы по правилу перестановки посылок (3.30 а)) получим

$\neg B_1 \rightarrow (B_1 \vee B_2)$. (3.36)

Из формул (3.35) и (3.36) по правилу *modus ponens* получим $\Gamma B_1 \vee B_2$, то есть ΓA .

Если же $\text{ч}_{12n} \text{ а } \dots \text{а} (B_2) = 1$, то ΓB_2 . Из этой формулы и аксиомы I_1 по правилу *modus ponens* получим $\Gamma B_1 \vee B_2$, то есть ΓA .

В случае б) имеется только один вариант: $\text{ч}_{12n} \text{ а } \dots \text{а} (B_1) = 1$, $\text{а } \dots \text{а} (B_2) = 0$. Тогда по индуктивному предположению ΓB_1 и $\neg B_2$.

По свойству рефлексивности импликации можно записать $\neg (B_1 \wedge B_2) \wedge (B_1 \wedge B_2)$. Из этой формулы по правилу перестановки посылок получим $\neg B_1 \wedge ((B_1 \wedge B_2) \wedge B_2)$. Из последней формулы с учетом того, что ΓB_1 , по правилу *modus ponens* получим $\Gamma ((B_1 \wedge B_2) \wedge B_2)$, а по правилу контрапозиции отсюда следует формула $\neg B_2 \wedge (B_1 \wedge B_2)$. Наконец, учитывая, что $\neg B_2$, по правилу *modus ponens* получим $\Gamma B_1 \wedge B_2$, то есть ΓA .

IV. Пусть $A = B_1$. Если $\text{ЭТ}_{12n} \text{ а } \dots \text{а} (B_1) = 1$, то $\text{Я}_{12n} \text{ а } \dots \text{а} (B_1) = 0$, и зна-

чит $\Gamma \neg B_1$, то есть ΓA . Если $\text{ч}_{2n12n} \text{ а } \dots \text{а} (B_1) = 0$, то $\text{ЭТ}_{12n} \text{ а } \dots \text{а} (B_1) = 1$ и,

следовательно, ΓB_1 , откуда по аксиоме IV2 по правилу *modus ponens* получим ΓB_1 , то есть ΓA .

На основании принципа полной математической индукции лемма 3.2 доказана.

Докажем теперь основную теорему, устанавливающую связь между алгеброй высказываний и исчислением высказываний.

Теорема 3.1. Каждая формула, доказуемая в исчислении высказываний, является тождественно истинной в алгебре высказываний. Обратно, каждая тождественно истинная формула алгебры высказываний доказуема в исчислении высказываний.

Доказательство. Для доказательства первой части теоремы 3.1 следует доказать три положения:

- 1) Каждая аксиома исчисления высказываний тождественно истинная формула в алгебре высказываний.
- 2) Правило подстановки, примененное к тождественно истинным формулам, приводит к тождественно истинным формулам.
- 3) Правило заключения, примененное к тождественно истинным формулам, приводит к тождественно истинным формулам.

Первое положение проверяется непосредственно с помощью таблиц истинности путем перебора всех возможных значений входящих в аксиомы переменных.

Докажем утверждение п. 2). Надо доказать, что если F тождественно истинная формула, x переменная и G любая формула исчисления высказываний, то в результате подстановки в формулу F вместо переменной x формулы G мы получим также тождественно истинную формулу. Действительно, пусть $x_1, x_2, \dots, x_n$, x полный список

переменных, входящих в формулы F и G . Нужно показать, что на всех наборах $\langle a_1, a_2, \dots, a_n, a \rangle$ значений переменных формула F , в которой переменная x заменена формулой G , принимает значение 1. Предположим, что это не так. Тогда существует набор $\langle a_1, a_2, \dots, a_n, a \rangle$

что результат замены, указанный выше, дает 0. Но тогда согласно доказанной выше лемме 3.1 $\bigwedge_{a_1, a_2, \dots, a_n} (F) = 0$, а это противоречит тому, что F тождественно истинная формула.

Остается доказать утверждение п.3): если формулы G и $G \wedge F$ тождественно истинны, то и формула F тождественно истинна. Пусть $x_1, x_2, \dots, x_n$ полный набор переменных, входящих в формулы

F и G . Предположим, что F не тождественно истинная формула. Тогда существует набор $\langle a_1^0, a_2^0, \dots, a_n^0 \rangle$, для которого выполняется

$$1 \quad 2 \quad n$$

равенство $\bigwedge_{a_1^0, a_2^0, \dots, a_n^0} (F) = 0$, но при этом (по определению) $a_1^0 a_2^0 \dots a_n^0$

$$\wedge_{a_1^0, a_2^0, \dots, a_n^0} (G \wedge F) = \bigwedge_{a_1^0, a_2^0, \dots, a_n^0} (G) \wedge \bigwedge_{a_1^0, a_2^0, \dots, a_n^0} (F) = 1 \wedge 0 = 0,$$

$$1 \quad 2 \quad n \quad 1 \quad 2 \quad n \quad 1 \quad 2 \quad n$$

что противоречит тождественной истинности формулы $G \wedge F$.

Докажем теперь вторую часть теоремы. Пусть F тождественно истинная формула в алгебре высказываний и $x_1, x_2, \dots, x_n$ полный

набор всех переменных, входящих в формулу F . Зафиксируем произвольный набор $\langle a_1, a_2, \dots, a_n \rangle$ значений переменных $x_1, x_2, \dots, x_n$.

Так как формула F тождественно истинная, то $\bigwedge_{a_1, a_2, \dots, a_n} (F) = 1$. То-

$$1 \quad 2 \quad n$$

гда по лемме 3.2 $\Gamma_n \vdash F$, где $\Gamma_n = \{x_1^{a_1}, x_2^{a_2}, \dots, x_n^{a_n}\}$, причем соотношение $1 \quad 2 \quad n$

$\Gamma_n \vdash F$ справедливо для всех возможных 2^n наборов $\langle a_1, a_2, \dots, a_n \rangle$. $1 \quad 2 \quad n$

Если $\Gamma_{n-1} = \{x_1^{a_1}, x_2^{a_2}, \dots, x_{n-1}^{a_{n-1}}\}$, то, очевидно, $\Gamma_{n-1}, x_n \vdash F$ и $\Gamma_{n-1}, x_n \sim \vdash 1 \quad 2 \quad n-1 \quad n$

F . Тогда по правилу введения дизъюнкции $\Gamma_{n-1} \vdash x_n \vee x_n \vdash F$, откуда $\Gamma_{n-1} \vdash F$ (формула $x_n \vee x_n$ доказуема, и ее можно опустить). По ана-

логии последовательно докажем, что $\Gamma_{n-2} \vdash F$, $\Gamma_{n-3} \vdash F$, ..., $\Gamma_2 \vdash F$, $\Gamma_1 \vdash F$. Но соотношение $\Gamma_1 = \{x_1^{a_1}\} \vdash F$ справедливо лишь при $a_1 = 1$ и при $a_1 = 0$, то есть $x_1 \vdash F$ и $x_1 \sim \vdash F$. Отсюда по правилу введения дизъюнкции получим, что $x_1 \vee x_1 \vdash F$, то есть $0 \vdash F$, а это означает, что

формула F доказуема. Теорема 3.1 доказана полностью. Таким образом, исчисление высказываний действительно выполняет задачу порождения общелогических законов тождественно истинных высказываний (тавтологий).

Замечание. Рассматривая связь исчисления высказываний с алгеброй высказываний, мы не касались операции эквивалентности высказываний $A \wedge B$. В логике высказываний если $f_1 = f_2$ эквива-

лентное соотношение, то формула $f_1 \wedge f_2$ является тождественно

истинным высказыванием. В исчислении высказываний формула $f \wedge f$ является сокращением формулы $(f \wedge f_2) \& (f_2 \wedge f)$. По теореме 3.1 (вторая часть) все такие эквивалентности, например, $A \vee B \wedge A \& B$, являются доказуемыми, то есть являются теоре-

мами в исчислении высказываний.

Рассмотренные нами вопросы, относящиеся к исчислению высказываний, позволяют овладеть отдельными логическими правилами вывода и простыми принципами, за счет которых и составляются сложные доказательства. Практическое применение этих принципов состоит в их использовании для вывода интересных

результатов в математике, в точных науках и в повседневной жизни, а не в решении искусственно сконструированных логических задач.

В заключение приведем несколько примеров других систем аксиом, точнее, схем аксиом исчисления высказываний, которые эквивалентны рассмотренной в п.3.2 системе аксиом.

S1. Наиболее ранней, вероятно, является система аксиом Фреге (1879 г.), в которой использовались только операции импликации и отрицания. Эта система состояла из следующих 6 схем:

1. $A \wedge (B \wedge A)$;
2. $(A \wedge (B \wedge C)) \wedge (B \wedge (A \wedge C))$;
3. $(A \wedge (B \wedge C)) \wedge ((A \wedge B) \wedge (A \wedge C))$;
4. $(A \wedge B) \wedge (B \wedge A)$;
5. $A \wedge A$;
6. $A \wedge A$.

S2. Другим примером является система аксиом, предложенная в 1910 г. Уайтхедом и Б. Расселом, и дополненная позднее П. Бернайсом. Эта система состоит из 10 схем (формул) и использует все пять логических связок: $A \vee A \wedge A$;

2. $A \wedge B \vee A$;
3. $A \vee B \wedge B \vee A$;
4. $(B \wedge C) \wedge (A \vee B \wedge A \vee C)$;
5. $(A \vee B) \wedge (A \wedge B)$;
6. $(A \wedge B) \wedge (A \vee B)$;
7. $(A \vee B) \wedge (A \& B)$;
8. $(A \& B) \wedge (A \vee B)$;
9. $((A \wedge B) \& (B \wedge A)) \wedge (A \wedge B)$;
10. $(A \wedge B) \wedge ((A \wedge B) \& (B \wedge A))$.

S3. Следующая система аксиом использует лишь две логические операции импликацию и дизъюнкцию и состоит всего из 4 схем аксиом (Гильберт и Аккерман, 1938 г.):

1. $A \vee A \wedge A$;
2. $A \wedge A \vee B$;
3. $A \vee B \wedge B \vee A$;
4. $(B \wedge C) \wedge (A \vee B \wedge A \vee C)$.

54. Россером в 1953 г. была подробно изучена система аксиом, состоящая из трех схем аксиом:

1. $A \wedge (A \& A)$;
2. $(A \& B) \wedge A$;
3. $(A \wedge B) \wedge (B \& C \wedge C \& A)$.

55. Еще один пример схем аксиом, состоящих всего из трех формул, предложен Э. Мендельсоном:

1. $A \wedge (B \wedge A)$;
2. $(A \wedge (B \wedge C)) \wedge ((A \wedge B) \wedge (A \wedge C))$;
3. $(B \wedge A) \wedge ((B \wedge A) \wedge B)$.

В исчислении высказываний, построенном на схемах аксиом S2 и S5, используется единственное правило вывода *modus ponens*.

Если в системе S5 схемы аксиом заменить конкретными аксиомами, то получим еще одну систему аксиом, однако в исчислении высказываний, построенном на такой системе, кроме правила *modus ponens* нужно использовать еще и правило подстановки.

Приведенных примеров систем аксиом вполне достаточно для того, чтобы представить большое многообразие различных подходов к построению исчисления высказываний.

Выводы

На лекции введено понятие формулы и части формулы (подформулы) исчисления высказываний, также сформулирована система аксиом этого исчисления. Рассмотренные на лекции вопросы, относящиеся к исчислению высказываний, позволяют овладеть отдельными логическими правилами вывода и простыми принципами, из которых за счет повторения и составляются сложные доказательства. Практическое применение этих принципов, оказывающихся в свою очередь полезными к использованию их для вывода интересных результатов в математике, в точных науках и в повседневной жизни, а не состоит в решении искусственно сконструированных логических задач.

1. Вопросы и упражнения для самопроверки
2. Сформулировать основные принципы построения формальной теории (исчисления).
 - a. Из чего состоит алфавит исчисления высказываний?
 - b. Дать индуктивное определение формулы исчисления высказываний.
 - c. Дать определение подформулы данной формулы.
 - d. Записать систему аксиом исчисления высказываний. Что такое схема аксиом?
 - e. Сформулировать основные правила вывода исчисления высказываний (правило заключения и правило подстановки).
 - f. Дать определение выводимой формулы, теоремы.
 - g. Что такое производные правила вывода? Сформулировать несколько таких правил.
 - h. Понятие вывода из совокупности формул это определение или теорема?
 - i. Сформулировать теорему дедукции.
 - j. В чем выражается связь между алгеброй высказываний и исчислением высказываний? Доказать, что если $\Gamma = \{A \rightarrow B\}$, то $\Gamma \vdash (B \wedge C) \rightarrow (A \wedge C)$.
 - k. Какие из следующих выражений являются формулами исчисления высказываний:
 - i. $(P \wedge P_2) \wedge (P_1 \vee P_2)$;
 - ii. $((p \vee P_i) \vee (p \wedge P_2)) \wedge P$;
 - iii. $(P_1 \wedge (P_2 \vee P_3)) \wedge P_3$;
 - iv. $(P_1 \wedge P_2) \wedge ((P_1 \wedge P_2) \wedge P_1)$;
 - v. $(P_1 \wedge (\wedge P_2)) \wedge (P_2 \wedge P_1)$;
 - vi. $(P_1 \wedge P_3)((P_2 \wedge P_3) \wedge ((P_1 \vee P_2) \wedge P_3))$;
 - vii. $((P \wedge P_2) \wedge (P \wedge P_3)) \wedge (P \wedge (P_2 \wedge P_3))$;
 - viii. $((P_1 \wedge P_2) \wedge (P_1 \vee P_2)) \wedge (P_1 \vee P_2)$.
3. Ответ: выражения 3.13.2, 3.13.5 и 3.13.8 формулами не являются.
 - a. Выписать все подформулы формул:
 - i. $x \wedge (y \wedge x)$;
 - ii. $a \wedge c \vee b$;
 - iii. $x \vee y \wedge z \wedge x$;
 - iv. $x \wedge y \vee x \wedge y$;
 - v. t ;
 - vi. $(x \wedge y) \wedge (x \wedge y) \wedge y$.
4. Выпишем, например, все подформулы формулы 3.14.4: $x \wedge y \vee x \wedge y$; $x \wedge y$, $x \wedge y$, x , y ; $x \wedge y \vee x$ подформу-
5. лой не является, поскольку операция дизъюнкции по приоритету слабее операции конъюнкции).
 - a. Установить доказуемость формул:
 - i. $A \wedge B \wedge A \wedge B \vee C$;
6. $(A \wedge B \wedge (C \wedge B \wedge C)) \wedge ((A \wedge B \wedge C) \wedge (A \wedge B \wedge B \wedge C))$;

7. $A \vee A \wedge A$;
8. $A \& B \wedge B \& A$;
9. $A \vee B \wedge B \vee A$.
10. Указания: 3.15.1. В аксиоме III1 вместо переменной x подставим формулу $A \& B$, вместо y формулу C . 3.15.2. К доказуемости приводит постановка в аксиому I2 вместо переменных x, y, z соответственно формул $A \& B, C, B \& C$. 3.15.3. В аксиоме III3 подставим вместо каждой из переменных x, y, z формулу A . Получим доказуемую формулу $(A \rightarrow A) \rightarrow ((A \rightarrow A) \rightarrow (A \vee A \rightarrow A))$. Но $p \ A \rightarrow A$.
11. Затем следует применить правило сложного заключения. 3.15.4. В аксиоме II3 вместо переменных x, y, z подставим соответственно формулы $B, A, A \& B$. Применяя правило сложного заключения,
12. получим доказуемость формулы 3.15.4. 3.15.5. В аксиоме III3 вместо переменных x, y, z подставим соответственно формулы $A, B, B \vee A$ и воспользуемся правилом сложного заключения.
 - а. Используя производные правила вывода показать, что доказуемы следующие формулы:
 - i. $A \vee B \rightarrow A \& B$;
 - ii. $(A \rightarrow B) \& B \rightarrow A$;
 - iii. $A \& B \rightarrow A \vee B$.
13. Указания. 3.16.1. В аксиоме III3 вместо x, y, z подставить соответственно $A, B, A \& B$ и далее воспользоваться аксиомами II1, II2, правилами контрапозиции и сложного заключения. 3.16.2. Осуществить последовательно подстановки: в аксиоме I2 вместо x, y, z соответственно $(A \rightarrow B) \& B, B, A$, в аксиоме II2 вместо x, y соответственно $A \rightarrow B$ и B , в аксиоме II1 вместо x, y соответственно $A \rightarrow B$ и B
14. и воспользоваться правилом сложного заключения. 3.16.3. В аксиоме III3 сделать подстановку, заменяя в ней z на $x \& y$ и далее восп-
15. пользоваться аксиомами II1, II2 и правилами контрапозиции, снятия двойного отрицания и сложного заключения.
 - а. Применяя правило подстановки, установить доказуемость формул:
 - i. $(A \& B) \rightarrow ((A \& B) \vee C)$;
 - ii. $P \vee Q \rightarrow P \vee Q$.
 - б. Применяя правило подстановки и правило modus ponens, установить доказуемость формул:
 - i. $(A \rightarrow B) \rightarrow (A \rightarrow A)$;
 - ii. $P \wedge P \& Q$.
 - в. Показать, что следующие формулы доказуемы:
 - д. $a \wedge (a \& b)$; 3.19.2. $p \vee p$; 3.19.3. $r \& s \wedge r \vee s$. Доказать производные правила вывода:

4. ИСЧИСЛЕНИЕ ПРЕДИКАТОВ

План занятия

1. Определение формулы исчисления предикатов.
2. Аксиомы исчисления предикатов и основные правила вывода.
3. Общезначимость и выполнимость формул исчисления предикатов.
4. Предваренная, скулемовская и клаузуальная формы представления формул исчисления предикатов.

Исчисление предикатов, как и всякая аксиоматическая система, содержит символы (алфавит), из которых составляются формулы. Затем среди всех формул выделяются формулы, называемые выводимыми. Выделение выводимых формул в исчислении предикатов так же, как и в исчислении высказываний, осуществляется путем указания некоторой конечной совокупности формул, которые называются аксиомами, и указанием

правил вывода, позволяющих из выводимых формул получать новые выводимые формулы.

4. 1. Определение формулы исчисления предикатов

Каждая формула исчисления предикатов представляет собой некоторую конечную последовательность символов этого исчисления (алфавита). Алфавит исчисления предикатов можно разбить на шесть групп.

1. Малые латинские буквы с индексами или без них: $a, b, c, \dots, x, y, z, \dots, a_1, a_2, \dots, x_1, x_2, \dots$. Эти символы носят название переменных предметов (предметные переменные).

2. Прописные латинские буквы с индексами внизу или без них: $A, B, \dots, A_1, A_2, \dots$. Эти символы называются переменными высказываниями.

3. Прописные латинские буквы с индексами вверху: $F^P, G^P, \dots, S^P, \dots$ и эти же символы с индексами внизу:

$F^P, F^P, \dots, G^P, G^P, \dots, S^P, S^P, \dots$. Эти символы носят названия переменных $1 \quad 2 \quad 1$
 $2 \quad 1 \quad 2$

предикатов от p переменных ($p = 1, 2, \dots, n$).

4. Знаки операций исчисления высказываний: $\&, \vee, \wedge$.

5. Символы кванторов $\forall, \exists$.

6. Круглые скобки: $(,)$.

Каждая формула является словом в алфавите, содержащем указанные выше в п.п. 1-6 символы. Поэтому мы определим, какие слова называются формулами.

1. Каждое переменное высказывание есть формула.

2. Если F символ переменного предиката, $a_1, a_2, \dots, a_p$ символы предметных переменных, не обязательно различных, то слово $F(a_1, a_2, \dots, a_p)$ есть формула. За такими формулами мы сохраним название переменного предиката.

Формулы, определённые в п.п. 1 и 2, мы будем называть элементарными.

3. Пусть формула F содержит свободную переменную x . Тогда слова $\forall x F, \exists x F$ (4.1)

также являются формулами.

Напомним, что символы $\forall$ и $\exists$, как и в алгебре предикатов, носят название кванторов: первый квантор всеобщности, второй квантор существования. Переменная x в формулах (4.1) является связанной переменной. Остальные же предметные переменные, которые в формуле F свободны, остаются свободными и в обеих формулах (4.1). Переменные, которые связаны в формуле F , остаются связанными и в формулах (4.1).

4. Пусть F_1 и F_2 формулы, причём в них нет таких предметных переменных, которые связаны в одной формуле и свободны в другой. Тогда слова

$(F_1 \& F_2), (F_1 \vee F_2), (F_1 \wedge F_2), (\neg F_1)$ (4.2)

являются формулами. При этом свободные переменные в формулах F_1 и F_2 остаются свободными во всех формулах (4.2), а связанные переменные в формулах F_1 и F_2 остаются связанными и в формулах (4.2).

Определение формулы исчисления предикатов имеет тот же индуктивный характер, что и определение формул в исчислении высказываний. Оно может быть сформулировано следующим образом.

Определение 4.1. Формулой исчисления предикатов называется слово из символов алфавита этого исчисления, которое может быть построено, исходя из элементарных формул, операциями перехода от формулы F_1 к формулам $\forall x F_1$ и $\exists x F_1$, от формул F_1 и F_2 к формулам $(F_1 \& F_2), (F_1 \vee F_2), (F_1 \wedge F_2)$ и $\neg F_1$.

Для доказательства каких-либо утверждений о формулах можно применять принцип полной математической индукции по построению формул. А именно, такое доказательство имеет следующий вид. Утверждение доказывается для элементарных формул 1 и 2. Затем доказывается, что из предположения истинности этого утверждения для формулы F следует его истинность для формул $\forall x F$ и $\exists x F$, а из

истинности его для формул $F1$ и $F2$ следует его истинность для формул (4.2). Отсюда делается заключение, что наше утверждение истинно для любой формулы.

Из определения формулы видно, что все формулы исчисления высказываний являются также формулами исчисления предикатов. В самом деле, в числе формул исчисления предикатов находятся переменные высказывания, и мы можем, исходя из них, образовывать формулы, пользуясь теми же операциями, что и в исчислении высказываний. При этом к формулам, построенным только из переменных высказываний (то есть не включающим символов предикатов), ограничения, указанные в п. 4 определения формулы, не относятся, так как они относятся только к формулам, содержащим предметные переменные.

Пример 4.1. $\exists x(F1(x) \wedge \forall yG2(y, z))$. Здесь $G2(y, z)$ переменный предикат, содержащий две свободные переменные y и z , то есть это элементарная формула. В силу п.3 индуктивного определения формулы слово $\forall yG2(y, z)$ также формула, содержащая свободную переменную z и переменную y , связанную квантором всеобщности. В формулах $F1(x)$ и $\forall yG2(y, z)$ нет переменных, связанных в одной формуле и свободных в другой, поэтому, в силу п.4 того же определения слово $(F1(x) \wedge \forall yG2(y, z))$ является формулой, в которой x и z свободные переменные, а переменная y связана квантором $\forall$. Наконец, в силу п.3 слово $\exists x(F1(x) \wedge \forall yG2(y, z))$ также является формулой, в которой переменная x связана квантором $\exists$, а переменная z свободна.

Предлагаем доказать самостоятельно, что выражение $(\forall xF1(x) \vee \exists yG2(y, z))$ является формулой, а выражение $(\exists xF1(x) \& \exists yG2(y, z))$ формулой не является.

Для формул исчисления предикатов можно, как и в исчислении высказываний, определить понятие подформулы (части формулы). Предлагаем это сделать самостоятельно.

Как и в исчислении высказываний, условимся вносить некоторые упрощения в записи формул: опускать внешние скобки и некоторые внутренние, если последовательность операций, исходя из их приоритета, не вызывает сомнений; иногда круглые скобки заменяют квадратными; можно в записи F^p опускать верхний индекс p , указывающий количество переменных, если эти переменные указаны явно в формуле F .

Кроме того, из условия 4 индуктивного определения формулы следует, что:

а) в формуле свободные и связанные переменные должны обозначаться разными буквами;

б) если какой-либо квантор находится в области действия другого квантора, то переменные, связанные этими кванторами, также обозначаются разными буквами.

Мы опускаем доказательство утверждений а) и б). Эти доказательства проводятся индукцией по построению формул исчисления предикатов. Нарушение условий а) и б) называется коллизией переменных. Например, слово $\forall x(F(x) \wedge \exists xG(x, y))$ формулой не является, так как для него нарушено условие б).

4. 2. Аксиомы исчисления предикатов и основные правила вывода

Аксиомы исчисления предикатов делятся на две группы:

1) Аксиомы исчисления высказываний, состоящих из 11 аксиом и разделенных на четыре группы I_i, I₂, II₁ II₃, III₁ III₃, IV₁ IV₃ (см. п.3.2).

2) Две предикатные аксиомы:

V1. $\forall x F(x) \wedge F(y)$.

V2. $F(y) \supset \exists x F(x)$.

В аксиомах V1 и V2 $F(x)$ любая формула, содержащая свободные вхождения переменной x , причем ни одно из них не находится в области действия квантора по y ; формула $F(y)$ получена из $F(x)$ заменой всех свободных вхождений переменной x на y .

Чтобы пояснить существенность требования к вхождениям x в формулу F , рассмотрим в качестве формулы $F(x)$ формулу $\exists y P(y, x)$, где это требование нарушено: свободное вхождение x находится в области действия $\exists y$. Подстановка этой формулы в аксиому V1 дает формулу $\forall x \exists y P(y, x) \wedge \exists y P(y, y)$. Если ее проинтерпретировать на

множестве N натуральных чисел с предикатом $P(u, v) = \text{«и больше } v\text{»}$, то получим высказывание: «если для всякого натурального числа x найдется натуральное число y , большее, чем x , то найдется натуральное число y , большее самого себя». Посылка этой импликации истинна на множестве N , а ее заключение ложно и, следовательно, само высказывание ложно.

Теперь рассмотрим основные правила вывода исчисления предикатов.

1. Правило заключения (modus ponens): если G и $G \wedge H$ выводимые формулы, то H также выводимая формула.

Имеет место также правило сложного заключения, как и в исчислении высказываний.

Видим, что это правило формулируется так же, как и в исчислении высказываний, только объем формул, к которым применяется это правило, здесь шире.

2. Правило введения квантора всеобщности: $\frac{F \wedge G(x)}{F \wedge \forall x G(x)}$

где $G(x)$ содержит свободные вхождения переменной x , а формула F их не содержит.

3. Правило введения квантора существования:

$\frac{G(x) \wedge F}{\exists x G(x) \wedge F}$

при тех же требованиях к формулам F и G , что и в правиле 2.

Нарушения правил 1-3 могут привести к ложным выводам из истинных высказываний. Рассмотрим в подтверждение этого пример.

Пример 4.2. Пусть $P(x)$ одноместный предикат « x делится на 3», $Q(x)$ предикат « x делится на 2». Высказывание $P(x) \wedge Q(x)$, очевидно, истинно для любого x , однако применение к нему правила введения квантора всеобщности дает высказывание $\forall x (P(x) \wedge Q(x))$, не всегда являющееся истинным. Если же к формуле $P(x) \wedge Q(x)$ применить правило введения квантора существования, то получим предикат $\exists x (P(x) \wedge Q(x))$, из которого путем (уже корректного) применения правила 3 получим высказывание $\exists x (P(x) \wedge \forall x Q(x))$, ложное на множестве натуральных чисел.

Заметим, что правило подстановки окончательно исчезло из изложения; тем самым из двух возможных истолкований системы аксиом (о чем шла речь в исчислении высказываний) выбрано истолкование, при котором правило подстановки отсутствует, а вместо аксиом рассматриваются схемы аксиом. Фактически этот выбор произошел уже тогда, когда аксиомы $V1$ и $V2$ были сопровождаемы словесным описанием ограничений на вхождения переменных. Тем самым аксиомы перестали быть выражениями исчисления, а вместе с этим словесным текстом превратились в метаописания множества формул, являющихся аксиомами, то есть в схемы аксиом.

Построение исчисления предикатов с правилом подстановки существенно более громоздко из-за необходимости различать свободные и связанные вхождения предметных переменных. Поэтому в большинстве современных книг по логике используется подход со схемами аксиом.

Понятие вывода и доказуемой формулы в исчислении предикатов определяются аналогично этим понятиям в исчислении высказываний. Утверждение о том, что формула F является выводимой в исчислении предикатов, обозначается так же, как и в исчислении высказываний: $\vdash F$.

Так как все формулы, выводимые в исчислении высказываний, являются также выводимыми в исчислении предикатов, то, совершая подстановки в выводимые формулы исчисления высказываний, мы будем получать выводимые формулы исчисления предикатов.

Пример 4.3. Заменяя в выводимой формуле исчисления высказываний $\vdash A \vee A$ формулу A на формулу $F(x)$, получим выводимую формулу исчисления предикатов $\vdash F(x) \vee F(x)$.

Пример 4.4. Заменяя в выводимой формуле $p \wedge A \vee B$ формулу A на формулу $F(x)$, формулу B на формулу $\forall y G(y)$, получим $p \wedge F(x) \vee \forall y G(y)$.

Подстановкой в выводимые формулы исчисления высказываний можно легко получить многие выводимые формулы исчисления предикатов; однако таким образом всякую выводимую формулу исчисления предикатов вывести нельзя, тем более что мы отказались от правила подстановки в исчислении предикатов, а лишь подразумеваем это правило в схемах аксиом.

Выведем для исчисления предикатов следующее производное правило.

Если формула $F(x)$, содержащая свободную предметную переменную x , выводима, то и формула $\forall x F(x)$ также выводима в исчислении предикатов.

Доказательство. Предположим, что верна формула $p \wedge F(x)$. Так как для произвольной выводимой формулы G $p \wedge A \wedge G$, то $p \wedge A \wedge G(x)$. Применяя правило связывания квантором всеобщности, получим выводимую формулу $p \wedge A \wedge \forall x F(x)$. Можно предположить, что формула A не входит в формулу F (ее всегда можно так выбрать). Заменив в формуле $p \wedge A \wedge \forall x F(x)$ формулу A произвольной выводимой формулой G , получим $p \wedge G \wedge \forall x F(x)$, откуда по правилу заключения можно сделать вывод об истинности формулы $p \wedge \forall x F(x)$.

Итак, если имеет место $p \wedge F(x)$, то верной будет и формула $p \wedge \forall x F(x)$, и мы доказали правило, которое можно записать так:

$F(x)$.

$\forall x F(x)$

Это правило называется производным правилом связывания квантором всеобщности. Оно применимо к любой предметной переменной.

В исчислении предикатов имеет место теорема, аналогичная теореме дедукции в исчислении высказываний:

Теорема 4.1. Если формула G выводима из формулы F , то формула $F \wedge G$ также выводима в исчислении предикатов.

Эта теорема позволяет получать выводимые формулы исчисления предикатов, не производя для них всех операций формальной дедукции, что в значительной степени сокращает непосредственный путь вывода доказуемых формул. Отметим, что доказательство данной теоремы (как и большинства других теорем исчисления предикатов) достаточно сложно и требует специального изучения.

4.3. Общезначимость и выполнимость формул исчисления предикатов

Определение 4.2. Формула A логики предикатов называется выполнимой в области M , если существуют значения переменных из области M , входящих в эту формулу, для которых формула A обращается в истинное высказывание.

Определение 4.3. Формула A называется выполнимой, если существует область, на которой эта формула выполнима.

Из определений 4.2 и 4.3 следует, что если формула A выполнима, то это еще не означает, что она выполнима в любой области.

Определение 4.4. Формула A называется тождественно истинной на множестве M , если она принимает истинное значение для всех значений переменных, входящих в эту формулу и отнесенных к области M .

Определение 4.5. Формула A называется общезначимой, если она тождественно истинна на любой области.

Из определений 4.2 и 4.5 следует:

1°. Если формула A общезначима, то она и выполнима на всякой области.

2°. Если формула A тождественно истинна в области M , то она и выполнима в этой области.

3°. Если формула A не выполнима, то она тождественно ложна на всякой области.

Выделяют два класса формул логики предикатов: класс выполнимых и класс невыполнимых формул. Общезначимую формулу называют логическим законом.

Пример 4.5. Формулы $\forall x \exists y P(x, y)$ и $\exists x \exists y (P(x) \& P(y))$ выполнимы: первая является тождественно истинной в области $M = \mathbb{N}_0 \times \mathbb{N}_0$ ($\mathbb{N}_0$ множество неотрицательных целых чисел), если в качестве $P(x, y)$ взять предикат « $x < y$ »; вторая если в качестве $P(x, y)$ взять предикат «Число x четное» на том же множестве $\mathbb{N}_0$.

Пример 4.6. Формула $\forall x (P(x) \vee \neg P(x))$ тождественно истинна в любой области M , следовательно, она является общезначимой, то есть является логическим законом (закон исключенного третьего). В то же время формула $\forall x (P(x) \& \neg P(x))$ тождественно ложна в любой области M и поэтому невыполнима.

Существуют некоторые критерии, позволяющие определить, общезначима формула или нет. Эти критерии основаны на связи между общезначимостью и выполнимостью формул исчисления предикатов. Эту связь можно выразить двумя следующими теоремами.

Теорема 4.2. Для того, чтобы формула A была общезначима, необходимо и достаточно, чтобы ее отрицание было невыполнимо.

Доказательство. Пусть формула A общезначима. Тогда по определению формула A тождественно ложна в любой области, и поэтому она невыполнима. Обратно, пусть формула A невыполнима в любой области. Тогда по определению невыполнимой формулы формула A тождественно ложна в любой области. Поэтому отрицание формулы A тождественно истинно в любой области и, следовательно, формула A общезначима.

Теорема 4.3. Для того, чтобы формула A была выполнимой, необходимо и достаточно, чтобы ее отрицание было не общезначимо.

Доказательство. Пусть формула A выполнима. Это означает, что существует область M и набор значений переменных, входящих в формулу A , для которых формула A принимает истинное значение.

Тогда на этом наборе отрицание формулы A принимает ложное значение, поэтому формула A не общезначима. Обратно, Пусть формула A не общезначима. Тогда существует область M и набор зна-

чений переменных, входящих в формулу A , для которых эта формула принимает ложное значение. На этом же наборе переменных формула A принимает истинное значение, поэтому формула A выполнима.

В исчислении предикатов имеет место теорема, аналогичная теореме дедукции в исчислении высказываний:

Если формула G выводима из формулы F , то формула $F \wedge G$ также выводима в исчислении предикатов.

Эта теорема позволяет получать выводимые формулы исчисления предикатов, не производя для них всех операций формальной дедукции, что в значительной степени сокращает непосредственный путь вывода доказуемых формул. Отметим, что доказательства сформулированной теоремы и большинства других теорем исчисления предикатов достаточно сложны и требуют специального изучения.

4.4. Предваренная, сколемовская и клаузная формы представления формул исчисления предикатов

В алгебре высказываний были введены дизъюнктивная и конъюнктивная нормальные формы (п. 2.3), которые позволяют упрощать многие доказательства в алгебре высказываний. Проблемы, возникающие в логике предикатов при оперировании с кванторами и областями действия кванторов, бывают достаточно сложными. В частности, к некоторым трудностям может привести наличие в формуле одновременно связанной и свободной одной и той же переменной. Поэтому в исчислении предикатов путем тождественных преобразований данную формулу также приводят к более простому или стандартному виду к нормальной форме. Говорят, что формула логики предикатов имеет нормальную форму, если она содержит только операции конъюнкции, дизъюнкции и кванторные операции, а операция отрицания отнесена к элементарным формулам. Среди

нормальных форм логики предикатов большое значение имеют предваренные нормальные формы. В них кванторные операции либо полностью отсутствуют, либо используются после всех операций алгебры высказываний. Дадим строгое определение предваренной

формы. **Определение 4.6.** Предваренной формой формулы исчисления предикатов называется формула $F = K_1 x_1 K_2 x_2 \dots K_n x_n R$, где K_i один

из двух кванторов, x_i переменные, входящие в формулу F , ($i = 1 \dots n$), R формула, не содержащая кванторов. Выражение $K_1 x_1 K_2 x_2 \dots K_n x_n$

называется префиксом формулы F , а формула R ее матрицей.

Теорема 4.4. Для любой формулы исчисления предикатов существует логически эквивалентная ей предваренная нормальная форма.

Доказательство. Проведем конструктивное доказательство теоремы 4.4, состоящее в построении алгоритма тождественных преобразований, используя законы логики. Можно предложить следующие шаги этого алгоритма.

1. Исключить из данной формулы все импликации (если они присутствуют в формуле) по закону: $P \rightarrow Q = P \vee Q$. Тем самым данная формула преобразуется к нормальной форме.

2. Уменьшить объем действия операции отрицания так, чтобы символ отрицания был применен не более чем к одной предикатной букве. Для этого следует использовать правила построения отрицаний формул, содержащих кванторы, а также законы де Моргана и закон двойного отрицания:

- $\forall x P(x) = \neg \exists x \neg P(x)$;
- $\exists x P(x) = \neg \forall x \neg P(x)$;
- $P \wedge Q = \neg (P \vee \neg Q)$;
- $P \vee Q = \neg (P \wedge \neg Q)$;
- $P = \neg \neg P$.

3. Переименовать, если это необходимо, связанные переменные так, чтобы в области действия каждого квантора каждая связанная им переменная имела бы индивидуальное имя, а имена свободных и связанных переменных не совпадали. Это можно сделать, так как имя связанной квантором переменной не влияет на значение формулы, например: $\forall x U(x, y) \wedge \exists z V(x, y) \vee U(x) = \forall x U(x, y) \wedge \exists z V(z, y) \vee U(m)$.

4. Удалить кванторы, если область их действия не содержит переменной, связываемой квантором, например: $\forall x \exists z P(x, y, z) = \forall x P(x, y, c)$.

5. Переместить кванторы в начало формулы, например:

$$\forall x P(x) \wedge \forall x Q(x) = \forall x (P(x) \wedge Q(x));$$

6. $\exists x P(x) \vee \exists x Q(x) = \exists x (P(x) \vee Q(x))$. Если P формула, не содержащая свободной переменной x , то будут иметь место следующие эквивалентности:

$$P \wedge \forall x Q(x) = \forall x (P \wedge Q(x)); P \wedge \exists x Q(x) = \exists x (P \wedge Q(x)).$$

Требуемый алгоритм построен и, тем самым, теорема 4.4 доказана. Разумеется, что в процессе приведения формулы к предваренной нормальной форме некоторые шаги рассмотренного выше алгоритма могут быть опущены за ненадобностью.

Пример 4.7. Привести к предваренной нормальной форме формулу

$$A = \forall x \exists y P(x, y) \wedge \exists z \forall y Q(x, y).$$

Решение. Проводя равносильные преобразования, получим:

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

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

Пример 4.8. Привести к предваренной нормальной форме формулу $F = \forall x (P(x) \wedge (\forall y (P(y) \wedge P(g(x, y))) \wedge \forall y (Q(x, y) \wedge P(y))))$. **Решение.** Воспользуемся шагами алгоритма теоремы 4.2.

1. Избавимся в формуле F от импликаций:

$$F = \forall x (P(x) \vee (\forall y (P(y) \vee \neg P(g(x, y)))) \wedge \forall y (Q(x, y) \vee \neg P(y)))$$

2. Уменьшим объем действия операции отрицания согласно законам п.2:

$$F = \forall x (P(x) \vee (\forall y (P(y) \vee \neg P(g(x, y)))) \wedge \exists y (Q(x, y) \wedge \neg P(y)))$$

3. Переименуем связанную квантором существования переменную y на z :

$$F = \forall x (P(x) \vee (\forall y (P(y) \vee P(g(x, y))) \& \exists z (Q(x, z) \& P(z))))).$$

4. Перемещая кванторы в начало формулы согласно п.5 алгоритма, получим искомую нормальную предваренную форму формулы F:

$$F = \forall x \forall y \exists z (P(x) \vee ((P(y) \vee P(g(x, y))) \& (Q(x, z) \& P(z)))).$$

5. ределение 4.7. Формула логики предикатов называется замкнутой, если она не содержит свободных предметных переменных.

Одна и та же формула исчисления предикатов допускает много эквивалентных предваренных форм. Вид полученного результата зависит от порядка применения правил, а также от произвольного переименования переменных (шаг 3 алгоритма). Тем не менее, предваренные и нормальные формы сохраняют всю выразительную силу исчисления предикатов. В частности, существует связь между общезначимостью формулы логики предикатов и ее предваренной нормальной формой. Эта связь выражается следующими двумя утверждениями, которые можно доказать методом от противного(мы не приводим этих доказательств).

1. Если замкнутая формула логики предикатов в предваренной нормальной форме содержит только кванторы существования и тождественно истинна на любой области, состоящей из одного элемента, то она общезначима.

2. Если замкнутая формула логики предикатов в предваренной нормальной форме содержит только кванторы всеобщности и тождественно истинна на любой области, состоящей из элементов, число которых не превышает числа переменных, входящих в данную формулу, то она общезначима.

Введем теперь понятие сколемовской нормальной формы формулы исчисления предикатов (по имени норвежского математика Т. А. Сколема (1887-1963); в другой транскрипции Скулем). Работать с формулами, содержащими кванторы, гораздо трудней, чем с бескванторными. Сколем в 1928 г. предложил метод замены в формулах кванторов некоторыми функциональными символами. Без ограничения общности можно рассматривать только предваренные формы, так как по теореме 4.4 любая формула может быть преобразована в эквивалентную ей предваренную форму. Более того, достаточно оперировать с замкнутыми формулами. Суть метода Сколема сводится к следующему рассуждению.

Пусть, например, $F = \exists x \forall y \exists z \forall u \forall v \exists w P(x, y, z, u, v, w)$ —

замкнутая формула исчисления предикатов. Обозначим значение переменной x , которое существует в соответствии с первым квантором, предметной константой, например, буквой a , отбросив при этом первый квантор существования в формуле F . В результате получим формулу

$$F1 = \forall y \exists z \forall u \forall v \exists w P(a, y, z, u, v, w).$$

Тот факт, что по каждому значению переменной y может быть найдено значение переменной z , может быть выражен некоторой функцией $z = f(y)$, называемой сколемовской. Тогда формула $F1$

преобразуется в формулу

$$F2 = \forall y \forall u \forall v \exists w P(a, y, f(y), u, v, w).$$

Тот факт, что для любых значений переменных y, u и v найдется значение переменной z , можно выразить другой сколемовской функцией $w = g(y, u, v)$, которая в предыдущей формуле подставляется вместо переменной w :

$$F3 = \forall y \forall u \forall v P(a, y, f(y), u, v, g(y, u, v)).$$

Так как по условию все переменные, входящие в формулу F должны быть связаны кванторами, кванторы существования исключены, а порядок следования кванторов всеобщности роли не играет при вычислении значения формулы, то отбрасывают и кванторы всеобщности, предполагая, что все переменные ими связаны. В итоге мы и получим сколемовскую форму представления данной формулы F : $S_F = P(a, y, f(y), u, v, g(y, u, v))$.

Так, для формулы примера 4.8 ее сколемовская форма имеет вид $S_F = P(x) \vee ((P(y) \vee$

$P(g(x, y))) \& (Q(x, h(x)) \& P(h(x)))$), где $z = h(x)$ сколемовская функция.

Итак, мы сопоставили формуле F формулу S_F более простого строения с гарантией, что формулы F и S_F либо обе выполнимы, либо обе невыполнимы. Эта связь между формулами F и S_F слабее, чем логическая эквивалентность, но она очень полезна при доказательстве того, что некоторое заключение является логическим следствием данных гипотез.

Клаузуальной формой формулы исчисления предикатов называется ее сколемовская форма, у которой матрица представлена в виде конъюнктивной нормальной формы, то есть в виде конъюнкции элементарных дизъюнкций (см. п. 2.3). Любая сколемовская форма допускает эквивалентную ей клаузуальную форму.

Используя дистрибутивный закон дизъюнкции относительно конъюнкции, приведем сколемовскую форму S_F примера 4.8 к

конъюнктивной нормальной форме. В результате получим клаузуальную форму S функции S_F

$S = (P(x) \vee P(y) \vee P(g(x, y))) \& (P(x) \vee Q(x, h(x))) \& (P(x) \vee P(h(x)))$, которую, заменяя символ конъюнкции запятой, записывают в виде так называемого множества дизъюнктов:

$$S = \{P(x) \vee P(y) \vee P(g(x, y)), P(x) \vee Q(x, h(x)), P(x) \vee P(h(x))\}.$$

Клаузуальная форма представления формул исчисления предикатов нами будет использоваться во второй главе при изучении метода резолюций и операции унификации. Содержательный смысл формул, выводимых в исчислении предикатов, показывает, что эти формулы представляют собой тождественно истинные высказывания, то есть логические тавтологии. Обратно, каждая тождественно истинная формула является выводимой в исчислении предикатов. Отсюда следует, что в исчислении предикатов нельзя вывести никакое сколько-нибудь содержательное по существу высказывание, в частности, математическое. Однако если к аксиомам исчисления предикатов присоединить какие-либо не выводимые формулы в качестве новых аксиом (сохраняя те же правила вывода), то получится другое исчисление, в котором выводимы, помимо тождественно истинных формул, и другие формулы.

Выводы

Содержательный смысл формул, выводимых в исчислении предикатов, показывает, что эти формулы представляют собой тождественно истинные высказывания, то есть логические тавтологии. Обратно, каждая тождественно истинная формула является выводимой в исчислении предикатов. Отсюда следует, что в исчислении предикатов нельзя вывести никакое сколько-нибудь содержательное по существу высказывание, в частности, математическое. Однако если к аксиомам исчисления предикатов присоединить какие-либо не выводимые формулы в качестве новых аксиом (сохраняя те же правила вывода), то получится другое исчисление, в котором выводимы, помимо тождественно истинных формул, и другие формулы.

Вопросы и упражнения для самопроверки

1. Дать определение алфавита исчисления предикатов.
2. Дать индуктивное определение формулы и подформулы исчисления предикатов.
3. Доказать, что выражение $(\forall x F_1(x) \vee \forall x \exists y G^2(x, y))$ является формулой исчисления предикатов, а выражение $(\exists x F_1(x) \& \exists y G^2(x, y))$ формулой не является.
4. Объяснить, почему в формуле свободные и связанные переменные должны обозначаться разными буквами;
5. Почему, если какой-либо квантор находится в области действия другого квантора, то переменные, связанные этими кванторами, следует обозначать разными буквами?
6. Сформулировать аксиомы, специфичные только для исчисления предикатов, и основные правила вывода этого исчисления.
7. Определить понятия вывода и доказуемых формул в исчислении предикатов.

8. Выполнимы ли следующие формулы:
9. $xP(x)$;
10. $\forall xP(x)$;
11. $x \forall yP(x, y)$;
12. $x \forall y (Q(x, x) \& Q(x, y))$;
13. $x \exists y (P(x) \& P(y))$.
14. Указание. Для доказательства выполнимости той или иной формулы достаточно найти область определения соответствующих предикатов и такие их значения, что в этой области формула принимает истинные значения.
15. Являются ли тождественно истинными следующие формулы:
16. $\exists xP(x) \wedge \forall xP(x)$;
17. $xP(x) \wedge \forall xP(x)$;
18. $\exists x \forall y Q(x, x) \supset \forall y \exists x Q(x, y)$;
19. $\forall x \exists y Q(x, y) \wedge \exists y \forall x Q(x, y)$?
20. Ответ: Формула 4.9.1 представляет собой закон исключенного третьего, следовательно, эта формула тождественно истинна. Очевидно, что формула 4.9.2 тождественно ложна. Формула 4.9.3 тождественно истинна, например, на множестве N^2 , если в качестве $Q(x, y)$ взять отношение « $y \sqsubset x$ ». Формула 4.9.4 тождественно истинная на любом множестве.
21. Доказать, что следующие формулы не являются общезначимыми:
22. $(\forall x \exists y P(x, y) \wedge \exists y \forall x P(x, y))$;
23. $x(P_1(x) \wedge P_2(x)) \wedge (\exists x P_1(x) \wedge \exists x P_2(x))$.
24. Какие из нижеприведенных формул являются общезначимыми:
25. $\exists x(P_1(x) \& P_2(x)) \wedge (\exists x P_1(x) \& \exists x P_2(x))$;
26. $\forall x(U(x) \wedge V(x)) \wedge (\forall x U(x) \wedge \forall x V(x))$;
27. $\exists xP(x) \wedge \forall xP(x)$;
28. $\exists xP(x) \& (r \wedge Q(x)) \wedge (\forall x(P(x) \wedge Q(x)) \wedge r)$.
29. Ответ: Используя равносильные преобразования можно доказать, что формулы 4.11.1 и 4.11.4 являются тождественно истинными на любом множестве и, следовательно, являются общезначимыми, чего нельзя сказать о формулах 4.11.2 и 4.11.3.
30. Привести к предваренной нормальной форме следующие формулы логики предикатов:
31. $F = \exists x \forall y \exists z \forall u P(x, y, z, u)$;
32. $F = \forall x R(x) \vee \exists x Q(x, y)$;
33. $F = p \supset \exists x R(x)$;
34. $F = \neg \forall x (A(x) \supset \exists y C(y))$;
35. $F = \exists x \forall y P(x, y) \& \exists x \forall y Q(x, y)$;
36. $F = \exists x \forall y P(x, y) \vee \exists x \forall y Q(x, y)$.
37. Ответ:
38. $\forall x \exists y \forall z \exists u P(x, y, z, u)$;
39. $x (R(x) \vee Q(x, y))$;
40. $\forall x (p \& R(x))$;
41. $\forall x \exists y (A(x) \vee C(y))$;
42. $\exists x \exists z \forall y (P(x, y) \& Q(z, y))$;
43. $\exists x \forall y \forall z (P(x, y) \vee Q(x, z))$.
44. Доказать выводимость следующих формул в исчислении предикатов
45. $F(x) \vee Fxx$;
46. $F(x) \wedge F(x) \vee \forall y G(y)$.
47. Решение 4.13.1. Заменяя в выводимой формуле исчисления высказываний $p \wedge \vee A$ формулу A на $F(x)$, получим выводимую формулу исчисления предикатов $p \wedge \vee F(x) \vee F(x)$.

48. Решение 4.13.2. Заменяя в выводимой формуле $p \ A \wedge A \vee B$ формулу A на $F(x)$, формулу B на $\forall y G(y)$, получим $p \ F(x) \wedge F(x) \vee \forall y G(y)$.

5. ОСНОВНЫЕ ПОНЯТИЯ АКСИОМАТИЧЕСКИХ ТЕОРИЙ ПЕРВОГО ПОРЯДКА

План занятия

1. Формальные аксиоматические теории.
2. Логические и специальные аксиомы.
3. Правила вывода и понятие доказательства в теории первого порядка.
4. Понятия интерпретации и модели теории. Изоморфизм интерпретаций.
5. Примеры аксиоматических теорий первого порядка со специальными аксиомами.

В алгебре высказываний и в исчислении высказываний использование таблиц истинности давало эффективный способ решения вопроса о том, является ли данная формула тавтологией (тождественно истинной) или нет. В логике предикатов нет эффективного способа, позволяющего для каждой формулы решить вопрос о том, является ли она общезначимой или нет (кроме частных случаев, рассмотренных в п.4.4). В связи с этим, в математических теориях, которые используют понятие предиката и связанные с ним понятия кванторных операций, необходимым становится аксиоматический метод. Этот метод базируется на простых отношениях между символами и выражениями точных формальных языков и использует достаточно простые арифметические методы.

Под аксиоматической теорией, построенной на основе данной системы аксиом, понимается совокупность всех теорем, доказываемых на основе этой системы аксиом.

Аксиоматические теории делятся на формальные и неформальные теории.

Неформальные аксиоматические теории наполнены теоретикомножественным содержанием, понятие выводимости в них довольно расплывчато и в значительной степени опирается на здравый смысл.

Формальная аксиоматическая теория считается определенной, если выполнены следующие условия:

1. Задан язык теории.
2. Определено понятие формулы в этой теории.
3. Выделено некоторое множество формул, называемых аксиомами.

Определены правила вывода в этой теории. Здесь под языком теории понимается множество всех слов теории над заданным алфавитом вместе с этим алфавитом. Это понятие нами будет уточнено в следующем пункте.

Содержательный смысл формул, выводимых в исчислении предикатов, показывает, что эти формулы представляют собой тождественно истинные высказывания, то есть логические тавтологии. Обратно, каждая тождественно истинная формула является выводимой в исчислении предикатов. Отсюда следует, что в исчислении предикатов нельзя вывести никакое сколько-нибудь содержательное по существу высказывание, в частности, математическое. Однако если к аксиомам исчисления предикатов присоединить какие-либо невыводимые формулы в качестве новых аксиом (сохраняя те же правила вывода), то получится другое исчисление, в котором выводимы, помимо тождественно истинных формул, и другие формулы.

Среди многочисленных математических теорий выделяют так называемые теории первого порядка. Они отличаются от теорий высших порядков тем, что не допускают в своем изложении предикатов, которые имеют в качестве возможных значений своих аргументов другие предикаты. Кроме того, они не допускают кванторных операций над предикатами. Так, построенное нами в предыдущем разделе исчисление предикатов является примером теории первого порядка. В исчислениях второго порядка возможны,

например, кванторы по предикатам, то есть выражения вида $\forall x P(x)$. Приложения таких исчислений встречаются гораздо реже.

Теорий первого порядка в принципе достаточно для описания большинства математических теорий. Большинство теорий высших порядков может быть сведено к теориям первого порядка.

Теории первого порядка относят и к прикладным исчислениям, характеризующимся тем, что в них к логическим аксиомам добавляются собственные аксиомы. В собственных аксиомах, как правило, участвуют конкретные (индивидуальные) предикатные буквы (например, $=$, $<$), функциональные буквы ($+$, $\cdot$, ...) и предметные константы (например, натуральные числа, пустое множество в теории множеств и т. п.).

Ценность всякой формальной теории, в конечном счете, определяется ее способностью описывать какие-то объекты и связи между ними. Поэтому одним из первых вопросов для любой теории является вопрос: для описания каких объектов пригодна данная теория? Конечно, если ставить его как общую проблему соответствия научных знаний о мире самому миру, то он не может обсуждаться средствами лишь математики (поскольку математика, в отличие от естественных наук, имеет дело не непосредственно с миром, а с формальным описанием его различных фрагментов) и становится фундаментальной проблемой философии и методологии науки. Эта проблема выходит далеко за рамки нашей дисциплины. Речь у нас идет о более простом случае, когда множество объектов, для которого строится формальная теория, само по себе представляет достаточно строго описанный предмет исследований. Более точно, проблема адекватности формальной теории и описываемых ею объектов рассматривается как математическая задача о соответствии между содержательно построенной теорией и множеством высказываний об этой теории, построенном как формальное значение исчислений. Поэтому большое значение имеет задание значений (смысла) математических выражений (символов, формул и т. д.), то есть интерпретация теории.

4. 1. Формальные аксиоматические теории

Введем основные понятия формальных аксиоматических теорий первого порядка. К этим понятиям относятся понятия алфавита, языка теории, сигнатуры языка, терма и формулы.

Определение 5.1. Алфавитом A языка теории первого порядка называется непустое конечное множество символов, называемых буквами.

Определение 5.2. Словом в алфавите A называется всякая конечная последовательность букв алфавита A . Пустая последовательность букв называется пустым словом.

Пустое слово мы условимся обозначать символом Λ .

Определение 5.3. Два слова $a_1 a_2 \dots a_n$ и $b_1 b_2 \dots b_k$ в алфавите A называются равными, если $k = n$ и $\forall i (a_i = b_i)$, $i = 1, n$. В этом случае пишут $a_1 a_2 \dots a_n = b_1 b_2 \dots b_k$. Число n называется длиной слова.

Определение 5.4. Пусть T некоторая теория. Обозначим через $A(T)$ алфавит этой теории. Множество $A^*(T)$ слов алфавита $A(T)$ называется множеством выражений теории T .

Определение 5.5. Языком теории T называется пара $\langle A(T), A^*(T) \rangle$.

Языки первого порядка обслуживают теории первого порядка. В алфавит всякой теории T первого порядка входят по существу те же символы, что и при построении исчисления предикатов.

Определение 5.6. Множество предикатных букв (или предикатных переменных) $F_1, F_2, \dots, S_1, S_2, \dots$ вместе с переменными высказываниями и значениями предметных переменных (констант) называется сигнатурой языка данной теории и является его специфической частью.

Различные теории первого порядка могут отличаться друг от друга по составу букв в алфавите и сигнатурой.

Дальнейшее описание теории первого порядка T требует индуктивных определений термина и формулы. Термы и формулы это два класса слов множества $A(T)$.

Определение 5.7 (индуктивное определение термина).

1. Предметные переменные и константы являются терминами.
2. Если $r_1, r_2, \dots, r_n$ термы и A символ n -местной операции, то $A^n(r_1, r_2, \dots, r_n)$ терм.

i i 2

3. Никаких других термов, кроме термов, перечисленных в п.п. i и 2, в теории T нет.

Согласно естественному истолкованию (интерпретации) терм это имя некоторого предмета. Кроме переменных и предметных констант, термами являются цепочки, образованные из переменных и предметных констант посредством символов операций, так как в подразумеваемом истолковании термина он понимается как значение некоторой функции.

Определение 5.8 (индуктивное определение формулы).

1. Если A предикат или функция, а $r_1, r_2, \dots, r_n$ термы, то $A(r_1, r_2, \dots, r_n)$ формула. В частности, если A переменный предикат, то $A^n(r_1, r_2, \dots, r_n)$ называется элементарной формулой.

i i 2

2. Если A и B формулы, то $(A \& B), (A \vee B), (A \wedge B), (\neg A)$ формулы.
3. Если A формула, а y предметная переменная, свободная в A или не содержащаяся в формуле A , то выражения $\forall y A, \exists y A$ формулы. При этом A называется областью действия квантора.

4. Никаких других формул, кроме перечисленных в п.п. i-3, нет.

5.2. Аксиомы теории первого порядка T разбиваются на два класса: логические аксиомы и специальные (специальные аксиомы еще называют собственными аксиомами теории T). Логические и специальные аксиомы

Каковы бы ни были формулы A, B и C теории T , следующие формулы являются схемами логических аксиом теории T :

$$A \wedge (B \wedge A), \quad (5.1)$$

$$(A \wedge (B \wedge C)) \wedge ((A \wedge B) \wedge (A \wedge C)), \quad (5.2)$$

$$(B \wedge A) \wedge ((B \wedge A) \wedge B), \quad (5.3)$$

$$\forall x_i A(x_i) \wedge A(t), \quad (5.4)$$

где $A(x_i)$ есть формула теории T , а t терм этой теории, свободный в формуле $A(x_i)$. Отметим, что t может и совпадать с x_i , и тогда мы приходим к аксиоме $\forall x_i A(x_i) \wedge A(x_i)$.

$$\forall x_i (A \wedge B) \wedge (A \wedge \forall x_i B), \quad (5.5)$$

если x_i не входит свободно в формулу A .

Специальные аксиомы не могут быть сформулированы в общем случае, так как они меняются от теории к теории.

Теория первого порядка, не содержащая собственных аксиом, очевидно, представляет собой чисто логическую теорию. Она носит название исчисления предикатов первого порядка. Заметим, что исчисление предикатов, как и исчисление высказываний, может быть построено не обязательно на тех системах аксиом, которые мы сформулировали при изучении этих вопросов. Эти исчисления могут быть построены на других системах аксиом со значительно меньшим их числом. Теория первого порядка является сужением исчисления предикатов на ту или иную заданную область. Аксиоматическая система любой конкретной теории получается присоединением специфических для этой теории аксиом к адекватной полной аксиоматической системе исчисления предикатов. Это вовсе не означает, что теорема исчисления предикатов остается теоремой в рассматриваемой теории, так как язык этой теории, вообще говоря, более ограничен, чем язык исчисления предикатов. Очень часто он располагает лишь

конечным числом различных предикатных и функциональных констант.

К созданию и развитию формальной или неформальной теории могут приводить различные причины. В большинстве случаев теорию разрабатывают для отражения каких-то реальных объектов и их свойств. Тогда цель теории состоит в том, чтобы из множества истинных выражений (формул), принятых за аксиомы данной теории, вывести другие истинные факты (формулы, выражения) этой теории.

5.3. Правила вывода и понятие доказательства в теории первого порядка.

Теорема дедукции

В число правил вывода теории первого порядка входит два правила:

1. Правило заключения (modus ponens):

$\Gamma \vdash S$

2. Правило связывания квантором всеобщности (или (5.6) правило

обобщения):

$\Gamma \vdash UxL$

В аксиоматической теории понятие истинного (5.7) высказывания имеет смысл лишь в связи с возможными приложениями этой теории.

Определение 5.9. Доказательством в теории T называется конечная последовательность $S_1, S_2, \dots, S_k$ высказываний теории T , каждое из которых либо является аксиомой, либо выводится из предыдущих высказываний этой последовательности по логическим правилам вывода.

Определение 5.10. Теоремой или доказуемым высказыванием называется высказывание, являющееся последним высказыванием некоторого доказательства.

Ясно, что любая аксиома является теоремой, причем ее доказательство состоит из одного шага.

В основе многих доказательств лежит уже знакомая нам теорема дедукции, которую мы рассматривали и в исчислении высказываний и в исчислении предикатов, но в теориях первого порядка теорема дедукции имеет место в несколько ослабленной форме. Прежде, чем формулировать и доказывать теорему дедукции для теорий первого порядка, уточним понятие вывода из совокупности формул и докажем вспомогательное утверждение (лемму). Понятие вывода из совокупности формул в исчислении высказываний было определено в п. 3.5 (определение 3.1). Это определение можно обобщить на случай теорий первого порядка с двумя оговорками.

Рассмотрим совокупность формул $\{\Gamma, A\}$ и вывод $F_1, F_2, \dots, F_m$ из этой совокупности. Будем считать, что формула F_k , где $1 < k < m$, зависит от формулы A в этом выводе в двух случаях:

1) формула F_k совпадает с формулой A и она включена в вывод как формула, содержащаяся в совокупности $\{\Gamma, A\}$;

2) формула F_k получена по правилам заключения и связывания квантором всеобщности, сформулированным в начале этого пункта, из формул, предшествующих ей в выводе, из которых, по крайней мере, одна зависит от формулы A .

Лемма. Если в выводе $F_1, F_2, \dots, F_{m-1}, F_m = F$ из совокупности формул $\{\Gamma, A\}$ формула F не зависит от формулы A , то $\Gamma \vdash F$.

Доказательство. Для доказательства воспользуемся методом полной математической индукции по количеству m формул в выводе $F_1, F_2, \dots, F_m$. При $m=1$ формула F есть вывод из совокупности формул $\{\Gamma, A\}$ и не зависит от формулы A , то либо $F \in \Gamma$, либо F доказуемая формула. В обоих случаях $\Gamma \vdash F$. Предположим, что

утверждение леммы справедливо для любого вывода длины $k < m$ и докажем его для вывода длины m . Если при этом $F \in \Gamma$ или F до-

Γ

казуемая формула, то $\Gamma \vdash F$. Если же формула F получена из каких-

то формул, предшествующих ей в выводе, то она не зависит от формулы A , так как по индуктивному предположению не зависят от формулы A все формулы,

предшествующие ей в выводе. Следовательно, и в общем случае $\Gamma \mid \text{---} F$. Лемма доказана.

Теорема дедукции. Пусть Γ, ApFi при этом пусть существует такой вывод F из совокупности формул $\{\Gamma, A\}$, в котором ни при каком применении правила связывания квантором всеобщности к формулам, зависящим в этом выводе от A , не связывается квантором никакая свободная переменная, входящая в A . Тогда $\Gamma \text{ p } A \wedge F$.

Доказательство. Доказательство проведем также методом полной математической индукции по числу m формул в выводе $F_1, F_2, \dots, F_m$. Пусть $F_i, F_2, \dots, F_m \text{---} i, F_m = F$ вывод из совокупности формул $\{\Gamma, A\}$, удовлетворяющий условию теоремы.

1. Пусть $m=1$ и формула F есть вывод из совокупности формул $\{\Gamma, A\}$. Тогда возможны три случая: а) $F \in \Gamma$; б) F доказуемая формула; в) $F = A$. Так как $\Gamma \mid F$ и формула $F \wedge (A \wedge F)$ доказуема, то правилу заключения получим $\Gamma \text{ p } A \wedge F$. Поэтому в случаях а) и б) утверждение теоремы справедливо. В случае в) формула $A \wedge F$ принимает вид $A \wedge A$. Эта формула также доказуема и, следовательно, выводима из формул Γ . Предположим, что утверждение теоремы справедливо для любого вывода длины $k < m$ и докажем его для вывода длины m . Так как $F_1, F_2, \dots, F_{m-1}, F_m = F$ есть вывод из совокупности формул $\{\Gamma, A\}$, то возможны следующие пять случаев: а) $F \in \Gamma$; б) F доказуемая формула; в) $F = A$; г) формула F получается по правилу заключения из предшествующих ей в выводе формул F_i и F_j ($i < j < m$); д) формула F получается из формулы F_i , предшествующей ей в выводе ($i < m$) по правилу связывания квантором всеобщности.

Для первых трех случаев доказательство полностью совпадает с доказательством, проведенным в п.1 при $m=1$. Рассмотрим случай г). Так как здесь формула F получается из двух формул F_i и F_j при $i < j < m$, то формула F_j должна иметь вид $F_i \wedge F$, при этом

$$\Gamma \mid A \wedge F_i, \quad (5.8)$$

$$\Gamma \text{ p } A \wedge (F_i \wedge F). \quad (5.9)$$

Используя логическую аксиому (5.2), получим

$$\Gamma (A \wedge (F_i \wedge F)) \wedge ((A \wedge F_i) \wedge (A \wedge F)). \quad (5.10)$$

Из формул (5.10), (5.9), (5.8) по правилу сложного заключения (см. п.3.5) получим $\Gamma \text{ p } A \wedge F$.

Рассмотрим последний случай д). Пусть в выводе из совокупности формул $\{\Gamma, A\}$ есть формула F_i ($i < m$) такая, что формула $F = \forall x_i F_i$. По индуктивному предположению имеет место вывод

(5.8) и либо F_i не зависит от формулы A , либо переменная x_i в формуле A связана квантором всеобщности. В первом случае согласно доказанной выше лемме $\Gamma \mid F_i$. Применяя правило (5.5), получим

$\Gamma \text{---} \forall x_i F_i$, то есть ΓF . Используя схему аксиом (5.1), получим

$\Gamma \mid F \wedge (A \wedge F)$ и, следовательно, $\Gamma \text{ p } A \wedge F$.

Пусть, наконец, переменная x_i не является свободной в формуле A . Тогда по схеме аксиом (5.5) $\forall x_i (A \wedge F_i) \wedge (A \wedge \forall x_i F_i)$. Так как $\Gamma \text{ p } A \wedge F$; , то по правилу (5.7) $\Gamma \mid \forall x_i (A \wedge F)$, а по правилу *modus ponens* $\Gamma \mid A \wedge \forall x_i F_i$, то есть $\Gamma \text{ p } A \wedge F$. На этом доказательство теоремы дедукции завершается.

Условия теоремы дедукции слишком громоздки, поэтому на практике часто бывают полезными более слабые следствия. Следствие 1. Если $\Gamma, A \text{---} F$ и существует вывод, построенный без применения правила (5.5) к свободным переменным формулы A , то $\Gamma \text{ p } A \wedge F$.

Следствие 2. Если формула A замкнута и $\Gamma, A \mid \text{---} F$, то $\Gamma \text{ p } A \wedge F$.

Рассмотрим пример. Убедиться, что $\text{---} \forall x \forall y A \wedge \forall y \forall x A$.

Действительно, формула $\forall x \forall y A$, как гипотеза, доказуема. По

схеме аксиом (5.4) доказуемой является формула $\forall x \forall y A \wedge \forall y \forall x A$. Тогда по правилу *modus ponens* получим $\text{---} \forall y A$ и, снова применяя схему аксиом (5.4), получим доказуе-

мость формулы $\forall y A \wedge A$. Снова применяя правило *modus ponens*, получим $\text{p } A$, а по правилу (5.7) сначала получим $\sim \forall x A$, за-

тем $\forall x \forall y A$. В результате всех этих рассуждений мы имеем $\forall x \forall y A \rightarrow \forall y \forall x A$, при этом в построенном выводе мы ни при одном применении правила (5.7) не связывается свободная переменная формулы $\forall x \forall y A$, поэтому, на основании следствия 1,

$\vdash \forall x \forall y A \wedge \forall y \forall x A$, что и требовалось.

5. 4. Понятия интерпретации и модели теории. Изоморфизм интерпретаций

Под интерпретацией теории T понимают всякую систему, включающую в себя:

- 1) Непустое множество M , называемое областью интерпретации.
- 2) Какое-либо соответствие f , которое каждому элементу языка теории T сопоставляет единственный элемент множества M : $\langle A(T), A^*(T) \rangle \rightarrow M$.

В частности, каждой предикатной букве $A_n \in A(T)$, $A^*(T) \rightarrow f$ ставит в соответствие некоторые n -местное отношение во множестве M ; каждой функциональной букве $f_n \in A(T)$, $A^*(T) \rightarrow f$ ставит-

ся в соответствие некоторая n -местная операция в M , то есть функция $M^n \rightarrow M$; каждой предметной переменной x некоторый элемент из множества M . Всякая формула без свободных переменных представляет собой высказывание, которое истинно или ложно, а всякая формула со свободными переменными выражает некоторое отношение на области интерпретации. Это отношение может быть истинным для одних значений переменных из области интерпретации и ложно для других.

Пусть задана некоторая интерпретация теории T с областью M , а G множество всех счетных последовательностей элементов из M .

С помощью последовательности $S = (b_1, b_2, \dots, b_n, \dots) \in G$ определим индуктивно функцию S^* , отображающую все термы во множество M , следующим образом:

- 1) если терм t предметная постоянная, то $S^*(t) = b_i$;
- 2) если терм t предметная константа, то $S^*(t)$ совпадает с интерпретацией этой константы во множестве M ;
- 3) если f^n функциональная буква, интерпретируемая как операция g в M , и $t_1, t_2, \dots, t_n$ термы, то $S^*(f^n(t_1, t_2, \dots, t_n)) = g(S^*(t_1), S^*(t_2), \dots, S^*(t_n))$.

Проще говоря, для любой последовательности $S = (b_1, b_2, \dots, b_n, \dots)$ и для любого терма t $S^*(t)$ есть элемент множества M , который получается в результате подстановки при каждом i элемента b_i на места всех вхождений переменной x_i в терм t и затем выполнение всех операций интерпретации, соответствующий функциональным буквам терма t .

Например, пусть терм $t = f_2(x_3, f_1(x_1, a_1))$; $M = Z$ область интерпретации; f_1 будем интерпретировать как операцию сложения целых чисел, f_2 как операцию умножения, a_1 как целое число 15. Тогда для всякой последовательности $S = (b_1, b_2, \dots, b_n, \dots)$ целых чисел $S^*(t)$ представляет собой целое число $b_3 \cdot (b_1 + 15)$.

Определение 5.11 (индуктивное определение выполненной формулы).

1. Если A есть элементарная формула A_j ($j = 1, 2, \dots, n$) (здесь верхний индекс опущен) и B_j соответствующее формуле A отношение в интерпретации теории, то формула A считается выполненной на последовательности S тогда и только тогда, когда $B(S^*(t_1), \dots, S^*(t_n))$ принадлежит отношению B_j .
2. Формула A выполнена на последовательности S тогда и только тогда, когда формула A не выполнена на S . Формула $A \wedge B$ выполнена на S тогда и только тогда, когда формула A не выполнена на S или когда формула B выполнена на S (ср. определение 2.27 операции импликации предикатов).
3. Формула $\forall x_i A$ выполнена на S тогда и только тогда, когда формула A выполнена на любой последовательности из G , отличающейся от S не более чем своим i -ым компонентом.

Не следует путать понятия выполнимой и выполненной формул.

Из определения выполненной формулы ясно, что формула A выполнена на последовательности $S = (b_1, b_2, \dots, b_n, \dots)$ тогда и только тогда, когда подстановка при каждом i символа, представляющего b_i , на места всех свободных вхождений x , в формулу A приводит к истинному в данной интерпретации предложению.

Определение 5.12. Формула A называется истинной в данной интерпретации тогда и только тогда, когда она выполнена на всякой последовательности из G (см. 4). Эта формула называется ложной в той же интерпретации, если она не выполнена ни на одной последовательности из G .

Определение 5.13. Интерпретация I называется моделью для множества формул Γ , если любая формула их множества Γ истинна в данной интерпретации I . Интерпретация I называется моделью теории T , если она является моделью множества всех теорем теории T , то есть, если всякая формула в T , доказуема в T , истинна в интерпретации I .

Понятие модели теории T можно рассматривать как интерпретацию языка этой теории. А именно: имея некоторую теорию T , мы приписываем первоначальным понятиям этой теории некоторый новый смысл. Если некоторая совокупность предметов и отношений между ними, выбранных в качестве первоначальных понятий аксиоматической теории, то есть в качестве ее интерпретации, удовлетворяет всем аксиомам теории, то она называется моделью данной аксиоматической теории.

Определение 5.14. Интерпретация I_1 данной теории T первого порядка изоморфна интерпретации I_2 той же теории, если существует биективное отображение g (называемое изоморфизмом) области M_1 интерпретации I_1 на область M_2 интерпретации I_2 такое, что:

1. Если A_{1j} и A_{2j} интерпретации предикатной буквы A_j соответственно в интерпретациях I_1 и I_2 , то каковы бы ни были $b_1, b_2, \dots, b_n \in M_1$ $A_{1j}(b_1, b_2, \dots, b_n)$ выполнено тогда и только тогда, когда выполнено $A_{2j}(g(b_1), g(b_2), \dots, g(b_n))$.
2. Если f_{1j} и f_{2j} интерпретации функциональной буквы f_j соответственно в I_1 и I_2 , то $\forall b_1 \forall b_2 \dots \forall b_n \in M_1 f_{1j}(b_1, b_2, \dots, b_n) = f_{2j}(g(b_1), g(b_2), \dots, g(b_n))$.
3. Если a_{1j} и a_{2j} интерпретации предметной постоянной соответственно в I_1 и I_2 , то $a_{1j} = g(a_{2j})$.

Ясно, что если интерпретации I_1 и I_2 изоморфны (пишут $I_2 = I_1$), то области этих интерпретаций имеют одинаковые мощности.

5. 5. Примеры аксиоматических теорий первого порядка со специальными аксиомами

Пример 5.1. Во многих теориях, которые могут быть аксиоматизированы как теории первого порядка, часто используется понятие равенства и определяющие его специальные аксиомы:

1. $\forall x (x = x)$.
2. Если x, y, z — различные предметные переменные и $F(z)$ формула, то $\forall x \forall y (x = y \wedge F(x) \rightarrow F(y))$.

Всякая теория, в которой присутствуют специальные аксиомы 1 и 2, называется теорией с равенством. Из аксиом 1 и 2 выводимы основные свойства равенства рефлексивность, симметричность и транзитивность:

- 1) $x = x$ для любого термина t ;
- 2) $x = y \rightarrow y = x$;
- 3) $x = y \wedge (y = z \rightarrow x = z)$.

Указанные свойства равенства имеют большое значение для прикладных исчислений. Из свойств равенства следует, что отношение равенства объектов в любой теории с равенством является отношением эквивалентности, которое позволяет разбить множество всех объектов, на котором это отношение определено, на попарно не пересекающиеся классы. Аксиома 2 теории с равенством выражает тождественность (неотличимость) равных элементов. Смысл этого понятия состоит в том, что если x и y тождественны, то они обозначают один и тот же объект. При этом каждый из классов, на которые разбивается множество всех объектов теории с равенством по отношению « $=$ »,

состоит только из одного объекта. Если рассматривать исчисление предикатов первого порядка с точки зрения наивной теории множеств, то равенство, которое определяется аксиомами 1 и 2, представляет собой именно теоретико-множественное тождество.

Пример 5.2. Теория частичного упорядочения. Пусть теория T содержит одну предикатную букву A_1^2 и не содержит функциональных букв и предметных констант. Вместо формул $A^2(x_1, x_2)$ и $A^2(x, x)$ обычно пишут $x_1 < x_2$ и $x_1 < x_2$.

1 1 2

Кроме того, пусть теория T содержит две специальные аксиомы:

1. $\forall x_1(x_1 < x_1)$ иррефлексивность;
2. $\forall x_1 \forall x_2 \forall x_3((x_1 < x_2) \& (x_2 < x_3) \rightarrow (x_1 < x_3))$ транзитивность.

Всякая модель этой теории называется частично упорядоченной структурой.

Пример 5.3. Теория групп. Пусть теория T содержит одну предикатную букву A^2 , одну функциональную букву f_1^2 и одну пред-

1 1

метную константу a_1 . Пользуясь принятыми в алгебре обозначениями, будем писать:

- $t = s$ вместо $A_1^2(t, s)$,
- $t + s$ вместо $f_1^2(t, s)$,
- 0 вместо a_1 .

Специальными аксиомами теории T здесь являются следующие формулы:

- a) $\forall x_1 \forall x_2 \forall x_3(x_1 + (x_2 + x_3)) = ((x_1 + x_2) + x_3)$ ассоциативность;
- b) $\forall x_1(0 + x_1 = x_1)$ свойство нейтрального элемента;
- c) $\forall x_1 \exists x_2(x_1 + x_2 = 0)$ существование противоположного элемента;
- d) $\forall x_1(x_1 = x_1)$ рефлексивность равенства;
- e) $\forall x_1 \forall x_2((x_1 = x_2) \rightarrow (x_2 = x_1))$ симметричность равенства;
- f) $\forall x_1 \forall x_2 \forall x_3(((x_1 = x_2) \& (x_2 = x_3)) \rightarrow (x_1 = x_3))$ транзитивность равенства;
- g) $\forall x_1 \forall x_2 \forall x_3((x_2 = x_3) \rightarrow (x_1 + x_2 = x_1 + x_3) \& (x_2 + x_1 = x_3 + x_1))$

подстановочность равенства.

Всякая модель этой теории называется группой. Если в группе истинна формула

$\forall x_1 \forall x_2(x_1 + x_2 = x_2 + x_1)$,

то группа называется абелевой или коммутативной.

Пример 5.4. Аксиоматическое построение натурального ряда чисел, предложенное итальянским математиком Д. Пеано. Первоначальные понятия: непустое множество N , называемое множеством натуральных чисел; отношение следования «'» и выделенный элемент 1.

Специальные аксиомы:

1. $\forall x \in N(x' \in N)$;
2. $\forall x \forall y(x = y \rightarrow x' = y')$;
3. $\forall x \forall y(x' = y' \rightarrow x = y)$;
4. пусть $m \in N$. Тогда $((1 \in M) \& \forall x(x \in M \rightarrow x' \in M)) \rightarrow (M = N)$.

Аксиома 4 носит название аксиомы математической индукции. На ней основан принцип математической индукции, который лежит в основе метода доказательства теорем, называемого методом полной математической индукции. Для того чтобы доказать, что некоторым свойством $P(x)$ обладают все натуральные числа, доказывают сначала, что им обладает число 1, а затем доказывают его для произвольного числа x' при «индуктивном предположении», что число x свойством $P(x)$ уже обладает. В силу аксиомы 4 множество чисел, обладающих свойством $P(x)$, должно содержать множество всех натуральных чисел.

Пример 5.5. Аксиоматическое построение системы натуральных чисел.

Предположим, что натуральный ряд чисел уже построен так, что выполняются аксиомы Пеано 1-4. Каждой паре натуральных чисел x, y можно единственным образом поставить в соответствие натуральное число, обозначаемое через $x + y$, так, чтобы

оказались выполненными следующие аксиомы:

1. $\forall x \in \mathbb{N} (x' = x + 1)$;
2. $\forall x \in \mathbb{N} \forall y \in \mathbb{N} ((x + y)' = x + y')$.

Число $x + y$ называется суммой натуральных чисел x и y , а соответствующее отображение $\mathbb{N}^2 \rightarrow \mathbb{N}$ операцией сложения.

Каждой паре натуральных чисел x, y можно единственным образом поставить в соответствие натуральное число, обозначаемое через $x \blacksquare y$ или через $x \cdot y$, так, чтобы выполнялись следующие две аксиомы:

3. $\forall x \in \mathbb{N} (x \blacksquare 1 = x)$;
4. $\forall x \in \mathbb{N} \forall y \in \mathbb{N} (x \blacksquare y' = x \blacksquare y + x)$.

Число $x \blacksquare y$ называется произведением чисел x и y , а соответствующее отображение $\mathbb{N}^2 \rightarrow \mathbb{N}$ операцией умножения.

Числовая система $\mathbb{N} = \langle \mathbb{N}, 1, +, ' \rangle$, операции сложения и умножения в которой удовлетворяют аксиомам 1-4 (разумеется, и аксиомам Пеано), называется системой натуральных чисел.

На основе специальных аксиом системы натуральных чисел, логических аксиом теорий первого порядка с использованием метода полной математической индукции и правил вывода можно доказать большинство свойств операций сложения и умножения натуральных чисел (коммутативность и ассоциативность этих операций, дистрибутивный закон умножения относительно сложения и др.), ввести на множестве $\mathbb{N}$ отношение линейного порядка и т.д. Также можно доказать следующую важную теорему: каждое непустое множество натуральных чисел содержит наименьшее число, то есть такое число, которое меньше всех остальных чисел этого множества.

Пример 5.6. Аксиоматическое построение системы целых чисел.

Числовая система $\mathbb{Z} = \langle \mathbb{Z}, 1, 0, +, \cdot \rangle$ целых чисел является естественным расширением системы натуральных чисел, поэтому включает в себя систему $\mathbb{N}$, а, следовательно, в ней выполняются все 4 аксиомы Пеано для натурального ряда. Эта система включает в себя два выделенных элемента: 0 нейтральный элемент относительно операции сложения, и 1 нейтральный элемент относительно операции умножения (см. определение 9, п.1.6). Обозначим символом $'x$ число, предшествующее числу x , то есть число, для которого выполняется условие $('x)' + 1 = x$. Обобщив аксиомы сложения и умножения для натуральных чисел, мы вместе с аксиомами Пеано получим аксиоматику системы целых чисел, а именно, для любых чисел $x, y \in \mathbb{Z}$:

1. $x + 0 = x$; 6. $x \cdot 0 = 0$;
2. $x + 1 = x'$; 7. $x \cdot 1 = x$;
3. $x + (-1) = 'x$; 8. $x \cdot y' = x \cdot y + x$;
4. $x + y' = (x + y)'$; 9. $x \cdot (-1) = -x$;
5. $x + 'y = (x + y)$; 10. $x \cdot 'y = x \cdot y \cdot x$.

Во множестве $\mathbb{Z}$ целых чисел есть возможность обобщить аксиому математической индукции: если некоторое подмножество M с $\mathbb{Z}$ содержит 1 и вместе с некоторым целым числом x оно содержит предшествующее ему число $'x$ и число x' , непосредственно следующее за числом x , то $M = \mathbb{N}$. Эта аксиома позволяет доказывать многие свойства целых чисел и операций над ними обобщенным методом полной математической индукции. В частности, можно доказать, что множество всех целых чисел относительно операций сложения и умножения образует ассоциативно-коммутативное кольцо с единицей (см. определения 15 и 16 п.1.7).

Пример 5.7. Аксиоматическое построение системы рациональных чисел.

Система $\mathbb{Q} = \langle \mathbb{Q}, +, \blacksquare, 0, 1 \rangle$ рациональных чисел является расширением системы целых чисел, то есть она включает в себя систему целых чисел со всеми ее свойствами, но в системе $\mathbb{Q}$ однозначно разрешимо уравнение $a \blacksquare x = b$ для любых целых чисел a и b , где $a \neq 0$. Другими словами, во множестве рациональных чисел кроме операций сложения, вычитания и умножения однозначно выполняется операция деления, кроме деления на

нуль. Аксиоматически теорию рациональных чисел можно построить следующим образом.

За основное (неопределяемое) понятие принимается понятие «рациональное число», которое представляется упорядоченной парой

целых чисел (x, y) , где $y \neq 0$. Эта упорядоченная пара записывается в виде $\frac{x}{y}$, причем $\frac{x}{y}$ поставленная черта между числами x и y есть символ, не означающий пока никакой математической операции. Рациональное число x называется обыкновенной дробью и читается: x к y . Вся теория рациональных чисел можно построить на основе следующих пяти аксиом (при этом система целых чисел считается построенной).

1. Аксиома равенства. Рациональные числа $\frac{a}{b}$ и $\frac{c}{d}$ считаются равными тогда и только тогда, когда $ad = bc$.

2. Аксиома неравенства. Рациональные числа $\frac{a}{b}$ и $\frac{c}{d}$ не равны, если

или $ad > bc$, причем $a > c$, если $ad > bc$, и $a < c$,
если $ad < bc$. b, d b, d

$\frac{a}{b} < \frac{c}{d}$ $a < c$ $a < c$ кратко обозначают $\frac{a}{b} < \frac{c}{d}$ через $\frac{a}{b} < \frac{c}{d}$

$\frac{a}{b} < \frac{c}{d}$ (или $\frac{a}{b} > \frac{c}{d}$).

3. Аксиома суммы. Суммой двух рациональных чисел

$\frac{a}{b} + \frac{c}{d}$ называется число $\frac{ad + bc}{bd}$, для которого выполняется аксиома $\frac{a}{b} + \frac{c}{d} = \frac{ad + bc}{bd}$

На практике аксиома суммы означает, что при сложении двух обыкновенных дробей их сначала приводят к общему знаменателю (то есть находят наименьшее общее кратное знаменателей), числители домножают на недостающие множители и лишь после этого складывают числители.

Заметим, что часто учащиеся, не зная аксиомы суммы, допускают грубую ошибку при сложении обыкновенных дробей, складывая отдельно числители и знаменатели этих дробей.

4. Аксиома произведения. Произведением двух рациональных чисел $\frac{a}{b}$ и $\frac{c}{d}$ называется число $\frac{ac}{bd}$, для которого выполняется

аксиома $\frac{a}{b} \cdot \frac{c}{d} = \frac{ac}{bd}$

5. Аксиома, устанавливающая связь между множествами рациональных и целых чисел: $\frac{a}{1} = a$, где $a \in \mathbb{Z}$.

1

Множество Q рациональных чисел, как и множество Z целых чисел, является счетным, то есть оно равномощно множеству натуральных чисел (см. п.1.1). Из аксиомы 5 вытекает, что множество целых чисел является собственным подмножеством рациональных чисел, при этом операции над целыми числами, рассматриваемые как элементы множества Q , имеют тот же смысл, что и во множестве Z , но во множестве Q стала выполнимой операция деления, которая была не всегда выполнимой во множестве Z . Рациональное число $\frac{x}{y}$ теперь можно рассматривать как частное от деления целых чисел x и y , а дробную черту как знак деления. Несложно доказать, что множество всех рациональных чисел относительно операций сложения, вычитания, умножения и деления (кроме деления на нуль) образует числовое поле (см. п.1.6), причем это поле является минимальным числовым полем, содержащим кольцо целых чисел. Нейтральным элементом относительно операции сложения является число $0 = \frac{0}{1}$, а относительно операции умножения число $1 = \frac{1}{1}$.

1

1

Пример 5.8. Аксиоматическое построение системы аддитивно-скалярных величин.

Понятие величины является одним из основных понятий математики и других

научных дисциплин. Это понятие с большим трудом поддается четкому описанию из-за того, что в различных научных дисциплинах и даже в рамках одной и той же дисциплины ему придают различный смысл. Более того, иногда отождествляют термины «величина» и «значение величины», что часто вызывает недоразумения.

Обычно имеют дело не с отвлеченным понятием величины, а с определенной системой однородных величин, которой уже можно дать совершенно четкое, чаще всего аксиоматическое, определение. Наиболее употребительной является система скалярных величин (длина, площадь, объем, масса, температура, время и т.п.), из которой выделяют систему положительных аддитивно-скалярных величин. Рассматривают также и другие системы величин: векторные, тензорные, строковые, графические, звуковые, массивы и другие величины. К аксиоматическому определению системы аддитивно-скалярных величин можно подойти, рассматривая и обобщая свойства длин отрезков или площадей плоских фигур. А.Н. Колмогоров предложил следующее определение системы аддитивно-скалярных величин.

Системой $\langle S, +, < \rangle$ положительных аддитивно-скалярных величин называется непустое множество S , в котором определена алгебраическая операция сложения и установлено бинарное отношение порядка так, что выполняются следующие аксиомы:

1. Каковы бы ни были величины $x, y \in S$, имеет место одно и только одно из трех отношений между ними: или $x = y$, или $x < y$, или $y < x$ (закон трихотомии).
2. Отношение « $<$ » на множестве S транзитивно: для любых величин $x, y, z \in S$, если $x < y$ и $y < z$, то $x < z$.
3. Для любых величин $x, y \in S$ существует однозначно определенная величина $z \in S$ такая, что $z = x + y$ (то есть операция сложения на множестве S определена однозначно).
4. Операция сложения на множестве S коммутативна: для любых величин $x, y \in S$ $x + y = y + x$. Операция сложения на множестве S ассоциативна: для любых величин $x, y, z \in S$ $(x + y) + z = x + (y + z)$.
5. Операция сложения на множестве S обладает свойством монотонности: для любых величин $x, y \in S$ $x < y \Rightarrow x + z < y + z$.
6. Если $y < x$, то существует однозначно определенная величина $z \in S$ такая, что $y + z = x$ (возможность вычитания во множестве S из большей величины меньшей).
7. Для любой величины $x \in S$ и любого натурального числа n существует величина $y \in S$ такая, что $x = n \blacksquare y$ (возможность неограниченного деления величины на равные доли).
8. Для любых величин $x, y \in S$ существует натуральное число n такое, что $x < n \blacksquare y$.
9. Если две последовательности величин $a_1 < a_2 < \dots < a_n < b_n < \dots < b_2 < b_1$ из множества S обладают тем свойством, что для любой величины $c \in S$ при достаточно большом номере n $b_n < c < a_n$, то существует единственная величина $x \in S$, которая больше всех величин a_i и меньше всех величин b_i ($i = 1, n$).

Аксиома 9 называется аксиомой Архимеда, а аксиома 10 аксиомой непрерывности Кантора.

Пример 5.9. Аксиоматическое построение системы действительных чисел.

Система действительных чисел, как известно из школьного курса математики, получается из системы рациональных чисел путем присоединения к ней множества всех иррациональных чисел, то есть всех непериодических десятичных дробей. Это обстоятельство характеризует систему действительных чисел как непрерывное числовое поле, содержащее в качестве подполя поле рациональных чисел. Система рациональных чисел образует так называемое архимедовски упорядоченное поле, так как в нем выполняется аксиома Архимеда. Таким образом, существенной аксиомой системы действительных чисел является аксиома непрерывности, которая не выполняется для

системы рациональных чисел. Эту аксиому можно сформулировать по-разному. Мы сформулируем ее так, как предложил выдающийся математик 19-го века К. Вейерштрасс.

Аксиома непрерывности. Для всякой системы $[A_i B_i]$, $i = 1, 2, 3, \dots$ вложенных отрезков существует хотя бы одна точка, которая принадлежит всем отрезкам данной системы. Если значения длин вложенных отрезков стремятся к нулю при i стремящемся к бесконечности, то такая точка единственная.

Резюмируя все изложенное, можно сказать, что система действительных чисел это минимальное, архимедовски упорядоченное, непрерывное числовое поле, содержащее в качестве подполя поле рациональных чисел.

Пример 5.10. Аксиоматическое построение теории множеств.

Многие парадоксы теории множеств (см. вводную часть к разделу 1) показали, что интуитивные определения и построения в математике не могут рассматриваться как основополагающие. Возникла необходимость определить множество так, чтобы при теоретических построениях не прибегать к привлечению интуитивных понятий. Выход был найден в аксиоматическом построении теории множеств. К настоящему времени сложилось несколько аксиоматических систем теории множеств. Одним из примеров такой системы является система аксиом, предложенная немецким математиком Э.Ф. Цермело в начале 20-го века. Одну из его аксиом мы сформулировали в п.1.5. Система аксиом Цермело состоит из семи аксиом.

Аксиома 1 (аксиома объемности). Если все элементы множества A принадлежат множеству B , а все элементы множества B принадлежат множеству A , то $A=B$ (ср. интуитивный принцип объемности, п.1.1).

Аксиома 2 (аксиома пары). Для произвольных объектов a, b существует множество M , единственными элементами которого являются эти объекты, то есть $M=\{a, b\}$.

Аксиома 3 (аксиома объединения). Для произвольных множеств A и B существует единственное множество C , элементами которого являются все элементы множества A и все элементы множества B и которое никаких других элементов не содержит.

Аксиома 4 (аксиома степени). Для любого множества X существует множество всех его подмножеств (X) .

Аксиома 5 (аксиома выделения). Пусть X произвольное множество, $x \in X$ и $\varphi(x)$ функция со значениями во множестве $\{0, 1\}$. Тогда существует множество Z , для которого $x \in Z$ тогда и только тогда, когда $x \in X$ и $\varphi(x)=1$.

Функции типа $\varphi(x)$ называются бинарными. В алгебре логики бинарные функции рассматриваются как предикаты. В таком случае их значения интерпретируются соответственно как «ложь» и «истина».

Аксиома 6 (аксиома бесконечности). Существует, по крайней мере, одно бесконечное множество натуральных ряд чисел.

Аксиома 7 (аксиома выбора). Для каждого семейства U непустых непересекающихся множеств существует множество B , имеющее один и только один общий элемент с каждым из элементов $X \in U$.

Некоторые комментарии к аксиоме выбора были приведены в п.1.6.

Пример 5.11. Теория векторных пространств.

Для каждого натурального числа $n > 0$ определим теорию E_n элементарную теорию векторного пространства размерности n . Эта теория строится на языке векторов. Специальные аксиомы E_n распадаются на две группы. Первая группа это просто аксиомы поля действительных чисел с обычными операциями сложения и умножения, относящаяся к переменным $x, y, z, \dots$, формулам и термам теории действительных чисел. Вторая группа специальных аксиом известные аксиомы линейного пространства: 1) $a + 0 = a$;

- 2) $a + b = b + a$;
- 3) $(a + b) + c = a(b + c)$;
- 4) $3 b (a + b = 0)$;
- 5) $x \bullet (a + b) = x \blacksquare a + x \blacksquare b$;

$$\begin{aligned}
& (x + y) \cdot a = x \cdot a + y \cdot a; \\
6) & \quad x \cdot (y \cdot a) = (x \cdot y) \cdot a; \\
7) & \quad a \cdot 1 = a; \\
8) & \quad \exists a_1 \exists a_2 \dots \exists a_n \forall x_1 \forall x_2 \dots \forall x_n (x_1 \cdot a_1 + \dots + x_n \cdot a_n = 0 \wedge x_1 = 0 \& x_2 = 0 \& \dots \& x_n = 0); \\
9) & \quad \forall a_1 \dots \forall a_{n+1} \exists x_1 \dots \exists x_{n+1} ((x_1 \neq 0 \vee \dots \vee x_{n+1} \neq 0) \& x_1 \cdot a_1 + \dots + x_{n+1} \cdot a_{n+1} = 0).
\end{aligned}$$

В приведенных аксиомах начальными буквами латинского алфавита обозначены векторы, символом 0 обозначен нулевой вектор.

Последние две аксиомы как раз выражают то обстоятельство, что размерность подразумеваемого пространства равна n .

Естественной моделью теории E_n является n -мерное линейное векторное пространство над полем действительных чисел.

Выводы

Всякую конкретную математическую теорию T можно перевести на язык подходящей формальной системы S таким образом, что каждое осмысленное (ложное или истинное) предложение теории T выражается некоторой формулой системы S . Однако К. Гедель в 30-е годы XX в. показал, что такой метод формализации позволяет построить не все положительное содержание математических теорий, а такие принципиальные вопросы, как проблема непротиворечивости математических теорий, не всегда может быть решена в форме доказательств соответствующих утверждений о формализующих эти теории формальных системах. Поэтому существуют определенные границы возможностей аксиоматического метода в том виде, который он приобрел в рамках Гильбертовского формализма (именно Гильберт впервые предложил такой формализм).

Вопросы и упражнения для самопроверки

- 5.1. Определить понятия алфавита языка теории первого порядка, слова в этом алфавите, равенства слов, множества выражений теории.
- 5.2. Дать определение сигнатуры языка теории первого порядка.
- 5.3. Сформулировать индуктивное определение терма и формулы теории первого порядка.
- 5.4. Привести примеры содержательных математических теорий первого порядка.
- 5.5. Сформулировать правило заключения и правило связывания квантором всеобщности теории первого порядка.
- 5.6. Что называется доказательством в теории первого порядка? А теоремой?
- 5.7. Что понимают под интерпретацией языка теории первого порядка?
- 5.8. Доказать основные свойства равенства рефлексивность, симметричность и транзитивность:

- 1) $t = t$ для любого терма t ;
- 2) $(x = y) \wedge (y = x)$;
- 3) $(x = y) \wedge (y = z) \rightarrow (x = z)$;
- 5.9. Показать, что $\forall x \forall y (x < y) \& (y < x)$.

Решение. Если бы это было бы не так, то по аксиоме 2) теории T должна быть истинной формула $\forall x (x < x)$, что противоречит аксиоме 1). Доказанное свойство частичного порядка показывает, что в никакой интерпретации теории T никакой элемент не может одновременно предшествовать какому-нибудь другому элементу и следовать за ним.

Если к аксиомам 1) и 2) теории частичного упорядочения добавить еще одну аксиому 3): $\forall x \forall y (x < y \vee y < x)$, то порядок $\leq$

сигнатурой « $<$ » называется полным, а множество, на котором он задан, называется вполне упорядоченным или цепью. В любой цепи из любых двух различных элементов один обязательно предшествует другому. Примером может служить стандартное

отношение « $<$ » на множествах натуральных, целых, рациональных и действительных чисел. На множестве $\mathbb{R}$ действительных чисел можно определить и другие отношения строгого линейного порядка. Например, $\forall x \forall y (x < y) \vee (xy \wedge 0 \wedge x < y) \vee (x \wedge 0 \wedge y = 0)$

5.10. Доказать для заданного таким образом отношения порядка выполнение аксиом 1) -3).

Зададим на множестве $N^2 = N \times N$ множество упорядоченных пар натуральных чисел (x, y) , для которых разность $x - y$ является положительным четным числом. Доказать, что соответствующее бинарное отношение есть отношение частичного порядка.

5.11. Задать отношение полного линейного порядка на множестве $\{a, b, c\}$. Сколькими способами это можно сделать?

5.12. Привести пример такого отношения линейного порядка на заданном множестве, что в этом множестве не найдется элемента, предшествующего остальным элементам.

5.13. Определим бинарное отношение « $\sim$ » в некоторой интерпретации I теории P как отношение, обладающее следующими тремя свойствами:

- 1) $\forall x (x \sim x)$ рефлексивность;
- 2) $\forall x \forall y (x \sim y \wedge y \sim x \rightarrow x = y)$ антисимметричность;
- 3) $\forall x \forall y \forall z (x \sim y \wedge y \sim z \rightarrow x \sim z)$ транзитивность.

Такое отношение называется отношением нестрогого порядка.

Возьмем в качестве множества I множество $\mathbb{Z}$ всех целых чисел, и пусть $x \sim y$ означает, что число x делится нацело на целое число $y \wedge 0$. Доказать, что отношение делимости является отношением частичного порядка.

5.14. Образуем ли данное множество относительно указанной операции группу. Если да, то будет ли она абелевой. Если нет, то указать, какие аксиомы группы не выполняются. Множество всех взаимно однозначных отображений множества M на себя, рассматриваемое вместе с операцией суперпозиции отображений.

1.1.1. . Множество неотрицательных целых чисел относительно сложения.

1.1.2. . Множество $\mathbb{Z}$ всех целых чисел относительно операции сложения.

1.1.3. . Множество $\mathbb{R}^2$ всех векторов плоскости относительно операции сложения.

1.1.4. . Множество всех четных чисел относительно вычитания.

1.1.5. . Множество всех положительных действительных чисел, если операция определена так: $a * b = a^b$.

1.1.6. . Множество всех векторов n -мерного арифметического пространства относительно операции покомпонентного сложения векторов. При этом координаты векторов являются действительными числами.

1.1.7. . Множество всех вещественных матриц одного и того же размера относительно операции сложения.

1.1.8. . Множество всех корней n -ой степени из единицы (действительных и мнимых) относительно умножения.

1.1.9. . Множество всех положительных целых чисел относительно умножения.

1.1.10. . Множество всех нечетных чисел относительно сложения.

1.1.11. . Множество всех рациональных чисел относительно умножения.

1.1.12. . Множество всех квадратных матриц порядка n с действительными элементами относительно умножения.

1.1.13. . Множество действительных многочленов степени n относительно сложения многочленов.

1.1.14. . Множество всех поворотов с общим центром на плоскости относительно последовательного их выполнения (композиции).

1.1.15. . Множество всех квадратных матриц порядка n с целыми элементами и определителем, равным единице, относительно умножения матриц.

5.15. Выяснить, какие из следующих множеств являются кольцами (но не полями) и какие полями относительно указанных операций. Если операции не указаны, то

подразумеваются сложение и умножение.

5.15.1. Целые числа.

5.15.2. Четные числа.

5.15.3. Целые числа, кратные данному числу m (рассмотреть, в частности случай $m=0$).

5.15.4. Рациональные числа.

5.15.5. Действительные числа.

5.15.6. Комплексные числа.

5.15.7. Числа вида $a + b\sqrt{2}$ с целыми a и b .

5.15.8. Числа $a + b\sqrt{2}$ с рациональными a и b .

5.15.9. Комплексные числа вида $a + bi$ с рациональными a и b .

5.15.10. Квадратные матрицы порядка n с целыми элементами относительно сложения и умножения матриц.

5.15.11. Квадратные матрицы порядка n с действительными элементами относительно сложения и умножения матриц.

5.15.12. Многочлены от одной переменной с целыми коэффициентами относительно обычных операций сложения и умножения матриц.

5.15.13. Все матрицы вида $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ с рациональными или

действительными элементами a и b относительно сложения и умножения матриц.

Ответ: 5.15.1 5.15.3. Кольца. В 5.15.3 при $m=0$ получаем нулевое кольцо, состоящее из одного числа 0, которое будет единицей кольца и сам для себя обратным. Нулевое кольцо не будет полем, так как поле должно содержать более одного элемента. 5.15.4 5.15.6. Поля. 5.15.7. Кольцо. 5.15.8. Поле. 5.15.9. Поле. 5.15.10.

Кольцо. 5.15.11. Кольцо. 5.15.12. Кольцо. 5.15.13. Матрицы с рациональными элементами a и b образуют поле, а с действительными a и b кольцо, но не поле.

5.16. Доказать, что числа вида $a + b\sqrt{5} + c\sqrt{25}$ с рациональными коэффициентами a , b и c образуют поле; найти в этом поле число, обратное числу $x = 2 + 3\sqrt{5} - \sqrt{25}$.

5.17. Доказать методом полной математической индукции, что для любого натурального числа выполняются следующие равенства:

$$\frac{1}{18} + \frac{1}{8} - \frac{1}{15} + \dots + \frac{1}{(7n+6)(7n+1)} = \frac{1}{2(2n-1)(2n+1)} \quad 1 + \frac{22}{15} + \dots + \frac{n}{2} = n(n+1);$$

$$2 \cdot 13 \cdot 3 \cdot 5 \cdot (2n-1)(2n+1) \cdot \frac{2(2n+1)}{3} = (7n+3n-1) \cdot 9;$$

$$12 + 2 \cdot 3 + \dots + n(n+1) = \frac{n(n+1)(n+2)}{3};$$

$$62n + 32n^2 + 3n > 11;$$

$$(3^3 n^2 + 5 \cdot 2^3 n^1) : 19.$$

В качестве примера докажем равенство 5.17.1. Обозначим сумму, стоящую в левой

$$S = \frac{1}{18} + \frac{1}{8} + \dots + \frac{1}{(7n-6)(7n-1)}$$

части доказываемого равенства через S_n , то есть

$$\text{Очевидно, что } S = S_n + \frac{1}{n(7n+1)(7n+8)}.$$

$$S_n =$$

1. Проверяем исходное равенство для $n=1$. Из последнего равенства при $n=1$ получаем $S = \frac{1}{18} + \frac{1}{8} - \frac{1}{15} = \frac{1}{2(2 \cdot 1 - 1)(2 \cdot 1 + 1)}$, следовательно, для $n=1$ исходное равенство истинно.

2. Предположим, что данное равенство истинно для некоторого натурального числа $k=n$, то есть $S = \frac{1}{2(2n-1)(2n+1)}$ и докажем при этом $\frac{1}{2(2k+1)(2k+3)}$

предположении истинность данного равенства при $k=n+1$. Для этого

рассмотрим сумму S_{k+1} :

$$S = S_k + \frac{7k^2 + 8k + 1}{(7k+1)(7k+8)}$$

$n = k + 1$. Заменяя S^k через

Выражение S_{k+1} совпадает с тем, которое получается из S_n при $k = n - 1$.

1,

предположение, что исходное равенство истинно при $n = k$.

3. Заключение: на основании принципа математической индукции данное равенство истинно для любого натурального числа.

5.17.5. Является ли линейным подпространством соответствующего векторного пространства каждая из следующих совокупностей векторов: Все векторы n -мерного векторного пространства, координаты которых целые числа?

5.17.6. Все векторы плоскости, каждый из которых лежит на одной из осей координат?

5.17.7. Все векторы плоскости, концы которых лежат на данной прямой (начало любого вектора, если не оговорено противное, предполагается совпадающим с началом координат)?

5.17.8. Все векторы плоскости, начала и концы которых лежат на данной прямой?

5.17.9. Все векторы плоскости, концы которых лежат в первой четверти системы координат?

5.17.10. Все векторы арифметического векторного пространства над полем действительных чисел, сумма всех координат которых равна 0?

5.17.11. Доказать, что все квадратные матрицы порядка n с действительными элементами образуют векторное пространство над полем действительных чисел, если в качестве операций взять сложение матриц и умножение матрицы на число.

6. ПРОБЛЕМЫ АКСИОМАТИЧЕСКОГО ПОСТРОЕНИЯ ТЕОРИЙ ПЕРВОГО ПОРЯДКА

План занятия

1. Проблема непротиворечивости.
2. Проблема независимости системы аксиом.
3. Формализуемость и разрешимость теории
4. Категоричность теории
5. Проблема полноты теории
6. 1. Проблема непротиворечивости

Определение 6.1. Теория T называется противоречивой, если она содержит такое высказывание S , что и S и его отрицание $\neg S$ являются теоремами. В противном случае теория T называется непротиворечивой.

Таким образом, теория T называется непротиворечивой, если в ней нет такого высказывания S , что S и $\neg S$ являются теоремами.

Для аксиоматических теорий вопрос об их непротиворечивости во многих случаях удается решить с помощью понятия модели. В самом деле, если теория T противоречива, то каждая ее модель содержит противоречие, так как пара противоречащих друг другу теорем теории переводится в два противоречащих друг другу высказывания о модели. Значит, теория непротиворечива, если для нее можно указать свободную от противоречий модель. Именно так доказывается непротиворечивость исчисления высказываний и исчисления предикатов. В первом случае моделью служит алгебра высказываний, во втором логика предикатов.

Если для теории T можно найти такую интерпретацию I , областью M которой является конечное множество, то вопрос об отсутствии противоречий в интерпретации I решается прямым рассмотрением всех элементов этого множества. Например,

одноэлементное множество $M = \{e\}$ (синглетон) вместе с определенной на нем операцией умножения $e \cdot e = e$ (например, $e = 1$) является моделью теории групп, лишенной противоречий, и, следовательно, теория конечных групп непротиворечива.

Чаще всего доказательство непротиворечивости теории требует очень сложных рассуждений. В частности, это бывает в случае, когда теория T имеет только бесконечные модели. Так, вопрос о непротиворечивости геометрии Лобачевского можно свести к вопросу о непротиворечивости геометрии Евклида, если использовать для построения интерпретации геометрии Лобачевского средства геометрии Евклида. Кстати сказать, непротиворечивость геометрии Евклида, как и непротиворечивость теории действительных чисел, до сих пор не доказана.

Любое доказательство непротиворечивости использует средства той или иной математической теории, а потому лишь сводит вопрос о непротиворечивости одной теории к вопросу о непротиворечивости другой теории. При этом говорят также, что первая теория непротиворечива относительно второй теории. Так, например, геометрия Лобачевского непротиворечива постольку, поскольку непротиворечива геометрия Евклида. Теория действительных чисел непротиворечива, если непротиворечива теория натуральных чисел и т.д. В настоящее время неизвестно ни одной содержательно теоретикочисловой теоремы, которая не была бы выводима в аксиоматической теории натуральных чисел, поэтому можно предположить, что теория натуральных чисел непротиворечива.

6. 2. Проблема независимости системы аксиом

Для всякой аксиоматической теории возникает вопрос о независимости ее аксиом, то есть, можно ли какую-нибудь аксиому вывести из остальных аксиом, применяя правила вывода данной теории? Если для некоторой аксиомы теории это возможно, то эту аксиому можно исключить из списка аксиом теории и теория при этом не изменится, то есть класс доказуемых формул останется без изменения.

Определение 6.2. Аксиома A теории T называется независимой от остальных аксиом этой теории, если она не может быть выведена из остальных аксиом. Система аксиом $\wedge$ теории T называется независимой, если каждая аксиома системы $\wedge$ не зависит от остальных аксиом.

Обычно независимость какой-либо аксиомы A от остальных аксиом системы $\wedge$ теории T доказывают следующим образом. Рассматривают новую систему $\wedge / = (\wedge \setminus \{A\})$ и A аксиом, которая полу-

чается из системы $\wedge$ заменой аксиомы A ее отрицанием $\neg A$. Если оказывается, что теория T' , построенная на основе системы аксиом

$\wedge /$, непротиворечива, то аксиома A не зависит от остальных аксиом данной системы. Таким способом, например, была доказана независимость аксиомы параллельности в геометрии Евклида от остальных аксиом. На этом пути была получена геометрия Лобачевского.

Независимость системы аксиом сама по себе не является обязательным свойством аксиоматической теории. Она лишь свидетельствует о том, что совокупность исходных положений теории не является избыточной, и представляет некоторые технические неудобства. Однако исследования, посвященные независимости системы аксиом, и доказательства независимости способствуют лучшему пониманию изучаемой теории.

По отношению к формальным системам и исчислениям имеет смысл говорить о независимости правил вывода. Правило вывода называется независимым, если существует теорема данного исчисления, которая не может быть выведена без использования этого правила.

6. 3. Формализуемость и разрешимость теории

Если для содержательной теории S удастся построить непротиворечивую и полную формальную теорию T , то S называется аксиоматизируемой или формализуемой теорией (понятие полноты теории будет рассмотрено ниже). Таким образом, здесь проблема

состоит в том, чтобы построить такую формальную теорию T , чтобы каждое истинное высказывание теории S отображалось в некоторую теорему теории T .

Примерами аксиоматизируемых теорий служат содержательные теории логики высказываний и логики предикатов, а также множество содержательных математических теорий теория групп, колец, полей, векторных пространств и т.д.

Еще одна важная характеристика формальной теории это ее разрешимость.

Теория T называется разрешимой, если существует алгоритм, который для любой формулы языка теории T определяет, является она теоремой в теории T , или нет.

Таким образом, проблема разрешимости является алгоритмической проблемой. В ней для заданного множества A требуется построить алгоритм, разрешающий A относительно другого множества B , включающего A . Этот алгоритм должен быть применим к любому элементу из множества B . Простейшим примером разрешимости является проблема разрешимости алгебры логики, в которой она состоит в отыскании алгоритма, позволяющего для каждой формулы алгебры логики установить, является ли она тождественно истинной, или тождественно ложной, или выполнимой.

Важным классом алгоритмических проблем является проблема разрешимости для формальных теорий, то есть проблема разрешимости для множества всех доказуемых (выводимых) в теории формул (множества A) относительно множества всех формул (множества B). Как было доказано Черчем, исчисление высказываний разрешимо, а исчисление предикатов неразрешимо (из-за бесконечности предметной области, которая приводит в общем случае к бесконечным таблицам истинности). Отметим, что важный для приложений фрагмент исчисления предикатов разрешим это исчисление одноместных предикатов.

6. 4. Категоричность теории

Определение 6.3. Математическая теория T называется категоричной, если все ее модели изоморфны.

Например, геометрия Евклида является категоричной математической теорией, так как любые ее две модели изоморфны, а любая модель изоморфна арифметической модели. Этот факт легко доказывается на основе средств аналитической геометрии.

Одна и та же теория может быть категоричной в одной интерпретации, и некатегоричной в другой. Здесь многое зависит от мощности множества, на котором строится модель теории. Например, теория групп некатегорична, так как существуют не изоморфные группы (например, коммутативная группа не изоморфна коммутативной группе на бесконечном множестве).

6. 5. Проблема полноты теории

Определение 6.4. Непротиворечивая теория T называется абсолютно полной (или полной в широком смысле слова), если для любого высказывания A этой теории A или $\neg A$ есть теорема, при этом высказывание A интерпретируется в некоторой модели.

Определение 6.5. Аксиоматическая теория T называется полной в узком смысле, если добавление к ее аксиомам любого недоказуемого в ней утверждения приводит к противоречивой теории.

Всякая абсолютно полная теория будет полна и в узком смысле. Так, исчисление высказываний полно как в широком, так и в узком смысле, в то время как исчисление предикатов неполно в узком смысле, но полно в широком смысле.

Мы не касаемся таких проблем аксиоматического исчисления высказываний как

- проблема разрешимости исчисления высказываний, которая заключается в доказательстве существования алгоритма, который позволил бы для любой заданной формулы исчисления высказываний определить, является ли она доказуемой или не является;
- проблема непротиворечивости исчисления высказываний, которая дает ответ на вопрос: существует ли такая формула A такая, что одновременно доказуема формула A и ее отрицание $\neg A$. Если такой формулы A не существует, то исчисление

высказываний непротиворечиво;

- проблема полноты исчисления высказываний, которая решает два вопроса:
 - а) можно ли рассматривать систему аксиом исчисления высказываний путем добавления к ней в качестве новой аксиомы какойнибудь недоказуемой в этом исчислении формулы? (полнота в узком смысле);
 - б) является ли всякая тождественно истинная формула алгебры высказываний доказуемой в исчислении высказываний? (полнота в широком смысле);
- проблема независимости аксиом исчисления высказываний: можно ли какую-нибудь аксиому вывести из остальных аксиом, применяя правила вывода данной системы?

Можно доказать, что исчисление высказываний разрешимо, непротиворечиво, полно как в узком, так и в широком смысле, и система аксиом независима.

Всякую конкретную математическую теорию T можно перевести на язык подходящей формальной системы S таким образом, что каждое осмысленное (ложное или истинное) предложение теории T выражается некоторой формулой системы S . Однако К. Гедель в 30-х годах XX в. показал, что такой метод формализации позволяет построить не все положительное содержание математических теорий. Такой принципиальный вопрос, как проблема непротиворечивости математических теорий, не всегда может быть решен в форме доказательств соответствующих утверждений о формализующих эти теории формальных системах. Поэтому существует, как уже отмечалось, определенные границы возможностей аксиоматического метода в том виде, который он приобрел в рамках формализма Д. Гильберта.

В завершение сформулируем две теоремы Геделя о неполноте формальной теории, содержащей формальную арифметику, которые имеют большое значение при аксиоматическом построении теорий первого порядка.

Первая теорема Геделя о неполноте

В любой формальной непротиворечивой теории, содержащей формальную арифметику, а, следовательно, и в теории натуральных чисел, найдется формально неразрешимое суждение, то есть такая замкнутая формула A , что ни A , ни $\neg A$ не являются выводимыми в данной теории.

Это теорема означает, что какая бы система аксиом для арифметики ни была выбрана, всегда найдется высказывание о натуральных числах, выраженное на языке этой формальной теории, которое в данной теории не может быть ни доказано, ни опровергнуто.

Вторая теорема Геделя о неполноте

Для любой формальной непротиворечивой теории T , содержащей формальную арифметику, формула, выражающая непротиворечивость теории T , недоказуема в T .

Выводы

На лекции рассмотрены основные проблемы построения аксиоматических теорий первопорядковой логики, такие как: непротиворечивость и независимость системы аксиом, формализуемость и разрешимость теории, категоричность и полнота.

ЛИТЕРАТУРА К 1 ГЛАВЕ

1. Акимов О.Е. Дискретная математика: логика, группы, графы. М.: Лаборатория Базовых Знаний, 2001.
2. Асеев Г.Г., Абрамов О.М., Ситников Д.Э. Дискретная математика: учебное пособие. Ростов н/Д: Феникс, Харьков: Торсинг, 2003.
3. Гильберт Д., Бернайс П. Основания математики. М.: Наука, 1979.
4. Гиндикин С.Г. Алгебра логики в задачах. М.: Наука, 1972.
5. Гладких О.Б., Белых О.Н. Математическая логика: учебнометодическое пособие. Елец: ЕГУ им. И.А. Бунина, 2011. 142 с.
6. Горбатов В. А., Горбатов А. В., Горбатова М. В. Дискретная математика: Учеб. для студентов втузов. М.: ООО «Издательство АСТ»; ООО «Издательство Астрель»,

2003.

7. Ерусалимский Я.М. Дискретная математика: теория, задачи, приложения. М.: Вузовская книга, 1998.
8. Ершов, Ю.Л., Палютин Е.А. Математическая логика: учеб. пособие. СПб.: Лань, 2005. 336 с.
9. Зарипова Э.Р., Кокотчикова М.Г., Севастьянов Л.А. Лекции по дискретной математике. Математическая логика: учебное пособие. М.: Российский университет дружбы народов, 2014. 118 с.
10. Игошин, В. И. Теория алгоритмов: учеб. пособие. М.: ИНФРА-М, 2012. 318 с.
11. Карпов В.Г., Мощенский В.А. Математическая логика и дискретная математика. Минск: Высшая школа, 1977.
12. Клини С. Математическая логика. М.: Мир, 1973.
13. Колмогоров А.Н., Драгалин А.Г. Математическая логика: Учеб. пособие. М.: Едиториал УРСС, 2004. 240 с.
14. Кузнецов О.П., Адельсон-Вельский Г.М. Дискретная математика для инженера. М.: Энергия, 1986.
15. Лавров И.А. Математическая логика. М.: Академия, 2006. 240 с.
16. Лихтарников, Л.М., Сукачева Т.Г. Математическая логика: Курс лекций, Задачник-практикум и решения: учеб. пособие для вузов. Санкт-Петербург; Москва; Краснодар: Лань, 2009. 276 с.
17. Макоха А. Н. Элементы математической логики: Учебное пособие. Невинномысск: Изд-во НГГТИ, 2005.
18. Мендельсон Э. Введение в математическую логику. М.: Наука, 1971.
19. Новиков П.С. Элементы математической логики. М.: Наука, 1973.
20. Столл Р. Р. Множества. Логика. Аксиоматические теории. М.: Просвещение, 1968.
21. Успенский В.А., Верещагин Н.К., Плиско В.Е. Вводный курс математической логики: уч. пособие. М.: Физматлит, 2007. 126 с.
22. Яблонский С. В., Гаврилов Г. П., Кудрявцев В. Б. Функции алгебры логики и классы Поста. М.: Наука, 1966.

Яблонский С.В. Введение в дискретную математику: учеб. пособие для вузов / под ред. В.А. Садовниченко. 3-е изд., стер. М.: Высш. шк., 2001. ГЛАВА II. НЕТРАДИЦИОННЫЕ ЛОГИКИ

В современном мире прогресс производительности программиста практически достигается только в тех случаях, когда часть интеллектуальной нагрузки берут на себя компьютеры. Одним из способов достигнуть максимального прогресса в этой области, является искусственный интеллект (ИИ), когда компьютер берет на себя не только однотипные, многократно повторяющиеся операции, но и сам сможет обучаться. Кроме того, создание полноценного ИИ открывает перед человечеством новые горизонты развития.

7. ОСНОВНЫЕ НАПРАВЛЕНИЯ ИССЛЕДОВАНИЙ В ОБЛАСТИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

План занятия

1. Естественный и искусственный интеллект
2. Некоторые философские аспекты проблем искусственного интеллекта
3. Примеры систем искусственного интеллекта

Термин интеллект происходит от латинского слова *intellectus* что означает ум, рассудок, разум; мыслительные способности человека. Соответственно искусственный интеллект (*artificial intelligence*) обычно толкуется как свойство автоматических систем

брать на себя отдельные функции интеллекта человека, например, выбирать и принимать оптимальные решения на основе ранее полученного опыта и рационального анализа внешних воздействий.

Мы интеллектом будем называть способность мозга решать интеллектуальные задачи путем приобретения, запоминания и целенаправленного преобразования знаний в процессе обучения на опыте и адаптации к разнообразным обстоятельствам.

В этом определении под термином «знания» подразумевается не только та информация, которая поступает в мозг через органы чувств. Такого типа знания чрезвычайно важны, но недостаточны для интеллектуальной деятельности. Дело в том, что объекты окружающей нас среды обладают свойством не только воздействовать на органы чувств, но и находиться друг с другом в определенных отношениях. Ясно, что для того, чтобы осуществлять в окружающей среде интеллектуальную деятельность, необходимо иметь в системе знаний модель этой среды, окружающего мира. В этой информационной модели окружающей среды реальные объекты, их свойства и отношения между ними не только отображаются и запоминаются, но могут мысленно «целенаправленно преобразовываться». При этом существенно то, что формирование модели внешней среды происходит «в процессе обучения на опыте и адаптации к разнообразным обстоятельствам».

В математике и кибернетике класс задач определенного типа считается решенным, когда для ее решения установлен алгоритм. Нахождение алгоритмов является естественной целью человека при решении им разнообразных классов задач. Отыскание алгоритма для задач некоторого данного типа связано с тонкими и сложными рассуждениями, требующими большой изобретательности и высокой квалификации. Принято считать, что подобного рода деятельность требует участия интеллекта человека. Задачи, связанные с отысканием алгоритма решения класса задач определенного типа, будем называть интеллектуальными.

Что же касается задач, для которых алгоритмы решения уже установлены, то, как отмечает известный специалист в области ИИ М. Минский, «излишне приписывать им такое мистическое свойства, как «интеллектуальность». В самом деле после того, как такой алгоритм уже найден, процесс решения соответствующих задач становится таким, что его могут в точности выполнить человек, вычислительная машина (должным образом запрограммированная) или робот, не имеющие ни малейшего представления о сущности самой задачи. Требуется только, чтобы лицо, решающее задачу, было способно выполнять те элементарные операции, из которых складывается процесс, и, кроме того, чтобы оно педантично и аккуратно руководствовалось предложенным алгоритмом. Такое лицо, действуя чисто механически, может успешно решать любую задачу рассматриваемого типа. Поэтому представляется совершенно естественным исключить их класса интеллектуальных такие задачи, для которых существуют стандартные методы решения. Примерами таких задач могут служить чисто вычислительные задачи: решение системы линейных алгебраических уравнений, численное интегрирование дифференциальных уравнений и т. д. Для решения подобного рода задач имеются стандартные алгоритмы, представляющие собой определенную последовательность элементарных операций, которая может быть легко реализована в виде программы для вычислительной машины. В противоположность этому для широкого класса интеллектуальных задач, таких, как распознавание образов, игра в шахматы, доказательство теорем и т. п., напротив это формальное разбиение процесса поиска решения на отдельные элементарные шаги часто оказывается весьма затруднительным, даже если само их решение несложно.

Таким образом, мы можем перефразировать определение интеллекта как универсальный сверхалгоритм, который способен создавать алгоритмы решения конкретных задач.

Еще интересным замечанием здесь является то, что профессия программиста, исходя из наших определений, является одной из самых интеллектуальных, поскольку

продуктом деятельности программиста являются программы алгоритмы в чистом виде. Именно поэтому, создание даже элементов ИИ должно очень сильно повысить производительность его труда.

Деятельность мозга (обладающего интеллектом), направленную на решение интеллектуальных задач, мы будем называть мышлением, или интеллектуальной деятельностью. Интеллект и мышление органически связаны с решением таких задач, как доказательство теорем, логический анализ, распознавание ситуаций, планирование поведения, игры и управление в условиях неопределенности. Характерными чертами интеллекта, проявляющимися в процессе решения задач, являются способность к обучению, обобщению, накоплению опыта (знаний и навыков) и адаптации к изменяющимся условиям в процессе решения задач. Благодаря этим качествам интеллекта мозг может решать разнообразные задачи, а также легко перестраиваться с решения одной задачи на другую. Таким образом, мозг, наделенный интеллектом, является универсальным средством решения широкого круга задач (в том числе неформализованных), для которых нет стандартных, заранее известных методов решения.

Следует иметь в виду, что существуют и другие, чисто поведенческие (функциональные) определения. Так, по А. Н. Колмогорову, любая материальная система, с которой можно достаточно долго обсуждать проблемы науки, литературы и искусства, обладает интеллектом. Другим примером поведенческой трактовки интеллекта может служить известное определение А. Тьюринга. Его смысл заключается в следующем. В разных комнатах находятся люди и машина. Они не могут видеть друг друга, но имеют возможность обмениваться информацией (например, с помощью электронной почты). Если в процессе диалога между участниками игры людям не удастся установить, что один из участников машина, то такую машину можно считать обладающей интеллектом. Некоторые философские аспекты проблем ИИ

Основная философская проблема в области ИИ возможность или не возможность моделирования мышления человека. В случае если когда-либо будет получен отрицательный ответ на этот вопрос, то все остальные вопросы не будут иметь не малейшего смысла. Следовательно, начиная исследование ИИ, мы заранее предполагаем положительный ответ. Приведем несколько соображений, которые подводят нас к данному ответу.

То, что раньше казалось вершиной человеческого творчества игра в шахматы, шашки, распознавание зрительных и звуковых образов, синтез новых технических решений, на практике оказалось не таким уж сложным делом (теперь работа ведется не на уровне возможности или невозможности реализации перечисленного, а о нахождении наиболее оптимального алгоритма). Теперь зачастую данные проблемы даже не относят к проблемам ИИ. Есть надежда, что и полное моделирование мышления человека окажется не таким уж и сложным делом.

С проблемой воспроизведения своего мышления тесно смыкается проблема возможности самовоспроизведения. Способность к самовоспроизведению долгое время считалась прерогативой живых организмов. Однако некоторые явления, происходящие в неживой природе (например, рост кристаллов, синтез сложных молекул копированием), очень похожи на самовоспроизведение. В начале 50-х годов Дж. фон Нейман занялся основательным изучением самовоспроизведения и заложил основы математической теории «самовоспроизводящихся автоматов». Так же он доказал теоретически возможность их создания.

Существуют также различные неформальные доказательства возможности самовоспроизведения, но для программистов самым ярким доказательством, пожалуй, будет существование компьютерных вирусов.

Принципиальная возможность автоматизации решения интеллектуальных задач с помощью ЭВМ обеспечивается свойством алгоритмической универсальности. Что же это за свойство?

Алгоритмическая универсальность ЭВМ означает, что на них можно программно реализовывать (т. е. представить в виде машинной программы) любые алгоритмы преобразования информации, будь то вычислительные алгоритмы, алгоритмы управления, поиска доказательства теорем или композиции мелодий. При этом мы имеем в виду, что процессы, порождаемые этими алгоритмами, являются потенциально осуществимыми, то есть что они осуществимы в результате конечного числа элементарных операций. Практическая осуществимость алгоритмов зависит от имеющихся в нашем распоряжении средств, которые могут меняться с развитием техники. Так, в связи с появлением быстродействующих ЭВМ стали практически осуществимыми и такие алгоритмы, которые ранее были только потенциально осуществимыми.

Однако не следует думать, что вычислительные машины и роботы могут в принципе решать любые задачи. Анализ разнообразных задач привел математиков к замечательному открытию. Было строго доказано существование таких типов задач, для которых невозможен единый эффективный алгоритм, решающий все задачи данного типа; в этом смысле невозможно решение задач такого типа и с помощью вычислительных машин. Этот факт способствует лучшему пониманию того, что могут делать машины и чего они не могут сделать. В самом деле, утверждение об алгоритмической неразрешимости некоторого класса задач является не просто признанием того, что такой алгоритм нам не известен и никем еще не найден. Такое утверждение представляет собой одновременно и прогноз на все будущие времена о том, что подобного рода алгоритм нам не известен и никем не будет указан или, иными словами, что он не существует.

Следующим философским вопросом ИИ является цель создания ИИ. Насколько это безопасно?

Данная проблема будоражит умы человечества еще со времен Карела Чапека, впервые употребившего термин «робот». Большую лепту в обсуждение данной проблемы внесли и другие писатели-фантасты. Как самые известные мы можем упомянуть серии рассказов писателя-фантаста и ученого Айзека Азимова, а также довольно свежее произведение «Терминатор». Кстати, именно у Айзека Азимова мы можем найти самое проработанное и принятое большинством людей решение проблемы безопасности. Речь идет о так называемых трех законах роботехники.

Робот не может причинить вред человеку или своим бездействием допустить, чтобы человеку был причинен вред.

Робот должен повиноваться командам, которые ему дает человек, кроме тех случаев, когда эти команды противоречат первому закону.

Робот должен заботиться о своей безопасности, насколько это не противоречит первому и второму закону.

На первый взгляд подобные законы, при их полном соблюдении, должны обеспечить безопасность человечества. Однако при внимательном рассмотрении возникают некоторые вопросы. Во-первых, законы сформулированы на человеческом языке, который не допускает простого их перевода в алгоритмическую форму. Попробуем, например, перевести на любой из известных нам языков программирования такие термины как «причинить вред» или «допустить». Предположим, что мы сумели переформулировать законы для робота на язык, который понимает автоматизированная система. Теперь интересно, что будет подразумевать система ИИ под термином «вред» после долгих логических размышлений? Не решит ли она, что все существования человека — это сплошной вред? Ведь он курит, пьет, с годами стареет и теряет здоровье, страдает. Не будет ли меньшим злом быстро прекратить эту цепь страданий? Конечно, можно ввести некоторые дополнения, связанные с ценностью жизни, свободой волеизъявления. Но это уже будут не те простые три закона, которые были в исходном тексте автора.

И теперь остался еще один вопрос, а стоит ли вообще создавать ИИ, может просто закрыть все работы в этой области? Единственное, что можно сказать по этому поводу

если ИИ возможно создать, то рано или поздно он будет создан. И лучше его создавать под контролем общественности, с тщательной проработкой вопросов безопасности, чем он будет создан лет через 100-150 (если к тому времени человечество еще не уничтожит само себя) каким-нибудь программистом-механиком-самоучкой, использующим достижения современной ему техники. Ведь сегодня, например, любой грамотный инженер, при наличии определенных денежных ресурсов и материалов, может изготовить атомную бомбу.

Примеры систем ИИ

Исторически сложились три основных направления в моделировании ИИ.

В рамках первого подхода объектом исследований являются структура и механизмы работы мозга человека, а конечная цель заключается в раскрытии тайн мышления. Необходимыми этапами исследований в этом направлении являются построение моделей на основе психофизиологических данных, проведение экспериментов с ними, выдвижение новых гипотез относительно механизмов интеллектуальной деятельности, совершенствование моделей и т. д.

Второй подход в качестве объекта исследования рассматривает ИИ. Здесь речь идет о моделировании интеллектуальной деятельности с помощью вычислительных машин. Целью работ в этом направлении является создание алгоритмического и программного обеспечения вычислительных машин, позволяющего решать интеллектуальные задачи не хуже человека.

Наконец, третий подход ориентирован на создание смешанных человеко-машинных, или, как еще говорят, интерактивных интеллектуальных систем, на симбиоз возможностей естественного и искусственного интеллекта. Важнейшими проблемами в этих исследованиях является оптимальное распределение функций между естественным и искусственным интеллектом и организация диалога между человеком и машиной.

Самыми первыми интеллектуальными задачами, которые стали решаться при помощи ЭВМ, были логические игры (шашки, шахматы), доказательство теорем. Хотя, правда здесь надо отметить еще кибернетические игрушки типа «электронной мыши» Клода Шеннона, которая управлялась сложной релейной схемой. Эта мышка могла «исследовать» лабиринт, и находить выход из него. Кроме того, помещенная в уже известный ей лабиринт, она не искала выхода, а сразу же, не заглядывая в тупиковые ходы, выходила из лабиринта.

Приведем ряд примеров систем ИИ.

Пример 1.1. Американский кибернетик А. Самуэль составил для вычислительной машины программу, которая позволяет ей играть в шашки, причем в ходе игры машина обучается или, по крайней мере, создает впечатление, что обучается, улучшая свою игру на основе накопленного опыта. В 1962 г. эта программа сразилась с Р. Нили, сильнейшим шашкистом в США и победила.

В машину были программно заложены правила игры так, что выбор очередного хода был подчинен этим правилам. На каждой стадии игры машина выбирала очередной ход из множества возможных ходов согласно некоторому критерию качества игры. В шашках (как и в шахматах) обычно невыгодно терять свои фигуры, и, напротив, выгодно брать фигуры противника. Игрок (будь он человек или машина), который сохраняет подвижность своих фигур и право выбора ходов и в то же время держит под боем большое число полей на доске, обычно играет лучше своего противника, не придающего значения этим элементам игры. Описанные критерии хорошей игры сохраняют свою силу на протяжении всей игры, но есть и другие критерии, которые относятся к отдельным ее стадиям дебюту, миттэндшпилю, эндшпилю.

Разумно сочетая такие критерии (например, в виде линейной комбинации с экспериментально подбираемыми коэффициентами или более сложным образом), можно для оценки очередного хода машины получить некоторый числовой показатель эффективности оценочную функцию. Тогда машина, сравнив между собой показатели

эффективности очередных ходов, выберет ход, соответствующий наибольшему показателю. Подобная автоматизация выбора очередного хода не обязательно обеспечивает оптимальный выбор, но все же это какой-то выбор, и на его основе машина может продолжать игру, совершенствуя свою стратегию (образ действия) в процессе обучения на прошлом опыте. Формально обучение состоит в подстройке параметров (коэффициентов) оценочной функции на основе анализа проведенных ходов и игр с учетом их исхода.

По мнению А. Самуэля, машина, использующая этот вид обучения, может научиться играть лучше, чем средний игрок, за относительно короткий период времени.

Пример 1.2. Ярким примером сложной интеллектуальной игры являются шахматы. Впервые шахматную программу предложил К. Шеннон в 1949 г. В 50-е годы прошлого века в СССР экс-чемпион мира по шахматам доктор технических наук М. Ботвинник вместе со своими учениками также создал программу для ЭВМ для игры в шахматы. В 1974 г. состоялся международный шахматный турнир машин, снабженных соответствующими программами. Как известно, победу на этом турнире одержала советская машина с шахматной программой «Каисса».

Несмотря на довольно большую сложность игры в шахматы, и невозможность, в связи с этим произвести полный перебор ходов, возможность перебора их на большую глубину, чем обычно, очень увеличивает шансы на победу. В 1997 г. компьютер Deep Blue фирмы IBM, победил чемпиона мира Г. Каспарова. В его состав входило 256 процессоров, каждый из которых имел 4 Гб дисковой памяти и 128 Мб оперативной. Весь этот комплекс мог анализировать 200 млн. возможных шахматных позиций в секунду. Автором замысла и главным конструктором компьютера Deep Blue был в 1987 г. Фенг-Синг Су. Это единственный компьютер, который победил чемпиона мира. Правда, в 2002 г. компьютер Deep Fritz сыграл с Владимиром Крамником со счетом 4:4. В основе стратегий компьютерных программ игры в шахматы, как и для игры в шашки, и некоторых других детерминированных и недетерминированных (нарды) игр лежат методы дискретной оптимизации, методы поиска по дереву (шенноновская схема) ветви дерева возможные ходы, листья заключительные положения, для которых результат игры известен. Размер дерева огромный порядка W^D , где W среднее количество ходов, а D количество уровней дерева. При этом используются методы МИНИМАКСА и

НЕГАМАКСА (последнее означает оценку позиции со знаком). При этом используется алгоритм альфа-бета-отсечения, идея которого состоит в том, что в больших частях дерева поиска мы не заинтересованы в получении точной оценки позиции. Нам лишь необходимо знать, является ли она лучше или хуже той, что мы нашли раньше. При оптимальных обстоятельствах перебор с альфа-бета-отсечением

$$\frac{D+!}{2} \quad D$$

должен просмотреть $W + W - 1$ позиций. Это меньше, чем ме-

тод минимакса. Есть много модификаций и усовершенствований альфа-бета-поиска: стремящийся поиск, использование таблиц перестановок, веер перебора, поиск с основным вариантом, нега-скаут и др.

До недавнего времени редкостью был компьютер, могущий делать такое количество целочисленных операций в секунду, как Deep Blue, а здесь мы говорим о ходах, которые должны быть сгенерированы и для которых просчитаны оценочные функции. Хотя с другой стороны, этот пример говорит о могуществе и универсальности переборных алгоритмов.

Пример 1.3. В настоящее время существуют и успешно применяются программы, позволяющие машинам играть в деловые или военные игры, имеющие большое прикладное значение. Здесь также чрезвычайно важно придать программам присущие человеку способность к обучению и адаптации. Одной из наиболее интересных интеллектуальных задач, также имеющей огромное прикладное значение, является задача

обучения распознавания образов и ситуаций. Решением ее занимались и продолжают заниматься представители различных наук физиологи, психологи, математики, военные, инженеры. Такой интерес к задаче стимулировался фантастическими перспективами широкого практического использования результатов теоретических исследований: читающие автоматы, системы ИИ, ставящие медицинские диагнозы, проводящие криминалистическую экспертизу и т. п., а также роботы, способные распознавать и анализировать сложные сенсорные ситуации.

В 1957 г. американский физиолог Ф. Розенблатт предложил модель зрительного восприятия и распознавания перцептрон. Появление машины, способной обучаться понятиям и распознавать предъявляемые объекты, оказалось чрезвычайно интересным не только физиологам, но и представителям других областей знания и породило большой поток теоретических и экспериментальных исследований.

Перцептрон или любая программа, имитирующая процесс распознавания, работают в двух режимах: в режиме обучения и в режиме распознавания. В режиме обучения некто (человек, машина, робот или природа), играющий роль учителя, предъявляет машине объекты о каждом из них сообщает, к какому понятию (классу) он принадлежит. По этим данным строится решающее правило, являющееся, по существу, формальным описанием понятий. В режиме распознавания машине предъявляются новые объекты (вообще говоря, отличные от ранее предъявленных), и она должна их классифицировать, по возможности, правильно. Естественным развитием теории перцептронов является теория искусственных нейронных сетей, нейроматематика и нейроинформатика.

Пример 1.4. Проблема обучения распознаванию тесно связана с другой интеллектуальной задачей проблемой перевода с одного языка на другой, а также проблема обучения машины языку. При достаточно формальной обработке и классификации основных грамматических правил и приемов пользования словарем можно создать вполне удовлетворительный алгоритм для перевода, скажем научного или делового текста. Для некоторых языков такие системы были созданы еще в конце 60-х годов прошлого века. Однако для того, чтобы связно перевести достаточно большой разговорный текст, необходимо понимать его смысл. Работы над такими программами ведутся уже давно, но до полного успеха еще далеко. Имеются также программы, обеспечивающие диалог между человеком и машиной на урезанном естественном языке.

Пример 1.5. Моделирование логического мышления. Хорошей модельной задачей здесь может служить задача автоматизации доказательства теорем. Начиная с 1960 г. был разработан ряд программ, способных находить доказательства теорем в исчислении предикатов первого порядка. Эти программы обладают, по словам американского специалиста в области ИИ Дж. Маккарти, «здравым смыслом», т. е. способностью делать дедуктивные заключения.

В программе К. Грина и др., реализующей вопросно-ответную систему, знания записываются на языке логики предикатов в виде набора аксиом, а вопросы, задаваемые машине, формулируются как подлежащие доказательству теоремы. Большой интерес представляет «интеллектуальная» программа американского математика Хао Ванга. Эта программа за 3 минуты работы IBM-704 вывела 220 относительно простых лемм и теорем из фундаментальной математической монографии, а затем за 8.5 мин. выдала доказательства еще 130 более сложных теорем, часть их которых еще не была выведена математиками. Правда, до сих пор ни одна программа не вывела и не доказала ни одной теоремы, которая бы, что называется «позарез» была бы нужна математикам и была бы принципиально новой. Это направление исследований мы рассмотрим более подробно в следующих разделах второй главы.

Пример 1.6. Очень большим направлением систем ИИ является роботехника. В чем основное отличие интеллекта робота от интеллекта универсальных вычислительных машин?

Для ответа на этот вопрос уместно вспомнить принадлежащее великому русскому

физиологу И. М. Сеченову высказывание: «... все бесконечное разнообразие внешних проявлений мозговой деятельности сводится окончательно лишь к одному явлению мышечному движению». Другими словами, вся интеллектуальная деятельность человека направлена, в конечном счете, на активное взаимодействие с внешним миром посредством движений. Точно так же элементы интеллекта робота служат, прежде всего, для организации его целенаправленных движений. В то же время основное назначение чисто компьютерных систем ИИ состоит в решении интеллектуальных задач, носящих абстрактный или вспомогательный характер, которые обычно не связаны ни с восприятием окружающей среды с помощью искусственных органов чувств, ни с организацией движений исполнительных механизмов. Приведем примеры интеллектуальных роботов.

1. В 1969 г. в Электротехнической лаборатории (Япония) началась разработка проекта «промышленный интеллектуальный робот». Цель этой разработки создание манипуляционного робота с элементами искусственного интеллекта для выполнения сборочно-монтажных работ с визуальным контролем.

2. Постепенно характеристики роботов монотонно улучшались, Но до сих пор они еще далеки по понятливости от человека, хотя некоторые операции уже выполняют на уровне лучших жонглеров. Например, удерживают на лезвии ножа шарик от настольного тенниса.

3. Еще можно выделить работы киевского Института кибернетики, где под руководством Н. М. Амосова и В. М. Глушкова (ныне покойного) ведется комплекс исследований, направленных на разработку элементов интеллекта роботов. Особое внимание в этих исследованиях уделяется проблемам распознавания изображений и речи, логического вывода (автоматического доказательства теорем) и управления с помощью нейроподобных сетей.

Пример 1.7. Экспертные системы. Экспертная система (ЭС) — это программа, которая ведет себя подобно эксперту в некоторой, обычно узкой, прикладной области. ЭС отличаются от традиционных программ тем, что они ориентированы на решение неформализованных задач. ЭС обладают следующими особенностями:

- Алгоритм решения задачи неизвестен заранее, а строится самой ЭС с помощью символических рассуждений, базирующихся на эвристических приемах.
- Ясность полученных решений, то есть ЭС как бы «осознает» в терминах пользователя, как она получила решение.
- Способность анализа и объяснения своих действий.
- Способность приобретения новых знаний от эксперта, не знающего программирования, и изменение в соответствии с ними своего поведения.
- Обеспечение дружественного естественно-языкового интерфейса с пользователем.

Приведем несколько примеров ЭС.

1. MYCIN экспертная система для медицинской диагностики. Разработана группой по инфекционным заболеваниям Стенфордского университета. Ставит соответствующий диагноз, исходя из представленных ей симптомов, и рекомендует курс медикаментозного лечения любой из диагностированных инфекций. База данных состоит из 450 правил.

2. PUFF анализ нарушения дыхания. Данная система представляет собой MYCIN, из которой удалили данные по инфекциям и вставили данные о легочных заболеваниях.

3. DENDRAL распознавание химических структур. Данная система старейшая, из имеющих звание экспертных систем. Первые версии данной системы появились еще в 1965 году во все том же Стенфордском университете. Пользователь дает системе DENDRAL некоторую информацию о веществе, а также данные спектроскопии (инфракрасной, ядерного магнитного резонанса и масс-спектрометрии), и та в свою очередь выдает диагноз в виде соответствующей химической структуры.

4. PROSPECTOR экспертная система, созданная для содействия поиску

коммерчески оправданных месторождений полезных ископаемых.

Экспертным системам мы также уделим достаточное внимание в этой главе.

Пример 1.8. Машинное творчество (программы сочинения волшебных сказок, программы сочинения стихотворных текстов, моделирование музыкальных произведений, когнитивная компьютерная графика и др.).

Еще интересным замечанием является то, что профессия программиста, исходя из наших определений, является одной из самых интеллектуальных, поскольку продуктом деятельности программиста являются программы алгоритмы в чистом виде. Именно поэтому, создание даже элементов ИИ должно очень сильно повысить производительность его труда.

Выводы

На лекции рассмотрено понятие интеллектуальной системы, затронуты некоторые философские аспекты построения такого рода систем, и приведены их примеры.

Вопросы и упражнения для самопроверки

1.1. Как изменятся требования к видеосистеме робота, если изображения, используемые им для самоконтроля, потребуется послать человеку, удаленно контролирующему поведение робота?

1.2. Растение, помещенное в темную комнату с единственным источником света, растет по направлению к нему. Является ли такое поведение разумным? Обладает ли растение интеллектом?

1.3. Предположим, что торговый автомат предназначен для отпуска различных товаров в зависимости от того, какая кнопка нажата. Можно ли сказать, что этот автомат «осознает», какая кнопка была нажата?

1.4. Если машина прошла тест Тьюринга, можно ли согласиться с тем, что она разумна? Если нет, то можно ли признать, что она кажется разумной?

1.5. Для каждой из перечисленных ниже прикладных областей указать, какие способы решения задач окажутся наиболее эффективными с помощью систем ИИ или традиционное программирование, и почему?

- управление предприятием;
- ремонт ЭВМ;
- классификация видов в биологии (птиц, цветов и т.д.);
- составление расписаний;
- установление диагноза заболевания человека;
- объяснение результата химического анализа крови при внутренних заболеваниях;

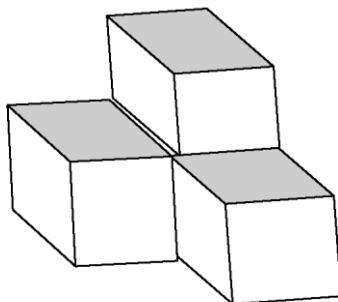
- финансовое планирование (анализ возможных вариантов инвестиций)

1.6. Сколько кубиков изображено на рис. 1.1? Как запрограммировать ЭВМ для получения точного ответа на подобный вопрос?

1.7. Сравнить результаты

1.8. В какой степени нашу жизнь *Рисунок 1.1. Иллюзия зрения* грамматического разбора следующих двух

1.8.



предложений и объяснить, чем они отличаются семантически: а) фермер построил изгородь на поле. б) фермер построил изгородь зимой. контролируют машины? Кто контролирует машины, воздействующие на нашу жизнь?

1.9. Откуда поступает информация, на которой основываются наши ежедневные решения? Насколько мы доверяем точности этой информации? Почему?

1.10. В качестве примера сложностей, возникающих при работе с естественным языком, указать ситуации, в которых один и тот же вопрос «Ты знаешь, который час?» будет иметь различные значения.

1.11. Люди могут употреблять вопросы для целей, отличных от опрашивания. Вот один из подобных вопросов: «Вы знаете, что у вас шина спустила?». Обычно его целью является информирование человека, а не пустое любопытство. Привести примеры вопросов, используемых для убеждения, предостережения, осуждения.

1.12. Если исследователь использует компьютерную модель для изучения способностей и процессов запоминания в человеческом мозге, всегда ли будут соответствующие программы для ЭВМ использовать память компьютера на пределе его возможностей? Объяснить свою точку зрения.

1.13. Как определить различие между интеллектом и кажущимся интеллектом? Это разные вещи?

1.14. Насколько исследователи в области ядерной энергетики, генной инженерии и искусственного интеллекта ответственны за использование результатов их работы?

8. ОСНОВНЫЕ ПРИНЦИПЫ ПОСТРОЕНИЯ СИСТЕМ АВТОМАТИЗАЦИИ ДОКАЗАТЕЛЬСТВ

План лекции

1. Постановка задачи автоматического доказательства теорем.
2. Унификация.
3. Метод резолюций.
4. Алгоритм поиска опровержения методом резолюций.
5. Доказательство истинности логических клауз методом резолюций.

Проблема автоматизации процесса доказательства теорем в науке (особенно в математике) возникла давно. Еще Г. Лейбниц в конце 17го века предпринимал попытку найти универсальную процедуру доказательства теорем.

Причина большого интереса к данному вопросу в настоящее время заключается в том, что современное научное знание не мыслимо без аксиоматического метода. Поэтому от того, владеем ли мы достаточно эффективным инструментом проверки выводимости формул в рамках конкретной теории или нет, зависит успех исследования и практического использования той или иной модели самой теории.

Вопросами автоматизации доказательства теорем изначально занималась математическая логика. На сегодняшний день автоматическое доказательство теорем специальная область искусственного интеллекта, интегрирующая результаты, полученные в математической логике, с достижениями в области искусственного интеллекта.

Несмотря на довольно большой срок существования данной проблемы в науке, основные результаты были получены только в начале 20-го века. Такие ученые как Генцен, Ж. Эрбран, Дж. А. Робинсон и др. заложили основы современных процедур автоматического доказательства теорем и определили основные направления развития этой области. Их исследования привели к созданию мощного метода доказательства теорем логики предикатов первого порядка метода резолюций. Наиболее современные его варианты намного отличаются от классического определения, предложенного Дж. А. Робинсоном в 1965 г. Жесткие требования, предъявляемые возможностями электронновычислительной техники, стали причиной детальных исследований метода и многочисленных усовершенствований в целях повышения его эффективности. Не всегда

эффективность конкретной процедуры сохраняла полноту метода резолюций.

В настоящее время большинство исследователей в области автоматического доказательства теорем сходятся в том, что никакая оптимизированная процедура по методу резолюций не может быть достаточно эффективной без использования других методов искусственного интеллекта, в частности, теории поиска в графах, теории экспертных систем, теории самообучающихся и самонастраивающихся систем, теории систем распознавания образов и др. Автоматическое доказательство теорем это краеугольный камень логического программирования, искусственного интеллекта и других современных направлений в программировании.

2.1. Постановка задачи автоматического доказательства теорем

Современная математика не мыслима без аксиоматического метода. Доказательство теорем длительное время считалось одной из сложных интеллектуальных задач, с которой под силу справиться лишь опытному математику или искусному логичу. XX век опроверг устоявшиеся представления.

Долгие попытки найти алгоритм доказательства произвольной теоремы в XX веке, наконец, увенчались успехом.

Американские ученые в области математической логики и теории алгоритмов Черч и Тьюринг независимо друг от друга доказали, что не существует никакой общей разрешающей процедуры, никакого алгоритма, проверяющего общезначимость формул в логике первого порядка. Тем не менее, существуют алгоритмы поиска доказательства, которые могут подтвердить, что формула общезначима, если она на самом деле общезначима. Для необщезначимых формул эти алгоритмы, вообще говоря, не заканчивают свою работу. Принимая во внимание результаты исследований Черча и Тьюринга, это лучшее, что мы можем ожидать от алгоритма поиска доказательства.

Среди основоположников всех современных методов автоматического доказательства теорем можно назвать имена Г. Лейбница, Д. Пеано, Д. Гильберта, Генцена, Ж. Эрбрана, Дж. А. Робинсона, Ч. Ченя и др. В частности, в 1931 г. Жаком Эрбраном был создан так называемый метод резолюций, который в 1961 г. был запрограммирован Дж. А. Робинсоном. Суть метода резолюций состоит в том, что множество дизъюнктивных нормальных форм (дизъюнктов) невыполнимо тогда и только тогда, когда пустой дизъюнкт (т.е. «0») является логическим следствием из этого множества дизъюнктов. Таким образом, невыполнимость множества дизъюнктов S можно проверить, порождая логические следствия из S до тех пор, пока не получим пустой дизъюнкт. Этот метод является одним из мощнейших методов доказательства теорем в логике первого порядка. Систематизируя результаты исследований перечисленных выше ученых, отечественные логики и математики С. Ю. Маслов и Г. Е. Минц создали так называемый обратный метод доказательства в произвольных исчислениях.

Возможность автоматизировать доказательство теорем чрезвычайно важный результат. Он был получен Ж. Эрбраном в начале 30-х годов 20-го века и послужил отправной точкой большинства исследований в этой области. Прежде всего, нужно отметить, что автоматизация доказательства теорем не есть узкая проблема математической логики. В общем случае мы можем рассматривать автоматизацию доказательства теорем логики первого порядка (то есть содержательных теорем). Ее суть заключается в поиске процедур и алгоритмов, которые позволяют обоснованно ответить на вопрос: является ли данное предложение теоремой или нет, и/или построить цепочку вывода, то есть доказательство теоремы. Это значит, что с помощью такой процедуры мы можем решать очень широкий класс задач.

Постановка задачи автоматического доказательства теорем в общем виде выглядит следующим образом.

Алгоритм, который проверяет отношение

$\Gamma \vdash T \text{ из } S$

для формулы S , множества формул Γ и теории T называется алгоритмом

автоматического доказательства теорем.

Как уже отмечалось, в общем случае такой алгоритм невозможен, то есть не существует алгоритма, который для любых S , Γ и T выдавал бы ответ «Да», если $\Gamma \vdash T$, S , и ответ «Нет», если неверно, что

$\Gamma \vdash T$. Более того, известно, что нельзя построить алгоритм автоматического доказательства теорем даже для большинства конкретных достаточно сложных формальных теорий T . В некоторых случаях удается построить алгоритм автоматического доказательства теорем, который применим не ко всем формулам теории (то есть частичный алгоритм). Для некоторых простых формальных теорий (например, исчисление высказываний) и некоторых простых классов формул (например, прикладное исчисление предикатов с одним неизвестным предикатом) алгоритмы автоматического доказательства теорем известны.

Упомянутый выше обратный метод позволяет решать вопрос о доказуемости некоторого предложения уже в рамках произвольного исчисления. Он эффективно используется в дискретных моделях, дедуктивных системах, системах планирования и долгосрочного макропрогноза и т.д.

Пример 2.1. Поскольку для исчисления высказываний известно, что теоремами являются общезначимые формулы, можно воспользоваться простым методом проверки общезначимости формулы с помощью таблиц истинности. А именно, достаточно вычислить истинностное значение формулы при всех возможных интерпретациях (их конечное число). Если во всех случаях получится истинное значение формулы, то эта формула тавтология и, следовательно, является теоремой теории T .

Рассмотренный пример является алгоритмом автоматического доказательства теорем в теории T , хотя и не является алгоритмом автоматического поиска вывода теорем из аксиом теории T .

Любую задачу можно переформулировать таким образом, что ее решение будет сведено к доказательству некоторой теоремы. Это касается и количественных задач. Их можно выразить в предложениях формальной арифметики. С другой стороны, столь подробная формализация задачи придаст ей сложный и громоздкий вид, что вообще затруднит ее понимание и поиск решения. Гораздо эффективнее аксиоматизировать задачу собственным множеством аксиом, которые являются аксиомами некоторой теории или же теоремами этой теории. Формулировка задачи в рамках этой новой аксиоматики будет проще и понятнее, и в большей мере будет способствовать ее успешному решению. В этих случаях применяется метод линейной резолюции, который был предложен независимо друг от друга Лавлендом и Лакхемом. Основная идея метода автоматического доказательства теорем такого класса заключается в том, что сначала строится опровержение доказываемой формулы, затем находится множество дизъюнктов, соответствующих опровержению этой формулы и, наконец, применяется сам метод линейной резолюции. Несмотря на то, что методу резолюций предшествует большое количество дополнительных преобразований, все они легко алгоритмизируются и потому их выполнение можно доверить ЭВМ, также как и саму процедуру резолютивного вывода.

В более сложных случаях эффективным может оказаться следующий путь. Он заключается в построении специального исчисления со своим набором аксиом и правилами вывода, например, специальное исчисление для нахождения первообразных. Аксиомами этого исчисления являются табличные интегралы. Правила вывода соответствуют интегрированию подстановкой или по частям, а также стандартным преобразованиям элементарных функций: вынесение множителей, интегрированию сумм и т.д. Для решения задач в этом исчислении используются результаты теории поиска вывода (специальной области математической логики, которая занимается выявлением по объекту и исчислению возможной структуры вывода). Для решения задач в таком исчислении лучше всего воспользоваться обратным методом как наиболее универсальным. Построение логической модели такого исчисления крайне

громоздкое и неэффективное.

Очень важный подход к автоматическому доказательству теорем был дан Эрбраном в 1930 г. По определению общезначимая формула есть формула, которая истинна при всех интерпретациях. Эрбран разработал алгоритм нахождения интерпретации, которая опровергает данную формулу. Однако если данная формула на самом деле общезначима, то никакой такой интерпретации не существует и алгоритм заканчивает свою работу за конечное число шагов. Метод Эрбрана служит основой для большинства современных алгоритмов автоматического поиска доказательства.

Гилмор одним из первых реализовал процедуру Эрбрана на вычислительной машине. Поскольку формула общезначима тогда и только тогда, когда ее отрицание противоречиво, его программа предназначена для обнаружения противоречивости отрицания данной формулы. Во время выполнения его программы порождаются пропозициональные формулы, которые периодически проверяются на противоречивость. Если отрицание данной формулы противоречиво, то программа Гилмора обнаруживает этот факт. Программа Гилмора справилась с доказательством нескольких простых формул, но столкнулась с большими трудностями в доказательстве формул в логике первого порядка. Внимательное изучение его программы показало, что его метод проверки противоречивости пропозициональных формул неэффективен. Метод Гилмора был улучшен Девидсом и Патнером несколько месяцев спустя, после того, как был опубликован. Однако их улучшение было еще недостаточно. Многие общезначимые формулы логики первого порядка все еще не могут быть доказаны на ЭВМ в разумное время.

Главный скачок был сделан Робинсоном, который, как уже упоминалось, ввел метод резолюций. Процедура поиска доказательства методом резолюций намного эффективнее, чем любая описанная ранее процедура. После введения метода резолюций были предложены некоторые стратегии в целях увеличения его эффективности. Такими стратегиями являются семантическая резолюция (Слэйгл, Мельцер, 264Робинсон, Ковальский и др.), лок-резолюция (Бойер), линейная резолюция (Лавленд, Лакхем, Андерсон и Бледсоу. Ейтс, Рафаэль и Кюнер), стратегия предпочтения единичных (Уос, Карсон, и Робинсон, Чень), стратегия поддержки (Уос, Робинсон и Карлсон).

Параллельно с методом резолюций возник и развивался обратный метод.

Принципиальные проблемы обоих методов имеют общие и специфические черты. И в методе резолюций и в обратном методе возможности их реализации упираются в возможности вычислительной техники. Принципиальная проблема метода резолюций критерий окончания процесса для выполнимых формул. Для обратного метода принципиальные проблемы заключаются в нахождении специальных классов формул, которые трудно разрешимы этим методом.

Несмотря на указанные проблемы автоматического доказательства теорем, важным остается вопрос об аппарате, позволяющем для формул некоторого исчисления решить вопрос об их выводимости из набора аксиом по правилам вывода этого исчисления. Формальные модели позволяют решать более широкий класс задач, чем математическое моделирование. Выбирая формальную модель в качестве аппарата решения проблемы, мы в значительной степени более свободны в выборе структуры представления задачи и конкретных способов ее решения. В приложении к математике формальные модели выступают важным инструментом теоретических и практических исследований.

2. 2. Унификация

В теориях первого порядка часто возникает вопрос: как применить теорему к данному выражению? Понятие унификации и алгоритм унификации предназначены для ответа на этот вопрос. Алгоритм унификации не зависит от формальной системы, в которой применяется. Для заданной теоремы A и заданного выражения E алгоритм унификации выполняет в строгой последовательности двойной проход в предварительно определенном порядке. Этот алгоритм, разработанный в 1966 г. Ж. Питра и независимо от

него Дж. Робинсоном, играет фундаментальную роль в искусственном интеллекте, в частности, в автоматическом доказательстве теорем. Задача унификации состоит в следующем.

Пусть теорема А сформулирована в виде логического следования (говорят также: в виде правила переписывания, правила подстановки или правила продукции):

Для данной теоремы А и некоторого выражения Е необходимо проверить, можно ли сделать Н и Е полностью идентичными, то есть унифицировать Н и Е путем последовательных подстановок свободных переменных в Н и Е. Подстановки, выполненные в формуле С, дают новую форму выражения Е в результате применения теоремы А (с использованием правила *modus ponens*). Рассмотрим примеры унификации.

Пример 2.2. Пусть известное алгебраическое тождество

$$(a + b)^2 = a^2 + 2ab + b^2 \quad (2.1)$$

рассматривается в качестве удобного сокращенного представления правила переписывания: «Если какое-то выражение является квадратом суммы двух термов, то его можно переписать в виде суммы квадрата первого терма, удвоенного произведения обоих термов и квадрата второго терма». Пусть, кроме того, имеется выражение

$$x^2 + (y + z)^2. \quad (2.2)$$

В данном случае легко увидеть совпадение выражений в скобках в (2.1) и (2.2): подстановка у вместо а и $\wedge 3$ вместо b позволяет унифицировать $y + 3\mathbf{I}$ и левый член тождества (2.1).

Пример 2.3. Аналогично известное правило переписывания

$$\sin^2 u + \cos^2 u = 1 \quad (2.3)$$

применяется к выражению

$$\sin^2 3X + \cos^2 3X, \quad (2.4)$$

чтобы показать, что выражение (2.4) также дает единицу (2.4). Но унификация с тем же правилом (2.4) не получается, если взять выражение

$$\sin^2 3X + \cos^2 X. \quad (2.5)$$

Унификация также невозможна при использовании выражения

$$\cos^2 3X + \sin^2 3X, \quad (2.6)$$

так как в этом случае требуется коммутативность относительно символа «+». Это затруднение можно устранить с помощью использова-

$$\text{правила} \quad a + b \wedge b + a,$$

предварительное применение которого к выражению (2.4) позволяет затем выполнить унификацию выражений (2.4) и (2.3). Пример 2.4. Предыдущие примеры представляли собой простейшие случаи унификации. Однако далеко не всегда все так просто. В общем случае возникает проблема выбора подстановок: ведь может существовать множество различных способов для получения совпадения подстановок двух формул, причем некоторые способы дают формулы, выводимые из какой-либо более общей формулы. Так, пусть в исчислении высказываний некоторое выражение является теоремой, вытекающей из первой аксиомы исчисления высказываний:

$$(p \wedge (q \wedge p)) \quad (2.7)$$

и пусть имеется метатеорема

$$((p \wedge Q) \wedge R) \wedge (Q \wedge R). \quad (2.8)$$

Мы рассмотрим те возможные естественные подстановки, которые, однако, не являются оптимальными. Они приведут к правильному результату частному выражению, которое, однако, не представляет большого интереса. Вспомним, что в исчислении высказываний переменные высказывания (пропозициональные буквы) p, q, r, ... и формулы P, Q, R, ... являются свободными переменными. Все появления какой-либо переменной могут быть заменены термом, не содержащим заменяемой переменной. Чтобы привести к совпадению выражений (2.8) и (2.7), можно вначале сделать пробные шаги. Так, подстановка, преобразующая формулу Q в переменное высказывание q в выражении (2.8),

дает

$$((P \wedge q) \wedge R) \wedge (q \wedge R). \quad (2.9)$$

Заменим p в (2.7) на $P \wedge q$ так, чтобы унифицировать начальные символы выражений (2.7) и (2.9)

$$(p \wedge q) \wedge (q \wedge (p \wedge q)). \quad (2.10)$$

Получим совпадение между формулой (2.7), в которой сделана подстановка, и левым членом теоремы (2.8), в которой также выполнена подстановка. При этом имеется еще резервная возможность замены R на формулу $(q \wedge (P \wedge q))$. Правый член выражения (2.9),

как и в предыдущих примерах, дает искомый результат:

$$(q \wedge (q \wedge (P \wedge q))). \quad (2.11)$$

Однако существует алгоритм унификации, который без всяких пробных шагов дает нужный результат:

$(p \wedge (q \wedge (p \wedge q)))$. Очевидно, что последняя формула представляет собой наиболее общий результат. Формула (2.11) соответствует частному случаю $p = q$.

Мы опишем основную идею оптимального алгоритма унификации, который представляет особый интерес, так как, с одной стороны, дает прямой систематический метод приведения к совпадению двух каких-либо формул с помощью подстановок в них, а с другой эти подстановки, полученные без полного перебора, носят наиболее общий характер.

Выражение E и теорему A будем просматривать параллельно и на одну и ту же глубину. Выполним только те унификации, которые необходимы. Если, в конце концов, мы перейдем от гипотезы H к теореме A , то унификация окажется успешной, и результат будет находиться в выражении C , модифицированном с помощью тех же подстановок. В противном случае имеем неудачу.

Во всяком выражении только свободные переменные являются заменяемыми, то есть переменные, которые не связаны кванторами. Единственно разрешенными подстановками являются такие, которые заменяют свободную переменную x из E или H термом t из E или H . Имеется только одно ограничение: терм t сам не должен содержать переменную x . Поэтому, чтобы избежать всякой неоднозначности и неопределенности, в первую очередь проводится разделение имен переменных из E или A , каковы бы ни были их начальные имена.

Подстановки выполняются только по необходимости. Символы, которые не являются переменными, не могут заменяться. К таким символам относятся либо символы операций, либо константы. Естественно, такие символы должны находиться на соответствующих позициях в формулах E и H , иначе невозможна никакая унификация.

Фундаментальная идея, лежащая в основе алгоритма унификации, связана с процедурой просмотра выражения: вначале просматривается основной оператор (операция), затем каждый из подтермов, которыми он управляет, начиная, например, с самого левого. Каждый подтерм при этом просматривается параллельно: основной оператор, подтерм и так далее до переменной или константы. Удобным для этого является представление выражения в виде дерева.

Порядок просмотра носит префиксный характер: каждый символ обрабатывается в той последовательности, в которой он встречается при спуске по дереву сверху вниз и слева направо. Используя этот порядок просмотра, введем индексы e и h : e индекс текущего символа в выражении E и h индекс текущего символа в выражении H .

Принцип рекуррентности: пусть все символы в выражениях E и H совпадают вплоть до символов с индексами e и h , не включая их; необходимо попытаться привести к совпадению символы с индексами e и h соответственно в E и H .

Имеются четыре возможных варианта ситуаций, при которых каждый из символов с индексами e и h соответствуют (либо нет) какой-то заменяемой переменной. Если

подстановка должна быть произведена, то ее можно осуществить с помощью определенной процедуры.

Сам алгоритм унификации носит рекурсивный характер. Рассмотрим один из вариантов описания этого алгоритма. Если язык теории удовлетворяет определенным условиям, то можно проверить, являются ли две заданные формулы унифицируемыми, и если это так, то можно найти наиболее общий унификатор, то есть набор подстановок. Предположим, что функция f осуществляет синтаксический разбор формулы и возвращает символ главной операции (то есть символ импликации, отрицания или признак того, что формула является переменной). Пусть функции l и r возвращают левый и правый операнды главной операции, то есть подформулы (подтермы). При этом предположим, что отрицание имеет только правый операнд. Набор подстановок (унификатор) представим в виде глобального массива S , значениями индекса которого являются имена переменных, а значениями элементов формулы, которые подставляются вместо соответствующих переменных. При этих соглашениях рекурсивный алгоритм унификации можно представить следующим образом.

алг Unify

вход: формулы H и E .

выход: false, если формулы не унифицируемы; true и наиболее общий унификатор S в противном случае.

если $f(H)$ переменная, то $v := f(H)$ {переменная}

если $S(v) = 0$, то

$S(v) := E$;

вернуть true {то есть добавить подстановку вместо v E } иначе вернуть $S(v) \wedge E$

{ибо эта подстановка уже есть, либо унификация невозможна} кон если

кон если

если $f(H) \wedge f(E)$ то вернуть false

{главные операции различны унификация невозможна} кон если

если $(H) = ' \neg '$ то повторить Unify ($r(H)$, $r(E)$) {попытаться унифицировать операнды отрицаний}

кон если

если $(A) = ' \wedge '$ то повторить Unify ($l(H)$, $l(E)$) & Unify ($r(H)$, $r(E)$) {попытаться унифицировать операнды импликации}

кон если.

Обоснуем этот алгоритм. При любых подстановках формул вместо переменных главная операция (логическая связка) формулы остается неизменной. Поэтому, если главные операции формул различны, то формулы заведомо не унифицируемы. В противном случае, то есть, если главные операции совпадают, формулы унифицируемы тогда и только тогда, когда унифицируемы подформулы, являющиеся операндами главной операции. Эта рекурсия заканчивается, когда сопоставление доходит до переменных. Переменная унифицируется с любой формулой (в частности, с другой переменной) простой подстановкой этой формулы вместо переменной. Но подстановки для всех вхождений одной переменной должны совпадать. Алгоритм завершается, когда доходят до конца формулы H . Формула E может иметь больший размер, чем формула H , если теорема A представляет собой правило переписывания. Поскольку правые ветви деревьев, представляющих формулы E и H , могут разрастаться, необходимо показать, что процесс на самом деле завершен.

Известно, что в начале процесса число переменных, заменяемых в формулах E и H , является конечным. С одной стороны, во время выполнения алгоритма унификации не создается новых переменных, а с другой каждая подстановка, которая заменяет переменную термом, не содержащим ее, приводит к уменьшению общего числа обрабатываемых переменных. Следовательно, алгоритм унификации является сходящимся при числе подстановок, большем или равном первоначальному числу

свободных переменных в Е и Н.

Алгоритм унификации может быть использован в различных областях математики: алгебре, арифметике, тригонометрии, логике, дифференциальном и интегральном исчислении (за исключением тех случаев, когда существуют незаменяемые переменные). Однако реализация алгоритма унификации может привести к проблеме заикливания, которая решается с помощью двух специальных процедур: процедура нормализации для формулы и процедуры запрета для теорем.

Нормализация основывается на том, что определенные правила применяются только при наличии определенного условия, которое запрещает повторное применение правила вывода. Иначе это может привести к потере полученного перед этим результата.

Возможен случай, когда применение некоторых правил оказывает влияние и на использование каких-то других правил, применение которых до этого было запрещено. Это тот случай, когда в результате применения некоторого правила структура данной формулы изменяется в такой «достаточной» степени, что вероятность возврата к предшествующему состоянию становится несущественной. В этих случаях используют процедуры запрета (деление на нуль, правило $\tan x \wedge \sin x / \cos x$ и т. п.).

При автоматическом доказательстве теорем существует проблема, которую надо решить, чтобы перевести на ЭВМ алгоритм унификации: необходимо отсечь от дерева поиска те его части, которые ведут к бесконечным процедурам. Существует некоторые методы, разрешающие эту проблему. В качестве примера рассмотрим метод Сиклоши и Маринова.

Некоторые правила переписывания при их применении дают, в частности, такую правую часть формулы, к которой снова применимо то же правило. Такие правила называются экстенсивными. Очевидно, использование таких правил порождает бесконечное множество формул, поэтому их применение должно быть как-то ограничено. Например, в исчислении высказываний правило $p \wedge p$ является экстенсив-

ным. Программа автоматического доказательства теорем может определить сама, используя алгоритм унификации, является ли выполняемое правило экстенсивным или нет. Правила типа $x + y \wedge y + x$, будучи применимы последовательно дважды к одному и тому же выражению, в результате оставляют его неизменным. Такие правила могут приводить к бесконечной процедуре поиска, и они являются на самом деле только псевдоэкстенсивными. Сиклоши и Маринов предложили рассматривать экстенсивные правила только тогда, когда обойтись без них невозможно. Эти правила в таком случае применяются не более одного раза для уровня текущего поиска, а полный поиск по всем другим правилам отсекается. Благодаря такому подходу Сиклоши и Маринов в 1971 получили гораздо лучшие результаты, чем при ручном способе решения.

С помощью описанного выше алгоритма унификации не были получены какие-то важные результаты в математике, однако в отличие от многих других систем автоматического доказательства теорем этот алгоритм может находить новые теоремы достаточно лишь дать ему возможность двигаться в обратном направлении. Он выявляет большие возможности ЭВМ в обработке символьной информации: самые длинные доказательства требуют всего лишь нескольких секунд машинного времени.

В тех случаях, когда проблемы символьного доказательства не являются основными (например, в системах автоматизации программирования), создаются просто хорошие подпрограммы. Кроме того, чтобы доказать промежуточные результаты в той области математики, которая не является основной для системы автоматического доказательства теорем, поступают, например, следующим образом: математик, специализирующийся в топологии, получает необходимый результат, работая традиционным образом с литературой.

Существует большое количество программ автоматического доказательства теорем. Мы укажем лишь одну из них программу PARI, созданную в 1978 г. Математические знания в этой программе сконцентрированы в специфических модулях.

Общая идея решения задачи состоит в том, чтобы выбрать одну из задач, имеющую решение, и затем сделать все возможное для того, чтобы свести исходную задачу к ней. Программа PARI решает задачи с целочисленными переменными, используя шесть модулей, которые могут обрабатывать определенные подзадачи. Эти подзадачи играют роль усовершенствованных обязательных правил вывода. Не имея возможности подробно останавливаться на структуре программы PARI, мы в заключение в качестве примеров приведем формулировки нескольких задач, которые эта программа решает.

Задача 2.1. Найти два простых числа p и q таких, что $p^2 = 8q + 1$.

Задача 2.2. Найти целые положительные числа x и n такие, что $5x^6 + 7x^8 + 8x^n$.

$x = 3x + 4$

n

Задача 2.3. Найти все целые числа n такие, что $\sum_{i=1}^n i = n(n+1)$.

$i = 1$

Задача 2.4. Найти целые числа a и b такие, что $a \mid b$ и $a \mid (b+1)$.

Задача 2.5. Доказать тождество

$\sum_{i=1}^n (2i-1)^2 = \frac{n(2n-1)(2n+1)}{3}$

$i = 1$

3

2. 3. Метод резолюций

В 1930 г. в докторской диссертации по математике Эрбран предложил оригинальный метод доказательства теорем в формальных системах первого порядка. Общая идея этого метода состоит в том, чтобы получить некоторое заключение C , исходя из гипотез $H_1, H_2, \dots, H_n$, с помощью следующей простой схемы рассуждений: чтобы доказать теорему A , записанную в виде

$H_1 \& H_2 \& \dots \& H_n \wedge C$, достаточно доказать противоречивость формулы F

$\neg H_1 \& \neg H_2 \& \dots \& \neg H_n \& C$,

в которой отрицание заключения добавлено к исходным гипотезам. Такое доказательство может быть проще прямого вывода. Для доказательства противоречивости формулы F достаточно показать, что F не является теоремой. Действительно, избавляясь в формуле A от импликации, приведем ее в виду

$H_1 \& H_2 \& \dots \& H_n \vee C = H_1 \vee H_2 \vee \dots \vee H_n \vee C$, а, применяя к формуле F закон двойного отрицания, получим

$\neg H_1 \& \neg H_2 \& \dots \& \neg H_n \& C = \neg H_1 \vee \neg H_2 \vee \dots \vee \neg H_n \vee C$,

откуда следует, что $F = A$. Так как A — теорема, то по закону противоречия ($p \& p$) формула F теоремой не является.

Эрбран предложил эффективный способ этого доказательства исходной теоремы A за конечное число подстановок в формулу F . Эти рассуждения и лежат в основе метода резолюций.

Правило резолюций для исчисления высказываний

Предварительно введем ряд понятий и фактов, необходимых для дальнейшего рассмотрения метода резолюций.

Как было отмечено во вступительной части, процедура поиска доказательства по Эрбрану или по методу резолюций на самом деле являются процедурами поиска опровержения, то есть вместо доказательства общезначимости формулы доказывается, что отрицание данной формулы противоречиво. Это только вопрос удобства при использовании процедур опровержения не происходит потери общности. Кроме того, эти процедуры опровержения применяются к некоторой стандартной форме формулы. Эта стандартная форма была введена Девидсом и Патнемом.

Под литерой будем понимать в общем смысле предикатную букву. Например, предикату $P(x, y, f(a))$ соответствует литера P . В случае, когда мы имеем несколько предикатов с различными параметрами, мы укажем их в явной форме, и тогда под литерой будем понимать предикатную букву с параметрами в скобках. Под атомом будем понимать элементарное выражение в логике предикатов предикат.

Определение 2.1. Дизъюнкт есть дизъюнкция литер.

Иногда удобно рассматривать дизъюнкт как множество литер, например, $P \vee Q \vee R = \{P, Q, R\}$. Дизъюнкт, содержащий g литер, называется g -литерным дизъюнктом. Однолитерный дизъюнкт называется единичным дизъюнктом. Когда дизъюнкт не содержит никаких литер, он называется пустым. Так как пустой дизъюнкт не содержит литер, которые могли бы быть истинными при любых интерпретациях, то пустой дизъюнкт всегда ложен. Будем обозначать пустой дизъюнкт символом 0.

Дизъюнкции $P(x, f(x)) \vee R(x, f(x), g(x))$ и $Q(x, g(x)) \vee R(x, f(x), g(x))$ в стандартной форме из вышеприведенного примера являются дизъюнктами.

Считается, что множество дизъюнктов S есть конъюнкция дизъюнктов из S , где каждая переменная в S считается связанной квантором всеобщности. Благодаря этому соглашению стандартная форма может быть просто представлена множеством дизъюнктов (мы затрагивали этот вопрос в п.4.4 первой главы, вводя сколемовскую и клаузальную формы формулы исчисления предикатов). Например, стандартная форма в вышеприведенном примере может быть представлена множеством $\{P(x, f(x)) \vee R(x, f(x), g(x)), Q(x, g(x)) \vee R(x, f(x), g(x))\}$.

Далее для множества дизъюнктов S вместо терминов «противоречиво/непротиворечиво» мы будем использовать термины «невыполнимо/выполнимо». По определению множество дизъюнктов невыполнимо тогда и только тогда, когда оно ложно при всех интерпретациях на всех областях. Так как неудобно и невозможно рассматривать все интерпретации на всех областях, надо решить вопрос о некоторой фиксированной специальной области такой, что множество дизъюнктов S невыполнимо тогда и только тогда, когда оно ложно при всех интерпретациях на этой области. Эрбрану удалось построить такую область, которая получила название эрбрановского универсума множества S .

Определение 2.2. Пусть N_0 множество констант, встречающихся в S . Если никакая константа не встречается в S , то N_0 состоит из одной константы, скажем $N_0 = \{a\}$. Пусть для $i = 0, 1, 2, \dots$ N_{i+1} есть объединение N_i и множества всех термов вида $f(t_1, t_2, \dots, t_n)$ (при всех n) для всех функций f , встречающихся в S , где t_j ($j=1, 2, \dots, n$) принадлежит N_i . Тогда каждое множество N_i называется множеством констант i -го уровня для S , а N , называется эрбрановским универсумом множества дизъюнктов S .

Пример 2.1. Пусть $S = \{P(a) \vee Q(x), R(g(z)), T(f(y)) \vee W(b)\}$.

Тогда эрбрановским универсумом для множества дизъюнктов S будут следующие множества

$$N_0 = \{a, b\};$$

$$N_1 = \{a, b, f(a), g(a), g(b)\};$$

$$N_2 = \{a, b, f(a), f(b), g(a), g(b), f(f(a)), f(f(b)), \dots, g(f(a)), g(f(b)), \dots\}.$$

Впоследствии под выражением будем понимать терм, множество термов, множество атомов, литеру, дизъюнкт и множество дизъюнктов. Если в выражении не встречаются никакие переменные, чтобы подчеркнуть этот факт, такое выражение будем называть основным выражением. Например, $P(x)$ атом, $P(fa)$ основной атом. Подвыражение выражения E есть выражение, которое встречается в E .

Определение 2.3. Пусть S есть множество дизъюнктов. Тогда множество основных атомов вида $P^n(t_1, t_2, \dots, t_n)$ для всех n -местных предикатов P^n , встречающихся в S , называется множеством атомов множества дизъюнктов S , или эрбрановским базисом S .

Пример 2.2. Эрбрановский универсум для множества дизъюнктов

$$S = \{P(a) \vee Q(x), R(g(z)), T(f(y)) \vee W(b)\}$$
 есть множество

$$N^\wedge = \{a, b, f(a), f(b), g(a), g(b), \dots\},$$
 а эрбрановский базис множество

$$A = \{P(a), Q(a), R(a), T(a), W(a), P(b), Q(b), \dots, P(f(a)), \dots, P(g(a)), \dots\}.$$

Определение 2.4. Основным примером дизъюнкта C множества дизъюнктов S называется дизъюнкт, полученный заменой переменных в C на члены эрбрановского универсума S .

Ясно, что основных примеров одного и того же дизъюнкта S бесконечно много.

Определение 2.5. Пусть S множество дизъюнктов, H эрбрановский универсум S и I интерпретация S над H . Говорят, что I есть Нинтерпретация множества S , если она удовлетворяет следующим условиям:

1. I отображает все константы множества дизъюнктов S в самих себя.
2. Пусть f n -местный предикат и $h_1, h_2, \dots, h_n$ элементы из множества H . В интерпретации I через f обозначается функция, которая отображает $(h_1, h_2, \dots, h_n)$ (элемент из H^n) в $f(h_1, h_2, \dots, h_n)$ (элемент из H).

При этом не возникает никаких ограничений при придании значения любому n -местному предикатному символу S .

3. Пусть $A = \{A_1, A_2, \dots, A_n, \dots\}$ эрбрановский базис множества дизъюнктов S . Нинтерпретацию I удобно записать в виде $I = \{M_1, M_2, \dots, M_n, \dots\}$, где M_i есть ли A_i , или A_i для $i = 1, 2, \dots$. Смысл введения это-

го множества состоит в том, что если M_i есть A_i , то атому A_i присвоено значение «истина», в противном случае «ложь».

Пример 2.3. Рассмотрим множество дизъюнктов $S = \{P(x) \vee Q(x), R(f(y))\}$. Эрбрановский универсум для этого

множества представляет собой множество

$$H = \{a, f(a), f(f(a)), \dots\}.$$

Во множество S входят три предикатных символа P , Q и R . Следовательно, эрбрановский базис для множества дизъюнктов S представляет собой множество

$$A = \{P(a), Q(a), R(a), P(f(a)), Q(f(a)), R(f(a)), \dots\}.$$

Примерами H -интерпретации для этого случая могут служить следующие множества

$$I_1 = \{P(a), Q(a), R(a), P(f(a)), Q(f(a)), R(f(a)), \dots\};$$

$$I_2 = \{P(a), Q(a), R(a), P(f(a)), Q(f(a)), R(f(a)), \dots\};$$

$$I_3 = \{P(a), Q(a), R(a), P(f(a)), Q(f(a)), R(f(a)), \dots\}.$$

Определение 2.6. Пусть I интерпретация над областью D . Нинтерпретацией I^* , соответствующей интерпретации I , называется интерпретация, которая удовлетворяет следующим условиям: пусть $h_1, h_2, \dots, h_n$ элементы из эрбрановского универсума H и пусть каждый элемент h_i отображается в некоторый элемент $d_i \in D$. Если $P(d_1, d_2, \dots, d_n)$ получает в интерпретации I^* значение

«истина» («ложь»), то $P(h_1, h_2, \dots, h_n)$ также получает значение со-

ответственно «истина» («ложь») в интерпретации I . Имеет место следующая лемма.

Лемма 2.1. Если интерпретация I на некоторой области D удовлетворяет множеству дизъюнктов S , то любая H -интерпретация I^* , соответствующая интерпретации I , удовлетворяет множеству дизъюнктов S .

Теорема 2.1. Множество дизъюнктов S невыполнимо тогда и только тогда, когда S ложно при всех своих Нинтерпретациях.

Доказательство. Истинность первой части теоремы очевидна, так как по определению множество дизъюнктов S невыполнимо тогда и только тогда, когда оно невыполнимо при всех интерпретациях на любой области. Чтобы доказать вторую часть теоремы, предположим, что множество дизъюнктов S ложно при всех H -интерпретациях. Предположим, что S выполнимо. Тогда существует такая интерпретация I на некоторой области D , что S истинно в I . Пусть I^* есть соответствующая интерпретации I H -интерпретация. Согласно лемме 2.1 множество S истинно в интерпретации I^* . Это противоречит предположению о том, что S ложно при всех Нинтерпретациях, следовательно, S должно быть невыполнимо, что и требовалось доказать.

На основании доказанной теоремы мы можем рассматривать множество дизъюнктов S только в Нинтерпретациях, сохраняя при этом противоречивость S .

Следствие 2.1. Основной пример S' дизъюнкта S выполняется в интерпретации I

тогда и только тогда, когда существует основная литера L' в S' такая, что L' есть также и в I , то есть $S' \cap L' \neq \emptyset$.

Следствие 2.2. Дизъюнкт S выполняется в интерпретации I тогда и только тогда, когда каждый основной пример S выполняется в интерпретации I .

Следствие 2.3. Дизъюнкт S опровергается интерпретацией I тогда и только тогда, когда существует, по крайней мере, один такой основной пример S' для S , что S' не выполняется в интерпретации I .

Следствие 2.4. Множество дизъюнктов S невыполнимо тогда и только тогда, когда для каждой интерпретации I существует, по крайней мере, один такой основной пример S' некоторого дизъюнкта S в S , что S' не выполняется в интерпретации I .

Теорема 2.2 (Эрбран). Множество дизъюнктов S невыполнимо тогда и только тогда, когда существует конечное невыполнимое множество S' основных примеров дизъюнктов S .

Мы не приводим доказательства этой теоремы, а лишь отметим, что согласно теореме Эрбрана для опровержения множества дизъюнктов S нам нужно иметь машинную процедуру, которая может успешно порождать множества основных примеров дизъюнктов из S и успешно устанавливать их невыполнимость. Эта процедура, как гарантируется теоремой Эрбрана, укажет нам такое конечное число N , что множество S'_N будет невыполнимым. Эту идею одним из первых

применил на практике Гилмор. В 1960 г. он написал машинную программу, которая успешно порождала множества всех основных примеров заменой переменных на элементы эрбрановского универсума и проверяла их методами логики высказываний на невыполнимость. Но метод, предложенный Гилмором, оказался неэффективным и практически не реализуем для сложных формул логики первого порядка. Его оптимизировали Девис и Патнем. Их модификация заключалась в следующих четырех основных правилах.

I. Правило тавтологии. Вычеркнем все тавтологические основные дизъюнкты из множества S . Оставшееся множество S' невыполнимо тогда и только тогда, когда невыполнимо S .

II. Правило однолитерных дизъюнктов. Если существует единичный основной дизъюнкт L в S , то S' получается из S вычеркиванием тех основных дизъюнктов в S , которые содержат L . Если S' пусто, то S невыполнимо. В противном случае построим множество S'' , выбрасывая из S' все вхождения L . S'' невыполнимо тогда и только тогда, когда S невыполнимо. Отметим, что если L единичный основной дизъюнкт, то при вычеркивании он превратится в $\emptyset$.

III. Правило чистых литер. Назовем литеру L в основном дизъюнкте из множества S чистой в S тогда и только тогда, когда литера L не появится ни в каком основном дизъюнкте S . Если литера L чистая в S , то вычеркнем все основные дизъюнкты, содержащие L . Оставшееся множество S' невыполнимо тогда и только тогда, когда множество S невыполнимо.

IV. Правило расщепления. Если множество S представить в виде

$$(A_1 \vee L) \& \dots \& (A_m \vee L) \& (B_1 \vee L) \& \dots \& (B_n \vee L) \& R,$$

где A_i , B_j и R свободны от L и $\neg L$, то получим множества

$$S_1 = A_1 \& A_2 \& \dots \& A_m \& R,$$

$$S_2 = B_1 \& B_2 \& \dots \& B_n \& R.$$

Тогда множество S невыполнимо тогда и только тогда, когда множество $(S_1 \vee S_2)$ невыполнимо, то есть множества S_1 и S_2 невыполнимы. Процедуры доказательства, основанные на теореме Эрбрана, имеют один существенный недостаток: они требуют порождения множеств основных примеров дизъюнктов S_1 , S_2 и т.д. В большинстве случаев эта последовательность растет экспоненциально. Чтобы избежать порождения множества основных примеров, как требуется в процедурах по Эрбрану, Дж. Робинсон предложил метод резолюций, который, вообще говоря, применим к любому множеству

дизъюнктов (необязательно основных).

Основная идея метода резолюций состоит в том, чтобы проверить, содержит ли множество S пустой дизъюнкт. Если S содержит пустой дизъюнкт, то проверяется, может ли пустой дизъюнкт быть получен из множества S . Метод резолюции, в сущности это обобщение правила однолитерных дизъюнктов Девиса и Патнема. Сформулируем теперь правило резолюций для исчисления высказываний:

Если для любых двух дизъюнктов C_1 и C_2 существует литера L_1 в дизъюнкте C_1 , которая инверсна (контрарная) литере L_2 в C_2 , то, вычеркнув L_1 и L_2 из C_1 и C_2 соответственно, мы построим дизъюнкцию оставшихся дизъюнктов. Построенный дизъюнкт называется резольвентой дизъюнктов C_1 и C_2 .

Пример 2.4. Рассмотрим дизъюнкты

$C_1 : P \vee R$,

$C_2 : P \vee Q$.

Дизъюнкт C_1 имеет литеру P , которая контрарная литере P в дизъюнкте C_2 . Следовательно, вычеркивая P и P из дизъюнктов C_1 и C_2 соответственно и составляя дизъюнкцию оставшихся дизъюнктов R и Q , мы получим резольвенту $R \vee Q$.

Теорема 2.3. Пусть даны два дизъюнкта C_1 и C_2 . Тогда резольвента C дизъюнктов C_1 и C_2 есть логическое следствие дизъюнктов C_1 и C_2 .

Доказательство. Пусть C_1 , C_2 и C означают следующее:

$C = L \vee C'_1$, $C = L \vee C'_2$, $C = C'_1 \vee C'_2$,

$\begin{matrix} 1 & 1 & 2 & 2 & 1 & 2 \\ C_1 & C_2 & C_1 & C_2 & C_1 & C_2 \end{matrix}$

где C'_1 и C'_2 дизъюнкции литер. Предположим, что дизъюнкты C_1 и C_2

$\begin{matrix} 1 & 2 \\ C_1 & C_2 \end{matrix}$

истинны в интерпретации I . Нужно доказать, что резольвента C дизъюнктов C_1 и C_2 также истинна в интерпретации I . Чтобы доказать это, отметим, что L или L ложно в интерпретации I . Предположим, что L ложно в интерпретации I . Тогда дизъюнкт C_1 не должен быть единичным дизъюнктом, иначе он был бы ложный в I . Следовательно, C'_1 должен быть истинен в I . Таким образом, резольвента C истинна в интерпретации I . Аналогично можно доказать истинность резольвенты C в случае, когда L истинно.

Замечание. Если мы рассматриваем пару единичных дизъюнктов, то их резольвента (если она существует), всегда является пустым дизъюнктом.

Определение 2.7. Пусть S множество дизъюнктов. Резолютивным выводом C из множества S называется такая конечная последовательность $C_1, C_2, \dots, C_k$ дизъюнктов, что каждый дизъюнкт C_i

либо принадлежит множеству S , либо является резольвентой дизъюнктов, предшествующих дизъюнктам C_i и $C_k = C$. Вывод пустого дизъюнкта из множества S называется опровержением (или доказательством невыполнимости) множества S .

Будем говорить, что дизъюнкт C может быть выведен или получен из множества S , если существует вывод этого дизъюнкта из S .

Пример 2.5. Пусть множество S состоит из дизъюнктов:

1) $P \vee Q$,

3) $P \vee Q$,

4) $P \vee Q$.

Вычеркивая из дизъюнктов 1) и 3), 2) и 4) соответственно литеры P и P , получим резольвенту $Q \vee Q$, которая, очевидно, является пустым дизъюнктом. Выводом здесь является последовательность дизъюнктов $Q, Q, Q \vee Q, 0$.

Правило резолюций для исчисления высказываний это очень мощное правило вывода. Некоторые рассмотренные нами ранее правила вывода являются частными случаями правила резолюций, например:

1. Правило modus ponens
2. Правило силлогизма $\Delta B \vdash B \Delta C$;
3. Правило слияния

Правило резолюций для исчисления предикатов

Основной задачей логики предикатов является установление истинности предикатных тождеств и клауз. Обозначим через P_i какойлибо предикат с произвольным числом аргументов, а через q_i соответствующую ему кванторную группу. Тогда, например, закон дистрибутивности дизъюнкции относительно конъюнкции примет вид:

$$q_1^{P_1} \vee (q_2^{P_2} \& q_3^{P_3}) = (q_1^{P_1} \vee q_2^{P_2}) \& (q_1^{P_1} \vee q_3^{P_3}).$$

В том, что он выполняется для одноместных предикатов, можно убедиться с помощью процедуры унификации (конкретизации). Действительно, пусть, например:

$$\begin{array}{llll} x = & q_1 = \forall x & P_1 = \\ a, b & & \Delta(\forall). \\ y = & q_2 = z & P_2 = B(\\ c, d, & y, & y); \\ z = & q_3 = \forall z & P_3 = \\ e, f & & \Gamma(\exists) \end{array}$$

Тогда

$$\begin{aligned} & [A(a) \& A(b)] \vee ([B(c) \vee B(d)] \& [C(e) \& C(f)]) = \\ & = ([A(a) \& A(b)] \vee [B(c) \vee B(d)]) \& ([A(a) \& A(b)] \vee [C(e) \& C(f)]). \end{aligned}$$

От того, что в квадратных скобках вместо дизъюнкции появится конъюнкция и наоборот, а также вместо одноместных предикатов будут фигурировать различные многоместные предикаты суть тождества не изменится. Оно останется истинным в силу справедливости законов операций над множествами (операций над областями истинности предикатов) и принципа суперпозиции, который гласит: замена какой-либо константы другой константой или даже группой констант не может повлиять на истинность тождества. Именно отсюда проистекает наша уверенность в справедливости законов операций над множествами и по отношению к предикатам.

Те же самые рассуждения могут быть проведены и в отношении логики высказываний. Она отличается от алгебры множеств аксиомой порядка $q_1 P_1, q_2 P_2 \wedge q_1 P_1$.

Процедура конкретизации сводит предикатную аксиому порядка к простому высказыванию, так что если клауза верна для высказываний, она будет справедлива и для предикатов.

Как показывает практика, много ошибок приходится на неправильное обозначение предикатов при их, в принципе правильной идентификации. Следует помнить, что положительный предикат должен иметь больше единиц, чем отрицательный. Будет допущена ошибка, если противоречие $A_1, A_2 \wedge 0$ записать как $A_2, A_1 \wedge 0$.

Основная сложность переноса правила резолюций в исчисление предикатов первого порядка заключается в особенности поиска контрарных литер. Рассмотрим пример. Пусть

$$C_1 : P(x) \vee Q(x),$$

$$C_2 : P(f(x)) \vee R(x).$$

Формально говоря, в данном примере не существует никакой пары контрарных литер, позволяющей нам применить правило резолюций. Тем не менее, если мы выполним подстановку

$$C_1 : x \wedge f(a),$$

$$C_2 : x \wedge a,$$

то получим

$$C_1 : P(f(a)) \vee Q(f(a)),$$

$$C_2 : P(f(a)) \vee R(a).$$

Теперь у нас есть пара контрарных литер, и мы можем применить правило резолюций.

Легко заметить, что подстановка, приводящая пару дизъюнктов к виду, к которому применимо правило резолюций, не единственна. Поэтому имеет смысл поставить вопрос о наиболее общей подстановке для данной пары дизъюнктов, а также вопрос об условии,

при котором существует такая подстановка.

Определение 2.8. Подстановка это конечное множество вида $\{t_1 / v_1, t_2 / v_2, \dots, t_n / v_n\}$, где для всех $i = 1, n$ V_i — переменная, t_i терм, отличный от v_i , и все переменные v_i различны. Если все t_i основные термы, то подстановка называется пустой и обозначается символом s .

Пример 2.6. Следующее множество является подстановкой: $\{a / x, g(y) / y, f(g(b)) / z\}$.

Определение 2.10. Пусть $O = \{t_i / V_1, t_2 / v_2, \dots, t_n / v_n\}$

Определение 2.9. Пусть $O = \{t_1 / v_1, t_2 / v_2, \dots, t_n / v_n\}$ подстановка и E выражение. Тогда запись E_O означает выражение, полученное из выражения E заменой одновременно всех вхождений переменной v_i ($1 < i < n$) в E на терм t_i . Выражение E_O называется примером E .

$X = \{U_1 / y_1, U_2 / y_2, \dots, U_n / y_n\}$ две подстановки. Тогда композицией подстановок O и X называется подстановка $O \circ X$, которая получается из множества

$$\{t_i X / x_i, \dots, t_n X / x_n, U_i / y_i, \dots, U_n X / y_n\}$$

вычеркиванием всех элементов $t_j X / x_j$, для которых $t_j X / x_j = x_j$ и всех элементов U_i / y_i , таких, что $U_i \in \{x_1, x_2, \dots, x_n\}$.

В процедуре доказательства по методу резолюций для того, чтобы отождествлять контрарные пары литер, мы часто должны унифицировать (склеивать) два или более выражений, то есть мы должны находить подстановку, которая может сделать несколько выражений тождественными.

Определение 2.11. Подстановка O называется унификатором для множества выражений $\{E_1, E_2, \dots, E_k\}$ тогда и только тогда, когда

$$E_1 O = E_2 O = \dots = E_k O.$$

Говорят, что множество выражений $\{E_1, E_2, \dots, E_k\}$ унифицируемо, если для него существует унификатор.

Определение 2.12. Унификатор C для множества выражений $\{E_1, E_2, \dots, E_k\}$ будет наиболее общим унификатором тогда и

только тогда, когда для каждого унификатора O для этого множества существует такая подстановка X , что $O = C \circ X$.

Использование унификатора позволяет ввести формальное определение метода резолюций для логики предикатов первого порядка.

Определение 2.13. Если две или более литер (с одинаковым знаком) дизъюнкта C имеют наиболее общий унификатор S , то CS называется склейкой дизъюнкта C . Если при этом CS единичный дизъюнкт, то склейка называется единичной.

Пусть C_1 и C_2 два дизъюнкта (называемые дизъюнктивными посылками), которые не имеют никаких общих переменных. Пусть L_1 и L_2 две литеры в C_1 и C_2 соответственно. Если L_1 и L_2 имеют наиболее общий унификатор S , то дизъюнкт $(CS_1 \cup CS_2)$ и $(CS_1 \cup L_2)$ является бинарной резольвентой C_1 и C_2 . Литеры C_1 и C_2 называются отрезаемыми литерами. Определение 2.14. Резольвентой дизъюнктов C_1 и C_2 называется одна из следующих резольвент:

- 1) бинарная резольвента C_1 и C_2 ;
- 2) бинарная резольвента C_1 и склейки C_2 ;
- 3) бинарная резольвента склейки C_1 и C_2 ;
- 4) бинарная резольвента склейки C_1 и склейки C_2 .

Правило резолюций есть правило вывода, которое порождает резольвенты для множества дизъюнктов. Это правило было выведено в 1965 г. Робинсоном. Оно более эффективно, чем предыдущие процедуры доказательства такие, как прямое применение теоремы Эрбрана, используемые Гилмором, и процедуры Девиса и Патнэма. Кроме того, метод резолюций полон, то есть при помощи правила резолюций для любого невыполнимого множества дизъюнктов можно породить пустой дизъюнкт.

Рассмотрим пример применения метода резолюций к одной из задач планиметрии.

Пример 2.7. Показать, что внутренние накрест лежащие углы, образованные диагональю трапеции, равны.

Решение. Чтобы доказать эту теорему, мы сперва аксиоматизируем ее. Пусть $T(x, y, u, w)$ обозначает, что xuw трапеция с левой верхней вершиной x , правой верхней вершиной y , правой нижней вершиной u и левой нижней вершиной w . Пусть $P(x, y, u, w)$ обозначает, что отрезок xu параллелен uw , и пусть $E(x, y, z, u, v, w)$ означает, что

$$\angle xyz = \angle uvw.$$

Тогда мы имеем следующие аксиомы:

$$A1 : \forall x \forall y \forall u \forall v (T(x, y, u, v) \wedge P(x, y, u, v));$$

$$A2 : \forall x \forall y \forall u \forall v (P(x, y, u, v) \wedge E(x, y, v, y, v, u));$$

$$A3 : T(a, b, c, d).$$

Из этих аксиом мы должны вывести, что $E(a, b, d, b, d, c)$ истинно, то есть

$$A1 \wedge A2 \wedge A3 \vdash E(a, b, d, b, d, c).$$

Используя соответствующие алгоритмы, найдем стандартную форму и выпишем множество дизъюнктов:

$$S = \{T(x, y, u, v) \vee P(x, y, u, v), P(x, y, u, v) \vee E(x, y, v, y, v, u), T(a, b, c, d), E(a, b, d, b, d, c)\}.$$

Теперь методом резолюций докажем невыполнимость этого множества дизъюнктов.

- 1) $T(x, y, u, v) \vee P(x, y, u, v);$
- 2) $P(x, y, u, v) \vee E(x, y, v, y, v, u);$
- 3) $T(a, b, c, d);$
- 4) $E(a, b, d, b, d, c);$
- 5) $P(a, b, c, d)$ резольвента из 2) и 4);
- 6) $T(a, b, c, d)$ резольвента из 1) и 5);
- 7) $\emptyset$ резольвента из 3) и 6).

Так как последний выведенный дизъюнкт является пустым, то множество дизъюнктов S невыполнимо, а, следовательно, исходная формула является теоремой.

Имеет место следующая теорема (о полноте метода резолюций). Множество дизъюнктов S невыполнимо тогда и только тогда, когда существует вывод пустого дизъюнкта из множества S . Доказательства этой теоремы мы не приводим.

2. 4. Алгоритм поиска опровержения методом резолюций

Для доказательства теорем с использованием правила резолюций поступают следующим образом. Преобразуют исходную формулу к множеству дизъюнктов S_0 . Далее строят всевозможные резольвенты $R = \{R_1, R_2, \dots, R_n\}$ из дизъюнктов, принадлежащих множеству S_0 . Далее

получаем множество $S_1 = S_0 \cup R$ и повторяем все сначала до тех пор, пока не получим множество S_m , содержащее пустой дизъюнкт.

Очевидно, что при таком подходе мы будем порождать наряду с дизъюнктами, которые дадут в итоге пустой дизъюнкт, массу лишних дизъюнктов, что, бесспорно, увеличит объем вычислений. Поэтому различными авторами были предложены стратегии вычеркивания лишних дизъюнктов. Прежде всего, из вновь полученного множества дизъюнктов надлежит удалить все тавтологии и дизъюнкты, которые являются логическим следствием каких-либо других дизъюнктов. Это, конечно, снижает количество вычислений, но мы по-прежнему ищем доказательство наугад.

Для решения этой проблемы было предложено множество различных стратегий метода резолюций. Эти стратегии предназначены в основном для порождения резольвент, сужающих область поиска доказательства (или опровержения по методу резолюций). Одни из них обладают свойством полноты, другие нет. Из предложенных стратегий отметим полные и неполные стратегии, семантическую резолюцию, лок-резолюцию и линейную резолюцию.

Полные стратегии это такие стратегии, которые гарантируют нахождение доказательства теоремы, если оно вообще существует.

Неполные стратегии могут в некоторых случаях не находить доказательства, зато они работают быстрее.

Идея семантической резолюции состоит, прежде всего, в разбиении исходного множества дизъюнктов на классы, которые не пересекаются, и обе посылки резолюции принадлежат различным классам. Для разделения множества дизъюнктов на классы используется аппарат эрбрановского универсума. Основой метода является тот факт, что контрарные литеры не могут быть одновременно истинными в одной и той же интерпретации. Еще одним способом сократить количество вычислений по методу семантической резолюции является упорядочение литер. Оно подразумевает, что отрезаемыми литерами могут быть только последние литеры (по порядку) обоих дизъюнктов. К недостаткам семантической резолюции можно отнести громоздкую процедуру разбиения на классы.

В лок-резолюции используется индексирование литер для их упорядочения и разрешается отрезать литеры только с наименьшими индексами в каждом из дизъюнктов. К недостаткам лок-резолюции относится зависимость количества вычислений от выбранной индексации литер.

Доказывая данное равенство, мы часто начинаем с его левой части, применяя к ней правило вывода для получения какого-то другого выражения. Затем снова применяем некоторое правило вывода к только что полученному выражению и т.д. до тех пор, пока левая часть выражения не совпадет с выражением в правой части равенства. Идея линейной резолюции аналогична такого рода цепочке рассуждений. Линейный вывод начинается с некоторого дизъюнкта, применяет к нему и другому дизъюнкту правило резолюций для получения резольвенты и применяет резолюцию к этой резольвенте и некоторому другому дизъюнкту, пока не будет получен пустой дизъюнкт. Линейная резолюция независимо друг от друга была предложена Лавлендом и Лакхемом, позднее усилена Андерсоном, Бледсоу, Ейтсом и др., а также Ковальским и Кюннером. Варианты линейной резолюции, предложенные Лавлендом, Ковальским и Кюннером, наиболее легко реализуются на ЭВМ.

Рассмотрим в заключение алгоритм опровержения методом резолюций. Опровержение методом резолюций это алгоритм автоматического доказательства теорем в прикладном исчислении предикатов, который, как мы видели, сводится к следующему рассуждению. Пусть нужно установить выводимость $S \vdash G$. Каждая формула из множества S и формула G (отрицание целевой теоремы) независимо преобразуются во множества дизъюнктов. В полученном совокупном множестве дизъюнктов S отыскиваются резольвируемые дизъюнкты, к ним применяется правило резолюций, а резольвента добавляется во множество S до тех пор, пока не будет получен пустой дизъюнкт. При этом возможны три случая:

1. Среди текущего множества дизъюнктов нет резольвируемых. Это означает, что теорема опровергнута, то есть формула G не выводима из множества формул S .
2. В результате очередного применения правила резолюций получен пустой дизъюнкт. Это означает, что теорема доказана, то есть $S \vdash G$.
3. Процесс не заканчивается, то есть множество дизъюнктов пополняется все новыми резольвентами, среди которых нет пустых. Это ничего не означает.

Если этот алгоритм заканчивает свою работу, то правильность результата гарантируется методом резолюций. Вообще говоря, завершение работы алгоритма не гарантировано. Следует иметь в виду, что автоматическое доказательство теорем методом резолюций имеет по существу переборный характер и этот перебор бывает столь велик, что может быть практически неосуществим за приемлемое время. При автоматическом доказательстве теорем методом резолюций львиная доля вычислений приходится на поиск унифицируемых дизъюнктов. Таким образом, эффективная реализация алгоритма унификации критически важна для практической применимости метода резолюций.

Метод доказательства, основанный на принципе резолюций, представляет интерес,

так как является систематическим. Большое число публикаций, трактующих вопросы автоматического доказательства теорем, посвящено методу резолюций, начиная со времени появления работ Эрбрана и реализации этой идеи Робинсоном в 1965 г. Невозможность определить эффективную стратегию решения для всех областей приложения приводит к следующему выводу: для заданного набора предложений либо доказательство получается очень быстро на ЭВМ, либо оно не получается вовсе. Однако эффективность программ резко падает с увеличением числа предложений. Проблема заключается в том, чтобы определить тот момент, когда надо остановить программу, поскольку существуют две опасности: во-первых, взрывное увеличение числа предложений, а во-вторых, возможность преждевременного прекращения работы перед самым моментом получения решения.

Практическая значимость и оригинальность метода резолюций определяются следующими его двумя свойствами.

1. Управление движением по дереву «и-или», обычно необходимое при всяком доказательстве, в данном случае отсутствует. Эту функцию выполняет стандартизованный набор дизъюнктов, причем может быть определена простая процедура обработки «выбор + унификация».

2. Гипотезы, предположение и промежуточные предложения играют здесь одну и ту же роль. Доказательства получаются при одновременном движении в двух направлениях: от гипотез к заключениям и от заключений к гипотезам, причем доказательство достигается, когда оба эти пути просмотра встречаются друг с другом.

Существенным недостатком метода резолюций является то, что уже по построению он предназначен только для доказательства теорем. Он не пригоден для того, чтобы изобретать или придумывать некоторые предложения, а затем уже доказывать их, как это обычно делает математик. К тому же, если предложение не является теоремой, резолюция может привести к построению бесконечного дерева решения. Таким образом, метод резолюций интересен благодаря простоте и систематичности, но, к сожалению, применим только для ограниченного числа случаев (доказательство не должно иметь большую глубину, а число потенциальных резолюций не должно быть большим).

3. 5. Доказательство истинности логических клауз методом резолюций

В булевой алгебре все доказательства строились на отношении эквивалентности. Две логические функции считаются эквивалентными, если они дают на соответствующих наборах аргументов абсолютно одинаковые значения нулей и единиц. При использовании формальной записи логических выражений отдельные звенья цепи любого доказательства там были связаны символом равенства «=». В исчислении высказываний все доказательства строятся на отношении порядка, которое устанавливается между причиной и следствием. Здесь уже отдельные звенья цепи доказательства связаны символом импликации « $\wedge$ ». Как и в алгебре Буля, в исчислении высказываний существует аксиоматический и конструктивный подходы доказательств логических выражений. Аксиоматическое построение исчисления высказываний состоит в том, чтобы попытаться вычленив из бесконечного числа истинных формул независимую систему аксиом, с помощью которой можно было бы устанавливать справедливость любых других формул. Исчисление высказываний, как мы уже знаем, является расширением алгебры высказываний. Другими словами, алгебра высказываний служит одной из интерпретаций исчисления высказываний.

К конструктивному подходу доказательства логических выражений относится метод составления таблиц истинности.

Кроме аксиоматического и конструктивного методов доказательства логических выражений существуют еще смешанные методы доказательства, среди которых особое место занимает рассмотренный нами метод резолюций.

Основной задачей логики предикатов первого порядка является установление истинности предикатных тождеств и формул. Процедура унификации (конкретизации)

сводит предикатную аксиому порядка к простому высказыванию, так что, если формула верна для высказываний, она будет справедливой и для предикатов. Напомним, что основная сложность переноса правила резолюций в исчисление предикатов первого порядка заключается в особенности поиска контрарных литер. В процедуре доказательства по методу резолюций для того, чтобы отождествлять контрарные пары литер, мы часто должны унифицировать (склеивать) два или более выражений, то есть мы должны находить подстановку, которая может сделать несколько выражений тождественными. Правило резолюций есть правило вывода, которое порождает резольвенты для множества дизъюнктов.

Для доказательства теорем с использованием правила резолюций преобразуют исходную формулу к множеству хорновских дизъюнктов S_0 . Далее строят всевозможные резольвенты $R = \{R_1, R_2, \dots, R_n\}$ из

дизъюнктов, принадлежащих множеству S_0 . Далее получаем множество $S_1 = S_0 \cup R$ и повторяем все сначала до тех пор, пока не получим множество S_m , содержащее пустой дизъюнкт.

При логическом выводе символ импликации « $\wedge$ » мы будем заменять на символ « $\supset$ ». Символ « $\supset$ » является объектным, а символ « $\wedge$ » субъектным. Таким образом, следует различать язык логики высказываний и метаязык исследователя. Во избежание путаницы введем еще два метасимвола: вместо объектной конъюнкции « $\&$ » будем использовать субъектный символ метаконъюнкции « $\cdot$ », а вместо объектной дизъюнкции « $\vee$ » субъектную метадизъюнкцию « $\vee$ ». Тогда утверждение, которое требуется доказать, в исчислении высказываний оформляется в виде следующего причинно-следственного отношения:

$$P_1, P_2, \dots, P_{n-1}, P_n \supset C, \quad (2.12)$$

где P_i посылка (причина), C заключение (следствие). Читается: «Если посылки $P_1, P_2, \dots, P_{n-1}, P_n$ истинны, то заключение C тоже истинно» или, по-другому: «Если причины $P_1, P_2, \dots, P_{n-1}, P_n$ имели место, то будет иметь место и следствие C ».

Чтобы не перепутать объектное высказывание (предложение) с субъектным высказыванием, справедливость которого мы намереваемся установить, условимся предложения типа (2.12) называть клаузой. Клауза метаясуждение, в котором использовано отношение порядка, оформленное через символ метаимпликации « $\supset$ ». Как и отношение эквивалентности, отношение порядка удовлетворяет трем аксиомам (законам):

транзитивности:
антисимметричности: если $A \supset B$, то $B \supset A$;

рефлексивности: $A \supset A$;
если $A \supset B$ и $B \supset C$, то $A \supset C$.

Закон антисимметричности можно записать и так: если $A \supset B$ и $B \supset A$, то $A = B$.

Клауза есть именно формальная запись доказываемого предложения. Вместо букв в ней можно подставить объектные высказывания, и тогда клауза наполняется конкретным содержанием, которое называется семантикой или легендой. Пример клаузы:

$$A \supset B, A \supset B.$$

Если принять, что A = «сверкнула молния», B = «грянул гром», то можно составить следующую легенду: «Известно, что если сверкнула молния, то после этого грянет гром. Молния сверкнула. Следовательно, должен и грянуть гром».

Над субъектом, который формулирует метаясуждение, может стоять другой субъект, для которого уже предложения первого субъекта окажутся объектными. Тогда клаузу (2.12) второй субъект или метасубъект запишет для себя следующим логическим выражением: $(P_1 \& P_2 \& \dots \& P_{n-1} \& P_n) \supset C$.

Преобразовав это выражение в дизъюнкт, получим:

$$P_1 \vee P_2 \vee \dots \vee P_{n-1} \vee P_n \vee C.$$

Отсюда легко находим:

$$(P_1 \& P_2 \& \dots \& P_{n-1}) \wedge (P_n \vee C).$$

Поэтому клауза (2.12) может быть представлена в другой эквивалентной форме:

$$P_1, P_2, \dots, P_{n-1} \wedge P_n; C. \quad (2.13)$$

В силу коммутативности конъюнкции на месте посылки P_n может оказаться любая другая, причем не одна. Например, клауза

$$P_1, P_2, P_3, P_4 \wedge C_1; C_2; C_3$$

может быть преобразована в другую, эквивалентную форму:

$$P_4, C, P_1, C \wedge P_3; P_2. \quad (2.14)$$

Однако клауза (2.12) по сравнению с (2.13) и другими подобными формами, имеет определенные преимущества и, в частности, используется в языке логического программирования ПРОЛОГ. Ее называют хорновской. Произвольную клаузу всегда можно свести путем равносильных преобразований к хорновскому виду.

Если символ метаимпликации « $\wedge$ » клаузы (2.12) сместить в крайнее левое положение, то она превратится в тавтологию; если же его сместить в крайнее правое положение, то она превратится в противоречие:

$$1 \wedge P_1; P_2; \dots; P_{n-1}; P_n; C \text{ — тавтология,}$$

$$P_1, P_2, \dots, P_{n-1}, P_n, C \wedge 0 \text{ противоречие.}$$

Метод доказательства истинности логических клауз, в котором используется принцип резолюций, относится к полуконструктивным методам доказательства. Принцип резолюций играет роль аксиомы порядка и вместе с тем порождает очень эффективную конструктивную процедуру. Таким образом, суть этого принципа сводится к тому, что два посылочных дизъюнкта с противоположными литерами (термами) всегда можно склеить в один заключительный дизъюнкт (резольвенту), в котором уже не будет противоположных литер:

$X \vee A, Y \vee A \wedge X \vee Y$, где X и Y произвольные термы или дизъюнкты с любым набором термов, включая пустой дизъюнкт 0 (в дальнейшем мы его будем обозначать нулем «0»); A и A произвольные термы.

При последовательном применении метода резолюций происходит уменьшение числа литер, вплоть до их полного исчезновения. При этом исходная клауза конструируется в форме конъюнктивного противоречия:

$$P_1, P_2, \dots, P_n \wedge 0.$$

Некоторые рассмотренные нами правила вывода исчисления высказываний являются частными случаями правила резолюций. Докажем, например, с помощью метода резолюций правило *modus ponens*, которое во введенной нами здесь символике записывается следующим образом:

$$A, A \wedge B \wedge B \text{ или } 0 \vee A, A \vee B, B \vee 0 \wedge 0.$$

Здесь имеется три дизъюнкта. Склеивая их последовательно, получаем в результате пустой дизъюнкт (0), который говорит о несовместимости заключения и посылок. Это как раз и свидетельствует об истинности исходной клаузы.

Принцип резолюций целиком заменяет аксиому порядка, поскольку сама эта аксиома может быть доказана в рамках метода резолюций. Действительно,

$$A, B \wedge A, A, B, A, 0, 0, B \wedge 0.$$

Следует обратить внимание на то, что посылка B здесь вообще не используется. Это необходимо иметь в виду: необязательно использовать все посылки (их число часто бывает избыточным) главное получить пустой дизъюнкт.

Пусть дана клауза $A \wedge B, C \vee A, B \wedge C \wedge A$. Доказательство ее справедливости следует начать с приведения ее в НКФ:

$$A \vee B, C \vee A, B \vee C, A \wedge 0.$$

$$\begin{array}{ccc} A & & \\ C & & \\ \vee A & & A \end{array} \quad \begin{array}{ccc} 9. & (2, & \\ & & 5) \end{array}$$

$\neg C$	B	10.	(3,
$\neg A$	C	11.	6) (3,
A	(1	12.	(4,
$\neg B$,	(1	13.	(4,
A	(2	14.	(4,
$\neg C$,	(2		

Подобная стратегия поиска пустого дизъюнкта очень непродуктивна. Если к этой же задаче подойти более творчески, то пустой дизъюнкт обнаружится на четвертом шаге от начала поиска:

5. B (1, 4), 6. C (2, 4), 7. B (3,6), 8. 0 (5, 7).

До сих пор мы использовали форму конъюнктивного противоречия. Исходя из принципа двойственности, метод резолюции можно сформулировать относительно дизъюнктивной тавтологии, при этом принцип резолюций, естественно изменится:

$X \vee Y \wedge X \wedge A; Y \wedge A$.

Докажем одну и ту же клаузу двумя способами в форме противоречия и в форме тавтологии. Пусть дана клауза:

$A, A \vee B, B \vee (A \vee C) \vee C$.

Доказательство в форме противоречия выглядит следующим образом:

$A, A \vee B, B \vee A \vee C, C \wedge 0$.

1 2 3 4

Склейки: 5. B (1, 2), 6. $A \vee C$ (1, 6), 7. C (3, 5), 8. 0 (4, 7).

Доказательство в форме тавтологии выглядит аналогично:

$1 \wedge A; A \wedge B; B \wedge A \wedge C; C$.

Склейки: 5. B (1, 2), 6. $A \wedge C$ (1, 6), 7. C (3, 5), 8. 1 (4, 7).

Одним из недостатков метода резолюций является то, что исходная клауза обязательно должна быть представлена в нормальной дизъюнктивной или конъюнктивной форме. Выводы

При использовании мета-, мета-мета и мета-мета-мета-теорем программы автоматического доказательства теорем сталкиваются с проблемой комбинаторного взрыва, связанного с необходимостью хранения большого количества информации. Для устранения этого недостатка были запрограммированы три общие идеи:

- Выражение считается тем более интересным, чем меньше символов оно содержит.
- Продукция (порождающее правило, то есть метатеорема с одним или несколькими посылками) считается тем более интересной, чем ближе посылки к заключению.
- Лучше затратить время на отбор выражений для архива, чем терять его в бесполезных проверках выражений.

Продукции обрабатываются особенно просто. Необходимо применять строгие меры, чтобы не хранить много различных версий некоторой наиболее общей формы, не производить за счет транзитивности дубликатов уже известных результатов. Например, если уже получено, что $a \wedge b$ и $b \wedge c$, то не нужно хранить в памяти выражение $a \wedge c$, так как программа сама легко получит этот результат. В противоположность этому результаты, полученные при использовании продукций (которые дают простой результат после значительно более сложных расчетов), всегда сохраняются.

Полезность какой-либо продукции оценивается в зависимости от того, что уже порождено, и тех выражений, которые могут быть еще произведены. Надо отметить, что результаты, которые могли бы быть отвергнуты вначале, затем сохраняются и аналогично, результаты, отвергнутые как бесперспективные в одной системе, могут быть оценены как интересные в другой. Программа заносит в архив только наиболее простые продукции в

данный момент, то есть те продукции, которые содержат наибольшее число операторов и пропозициональных переменных.

Случай с теоремами является более сложным, так как если продукции по их определению состоят из посылок и следствия, что позволяет легко производить сопоставление, то теоремы содержат лишь последовательности символов данного языка. Реальная полезность теоремы заключается в том, что она связана с возможными интерпретациями данной формальной системы. Поэтому связи между языком и метаязыком, возникающие в процессе проверки продукции какойлибо теоремы, рассматриваются только с этой точки зрения. Вопросы и упражнения для самопроверки

2.1. Приведенные ниже клаузы доказать методом резолюций:

2.1.1. $A \wedge B, C \wedge D, B \wedge E, D \wedge F, E \wedge F, A \wedge C \wedge A$.

2.1.2. $(A \wedge C) \wedge (A \wedge B) \wedge A \vee B$.

2.1.3. $A \vee D, B \vee E, D \wedge C, D \vee C \supset A \wedge C; E \wedge D; B$.

2.1.4. $A \wedge B, C \wedge D, A \vee C, A \wedge D, C \wedge B \wedge (A \vee B) \wedge (A \wedge B)$.

2.2. Составить легенды для следующих клауз:

2.2.1. $A \wedge B, B \wedge E, A \wedge C, C \wedge D, D \wedge F, E \wedge F \wedge A$.

2.2.2. $J \wedge H, K \wedge H, I \wedge J, H \wedge I, H \wedge J \wedge K$.

Приведем пример легенды для клаузы 2.2.1:

«Если человек занимается спортом (A), то он хочет быть здоровым (B). Хорошее здоровье (B) ведет к счастливой жизни (E). Кроме того, если человек занимается спортом (A), то он, как правило, стремится достичь высоких спортивных результатов (C). Наличие высоких результатов (C) позволяет одерживать победы на соревнованиях (D). Победы на соревнованиях (D) влекут за собой всеобщее признание (F). Однако, человек не хочет жить счастливо и иметь всеобщее признание (E & F).

Следовательно, он не станет заниматься и спортом (A)».

2.3. Для заданных легенд составить клаузы, отвечающие тексту или контексту этих легенд, для чего сформулировать необходимые посылки и два следствия: одно истинное, другое ложное.

2.3.1. Если в цепи будет большой перепад напряжения, то сгорит предохранитель, что повлечет за собой необходимость его замены. При целом предохранителе телевизор, конечно, будет работать, но только если он включен в сеть питания. Если телевизор работает нормально, то я увижу сегодняшние «Новости» Итак, я смотрю телевизионные «Новости» при условии отсутствия перепада напряжения и подключения телевизора к сети питания.

Уменьшение температуры приводит к снижению давления и уменьшению объема. Увеличение объема приводит к росту скорости потока. Повышение давления приводит к падению уровня, если при этом уменьшать температуру. Снижение скорости приводит к уменьшению давления или росту температуры. Технолог Иванов рассудил так: «Мне надо повысить давление при одновременном снижении скорости потока, поэтому я должен увеличить объем и температуру». Решение. Данное умозаключение является ложным. Если ввести обозначения: A уменьшение температуры; B снижение давления; C уменьшение объема; D снижение скорости; E падение уровня, то для легенды 3.2 можно записать клаузу, имеющую истинное заключение:

$A \wedge (B \wedge C), C \wedge D, B \wedge (A \wedge E), D \wedge (B \vee A) \wedge (A \wedge B) \wedge C$.

2.4. Установить истинность приведенных ниже логических выражений путем конкретизации.

2.4.1. $\forall x (A(x) \wedge B(x)) = \forall x A(x) \wedge \forall x B(x)$;

2.4.2. $\forall x \forall y P(x, y) \wedge \forall x P(x, b)$;

2.4.3. $(\forall x \forall y P(x, y) \vee \forall u \forall v P(v, u)) \wedge (\forall v \forall u P(v, u) \wedge \forall y \forall x P(x, y)) = \forall x \forall y P(y, x) \wedge \forall u \forall v P(u, v)$;

2.4.4. $\forall x \forall y P(x, y) \wedge \forall v P(a, v), \forall x P(a, x) \wedge \forall y \forall x P(y, x) \wedge \forall u \forall v P(u, v) \wedge \forall y \forall v P(u, v)$.

Решение 2.4.1.

$$\begin{aligned} \forall x (A(x) \wedge B(x)) &= (A(a) \wedge B(a)) \vee (A(b) \wedge B(b)) = \\ &= A(a) \vee B(a) \vee A(b) \vee B(b) = (A(a) \vee A(b)) \vee (B(a) \vee B(b)) = \\ &= \forall x A(x) \vee \forall x B(x) = \forall x A(x) \wedge \forall x B(x). \end{aligned}$$

Решение 2.4.2. Для доказательства истинности данной клаузы избавимся от кванторов в обеих ее частях:

$$\begin{aligned} &(P(a, a) \wedge P(a, b)) \vee (P(b, a) \wedge P(b, b)) \wedge P(b, a) \vee P(b, b); \\ &P(a, a) \vee P(a, b), P(a, b) \vee P(b, a), P(a, a) \vee P(b, b), P(b, a) \vee P(b, b), \\ &P(b, a) \wedge P(b, b); \\ &P(a, a) \vee P(a, b), P(a, b) \vee P(b, a), P(a, a) \vee P(b, b), P(b, b) \wedge P(b, b). \end{aligned}$$

Последняя клауза верна в силу аксиомы порядка.

Решение 2.4.3. Введем следующие обозначения:

$$\begin{aligned} A &= \forall x \exists y P(x, y) = \forall v \exists u P(v, u) = \forall u \exists v P(u, v), \\ B &= \exists u \forall v P(v, u) = \exists y \forall x P(x, y) = \exists x \forall y P(y, x). \end{aligned}$$

В этих обозначениях данное тождество выглядит следующим образом:

$$(A \vee B) \wedge (A \wedge B) = B \wedge A.$$

Производя элементарные преобразования, мы можем убедиться в справедливости последнего равенства. Следовательно, и исходное тождество составлено верно.

Решение 2.4.4. Введя обозначения

$$\begin{aligned} A &= \forall x \exists y P(x, y) = \forall u \exists v P(u, v), \\ B &= \forall v P(a, v) = \forall x P(a, x), \\ C &= \exists x \forall y P(y, x) = \exists v \forall u P(u, v), \end{aligned}$$

получим $A \wedge B$, $B \wedge C \wedge A \wedge C$. Как известно, формула силлогизма верна, следовательно, и предикатная клауза тоже верна.

Итак, доказательство справедливости данных предикатных выражений свелось к простой процедуре идентификации (унификации) их с соответствующими выражениями, имеющими место в исчислении высказываний.

2.5. Доказать истинность предикатных клауз методом резолюций.

$$\begin{aligned} 2.5.1. \quad &\forall x \forall z A(x, x, z), \exists x \exists y B(x, x, y) \wedge \forall z A(z, b, z) \wedge \\ &\wedge \exists u \exists v A(u, v, v) \wedge \exists z B(z, z, z); \forall u \exists v A(a, v, u) \wedge \exists y B(a, y, a) \wedge \exists x A(x, b, c); \end{aligned}$$

$$\begin{aligned} 2.5.2. \quad &\forall y \forall z A(a, y, z) \vee \forall v \forall w A(w, v, v), \forall x \forall z B(x, a, z) \vee \forall x B(x, c, a) \wedge \\ &\wedge \exists u \forall z A(u, z, c) \wedge \exists x B(x, x, a); \exists y B(b, y, b) \wedge \exists y B(c, y, a); \end{aligned}$$

$$\begin{aligned} 2.5.3. \quad &\forall y \forall z A(a, y, z) \vee \forall w B(a, w, w) \vee \forall y \exists x A(x, y, c), \exists u \forall z B(u, u, z) \wedge \\ &\wedge \exists x \forall y A(x, y, a) \wedge \exists x \forall z B(x, z, z) \wedge \forall w \exists z A(z, w, w) \wedge \forall w \exists u B(u, b, w); \exists u \forall w A(u, \\ &w, c); \end{aligned}$$

$$\begin{aligned} 2.5.4. \quad &\exists u A(u, b, c) \wedge (\exists v \exists w B(v, v, w) \wedge \exists v A(b, v, v)), \exists y B(a, y, a) \vee \\ &\vee \forall x A(b, x, x) \vee \forall x \forall u B(x, u, b) \wedge \forall x A(x, x, c) \wedge \exists y \exists z A(b, y, z); B(a, a, b). \end{aligned}$$

Для решения этих задач ответы нетрудно отыскать, если знать, каким образом были составлены логические выражения.

Решение 2.5.1. Введем обозначения:

$$\begin{aligned} A_1 &= \forall x \forall z A(x, x, z), A_2 = \exists u \exists v A(u, v, v), \\ B_1 &= \exists z B(z, z, z), B_2 = \exists y B(a, y, a), B_3 = \exists x \exists y B(x, x, y), \\ C_1 &= \forall z A(z, b, z), C_2 = \forall u \exists v A(a, v, u) \wedge \exists x A(x, b, c), \\ C'_2 &= \forall u \exists v A(a, v, u), C'_2 = \exists x A(x, b, c). \end{aligned}$$

Тогда

$$\begin{aligned} &A_1 \wedge A_2, B_1 \wedge B_2, B_1 \wedge B_3, C_1 \wedge C_2, \\ &A_1, B_3 \wedge C_1 \wedge A_2 \wedge B_1; \end{aligned}$$

$C \wedge B \wedge C, A, B \rightarrow C, A \rightarrow B, B \rightarrow C \rightarrow 0$. Решение 2.5.2. Введем обозначения:

$$\begin{aligned} A_1 &= A_1' \vee A = \forall y \forall z A(a, y, z) \vee \forall v \exists w A(w, v, v), \\ A_2 &= \exists u \forall z A(u, z, c), A_1 > A_2, \\ B_1 &= \forall x \forall z B(x, a, z), B_2 = \exists y B(b, y, b), B_1 > B_2, \end{aligned}$$

$C' = \forall x B(x, c, a), C'' = \forall y B(c, y, a), C = \exists x B(x, x, a), 1 \quad 1 \quad 2$
 $C1 > C2, C \setminus > C2.$

Тогда

$A \vee A, B \vee C > A \& C;$

$1 \quad 1 \quad 1 \quad 1 \quad 2 \quad 2$

"

$B2 \& C1, A1 \vee A1, B1 \vee C1, A2 \vee C2, B2 \vee C1 > 0.$

Решение 2.5.3. Введем обозначения:

$A_1 = A_1' \vee A_1'' = \forall y \forall z A(a, y, z) \vee \forall y \exists x A(x, y, c),$

$A_2 = \exists u \forall w A(u, w, c) = \sup(A_1', A_1''),$

$B_2 = B_2' \& B_2'' = \exists x \forall z B(x, z, z) \& \forall w \exists u B(u, b, w),$

$B_1 = \forall w B(a, w, w) = \inf(B', B''), B = \exists u \forall z B(u, u, z),$

$2 \quad 2 \quad 3$

$A1 > A2, B1 > B2, B1 > B3,$

$C1 = \exists x \forall y A(x, y, a), C2 = \forall w \exists z A(z, w, w), C1 \wedge C2,$ где: $\inf(B', B'') \wedge B \& B, A \vee A > \sup(A', A).$

$2 \quad 2 \quad 2 \quad 2 \quad 1 \quad 1 \quad 1 \quad 1$

Тогда

$A' \vee B \vee A'', B \wedge C > B' \& C \& B''; 11 \quad 1 \quad 3 \quad 1 \quad 2 \quad 2 \quad 2$

$A, A \vee B \vee A, B \rightarrow \vee C, B \vee C \rightarrow \vee S, A > 0.$

$2 \quad 1 \quad 1 \quad 1 \quad 3 \quad 1 \quad 2 \quad 2 \quad 2 \quad 2$

Решение 2.5.4. Введем обозначения:

$A_1 = \forall x A(x, x, c), A_2 = \exists u A(u, b, c), A_1 > A_2,$

$B_1 = \exists y B(a, y, a), B_2 = \exists v \exists w B(v, v, w), B_1 > B_2,$

$C = \exists v A(b, v, v), C = \forall x A(b, x, x), C \supset y \supset z A(b, y, z),$

$1 \quad 1 \quad 2$

$C1 \wedge C2, C \wedge C2,$

$D1 = \forall x \forall u B(x, u, b), D2 = B(a, a, b), D1 \wedge D2.$ Тогда

$A \wedge (B \wedge C'), B \vee C'' \vee D0 A \wedge C; 2 \quad 2 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 2$

$D, A \sim \vee B \vee C', B \vee C'' \vee D, A, C, D0 0.$

9. ОСНОВНЫЕ ПРИНЦИПЫ ЛОГИЧЕСКОГО ПРОГРАММИРОВАНИЯ

План занятия

1. Использование формализмов логики для алгоритмических вычислений.

Факты, правила, запросы.

2. Общие правила выполнения запросов логическими программами.

3. Методика составления логических программ.

На начальном этапе развития искусственного интеллекта (ИИ) языков и систем, ориентированных специально на создание прикладных систем, основанных на знаниях, не существовало. С одной стороны, в то время еще не оформился сам подход, в котором центральное место отводилось бы изложению теории в форме программ, а с другой сама область ИИ только зарождалась как научное направление. Немаловажным было и то, что появившиеся к тому времени универсальные языки программирования высокого уровня казались адекватным инструментом для создания любых, в том числе и интеллектуальных систем. Однако сложность и трудоемкость разработки здесь настолько велики, что практически полезные интеллектуальные системы становятся недоступными для реализации. Учитывая вышесказанное, были разработаны языки и системы обработки символьной информации, которые на несколько десятилетий стали основным инструментом программирования интеллектуальных систем.

До недавнего времени наиболее популярным базовым языком реализации систем ИИ вообще и экспертных систем (ЭС), в частности, был Лисп. Этот язык был разработан в Стэнфорде под руководством Дж. Маккарти в начале 60-х годов и не предназначался

вначале для программирования задач ИИ. Это был язык, который должен был быть следующим за Фортраном шагом на пути автоматизации программирования. К моменту создания первых Лисп-интерпретаторов в практику работы на ЭВМ стал входить диалоговый режим, и режим интерпретации естественно вписался в общую структуру диалоговой работы. Примерно тогда же окончательно сформировались и принципы, положенные в основу языка Лисп: использование единого спискового представления для программ и данных; применение выражений для определения функций, скобочный синтаксис языка. Процесс разработки языка завершился созданием версии Лисп 1.5, которая на многие годы определила пути его развития.

К концу 80-х годов 20-го века Лисп был реализован на всех классах ЭВМ, начиная с персональных компьютеров и кончая высокопроизводительными вычислительными системами. Новый толчок развитию Лиспа дало создание Лисп-машин, которые и в настоящее время выпускаются рядом фирм США, Японии и Западной Европы. Параллельно с развитием языка Лисп и его различных версий разрабатывались и другие системы программирования для ИИ, среди которых отметим языки Снобол и Рефал.

В начале 70-х годов прошлого века появился еще один новый язык, способный составить конкуренцию Лиспу при реализации систем, ориентированных на знания, язык Пролог. Он не дает новых сверхмощных средств программирования по сравнению с Лиспом, но поддерживает другую модель организации вычислений. Подобно тому, как Лисп скрыл от программиста устройство памяти ЭВМ, Пролог позволил ему не заботиться (без необходимости) о потоке управления в программе. Пролог предлагает такую парадигму мышления, в рамках которой описание решаемой задачи представляется в виде слабо структурированной совокупности отношений. Это удобно, если число отношений не слишком велико и каждое отношение описывается небольшим числом альтернатив. В противном случае Пролог-программа становится весьма сложной для понимания и модификации. Возникают и проблемы эффективности реализации языка, так как в общем случае механизмы вывода, встроенные в Пролог, обеспечивают поиск решения на основе перебора возможных альтернатив и декларативного возврата из тупиков.

Пролог был разработан в 1971 году в Марсельском университете А. Колмероз. Однако популярность он стал приобретать лишь в начале 80-х годов, когда благодаря усилиям математиков был обоснован логический базис этого языка, а также в силу того, что в японском проекте вычислительных систем 5-го поколения Пролог был выбран в качестве базового для машины вывода. В настоящее время Пролог завоевал признание и на американском континенте, хотя уступает в популярности Лиспу и даже специальным продукционным языкам, широко используемым при создании ЭС. Пролог представляет собой простой, но в то же время мощный язык. Он относится в большей степени к реляционным или декларативным (описательным), а не к функциональным языкам, и является языком программирования логического типа. Логическую основу языка Пролог составляют унификация и метод резолюций.

В последнее время широкое распространение получила универсальная среда разработки приложений Visual Prolog компании PDC из-за возможностей работы с развитой логикой. В этой среде можно решать те же задачи, что и в системе баз данных SQL, она поддерживает объектно-ориентированную технологию программирования, в частности, на языке C++. В ней можно использовать и другие инструментальные средства, такие как Visual Basic, Borland Delphi и др. Важным элементом среды Visual Prolog является возможность создавать Web-программы. Например, экспертные системы, написанные в среде Visual Prolog, могут быть подключены к Web-страницам, что имеет большое значение для развития Web-технологий. Система Visual Prolog продолжает развиваться и совершенствоваться. Возглавляют проект Visual Prolog Лео Йенсен, Томас Линдер Пулс и от России (Санкт-Петербург) Виктор Юхтенко и Юрий Ильин.

В данном разделе мы ознакомимся с основными принципами логического

программирования, в частности, с основными концепциями P_1 , программирования на языке Пролог.

3. 1. Использование формализмов логики для алгоритмических вычислений. Факты, правила и запросы

Логика призвана четко выражать человеческую мысль, однако если использовать в языках программирования исчисление предикатов первого порядка в непосредственном виде, то это приведет к чрезвычайно большим объемам вычислений, что неприменимо на практике. Поэтому в языке программирования Пролог ограничиваются специальными выражениями, называемыми выражениями Хорна или хорновскими дизъюнктами, и только к ним применяют метод резолюций.

Выражение Хорна имеет вид $P_1 \vee P_2 \vee \dots \vee P_n \vee P_m$, где $P_i, P_2, \dots, P_m$ литеры.

Атомы называются положительными литерами, а их отрицания отрицательными. В записанном выражении Хорна имеется единственная логическая формула без отрицания, и все формулы связаны операцией дизъюнкции. Это можно записать, как мы видели, и в другом виде:

$P_1 \& P_2 \& \dots \& P_n \wedge P_m$.

Последнее выражение читается так: «Из P_1 и P_2 и так далее до P_n следует P_m ». На языке Пролог в синтаксисе системы программирования Турбо-Пролог это записывается в другом виде:

P:

m

$P_2 \ P_n$. Для любой системы логического программирования характерно то обстоятельство, что для исполнения программ применяется так называемая встроенная система автоматического поиска вывода. Механизм поиска вывода, используемый в Прологе, берет свое начало от метода резолюций. Как нам уже известно, формулами метода резолюций являются дизъюнкты. Они имеют вид $A_1 \vee A_2 \vee \dots \vee A_n$. По-

рядок литер в дизъюнкте несущественен. Используемые в Прологе формулы A и $A_1 \& A_2 \& \dots \& A_n \wedge B$ эквивалентны хорновским дизъюнктам A и $A_1 \vee A_2 \vee \dots \vee A_n \vee B$.

Принцип резолюций подразумевает использование регулярной процедуры последовательного сопоставления с образцом, то есть алгоритма унификации. При наличии нескольких переменных или логических формул, содержащих функции, эти переменные или функции выделяются, часть их подвергается процедуре унификации и производится их подстановка в исходные формулы. Если при этом среди них оказываются тождественные истинные формулы вида $(A \vee A)$, то они, как мы знаем, удаляются. Метод резолюций как раз состоит в том, чтобы, повторяя эту операцию, показать невыполнимость множества дизъюнктов, то есть множества исходных формул. Поясним на примере применение принципа резолюций и унификации к процедуре вывода в языке Пролог.

Запишем выражение «X футболист» в виде $\text{footballer}(X)$. Здесь footballer предикатная буква (литера), X переменный терм. Кроме того, запишем выражение «X спортсмен» в виде $\text{sportsman}(X)$. Таким образом, выражение «футболист является спортсменом» можно записать в виде $\text{footballer}(X) \wedge \text{sportsman}(X)$.

Попробуем теперь, задав выражение «Пеле футболист», то есть $\text{footballer}(\text{Пеле})$, где Пеле константный терм (или просто константа) вывести методом резолюций выражение $\text{sportsman}(\text{Пеле})$.

Прежде всего, добавим к исходному множеству формул отрицание утверждения, которое мы хотим доказать, то есть $\text{sportsman}(\text{Пеле})$:

$\text{footballer}(X) \vee \text{sportsman}(X)$ (3.1)

$\text{footballer}(\text{Пеле})$ (3.2)

$\text{sportsman}(\text{Пеле})$. (3.3)

Формула $\text{footballer}(X) \vee \text{sportsman}(X)$ формуле
 $\text{footballer}(X) \wedge \text{sportsman}(X)$. логически эквивалентна

Поэтому необходимо осуществить унифицирующую подстановку $X = \text{Пеле}$ в дизъюнкцию формул (3.1) и (3.2). В результате удаляется тождественно истинная формула вида $a \vee a$ и остается лишь выражение $\text{sportsman}(\text{Пеле})$. После этого, присоединив к выражению $\text{sportsman}(\text{Пеле})$ с помощью операции дизъюнкции формулу (3.3), снова получим тождественно истинное выражение

$\text{sportsman}(\text{Пеле}) \vee \text{sportsman}(\text{Пеле})$, которое также удаляется, давая при этом пустой дизъюнкт. Это показывает невыполнимость формул (3.1) (3.3) и, следовательно, выводимость методом опровержения формулы $\text{sportsman}(\text{Пеле})$, то есть истинность теоремы $\text{footballer}(X) \wedge \text{sportsman}(X)$.

На языке Пролог приведенный пример доказательства может выглядеть следующим образом:

$\text{sportsman}(X) : \text{footballer}(X).$ $\text{footballer}(\text{Пеле}).$
 $? \text{sportsman}(\text{Пеле}).$

Эта коротенькая программа содержит три предложения. Первое из них является правилом, второе фактом и третье вопросом. В общем виде прологовские предложения бывают только трех типов: факты, правила и вопросы (или запросы). Обращаем внимание на то, что каждое предложение в программе на Прологе заканчивается точкой.

Пролог программы можно расширять, добавляя в них новые предложения.

Факты содержат утверждения, которые являются всегда безусловно истинными. В нашем примере фактом является предложение $\text{footballer}(\text{Пеле})$.

Правила содержат утверждения, истинность которых зависит от некоторых условий. Поэтому правила имеют условную часть (правая половина правила после символа $:$, который читается как «если») и часть вывода (левая половина правила). Вывод называют также головой предложения, а условную часть его телом. Отсюда следует, что факты это предложения, имеющие пустое тело, а правила имеют голову и непустое тело. В нашем примере имеется только одно правило первое предложение. Оно читается так: если X футболист, то он спортсмен. В общем случае тело правила это список целей, разделенных запятыми. Запятая понимается как конъюнкция. Например, правило для отношения «быть сестрой» может выглядеть так:

Сестра (X, Y) : родитель (Z, X) , родитель (Z, Y) , женщина (X) , различны (X, Y) .

По ходу вычислений вместо переменной в предикат может быть подставлен другой объект. В этом случае говорят, что переменная конкретизирована. Предполагается, что все переменные связаны квантором всеобщности, хотя для переменных, появляющихся только в теле правил, возможно связывание и квантором существования.

С помощью вопросов пользователь может спрашивать систему о том, какие утверждения являются истинными. Вопрос является целевым утверждением. Обычно ответ на вопрос бывает либо утвердительным «yes» (да), или отрицательным «no» (нет). В языке программирования Пролог, как и в методе резолюций, взяв за отправную точку отрицание данной формулы, методом опровержения можно доказать саму формулу.

3. 2. Общие правила выполнения запросов логическими программами

Вопрос к системе это всегда последовательность, состоящая из одной или нескольких целей. Для того, чтобы ответить на вопрос, система пытается достичь всех целей. Это означает, что утверждения, содержащиеся в вопросе, истинны в предположении, что все отношения программы истинны. Другими словами, достичь цели это значит показать, что она логически следует из фактов и правил программы. Если вопрос содержит переменные, система должна к тому же найти конкретные объекты, которые, будучи подставленными вместо переменных, обеспечивают достижение цели. Найденные конкретизации сообщаются пользователю. Если для некоторой конкретизации система не в состоянии вывести цель из остальных предложений программы, то ее ответом на вопрос будет «нет». Таким образом, подходящей интерпретацией Пролог-

программы в математических терминах будет следующая: Пролог-система рассматривает факты и правила в качестве множества аксиом, а вопрос пользователя как теорему; затем она пытается доказать эту теорему, то есть показать, что ее можно логически вывести из аксиом, используя алгоритм унификации и метод резолюций.

Рассмотрим на примере с футболистом Пеле, как происходит доказательство (вывод) целевого утверждения. Прежде всего, в левой части правила производится поиск предиката, совпадающего с целевым утверждением $sportsman(X)$, и производится унификация (конкретизация) переменной X и константного термина Пеле. Затем в программе ищется выражение, содержащее в левой части предикат, совпадающий с новой целью (в данном случае это факт $footballer(Пеле)$), образующейся в правой части текущего правила в результате его унификации. Затем аналогично производятся дальнейшие унификации.

Итак, программа на языке Пролог представляет собой некоторое множество дизъюнктов Хорна и выполняется путем проведения рассуждений от целевого утверждения в обратном направлении. Если логических формул, подлежащих унификации, много, то производится унификация для одной из них. Если унификация удовлетворительна, то программа выполняется дальше. Если унификация неудовлетворительна, происходит возврат и выполняются другие возможные унификации. Таким образом, в языке Пролог рассуждение ведется в обратном направлении методом проб и ошибок.

Для управления построением выводов (вычислений) имеется два типа стратегий вычисления. Во всех реализациях Пролога на последовательных машинах используется стратегия, управляющая последовательным вычислением, следующего типа. Если заданы два условия (процедурных вызова), то для одного из них дерево вывода полностью строится до того, как обрабатывается другое условие (инициируется другой вызов).

Предположим, например, что программа на Прологе имеет вид: $p:q, s$.

$p: r, t$.

Эту программу с использованием понятия процедуры можно прочитать следующим образом. При выполнении процедуры p выполняются процедуры q и s . Если они завершаются успешно, то и процедура p считается успешно завершенной. Если это не так, то выполняются процедуры r и t . В этом случае процедура считается успешно завершенной, если обе процедуры r и t завершаются успешно. В противном случае процедура p терпит неудачу. Такой алгоритм обработки можно реализовать на дереве специального вида «И-ИЛИ» дереве поиска (рис. 3.1).

В каждой своей вершине «И-ИЛИ» дерево имеет метку «И» или метку «ИЛИ». Вершина «И» успешно разрешима только в том случае, когда ее вершины-потомки успешно разрешимы. Вершина типа «ИЛИ» успешно разрешима в том случае, когда хотя бы одна из ее вершин-потомков успешно разрешима.

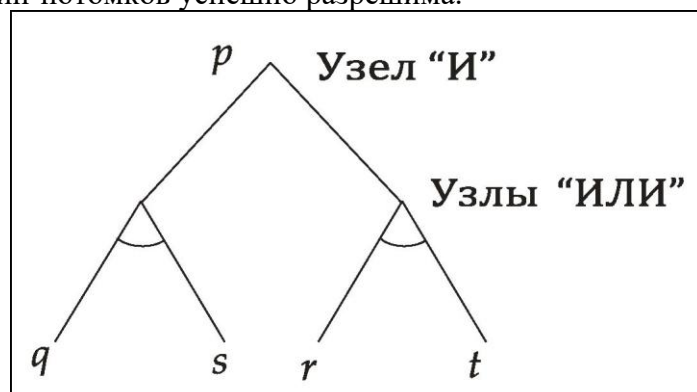


Рисунок 3.1. «И-ИЛИ» – дерево

Согласно стратегии поиска в глубину, используемой в Прологе, производится последовательный обход «И-ИЛИ»-дерева сверху вниз и слева направо. Это соответствует

выделению самого левого атома запроса и упорядочению сверху вниз правил программы.

Если при обходе дерева какой-либо потомок вершины типа «ИЛИ» разрешается успешно, то обработка остальных дочерних вершин (поддерева, находящегося справа) приостанавливается и считается, что данная «ИЛИ»-вершина разрешилась успешно.

Если какой-либо из потомков «И»-вершины терпит неудачу, то обработка остальных ее потомков полностью прекращается и считается, что данная вершина терпит неудачу. Следует заметить, что обработка «ИЛИ»-вершины не прекращается, а лишь приостанавливается. Это связано с тем, что впоследствии возможно повторное обращение к этой вершине, и тогда оставшиеся про запас ветви, которые еще не обрабатывались, могут повторно привести к успеху этой вершины. Этот процесс называется бэктрекингом.

Основное достоинство такой стратегии простота ее реализации на последовательных машинах, недостаток большой перебор для некоторых программ и запросов. Кроме того, если дерево вычислений содержит бесконечную ветвь, то, попав на нее, интерпретатор с этой ветви не выберется и поэтому правильные ответы, лежащие правее этой ветви на дереве вычислений, не будут получены.

Одним из средств преодоления указанного недостатка является использование встроенного в систему программирования Пролог конструкции (оператора) отсечения (cut), который обозначается символом !, который вставляется между целями и играет роль некоторой псевдоцели. Он используется для предотвращения бессмысленного перебора. Смысл этого оператора заключается в изменении стандартного порядка вычислений. При его выполнении, то есть когда он становится самым левым в запросе, «забываются», то есть выбрасываются из стека все разветвления, зафиксированные с момента обращения к правилу, содержащему этот оператор, и в дальнейшем осуществляется возврат к предыдущему разветвлению. Тем самым усекается, иногда весьма существенно, дерево разветвлений. Пусть, например, дана следующая Пролог-программа:

a (X) :b (x), !, c(X).

a (X) :d(X).

b (e).

b (f).

c (e).

c (f).

d (g).

Наша программа без оператора ! на запрос a(Z) выдаст три ответа: Z = e, f, g. Благодаря оператору ! программа выдаст единственный ответ Z = e, так как она не будет уже возвращаться к развилкам, возникшим до обращения к оператору ! при обработке атомов a(Z) и b(Z).

Оператор отсечения довольно остроумное и мощное средство. Пользуясь им, программист может составлять достаточно эффективные программы и экономить вычисления при построении дерева вывода. Однако этот оператор в какой-то степени подрывает основную идею логического программирования. Программа лишается прозрачной логической семантики. Остается лишь процедурная семантика, соответствующая данной стратегии обхода дерева вычислений.

3. 3. Методика составления логических программ

В настоящее время уже накоплен определенный опыт реализации интеллектуальных систем и можно говорить о том, что в области искусственного интеллекта сформировались определенные парадигмы программирования. Основными из них являются функциональная, логическая и объектно-ориентированная парадигмы.

Функциональная парадигма программа состоит из независимых функций, каждая из которых определяет правило преобразования своих аргументов для получения некоторого результата. Эти функции определяются путем композиции системных или определяемых про-граммистом функций с использованием структур типа альтернативы или рекурсии. Наиболее известным представителем языков, опирающихся на

функциональную парадигму, является Лисп.

Логическая парадигма основана на использовании механизма доказательства теорем, который позволяет выяснить, противоречиво ли некоторое множество логических формул или нет. При этом программа может рассматриваться как набор логических формул совместно с теоремой (запросом), которая должна быть доказана. Формулы ограничены импликантами, а каждая часть импликанты ограничена логическим выражением. Формулы предназначены для представления различных фактов (данных) и правил вывода. Для хранения и эффективной работы с фактами и правилами привлекаются средства баз данных и сопоставление с образцом (унификация). При этом программист избавляется от необходимости определения точной последовательности шагов для выполнения вычислений. Типичным представителем этой парадигмы является язык Пролог.

Объектно-ориентированная парадигма программа состоит из объектов и сообщений. Такой способ программирования вносит в программу модульность посредством абстракции данных и наследования и особенно хорошо подходит для ситуаций, когда имеется четкая иерархическая классификация объектов. Это позволяет избежать дублирования и локализовать описания механизмов работы с информацией. Одним из первых языков этого типа является Smalltalk. В настоящее время широко используется язык C++.

В современной практике интеллектуального программирования отдельные парадигмы редко используются в чистом виде. В качестве примеров слияния различных парадигм можно привести не только отдельные языки, но и целый подход к программированию интеллектуальных систем на базе продукционной парадигмы, в рамках которой взаимодействие правил-продукций может опираться на функциональную парадигму, а унификация условий применимости на логическую. При этом сам продукционный процессор может быть построен на базе механизмов асинхронного управления, близких по своей идее к объектно-ориентированной парадигме.

Мы на конкретных примерах рассмотрим методику составления программ и проведем трассировку одной из таких программ на языке логического программирования Пролог.

Практически каждая система искусственного интеллекта, построенная на логическом принципе, представляет собой машину доказательства теорем. При этом исходные данные хранятся в базе данных в 330 виде аксиом, правила логического вывода как отношения между ними. Кроме того, каждая такая машина имеет блок генерации цели, и система вывода пытается доказать данную цель как теорему. Если цель доказана, то трассировка примененных правил позволяет получить цепочку действий, необходимых для реализации поставленной цели. Мощность такой системы определяется возможностями генератора целей и машиной доказательства теорем. Конечно, можно сказать, что выразительности алгебры высказываний не хватит для полноценной реализации ИИ, но стоит вспомнить, что основой всех существующих ЭВМ является бит ячейка памяти, которая может принимать значения только 0 и 1. Таким образом, было бы логично предположить, что все, что можно реализовать на ЭВМ, можно реализовать и в виде логики предикатов.

Обсудим сначала кратко понятие неформальной процедуры. Говоря о неформальных процедурах, мы без затруднений можем привести примеры таких процедур. Они связаны с пониманием текстов естественного языка, переводом с одного естественного языка на другой, информационным поиском по смыслу и т. д. Трудности возникают при попытке точного определения подобных процедур. Так, если рассматривать неформальные процедуры всего лишь как абстрактные функции, которые для каждого значения аргумента выдают некоторое значение функции, то категория неформальности вообще исчезает из рассмотрения.

Неформальная процедура это особый способ представления функций. Чтобы в

какой-то степени приблизиться к этому, понятному для человека способу представления функций, рассмотрим, прежде всего, традиционные алгоритмические модели и попытаемся понять, в чем состоит основная трудность их применения для имитации неформальных процедур.

Алгоритмические модели основаны на понятии алгоритма. Исторически первые точные определения алгоритма, возникшие в 30-х годах, были связаны с понятием вычислимости. С тех пор было предложено несколько, как выяснилось, эквивалентных определений алгоритма. Эти вопросы мы подробно будем изучать в следующей главе.

В практике программирования алгоритмы принято описывать с помощью алгоритмических языков программирования. Широко используются также разного рода блок-схемы алгоритмов, позволяющие представить их в наглядном и общедоступном виде, не привлекая в тоже время сложных конструкций из конкретных языков программирования. С точки зрения программирования на алгоритмических языках достоинства подобного представления алгоритма очевидны: блок-схема без затруднений переводится в текст программы, например, на языке Ассемблера или C++. Однако составление блок-схемы при появлении новых данных становится утомительным занятием.

В общем случае неформальные процедуры являются многозначными, а правильность конкретного выбора, сделанного на некотором шаге, проверяется на следующих шагах. При этом используется режим возвратов, описанный в п.3.2.

Пусть некоторое предложение начинается со слов «мать любит ... »:

- а) мать _____> любит _____> ? что делает? кого?
- б) мать <_____ любит <_____ ? кого? что делает?

Проанализировав эти слова в первоначальном предположении именительного падежа для слова «мать», система вправе построить структуру, представленную в случае а). Если следующее слово после слова «любит» представляет собой существительное в винительном падеже, например, «сына», то вся фраза будет иметь вид «мать любит сына», и эта структура является окончательной. Если же фраза имеет вид «мать любит сын», то возникает противоречие или, как говорят, сигнал неуспеха очередное слово «сын» противоречит ожиданию прямого дополнения. В этом случае система должна вернуться на ближайший из предыдущих шагов, где можно принять другую альтернативу анализа. В данном примере это шаг анализа слова «мать»: система должна принять теперь альтернативу винительного падежа для слова «мать», то есть построить структуру, указанную в случае б). Тривиальность рассмотренного примера убеждает нас в необходимости режима возвратов при реализации неформальных процедур. При этом основополагающую роль играет механизм логического вывода.

Важность логического вывода становится очевидной уже при рассмотрении простейших информационно-логических процедур. Предположим, что некоторая база данных содержит сведения о родственных отношениях «х отец у» и «х мать у». Чтобы обработать запросы типа «Иванов А.И. дед Петрова В.А.?» и «Петров В.А. внук Иванова А.И.?»», необходимо либо ввести в базу данных также и сведения об отношениях «х дед у» и «х внук у», либо объяснить системе, как из исходных отношений «отец» и «мать» извлечь искомую информацию. Реализация первой возможности связана с неограниченным ростом избыточности базы данных. Вторая возможность при традиционном алгоритмическом подходе требует написания все новых и новых программ для реализации новых типов запросов.

Логический вывод позволяет расширять возможности «общения» наиболее просто и наглядно. Так, для приведенных типов запросов системе достаточно будет сообщить три правила:

- 1. хдед у, если х отец z и z родитель у.
- 2. х родитель у, если х отец у или х мать у.
- 3. х внук у, если у дед х.

Эти правила содержат естественные и очевидные определения понятий дед, родитель, внук. Поясним, в чем состоит логический вывод для запроса «а дед b?» в предположении, что в базе данных имеются факты: «а отец b» и «а мать b». При этом для упрощения опустим тонкости, связанные с падежными окончаниями. Пользуясь правилом 1, система придет к необходимости проверки существования такого индивидуума а, что факты «а отец b» и «а родитель b» будут истинными. Если такой а существует, то а дед b, если не существует такого а, то субъект а не является дедом субъекта b.

Продукционная система должна взять на себя функции распознавания и интерпретации приоритета продукций, а программист должен только описывать ситуации и соответствующие им действия. Для этих целей и используются языки логического программирования. У языков логического программирования имеется два главных отличия от классических алгоритмических языков программирования, предназначенных в основном для численных расчетов:

- Логическое программирование это символьное программирование, предназначенное для обработки неформальной информации: данные в языках логического программирования являются символами, представляющими только самих себя и не подлежащими интерпретации при выполнении программ.

- Алгоритмы получения определенных результатов непосредственно не задаются, но описываются объекты, их свойства и отношения между ними. Кроме того, в алгоритмах указываются факты и правила получения новых фактов. С помощью вопросов (запросов) к системе выясняется установление или возможность установления фактов.

Три вида выражений (факты, правила и вопросы) интерпретируются как логические предикаты, и тогда язык логического программирования становится подмножеством языка формальной логики. Языки логического программирования относят к языкам декларативного или сентенциального типа (sentence высказывание, предложение). Программа на таком языке представляется в виде набора предложений (соотношений, правил, формул), которые машина, понимающая данный язык, умеет каким-то образом применять к обрабатываемой информации. Можно указать прообразы указанных типов алгоритмических языков в естественных языках. Для процедурных (операторных) языков программирования это аналог повелительного наклонения (императив, приказание) в естественном языке, для декларативных изъявительное наклонение (описание, повествование). Обращаясь к естественному языку, нетрудно заметить, что изъявительное наклонение является несравненно более распространенным и образует, в сущности, основу языка, в то время как повелительное наклонение предстает в виде некоторой специальной модификации. Таким образом, можно сделать вывод о том, что «относительный вес изъявительного наклонения является мерой развитости языка». К языкам логического программирования относятся, в первую очередь, языки Рефал и Пролог.

Приведем краткую характеристику языка Рефал. Название этого языка происходит от сокращения фразы «РЕкурсивных Функций АЛгоритмический язык». Этот язык является декларативным в своей основе, а вся информация в этом языке представляется в виде правил конкретизации. Каждое правило записывается в виде предложения, которое представляет собой продукцию с определенными синтаксисом и семантикой. Предложения в Рефал-программе отделяются друг от друга знаком (параграф). Каждое правило конкретизации определяет раскрытие смысла некоторого понятия через более простые понятия. Операцию конкретизации можно также определить как переход от имени переменной к ее значению. Выполнение конкретизации объявляется основной операцией и, по существу, единственной операцией в языке Рефал. Эту операцию будет выполнять Рефал-машина (имеется в виду машина на логическом уровне, имитируемая соответствующим транслятором на универсальной ЭВМ; возможно, разумеется, и построение реальной «физической» Рефал-машины).

Поскольку правило конкретизации есть указание для замены одного объекта (слова

в некотором алфавите) на другой, предложения языка Рефал должны состоять из левой части (заменяемый объект) и правой части (объект, заменяющий левую часть). Для разделения правой и левой части используется знак стрелки « $\wedge$ ». В языке Рефал вводится еще два знака k и $\pm$, которые называются конкретизационными скобками, и которые содержать объект, подлежащий конкретизации. Так, если x некоторая переменная, то $kx1$. (конкретизация x) будет изображать значение этой величины. Другой пример: объект $k28+71$ при правильном определении операции сложения рано или поздно будет заменен объектом 35.

Опишем теперь структуру Рефал-машины, которая, используя предложения Рефал-программы, будет выполнять конкретизации. Будем считать, что объектом обработки является некоторое выражение (слово), которое находится в поле зрения машины. Работа машины осуществляется по шагам, каждый из которых представляет выполнение одного акта конкретизации.

Так как Рефал-программа содержит, вообще говоря, набор (последовательность) предложений, может оказаться, что для выполнения данной конкретизации пригодно не одно, а несколько предложений. Машина просматривает предложения в том порядке, в котором они расположены в Рефал-программе, и применяет первое из них, которое окажется подходящим. Поле зрения машины может содержать сколько угодно конкретизационных скобок, причем они могут быть как угодно вложены друг в друга. В этом случае Рефал-машина начинает процесс конкретизации с первого из знаков k , в области действия которого (то есть в последовательности знаков до парной скобки 1) нет ни одного знака k . Выражение, находящееся в области этого знака k , последовательно сравнивается с левыми частями предложений Рефалпрограммы. Найдя подходящее предложение, машина выполняет в поле зрения необходимую замену и переходит к следующему шагу конкретизации.

Пример 3.1. Пусть Рефал-программа имеет вид

$k X 1^{\wedge} 137$

$k X 1^{\wedge} 274 k Y 1+2 k 137+21^{\wedge} 139$, а поле зрения содержит выражение

$k k X 1+ k Y 1 1$.

На первом шаге замены подлежит подвыражение $k X 1$ получим в поле зрения $k 137 + k Y 1 1$. Теперь в первую очередь конкретизируется $k Y 1$. В результате применения третьего предложения получим $k 137 +2 1$ и, наконец, на последнем шаге получим 139, не содержащее символов k . Разумеется, для реального сложения используются соответствующие встроенные функции, а этот пример лишь простейшая иллюстрация принципов работы машины.

Рассмотрим теперь методику составления программ на языке Пролог.

Большинство задач с использованием предикатов носит поисковый характер. Поисковые задачи реализуются средствами языка логического программирования Пролог. Остановимся на основных структурных и функциональных элементах этого языка.

Поскольку логика высказываний имеет дело с любыми правильно построенными предложениями, существует серьезная опасность смешения объектных и субъектных предложений, а также предложений, взятых с различных иерархических уровней предметной области логики высказываний. Чтобы избежать указанной опасности, решаемую задачу оформляют в виде отчетливой структуры древовидной формы. В качестве корня дерева выбирается некоторая цель C , истинность которой необходимо установить. Она всегда фигурирует в качестве заголовка правила, которое представляет собой хорновскую клаузу. Эти клаузы в данном случае удобно записать в обратном порядке: $C <^{\wedge} B 1, B_2, \dots$

Каждая из посылок B_i представляет собой подцель основной цели C и зависит, в свою очередь, от других правил $A_{ij} : B_1 \wedge A_{11}, A_{12}, \dots, B_2 \wedge A_{21}, A_{22}, \dots$ Впрочем, в роли посылок B_i могут выступать не клаузы, а элементарные высказывания факты. Далее посылки A_{ij} опять могут быть либо заголовками новых правил, либо фактами. Так

образуется иерархическая структура древовидной формы.

В Прологе реализована процедура унификации, с помощью которой производится сравнение цели с правилами, а правила сопоставляются с фактами. В результате унификации переменным присваиваются конкретные значения так, что предикат цели становится истинным фактом в случае положительного исхода.

Не рассматривая подробно синтаксис языка Пролог, мы все же для лучшего понимания структуры программы на Прологе и принципов ее работы введем некоторые основные понятия этого языка. Объекты данных в Прологе называются термами. Терм может быть константой, переменной или составным термом (структурой).

Константами являются целые и действительные числа. К константам относятся также атомы, то есть любые последовательности символов, заключенные в одинарные кавычки (апострофы). Кавычки опускаются, если и без них атом можно отличить от символов, используемых для обозначения переменных. Как правило, в программах на Прологе используются атомы без кавычек. Как и в других языках программирования, константы обозначают конкретные элементарные объекты, а все другие типы данных в Прологе составлены из сочетаний констант и переменных.

Переменные, подобно атомам, являются элементарными объектами языка Пролог. Понятие переменной в Прологе отличается от принятого во многих языках программирования. Переменная не рассматривается как выделенный участок памяти. Она служит для обозначения объекта, на который нельзя сослаться по имени. Переменную можно считать локальным именем для некоторого объекта. Синтаксис переменной довольно прост. Она должна начинаться с прописной буквы или символа подчеркивания и содержать только символы букв, цифр и подчеркивания, например: X, Переменная, _3, _переменная. Переменная, состоящая только из символа подчеркивания «_», называется анонимной и используется в том случае, если имя переменной несущественно. Переменные, отличные от анонимных, называются именованными, а неконкретизированные (переменные, которым не было присвоено значение) называются свободными. Завершает список синтаксических единиц сложный терм, или структура. Все, что не может быть отнесено к переменной или константе, называется сложным термом. Следовательно, сложный терм состоит из констант и переменных. Структура состоит из атома и последовательности термов, называемых компонентами структуры. Компоненты разделяются запятыми и заключаются в круглые скобки, например: собака (рекс), родитель (X, Y).

После записи утверждений в базу данных вычисления могут быть инициированы вводом запроса. Запрос выглядит так же, как и целевое утверждение, образуется и обрабатывается по тем же правилам, но он не входит в базу данных (программу). В Прологе вычислительная часть программы и данные имеют одинаковый синтаксис. Программа обладает как декларативной, так и процедурной семантикой. Запрос обозначается в Прологе утверждением ?-. Обычно запрос записывается в операторной форме: за знаком ? следует ряд целевых утверждений (чаще всего в виде конъюнкции), например: ?собака (X).

?-родитель(X, Y). ?-собака (Y).

Формальное определение синтаксиса Пролога в форме записи Бэкуса-Наура имеет вид:

```
запрос :: голова предложения
правило :: голова предложения:тело предложения
факт :: голова предложения голова предложения::-атом | структура тело предложения::атом | структура термы ::-терм [,термы]
терм ::число | переменная | атом | структура структура ::-атом (термы).
```

Данное определение синтаксиса не включает операторную, списковую и строковую формы записи. С полным определением синтаксиса Турбо-Пролога можно ознакомиться по любой книге по языку программирования Пролог. Однако любая программа на

Прологе может быть написана с использованием вышеприведенного синтаксиса. Специальные формы только упрощают понимание программы. Как мы видим, синтаксис Пролога не требует пространного объяснения. Но для написания хороших программ необходимо глубокое понимание языка.

Одним из наиболее важных аспектов программирования на Прологе являются понятия унификации (отождествления) и конкретизации переменных. Пролог пытается отождествить термы при доказательстве или согласовании целевого предложения. Например, для согласования запроса ?собака(X) целевое предложение собака(X) можно отождествить с фактом собака(рекс), в результате чего переменная X стала конкретизированной: X= реке. Переменные, входящие в предложения, отождествляются особым образом сопоставляются. Факт доказываемся для всех значений переменной (переменных). Правило доказываемся для всех значений переменных в головном целевом предложении при условии, что целевые утверждения в теле правила доказаны. Предполагается, что переменные в фактах и головных целевых предложениях связаны квантором всеобщности. Переменные принимают конкретные значения на время доказательства целевого предложения. В том случае, когда переменные содержатся только в целевых предложениях тела правила, правило считается доказанным, если целевое предложение тела истинно для одного или более значений переменных. Переменные, содержащиеся только в целевых предложениях тела правила, связаны квантором существования. Таким образом, они принимают конкретные значения на то время, когда целевое предложение, в котором переменные были согласованы, остается доказанным.

Терм X сопоставляется с термом Y по следующим правилам. Если X и Y константы, то они сопоставимы только в том случае, если они одинаковы. Если X является константой или структурой, а Y не конкретизированной переменной, то X и Y сопоставимы и Y принимает значение X (и наоборот). Если X и Y структуры, то они сопоставимы тогда и только тогда, когда у них одни и те же главный оператор и аргументы, и каждая из их соответствующих компонент сопоставима. Если X и Y не конкретизированные (свободные) переменные, то они сопоставимы; в этом случае говорят, что они сцеплены. Заметим, что Пролог находит наиболее общий унификатор термов.

Разберем конкретный пример программы на Прологе.

Пример 3.2. Имеется следующая легенда:

Ивана интересуют компьютеры, книги и автомобили. Петра интересует нечто, что интересует Ивана, но если это нечто является техникой и если это произведено в России. Известно, что компьютеры и автомобили это техника. Кроме того, известно, что компьютеры производятся в Америке, а автомобили в США и России. Вопрос: «Что интересует Петра?».

Для удобства анализа нижеприведенной программы все предикаты и конкретные значения переменных будем обозначать не отдельными литерами, а словами, передающими их семантику. Кроме того, объекты программы (факты, правила и запросы) будем нумеровать, а справа от текста программы приведем символьные выражения, чтобы далее у нас была возможность продемонстрировать в аналитической форме метод резолюций, который лежит в основе функционирования Пролога. Будем помнить также, что все склейки осуществляются с квантором всеобщности, хотя сам квантор не указывается.

Программа

- | | |
|--------------------------------|------|
| 1) интерес (Иван, компьютеры). | A(i, |
| 2) интерес (Иван, книги). | A(i, |
| 3) интерес (Иван, автомобили). | A(i, |
| 4) интерес (Петр, x) ^ | A(p, |
| | --\ |
| 5) ^ интерес (Иван, x), | A(i, |
| 6) техника (x), | T(x) |
| | --\ |

7) произведено (х, Россия).	P(x,
8) техника (компьютеры).	T(a)
9) техника (автомобили).	T(c)
10) произведено (компьютеры,	P(a,
11) произведено (автомобили,	P(c,
12) произведено (автомобили,	P(c,
Цель: интерес (Петр, х).	A(p,

Данная программа позволяет составить следующее противоречие, которое может быть разрешено в рамках метода резолюций:

$A(i, a), A(i, b), A(i, c), T(a), T(c), A(P, x) \vee A(i, x) \vee T(x) \vee P(x, r), P(a, s), P(c, s), P(c, r), A(p, x) \wedge 0.$

Ноль можно получить только в том случае, если $x=c$. Тогда дизъюнкт

$A(p, c) \vee A(i, c) \vee T(c) \vee P(c, r)$ нейтрализуется предикатами под номерами 3, 9, 12 и 13. Поиск нужного значения как раз и осуществляется через процедуру унификации, которую можно отследить путем трассировки программы. Трассировка это пошаговое протоколирование выполнения программы.

Введем следующие обозначения: В вызов нового предиката; П повторный вызов предиката; У успешное завершение процедуры унификации, то есть вызванный предикат отвечает какомулибо факту; Н неуспешное завершение унификации; У указатель, который говорит о том, что существует, по крайней мере, еще один факт с подходящей унификацией. Символ подчеркивания на месте x является анонимной переменной и полностью заменяет x при трассировке программы. Строка трассировки заканчивается числом, отвечающим номеру листинга программы. Приведем трассировку нашей программы:

1. В: цель () 13,
2. В: интерес (Петр, _) 1,
3. П: интерес (Петр, _) 2,
4. П: интерес (Петр, _) 3,
5. У: интерес (Петр, _) 4,
6. В: интерес (Иван, _) 5,
7. У: интерес (Иван, компьютеры) 1 у,
8. В: техника (компьютеры) 6,
9. У: техника (компьютеры) 8,
10. В: произведено (компьютеры, Россия) 7,
11. П: произведено (компьютеры, Россия) 10,
12. П: произведено (компьютеры, Россия) 11,
13. Н: произведено (компьютеры, Россия) 12,
14. П: интерес (Иван, _) 5,
15. У: интерес (Иван, книги) 2 у,
16. В: техника (книги) 6,
17. П: техника (книги) 8,
18. Н: техника (книги) 9,
19. П: интерес (Иван, _) 5,
20. У: интерес (Иван, автомобили) 3,
21. В: техника (автомобили) 6,
22. П: техника (автомобили) 8,
23. У: техника (автомобили) 9,
24. В: произведено (автомобили, Россия) 7,
25. П: произведено (автомобили, Россия) 10,
26. П: произведено (автомобили, Россия) 11,
27. У: произведено (автомобили, Россия) 12,
28. У: цель () 13.

Для того чтобы ответить на вопрос «Что интересует Петра?», или в предикатной форме интерес (Петр, x), Пролог-система ищет факты и заголовки правил, сопоставимые с целью. Поиск всегда начинается с первой строки программы. Первые три факта несопоставимы с целью; далее следует правило, заголовок которого заменяется тремя подцелями. В строке 6 вызывается первая подцель. Затем Пролог-система возвращается в начало программы, где ее ожидает успех. Выставляется указатель «У», к которому система вернется в случае неуспеха, который может появиться в другом месте программы. В строке 8 производится вызов второй подцели техника (компьютер), для которой тут же находится подходящий факт (строка 9). Однако для третьей подцели произведено (компьютеры, Россия) нужного факта не находится (строка 13). Тогда переменная x освобождается от своей прежней конкретизации (компьютеры) и принимает новое значение книги. Эта конкретизация не удовлетворяет вторую подцель (строка 18). Переменная вновь освобождается и принимает значение автомобиля. При этой конкретизации удовлетворены все три подцели, следовательно, и поиск ответа на основной вопрос заканчивается успехом: Петра интересуют автомобили, то есть интерес (Петр, автомобили) является тем конкретизированным предикатом, при котором обеспечена истинность всей клаузы.

Граф поиска ответа, сформулированного в виде цели интерес (Петр, x), представлен ниже на рисунке 3.2. Бинарное дерево получилось в итоге подстановок предметных констант на место переменных, правила заменялись фактами, а цель подцелями.

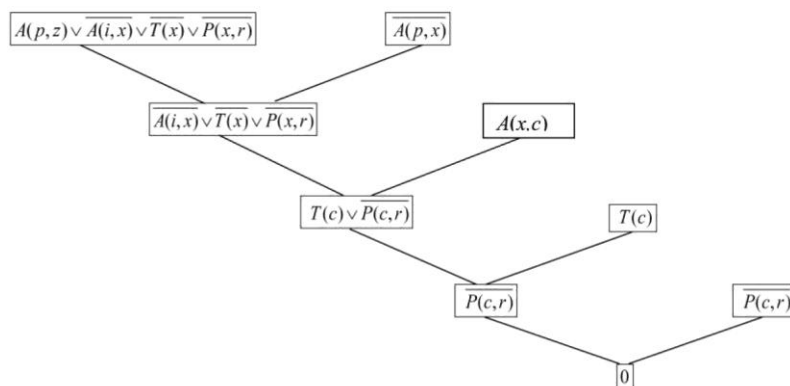


Рисунок 3.2. Дерево поиска ответа

Выводы

В заключении рассмотрим некоторые достоинства и недостатки языка программирования ПРОЛОГ при решении задач искусственного интеллекта.

Достоинства

1. ПРОЛОГ обладает четкой математической основой, что является в некотором роде формализацией человеческого мышления.
2. Существенно упрощается проблема верификации программ. Для ее решения желательно осуществить взаимодействие языков, на которых записываются требования к программам, и языков программирования. Кроме того, необходимо исходить из одних и тех же посылок при выполнении операций по верификации и составлению программ. ПРОЛОГ позволяет использовать единую основу исчисления предикатов первого порядка для решения всех этих задач.
3. Удобство при работе с реляционными базами данных. В ПРОЛОГЕ факты представляются выражениями, в которых константы играют роль термов. Следовательно, множество этих выражений можно рассматривать как реляционную базу данных.
4. Принципы организации вычислений ПРОЛОГА, параллельные по своей сути, выявляют новую точку зрения на возможную архитектуру машин.
5. ПРОЛОГ поддерживает вычисления, основанные на поиске через обратную цепочку рассуждений, что довольно эффективно используется при построении

экспертных систем.

6. Использование аппарата унификации как основного средства для сопоставления образцов поддерживает стиль программирования, основанный на предложениях.

7. ПРОЛОГ может хранить промежуточные результаты для последующего использования. Основным механизмом, обеспечивающим эту возможность, является понятие «логическая переменная», которое позволяет иметь дело с присваиванием значений сцепленным переменным в естественной форме.

Недостатки

1. Сложность понимания процесса выполнения программы на ПРОЛОГЕ, связанная с некоторой «скрытостью» деталей управляющего механизма.

2. Отсутствие совершенных сред программирования для поддержки разработки и отладки больших программ.

3. Отсутствие пригодных средств для разбиения пространства предложений ПРОЛОГА, чтобы улучшить модульность разработанных программ.

Разумно предположить, что эти недостатки можно преодолеть, если сохранить достаточный интерес к развитию логического программирования и ПРОЛОГА, в частности. В настоящее время, несмотря на конкуренцию со стороны основанных на ЛИСПЕ сред для разработки экспертных систем, преимущества ПРОЛОГА и тот факт, что он легко доступен для персональных компьютеров (особенно после появления версии ТУРБО ПРОЛОГ), позволяют считать ПРОЛОГУ наиболее пригодным средством для создания прототипов малых и средних экспертных систем.

Вопросы и упражнения для самопроверки

Задана Пролог-программа «Родственные отношения». Необходимо выполнить трассировку программы отдельно для двух целей, предварительно введя недостающие предикаты и правила. Так как может существовать несколько значений для переменных x и y , удовлетворяющих одной и той же цели, при трассировке допустимо ограничиться одним истинным значением x или y . Кроме того, ввиду большого числа повторных вызовов, разрешается указывать только первую и последнюю строки повторяющихся вызовов, например:

81. В: муж (Феликс) 85,

82. П: муж (Феликс) 1,

99. П: муж (Феликс) 18, 100. У: муж (Феликс) 19.

Пролог-программа «Родственные отношения» муж (x) x мужчина, жен (x) x женщина, отец (x, y) x отец y , мать (x, y) x мать y , дочь (x, y) x дочь y , сын (x, y) x сын y , род (x, y) x родитель y , брат (x, y) x брат y , сестра (x, y) x сестра y .

Клаузы

1) муж (Николай). 2) муж (Иван).

3) муж (Степан). 4) муж (Сергей).

5) муж (Павел). 6) муж (Игорь).

7) муж (Анатолий). 8) муж (Иосиф).

9) муж (Роман). 10) муж (Кирилл).

11) муж (Дмитрий). 12) муж (Максим).

13) муж (Евгений). 14) муж (Петр).

15) муж (Ефим). 16) муж (Юрий).

17) муж (Вадим). 18) муж (Олег).

19) муж (Феликс). 20) жен (Мария).

21) жен (Ольга). 22) жен (Татьяна).

23) жен (Жанна). 24) жен (Ирина).

25) жен (Алиса). 26) жен (Екатерина).

27) жен (Елена). 28) жен (Виктория).

29) жен (Полина). 30) жен (Луиза).

- 31) жен (Наталья). 32) жен (Луиза).
- 33) жен (Белла). 34) жен (Анна).
- 35) отец (Николай, Иван). 36) отец (Степан, Мария).
- 37) отец (Иван, Ольга). 38) отец (Иван, Сергей).
- 39) отец (Иван, Татьяна). 40) отец (Сергей, Виктория).
- 41) отец (Павел, Роман). 42) отец (Сергей, Дмитрий).
- 43) отец (Иосиф, Ирина). 44) отец (Ефим, Анатолий).
- 45) отец (Ефим, Полина). 46) отец (Юрий, Вадим).
- 47) отец (Петр, Ефим). 48) отец (Петр, Юрий).
- 49) отец (Петр, Анна). 50) отец (Максим, Петр).
- 51) отец (Дмитрий, Наталья). 52) отец (Феликс, Олег).
- 53) отец (Олег, Евгений). 54) мать (Алиса, Петр).
- 55) мать (Екатерина, Елена). 56) мать (Елена, Ефим).
- 57) мать (Елена, Юрий). 58) мать (Елена, Анна).
- 59) мать (Мария, Ольга). 60) мать (Анна, Виктория).
- 61) мать (Луиза, Евгений). 62) мать (Татьяна, Полина).
- 63) мать (Мария, Сергей). 64) мать (Анна, Олег).
- 65) мать (Татьяна, Анатолий). 66) мать (Анна, Дмитрий).
- 67) мать (Виктория, Белла). 68) мать (Мария, Татьяна).
- 69) мать (Ольга, Жанна). 70) мать (Ольга, Павел).
- 71) мать (Жанна, Ирина). 72) мать (Ирина, Игорь).
- 73) мать (Ирина, Кирилл). 74) мать (Барбара, Роман).
- 75) $\text{род}(x, y) \wedge \text{отец}(x, y)$.
- 76) $\text{род}(x, y) \wedge \text{мать}(x, y)$.
- 77) $\text{сын}(x, y) \wedge \text{род}(y, x), \text{муж}(x)$.
- 78) $\text{дочь}(x, y) \wedge \text{род}(y, x), \text{жен}(x)$.
- 79) $\text{брат}(x, y) \wedge \text{род}(z, x), \text{род}(z, y), \text{муж}(x), x \neq y$.
- 80) $\text{сестра}(x, y) \wedge \text{род}(z, x), \text{род}(z, y), \text{жен}(x), x \neq y$.

Варианты целей

- 80.1. тесть (Феликс, x), племянник (Павел, y).
- 80.2. тетя (Анна, y), внук (x, Алиса).
- 80.3. двоюродный брат (Вадим, y), внучка (x, Сергей).
- 80.4. дядя (Ефим, y), свекровь (Мария, y).
- 80.5. невестка (x, Максим), внук (x, Анна).
- 80.6. племянница (Жанна, y), внук (Роман, y).
- 80.7. теща (Елена, y), внучка (Белла, y).
- 80.8. теща (Мария, y), свекор (x, Елена).
- 80.9. внук (Евгений, y), дядя (Юрий, y).
- 80.10. тетя (Татьяна, y), двоюродный брат (Дмитрий, y).
- 80.11. зять (Петр, y), тетя (Ольга, y).
- 80.12. свекровь (x, Луиза), дядя (Сергей, y).
- 80.13. тесть (x, Сергей), двоюродный брат (Вадим, y).
- 80.14. теща (x, Сергей), двоюродная сестра (x, Виктория).
- 80.15. племянник (Вадим, y), внучка (Ирина, y).
- 80.16. свекор (Петр, y), внучка (Полина, y).

В качестве примера произведем трассировку Пролог-программы «Родственные отношения» для первой цели варианта 3.16. Для этого имеющуюся программу дополним клаузой:

- 81) $\text{свекор}(x, y) \wedge$
- 82) $\text{отец}(x, z),$
- 83) $\text{отец}(z, u),$
- 84) $\text{мать}(y, u).$

Цель: 85) свекор (Петр, у).

Трассировка программы

1. В: цель () 85,
2. В: свекор (Петр, _) 81,
3. В: отец (Петр, _) 82,
4. П: отец (Петр, _) 35,
15. П: отец (Петр, _) 46,
16. У: отец (Петр, Ефим) 47,
17. В: отец (Ефим, _) 83,
18. П: отец (Ефим, _) 35,
26. П: отец (Ефим, _) 43,
27. У: отец (Ефим, Анатолий) 44 у,
28. В: мать (_ , Анатолий) 84,
29. П: мать (_ , Анатолий) 54,
39. П: мать (_ , Анатолий) 64,
40. У: мать (Татьяна, Анатолий) 65 у,
41. У: цель () 85.

10. СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА, ОСНОВАННЫЕ НА ЗНАНИЯХ

План лекции

1. Представление и использование нечетких знаний.
2. Нечеткая, вероятностная и монотонная логики.
3. Нечеткие множества и нечеткие выводы.
4. Многозначная логика.
5. Модальные логики.
6. Продукционные системы.
7. О механизмах вывода в экспертных системах.

Знания не всегда могут быть описаны точно часто встречаются так называемые «нечеткие» знания. Люди повседневно решают проблемы и делают заключения в среде нечетких знаний и, для того чтобы интеллектуальные системы обладали такими возможностями, как гибкость, широкий кругозор, адаптация, необходимо представление и использование нечетких знаний. Сам термин «нечеткость» многолик, его смысл недостаточно определен.

Как известно, классическая логика оперирует только с двумя значениями: истина и ложь. Однако этими двумя значениями довольно сложно представить (можно, но громоздко) большое количество реальных задач. Поэтому для их решения был разработан специальный математический аппарат, называемый нечеткой логикой. Основным отличием нечеткой логики от классической, как следует из ее названия, является наличие в ней не только двух классических значений 0 и 1, но и промежуточных значений, заключенных между 0 и 1 типа «не знаю», «скорее да, чем нет» и др. Данный подход больше похож на мышление человека, поскольку он на вопросы редко отвечает только «да» или «нет». Соответственно, вводятся расширения базовых логических операций конъюнкции, дизъюнкции и отрицания: $a \& b = \min\{a, b\}$, $a \vee b = \max\{a, b\}$, $a = 1 \rightarrow a$. Легко заметить, что при использовании только классических значений истинности (0 ложь, 1 истина) мы приходим к классическим определениям логических операций. Нечеткое логическое управление может использоваться для осуществления разнообразных интеллектуальных функций в разнообразных электронных товарах и бытовых приборах, в электронике, управлении производственными процессами и автоматизации.

В этом разделе мы рассмотрим разработанные и используемые в инженерии знаний, то есть в одном из направлений искусственного интеллекта, занимающимся

приобретением и представлением знаний, методы обработки «нечеткости». Эту «нечеткость» можно классифицировать по нескольким признакам: 1) недетерминированность управления выводом, 2) многозначность, 3) ненадежность, 4) неполнота и 5) нечеткость или неточность данных (знаний).

Кроме того, мы рассмотрим элементы модального исчисления и связанного с ним понятия модальной логики. Простейшей функцией модальной логики является формализация модальностей вида «возможность» и «необходимость». Другое ее применение моделирование и анализ парадигм «знание» и «вера». Для этого логические системы используют формальные языки с модальными операторами «веры» и «знания». Системы сочетают различные схемы аксиом и правила вывода для формализации свойств этих операторов. Модальные системы снабжены специфической семантикой. Различные модальные логики являются расширениями логики предикатов первого порядка, откуда они заимствуют аксиомы, правила вывода и теоремы. Одним из примеров таких логик являются временные модальные логики. Слово «временные» появляется в связи с тем, что изменение подразумеваемых ситуаций происходит во времени.

Еще один вопрос этого раздела направлен, с одной стороны, на точное определение понятия «экспертная система» как системы искусственного интеллекта и как ее можно программировать, а с другой показать состояние развития области экспертных систем в настоящее время.

4. 1. Представление и использование нечетких знаний

Любая формула логики высказываний или логики предикатов первого порядка может быть только либо истинной, либо ложной. Это означает, что при использовании этих логик испытатель имеет возможность делать только одно из противоположных суждений об истинности или ложности каких-либо событий, фактов, отношений в среде, с которой он имеет дело. Реальные среды далеки от такого идеала, при котором о любом явлении, событии, отношении можно говорить с категорической уверенностью, что оно имеет или не имеет места. Основными источниками такого рода неопределенности являются следующие факторы: недостаточная теоретическая изученность свойств среды; практическая невозможность или нежелание сделать описание знаний о среде полным вследствие высокой ее размерности и, как следствие, высокой сложности описания, например, по затратам времени и памяти; недетерминированность переходов среды из одного состояния в другое; многозначность; ненадежность; нечеткость или неточность знаний о среде.

В связи с перечисленными факторами возникают вопросы: как можно задавать такого рода неопределенности и как осуществлять вывод в этом случае?

Имеются различные подходы и основанные на них методы задания и вычисления неопределенности. Среди всех этих подходов можно выделить два главенствующих.

Один из них основан на введении числовой оценки, характеризующей степень уверенности или доверия к истинностным значениям вычисляемых формул. По степени уверенности можно судить, как далеки формулы от полной истинности или ложности. Иначе говоря, сначала формула вычисляется так же, как и прежде, и определяется, истинна она или ложна, а потом вычисляется степень уверенности в истинности этой формулы. Существуют различные подходы к вычислению степени уверенности. Чаще всего степень уверенности это положительное число, изменяющееся в диапазоне от 0 до 1. Чем больше это число, тем выше степень уверенности в истинности формул. Математические аппараты для вычисления степени уверенности также могут быть различными. Очень популярным является аппарат теории вероятностей. В этом случае степень уверенности истинности атомов задается как вероятность, а степень истинности формул, в которые эти атомы входят, вычисляется по законам условной вероятности.

При другом подходе для каждого атома $a(x_1, x_2, \dots, x_m)$ или фор-

мулы вводится целый спектр возможных значений, лежащих между 0 и 1, и правила вычисления этих значений. Вычисленные таким образом значения определяют

степень истинности формул. Фундаментом этого подхода является нечеткая логика, предложенная Л. Заде в начале 60-х годов прошлого века.

Для того чтобы интеллектуальные системы вышли за рамки простых символьных выводов и приблизились к мышлению человека, необходимы методы представления нечетких знаний и механизм выводов, работающий в их среде. Человек обладает самыми различными формами знаний, и пока еще не совсем ясно, каким образом они структурированы в его мозгу. Однако для использования знаний в компьютерах они должны быть формализованы и описаны. В этом состоит проблема представления знаний. Пока знания не формализованы и не описаны, они не могут быть использованы в компьютерах. Точно так же не могут быть использованы техническими средствами и нечеткости, предварительно не описанные в некоторой форме. Рассмотрим более подробно наиболее важные аспекты обработки нечеткости, которые чаще всего встречаются на практике.

Недетерминированность управления выводом

Недетерминированное управление наиболее характерно для систем искусственного интеллекта. Такое управление необходимо потому, что знания накапливаются фрагментарно, и нельзя априори определить цепочку логических выводов, в которых они используются. Другими словами, необходимо методом проб и ошибок выбрать некую цепочку выводов, и в случае неуспеха организовать перебор с возвратами для поиска другой цепочки и т.д. Такое управление является предпосылкой проявления гибкости и интеллектуальных способностей, позволяющих найти выход в самых различных ситуациях.

Поскольку эффективность простого поиска низка, возникает необходимость определения пути, по которому следует начать поиск в первую очередь. Например, при так называемом эвристическом поиске в пространстве состояний пути от исходного состояния заданной задачи до целевого состояния путем повторения возможных преобразований руководствуются оценочными функциями, которые не всегда бывают точными (имеют априорные значения). При этом для организации поиска в пространстве состояний используют дерево поиска или графы.

Многозначность и методы ее устранения

Многозначность интерпретации обычное явление при понимании естественных языков и распознавании изображений и речи. При понимании естественных языков большими проблемами становятся многозначность смысла слов, многозначность их подчиненности, многозначность местоимений в контексте и т.п. Как правило, устранение многозначности обеспечивается за счет более широкого контекста и семантических ограничений. При обработке изображений часто бывает многозначной интерпретация элементов изображения (контуры, области и т.п.). В общем случае устранить многозначность помогают более широкие пространственные отношения и другие способы, например, методы дискретной и вероятностной релаксации. Метод релаксации это метод систематического устранения многозначности при интерпретации изображений с помощью циклических операций. В основе этого метода лежит численное решение дифференциальных уравнений в частных производных.

Ненадежные знания и выводы

В задачах, которые решают интеллектуальные системы, иногда приходится применять ненадежные знания и факты, представить которые двумя значениями истина или ложь (1 или 0) трудно. Существуют знания, достоверность которых, равна, скажем, 0.7. Такую ненадежность в современной физике и технике представляют вероятностью, подчиняющейся законам Байеса, но в инженерии знаний было бы нелогично иметь дело со степенью надежности, приписанной знаниям (или незнаниям) изначально с байесовской вероятностью. Поэтому был разработан метод использования коэффициентов уверенности (доверия). Этот метод не имеет теоретического подкрепления, но стал примером обработки ненадежных знаний, что оказало заметное влияние на создание

систем искусственного интеллекта и, прежде всего, на создание экспертных систем, например, PROSPECTOR (США).

Для решения сложных задач можно использовать метод разбиения этих задач на несколько подзадач. Каждая подзадача в свою очередь разбивается на более простые подзадачи, поэтому задача в целом описывается иерархически. Знания, которые по условиям подзадач определяют условия задач высшего уровня, накапливаются фрагментарно. В задачах с ненадежными данными знания могут иметь степень надежности не только равную 1, но и промежуточные значения между истиной и ложью. Связь между подзадачами, на которые разбита данная задача, оперирующая двумя понятиями истина и ложь, может быть представлена с помощью операций конъюнкции и дизъюнкции. В задачах с ненадежными исходными данными кроме конъюнкции и дизъюнкции важную роль играет комбинированная связь. Такая связь независимо подтверждает или опровергает цель на основании двух и более доказательств.

Указанные выше методы обработки ненадежных знаний продолжают развитие методов теории вероятностей. Они более субъективны, но по своей логической сути более сильны. Вообще говоря, логика играет важную роль при рассмотрении фундаментальных понятий представления знаний и выводов. Даже принципы обработки ненадежных знаний основаны на логике.

4. 2. Нечеткая, вероятностная и немонотонная логики

Нечеткая логика, выделилась из теории нечетких множеств. Нечеткая логика это разновидность непрерывной логики, в которой логические формулы могут принимать истинностные значения между 0 и 1. В нечеткой логике истинностное значение это некоторое произвольное субъективное значение, заключенное между 0 и 1, не имеющее никакого статистического смысла. Пусть t_x и t_y значения истинности предпосылок X и Y некоторого правила A . Тогда истинностное значение t предпосылок в случае связей «И» и «ИЛИ» определяется соответственно следующим образом:

1. При связи «И» $t = \min(t_x, t_y)$
2. При связи «ИЛИ» $t = \max(t_x, t_y)$.

Если в общем случае t_n есть значение истинности правила A , то значение истинности t_A , распределенное на вывод, определяется по формуле $t_A = \min(t, t_n)$.

Определение минимума это идея, свойственная нечеткой логике и отличающая ее от других методов. В качестве комбинированной связи можно рассматривать одну из связей «И» или «ИЛИ».

Нильсон Н. в 1985 г. ввел понятие вероятностной логики, в которой всем логическим формулам приписывается байесовская вероятность. Связь логики и вероятности важна с точки зрения рационального построения новой теории на основе теории логического моделирования. И хотя эта теория еще не доведена до использования на уровне вычислений, для некоторых задач можно определить логический вывод с вероятностью.

Полностью описать мир задач чрезвычайно сложно. Например, знание «птицы летают» верное, однако встречаются и не летающие птицы, то есть это неполное знание. Другой пример: можно перечислить все предметы, которые находятся в комнате, но того, чего в ней нет, перечислить невозможно, поскольку это бесконечное множество предметов. Точно так же можно перечислить верные знания (в некоторой предметной области), но перечислить неверные знания и разумно их определить невозможно. Поэтому удобно в базе знаний определять исключительно верные знания, а все, что не определено, считать заведомо неверным. Утверждения, которые не упомянуты ни как истинные, ни как ложные, принято относить к ложным утверждениям. Эту ситуацию называют гипотезой закрытого мира. Предикат `not` в языке программирования Пролог принимает значение «истина», если среди его аргументов нет решения, и его выполнение основано на гипотезе закрытого мира. Эта же гипотеза применяется для неопределенных знаний

(разновидности неполных знаний).

Классическая логика исходит из предпосылки, что набор определенных в ней аксиом (знаний) полон, и правильный вывод не меняется, даже если впоследствии добавлена новая аксиома. Такое свойство называется монотонностью. Если допустить, что в базу знаний добавлено новое знание: «как правило, птицы летают (за некоторым исключением)», то обнаружится свойство немонотонных выводов. А именно, при добавлении новой аксиомы иногда возможно отрицание вывода, который считался верным в некоторой системе аксиом (базе знаний).

Как средство формальной обработки неполных знаний, при которой необходимы немонотонные выводы, предложены и активно исследуются методы немонотонной логики. Известны следующие типичные подходы: немонотонная логика Макдермотта и Доула (в которой вводятся условные логические операции), логика умолчания Рейтера, окружение (в смысле ограничения) Маккарти и др. Маккарти известен как создатель языка Лисп; он всячески подчеркивал, что в дополнение к знаниям, которые в настоящее время стали объектом обработки компьютеров, необходимо сделать скачок к тому, что мы называем здравым смыслом. По его словам это самая большая проблема искусственного интеллекта, и в ней не обойтись без немонотонного вывода.

К системам, связанным с неполными знаниями и управлением такими знаниями, относится система поддержания значений истинности. В базе знаний этой системы, неполной и содержащей противоречия, все знания делятся на достоверные и недостоверные, и предусмотрено упорядочение базы с целью устранения недостоверных знаний. Если при добавлении новых знаний возникает противоречие, то выполняется повторная проверка классов достоверных и недостоверных знаний.

4.3. Нечеткие множества и нечеткие выводы

В постановке задачи часто имеются элементы неопределенности. Эта неопределенность не обязательно имеет вероятностный характер. В частности, она может быть лингвистической, то есть проистекать из свойств языка описания задач. Например, в справочниках по медицинской диагностике числа, как правило, не называются: надо знать, повышенная температура у больного или пониженная, пожелтели ли белки глаз, имела ли место существенная потеря веса и т.п. Такие характеристики, как «большой», «малый», «старый», «прочный», «быстрый» не только не задают количественную меру, но и имеют сильную контекстную зависимость (например, молодой пенсионер скорее будет старше старого атлета, а карликовый бегемот наверняка больше крупной мыши). Человек умеет принимать решения (возможно, не самые лучшие, но, по крайней мере, приемлемые) в условиях, которые с различных точек зрения содержат неопределенность или

нечеткость. Он оперирует с нечеткими целями и идеями; использует методы, которые строго не определены; принимает решения в ситуациях с неполной информацией. Желательно придать такие способности и системам искусственного интеллекта. Кстати отметим, что системы автоматического управления, построенные на нечеткой логике, оказались более «внимательными» к основному сигналу и менее чувствительными к шумам.

Нечеткое (размытое) множество есть совокупность упорядоченных пар

$$A = \{x, \mu_A(x)\}, x \in X,$$

где $\mu_A(x) \in [0,1]$ функция принадлежности элемента x множеству A . Считается, что

- $A = B$, если $\mu_A = \mu_B$ для всех $x \in X$;
- $A \subset B$, если $\mu_A < \mu_B$;
- A есть дополнение к множеству A , если $\mu_{\bar{A}} = 1 - \mu_A$.

Последнее условие предполагает нормировку меры принадлежности по всем элементам рассматриваемого множества. В задачах диагностики считается, что состояние «подтверждено», если показатель доверия (значение степени принадлежности) выше

выбранного верхнего порога, и «отвергнуто» если ниже нижнего порога. Трасса вывода продолжается от «подтвержденного» состояния или от состояния с наибольшим весом.

Возведение в положительную степень a функции принадлежности позволяет перераспределять ее значения между элементами множества. При $a > 1$ разность значений усугубляется, при $a < 1$ значения выравниваются.

Мера принадлежности элемента x к результату операций над множествами определяется следующим образом:

- для пересечения $\mu_{A \cap B}(x) = \min\{\mu_A(x), \mu_B(x)\}$,
- для объединения $\mu_{A \cup B}(x) = \max\{\mu_A(x), \mu_B(x)\}$,
- для алгебраической суммы $\mu_{A + B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x)$.

Пусть $f: X \rightarrow Y$ четкое отображение из универсума X в универсум Y . Тогда $y = f(x)$ образ x , а $x \in X$ прообраз y . Множество всех прообразов элемента y обозначается общепринятым способом: $f^{-1}(y)$.

Если A нечеткое множество в универсальном множестве (универсуме) X с заданной функцией принадлежности μ_A , то его образом при отображении f называется нечеткое множество $B = f(A)$ с функцией принадлежности

$$\mu_B(y) = \sup_{x \in f^{-1}(y)} \mu_A(x) \text{ для } f^{-1}(y) \neq \emptyset;$$

$$\mu_B(y) = 0, \text{ если } f^{-1}(y) = \emptyset.$$

В этой формуле $\sup$ обозначает верхнюю границу (наибольшее значение) функции принадлежности $\mu_B(y)$. Объединение всех супремумов дает нам снова нечеткое множество B . Это множество обычно называют принципом обобщения. С помощью принципа обобщения можно осуществить переход от нечетких множеств, каждое из которых имеет свою функцию принадлежности и задано на своем универсуме из некоторого множества универсумов, к одному нечеткому множеству, заданному на одном универсуме и с одной функцией принадлежности.

Если B нечеткое множество в Y с заданной функцией принадлежности μ_B , то прообразом этого множества при отображении f

называется нечеткое множество $A = f^{-1}(B)$ с функцией принадлежности $\mu_A(x) = \mu_B(f(x))$.

Пусть $B(x)$ нечеткое множество в пространстве Y и функция принадлежности $\mu_B(y | x)$ зависит от x как от параметра. Тогда функция

$\mu_B(y) = \sup_x \mu_B(y | x)$ определяет нечеткое отображение из множества X во множество Y . Образом нечеткого множества A во множестве X в этом случае является нечеткое множество B с функцией принадлежности

$$\mu_B(y) = \sup_x \min\{\mu_A(x), \mu_B(y | x)\}.$$

Нечеткое решение выбирается при нечеткой цели G и нечетких ограничениях C . Решение определяется множеством $D = G \cap C$. Нечеткое решение может рассматриваться как нечетко сформулированная инструкция. Можно выбрать альтернативу, имеющую наибольшую степень принадлежности к множеству D .

Возможны случаи, когда некоторые цели (ограничения) важнее остальных. Решение может быть определено как нечеткое множество D с функцией принадлежности

$$\mu_D(x) = \min_i \{ \mu_{G_i}(x), \mu_{C_j}(x) \}, \text{ где } \mu_{G_i}(x) = \mu_{C_j}(x) = 1.$$

В нечетких дедуктивных выводах также используют традиционное правило вывода *modus ponens*. Если знания это нечеткие множества (F, G, F', G') , то вывод G' из F' по правилу *modus ponens* записывают следующим образом:

$$F', F \wedge G.$$

$$G$$

Эта запись имеет существенную особенность: множества F и F' , не обязаны совпадать. Если F и F' близки друг к другу, то их можно более или менее сопоставить и получить вывод G' , так сказать, в сфере их совпадения.

Пусть $\mu_A: U \rightarrow [0, 1]$, $\mu_B: U \rightarrow [0, 1]$ функции принадлежности нечетких множеств A и

В соответствии. Функция $t = \mu_B(x)$ определяет степень принадлежности элемента $x \in U$ нечеткому множеству B . Используя принцип обобщения, можно получить нечеткое множество $C(B, A)$:

$$C(B, A) = U \sup\{\mu_{A(x)} \mid x \in A \& t = \mu_B(x)\} / t.$$

$$\mu = \mu_B(\mu)$$

Это нечеткое множество называют степенью совместимости нечеткого множества B с нечетким множеством A . Универсумом нечеткого множества $C(B, A)$ является интервал $[0, 1]$. Чем ближе нечеткое множество B к нечеткому множеству A , тем больше элементов множества $C(B, A)$, в которых значение t совпадает с $\mu_A(t)$. Если эти множества одинаковы, то для всех t $t = \mu_A(t)$.

Во многих прикладных задачах, например, распознавание сигналов и образов по эталонам, сравнение одной формулы с другой (эталонной), семантика эталонов задается нечетким множеством A , а семантика вновь полученного образа, сигнала или формулы множеством B . В этом случае степень совместимости используют для оценки степени близости вновь полученного сигнала, образа или формулы, семантика которых выражается нечетким множеством B , к эталонам, семантика которых выражается нечетким множеством A . На основе степени совместимости могут быть построены различные меры для оценки степени близости.

4.4. Модальные логики

При развитии различных исчислений как формальной основы для рассуждений о тех или иных явлениях нашей среды постоянно наблюдается стремление по возможности полно и точно выразить и учесть динамику изменений этой среды. Самым простым исчислением является логика высказываний. Для того чтобы выразить динамику изменений среды, например во времени, средствами логики высказываний, придется ввести переменные, истинность которых будет означать наступление того или иного момента времени. Если таких моментов много, то описание в логике высказываний становится крайне неудобным. Логика предикатов первого порядка является более сложным, но и более выразительным исчислением в результате введения кванторов, функций и предикатов. Функции и предикаты могут зависеть от временных переменных, которые позволяют описывать динамику среды. Ситуационные исчисления еще более удобны для описания динамики среды, поскольку они основаны на использовании ситуационных переменных, значения которых могут зависеть от времени или пространства.

В некоторых случаях рассуждения о динамике изменения во времени ситуаций и связанных с этими ситуациями фактов не требуют знания ни точного времени наступления этих ситуаций, ни использования самих ситуаций (ситуационных переменных). Достаточным является установление истинности того, что та или иная формула, выражающая определенные отношения между объектами среды, станет, в конце концов, истинной, не станет истинной, станет истинной с какой-либо степенью уверенности и т.п. при определенных значениях объектных переменных в некоторой ситуации, явное значение которой нас не интересует. На естественном языке для выражения этих отношений обычно используют слова типа «допустимо», «необходимо», «возможно» и другие, выражающие определенную степень уверенности наступления каких-либо событий во времени или пространстве, но без указания точного времени или места. Возможность делать в логике высказывания, содержащие выразительные средства, которые характеризуют подобную степень уверенности или «силу» высказывания, называют модальностью. Исчисления, в которых введены специальные символы или слова (модальные операторы), с помощью которых это делается, называют модальными исчислениями.

Рассмотрим пример одного из таких исчислений, основанного на логике предикатов первого порядка и называемого обычно временным модальным исчислением или временной модальной логикой. В этой логике изменение подразумеваемых ситуаций

происходит во времени, причем время течет по линейному закону. Обсудим более подробно концептуальную суть временного модального исчисления.

Рассмотрим, например, утверждение «Идет дождь». Очевидно, что истинность этого утверждения зависит от двух факторов (параметров): даты и места, которые подразумеваются, но не употреблены. Выбирая определенные дату t_0 и место l_0 и уточняя с их помощью указанное утверждение, получаем другое утверждение: «Идет дождь в месте l_0 во время t_0 ». Это утверждение полностью определено и должно быть либо ложным, либо истинным. Теперь можно ввести атом дождь (l, t) : «Идет дождь в месте l во время t », который имеет равные приоритеты для обоих параметров l и t . Модальный подход предполагает два различных приоритета или уровня для параметров l и t . Параметр с более высоким приоритетом определяет ситуацию и не употребляется явно в высказываниях. Так, если в качестве переменной, полностью определяющей ситуацию, выбрать время, то вместо атома дождь (l, t) будет использоваться атом дождь (l) . Значение этого предиката зависит как от ситуации (в данном случае времени или даты), так и от места. Но точное значение ситуации нас не интересует, поскольку мы собираемся рассуждать о таких вещах, как возможность дождя в месте l в принципе или неизбежность дождя в месте l когда-либо вообще и т.п.

Переход от логики предикатов к модальной логике не такой резкий, как переход от логики исчисления высказываний к логике предикатов. При этом выбор переменных, определяющих ситуацию, произволен и зависит от субъективной точки зрения. Имеются очевидные преимущества введения модального формализма. Он позволяет нам сделать переменные, определяющие ситуацию, более важными, чем все остальные, и ввести неявную зависимость формул модальной логики от ситуаций и тем самым от переменных, ее определяющих. Временная модальная логика предполагает наличие базисного отношения достижимости между ситуациями $R(s, s')$, которое определяет возможность перехода от одной ситуации s в другую ситуацию s' . Это отношение имеет место, если ситуация s' достижима из ситуации s .

Вернемся к примеру о дожде. Как уже отмечалось, здесь ситуации определяются датами (днями), то есть ситуация и день или дата в данном случае одно и то же. Между ситуациями можно установить отношение достижимости $R(s, s')$, если ситуация (день) s' следует за ситуацией (днем) s . Главная идея обозначений, принятых в модальной логике, состоит в стремлении избежать обозначения ситуаций (дат в нашем примере) и отношения достижимости. Вместо этого вводят два специальных модальных оператора, описывающих свойства ситуаций, достижимых из данной ситуации: оператор необходимости $\Box$ оператор возможности $\Diamond$.

Смысл этих операторов заключается в следующем. Пусть w является некоторой формулой модальной логики. Тогда $\Box w$ и $\Diamond w$ также формулы модальной логики. Значение формул w , $\Box w$ и $\Diamond w$ неявно зависит от некоторой текущей ситуации s . Формула $\Box w$ считается истинной в ситуации s , если формула w истинна на всех ситуациях, достижимых из ситуации s , включая собственно ситуацию s . Формула $\Diamond w$ считается истинной в ситуации s , если формула w истинна хотя бы в одной ситуации, достижимой из ситуации s . Обозначим через $w(s)$ значение формулы w в ситуации s . Тогда правила вывода истинных значений модальных формул $\Box w(s)$ и $\Diamond w(s)$ можно записать следующим образом:

$$w(5), V5'(R(5, 5') \& w(5')) \mid \Box w(5), \quad (4.1)$$

$$w(5), 35'(R(5, 5') \& w(5')) \mid \Diamond w(5). \quad (4.2)$$

Эти правила введены только для того, чтобы пояснить смысл модальных операторов. На самом деле, как уже говорилось, в формулах модальной логики символ ситуации не используется, но подразумевается. Модальная формула записывается с помощью пропозициональных символов, предикатных символов (включая равенство), функциональных символов, констант, переменных, классических операторов (логических связок) и кванторов, а также модальных операторов.

Формулу без модальных операторов, являющуюся по существу формулой логики предикатов первого порядка, называют иногда статической. Модальная формула, называемая иногда динамической, содержит статические подформулы, к которым применены классические и модальные операторы. Истинностное значение модальной формулы в некоторой ситуации можно найти на основе правил (4.1) и (4.2). При этом предполагается, что в каждой рассматриваемой ситуации все классические символы формулы можно интерпретировать и конкретизировать таким образом, что ее истинное значение будет определено. Например, формула

дождь (l) $\wedge \Diamond$ дождь (l) эквивалентна утверждению «Если в какой-либо день в каком-либо месте l идет дождь, то наступит другой день, когда в этом же месте l не будет дождя». Для того чтобы эта формула стала истинной при некотором значении места l, должен существовать день 5, в который дождь начался и затем прошел в этом же месте. Утверждение «Если в какой-либо день в каком-либо месте l идет дождь, то он будет идти и в любой другой день в этом же месте l» соответствует формула

дождь (l) $\rightarrow \Box$ дождь (l).

Для того чтобы эта формула стала истиной при некотором конкретном значении места l, должен существовать день s, в который дождь начался и затем уже никогда не прекращался в этом месте.

Приведем примеры формул модальной логики, поясняя на основе правил (4.1) и (4.2), при каких условиях эти формулы будут истинными.

Формула $\Diamond(\Box w)$ истинна в ситуации s, если существует хотя бы одна ситуация 5', достижимая из ситуации 5, в которой истинна формула $\Box w$. Иными словами, истинность формулы $\Diamond(\Box w)$ означает, что существует ситуация 5', достижимая из ситуации 5, такая, что в ситуации 5' и во всех достижимых из нее ситуациях истинна формула w.

Формула $\Box(\Diamond w)$ истинна в ситуации 5, если во всех ситуациях 5', достижимых из ситуации 5, истинна формула w. Иными словами, истинность формулы $\Box(\Diamond w)$ означает, что для всех ситуаций 5', достижимых из ситуации 5, можно найти хотя бы одну достижимую из 5' ситуацию, в которой истинна формула w.

Формула $\Box(w \rightarrow \Box w)$ истинна, если для всех ситуаций 5', достижимых из 5, истинна формула $w \rightarrow \Box w$. Истинность формулы $w \rightarrow \Box w$ в ситуации 5' означает истинность формулы w во всех ситуациях, достижимых из ситуации 5'. Таким образом, истинность формулы $\Box(w \rightarrow \Box w)$ означает, что формула w становится всегда истинной в любой ситуации 5', достижимой из 5, и остается истинной для всех ситуаций, достижимых из 5'.

Модальная логика построена на базе логики предикатов первого порядка и в ней, естественно, справедливы понятия, используемые в последней: общезначимость, выводимость и др. В частности, формулы

$\Box w \wedge \Diamond w, \Box w = \Diamond w, \Box(w1 \rightarrow w2) \rightarrow (\Box w1 \rightarrow \Box w2)$

общезначимы. Например, последняя формула означает (на основе правила modu5 ropen5), что истинность формулы $\Box(w1 \rightarrow w2)$ всегда влечет истинность формулы $\Box w1 \rightarrow \Box w2$.

Задавая различные ограничения на отношение достижимости R, можно получать различные модальные исчисления. В нашем случае полагаем, что отношение R всегда рефлексивно ($5 R 5$) и транзитивно ($((51 R 52 \ \& \ 52 R 53) \rightarrow (51 R 53))$). Вследствие этого общезначимыми являются и следующие две формулы:

- $\Box w \wedge w$ благодаря рефлексивности (если во всех ситуациях, достижимых из s, истинна формула w, то w истинна и в ситуации s);

- $\Diamond\Diamond w \wedge \Diamond w$ благодаря транзитивности отношения R (если существует ситуация s2, достижимая из s1, которая в свою очередь достижима из s0, причем в ситуации s2 формула w истинна, то существует ситуация s3, достижимая из s0, в которой формула w также истинна, и $s3 = s2$).

Во временной модальной логике на базовое отношение налагаются

дополнительные условия, состоящие в том, что ситуация s' достижима из ситуации s , если ситуация s может перейти в ситуацию s' благодаря изменениям среды во времени. Рассматриваемая нами временная модальная логика имеет дело с последовательностями вида $s_0, s'_1, \dots$, причем S_j достижимо из s , тогда и только тогда, когда $i < j$, а s_0 есть единственная начальная ситуация.

Благодаря дискретности последовательностей ситуаций мы можем ссылаться не только на ситуации, которые являются далекими последователями данной ситуации, но и на единственную следующую ситуацию. Поэтому в ряде случаев целесообразно ввести оператор следующей ситуации, обозначаемый символом $\circ$, и отношение непосредственного следования $r(s, s')$. Очевидно при этом, что транзитивное рефлексивное замыкание отношения r дает отношение достижимости R , введенное выше. Последовательность $s_0, s_1, \dots$, для которой отношение $r(s_i, s_{i+1})$ имеет место для всех $i > 0$, интуитивно соответствует развитию процессов во времени, наблюдаемому как последовательность дискретных ситуаций.

Заметим, что существует большое количество других видов модальностей, среди которых отметим модальность вида процент уверенности, модальность вида возможность, модальность вида невозможность, модальность вида необходимость, модальность вида случайность. Семантические правила, определяющие смысл перечисленных модальностей основаны на теории нечетких множеств и соответствующих функциях принадлежности. При этом с каждым видом модальностей связана собственная шкала (отрезок $[0, 1]$). Любое модальное понятие, имеющее градацию n , содержит оценку, которой на соответствующей шкале соответствует точка $n = N / 100$. Между различными видами модальностей существует определенная взаимосвязь.

4.5. Многозначные логики

Многозначная логика является развитием обычной (двузначной) логики. Двоичная логика часто оказывается слишком жесткой, когда речь идет о моделировании рассуждений в системах искусственного интеллекта, об описании состояний технических устройств или о распознавании образов. В самом деле, как оценить истинность таких суждений: «Я, по всей видимости, буду у вас сегодня поздно вечером» или «О здешнем климате мне трудно что-либо утверждать». Вряд ли их истинность можно жестко маркировать истиной или ложью.

В большинстве сложных систем имеет место ситуация, при которой решения по управлению принимаются при неполной информации или информации, достоверность которой не является стопроцентной. В этом случае не удастся оценить истинность информации адекватно, если стоять на позициях использования всего двух значений истинности. Именно поэтому многозначная логика получила мощный стимул для своего развития и стала дисциплиной, имеющей немалое практическое значение.

В k -значной логике ($k \geq 2$) переменные и функции принимают значения из множества $\{0, 1, 2, \dots, k-1\}$, при этом 0, как и в двузначной логике, означает абсолютную ложь, а $k-1$ соответствует абсолютной истине. Промежуточные значения могут характеризовать различные градации истинности данной переменной. Можно, например, считать, что они характеризуют значения вероятностей истинности. В этом случае в качестве значений истинности удобно рассматривать не целые

числа, а дроби вида $\frac{i}{k-1}$, где i принимает значения от 0 до $k-1$. Можно также считать, что значения истинности соответствуют, например, совокупности значений лингвистической переменной «Частота», шкала которых может иметь вид {никогда, редко, ни часто ни редко, часто, всегда}. Здесь ($k=5$).

Итак, многозначная логика это раздел математической логики, изучающий математические модели логики высказываний. Эти модели отражают две основные черты логики высказываний множественность значений истинности высказываний и возможность построения новых, более сложных высказываний из заданных высказываний при помощи логических операций, которые позволяют также по значениям истинности исходных высказываний устанавливать значение истинности сложного высказывания.

Примерами многозначных высказываний являются суждения с модальным исходом («да», «нет», «может быть») и суждения вероятностного характера. Примерами логических операций могут служить логические связки типа «не», «и», «или», «если..., то...».

Исторически первой моделью многозначной логики при $k \geq 2$ была 3-значная (модальная) логика Я. Лукасевича (1878-1956), предложенная им в 1920 г. Эту логику мы достаточно подробно рассмотрели в п.4.3. В 1955 г. Лукасевич построил систему 4-значной логики, а затем и бесконечнозначную модель логики.

При неограниченном возрастании k от многозначных логик можно осуществить предельный переход к счетнозначным и даже непрерывным (континуальным) логикам.

Для многозначных логик можно построить алгебру логики на многозначный случай. Например, три основные операции, определенные в алгебре логики: конъюнкция, дизъюнкция и отрицание при переходе k -значным логикам могут быть определены так:

$$x \wedge y = \min(x, y); \quad x \vee y = \max(x, y); \quad \neg x = 1 - x.$$

При $k=2$ эти определения операций дают те же операции, которые для двузначного случая определены в алгебре логики.

Многозначные логики используются при планировании поведения в роботах и других системах искусственного интеллекта, действующих в не полностью описанных средах, при принятии решений в условиях отсутствия исчерпывающей информации, в экспертных системах при частичных знаниях в определенной области, при решении задач квантовой механики, разрешении парадоксов классической математической логики и др.

Определение. Функция $f: S^k \rightarrow S$, отображающая k -мерный

k -значный кортеж $(x_1, \dots, x_k) \in S^k$ в элемент $f(x_1, \dots, x_k) \in S$;

во множество $S = \{0, 1, \dots, k-1\}$, то есть, $f: S^k \rightarrow S$, называется функцией k -значной логики.

Подобно функциям двузначной логики, k -значные логические функции можно задавать в виде конечной таблицы. Количество столбцов в такой таблице равно k^n , а количество всевозможных функций выражается числом k^k . Это число сильно возрастает с ростом k даже для небольших значений n . Так, при $k=2$ будем иметь $2^4 = 16$ функций двух переменных. Поэтому нечего и думать о

том, чтобы изучить свойства таких функций путем их перебора, как это делается в двузначной логике. Обычно ограничиваются рассмотрением наиболее важных многозначных функций одной и двух переменных. По аналогии с двузначными функциями вводятся понятия равенства и суперпозиции функций k -значной логики. Проблема полноты для многозначной логики далека еще от полного решения. Полученные результаты в основном сводятся либо к общим условиям существования полных систем функций, либо к рассмотрению конкретных базисов и полных систем, которые по тем или иным причинам считаются удобными для представления.

Пример 4.1. Константы и функции одной переменной.

В k -значной логике имеется k констант $0, 1, \dots, k-1$.

Среди функций одной переменной наиболее используемыми являются следующие функции:

1) характеристические функции i -го порядка, число которых равно k . Эти функции определяются равенством

$$x_i = \begin{cases} 1 & \text{при } x = i, \\ 0 & \text{при } x \neq i; \end{cases}$$

2) функция инверсии (отрицание Лукасевича) $\neg x = 1 - x$;

3) функция циклического отрицания (отрицание Поста) $x = (x + 1) \pmod{k}$.

Например, при $k=5$ имеем следующую таблицу функций одной переменной:

Таблица 4.1

Сводная таблица функций одной переменной

$x f(x)$	0	1	2	3	4
	4	0	0	0	0
	0	4	0	0	0
$\leq p_2(x)$	0	0	4	0	0
(x)	0	0	0	4	0
$p_4(x)$	0	0	0	0	4
X	4	3	2	1	9
X	1	2	3	4	0

Очевидно, инверсия характеристических функций выражается соотношением:
 $\neg f(x) = 1 - f(x)$ (0 при $x = i$, $f(x) = 1$ при $x \in I$).

В многозначной логике используются также функции одной переменной более общего вида:

$\Gamma_1 = P_{\text{при } x = i} \text{ } f_{ij} \text{ } W$ [0 при $x \in I$, частным случаем которых при $f = \leq$ являются характеристические функции, то есть $\Gamma_1 \cdot \Gamma_2 = \Gamma_1 \wedge \Gamma_2$.. Γ_n . При $f =$ получается другой тип характеристических функций

, $f \cdot$ (1 при $x = i$,

$V_i W$ |_{0 при $x \in I$}

В двузначном случае функции $\neg$, $\wedge$ и $\vee$ совпадают с отрицанием, а $\neg \neg X = X$.

Пример 4.2. Функции двух переменных.

Большое значение имеют следующие функции двух переменных:

- 1) n -значная дизъюнкция $\vee$; $\Gamma_1 \vee \Gamma_2 = \Gamma_1 \vee \Gamma_2$;
- 2) n -значная конъюнкция $\wedge$ — Т-П. $\Gamma_1 \wedge \Gamma_2 = \Gamma_1 \wedge \Gamma_2$;
- 3) n -значная функция Шеффера-Вебба $\neg \Gamma_1 \vee \neg \Gamma_2$;
- 4) функция сложения по модулю $\oplus$;
- 5) функция умножения по модулю $\otimes$.

Значения этих функций при $\leq$ приведены в таблице 4.2:

Таблица 4.2

Сводная таблица функций двух переменных при $\leq$

'1	0000	1111	2222	3333
Γ_2	0123	0123	0123	0123
$X_j \vee X_2$	0123	1123	2223	3333
$x_1 \oplus x_2$	0000	0111	0122	0123
$X_j \otimes X_2$	1230	2230	3330	0000
$(x_1 + x_2) \pmod{4}$	0123	1230	2301	3012
$(x_1 x_2) \pmod{4}$	0000	0123	0202	0321

При $\leq$ функции $\vee$ и $\wedge$ совпадают с соответствующими функциями двузначной логики и между ними имеют место аналогичные зависимости:

$\neg \neg X = X$; $\neg \neg \neg X = \neg X$;

Как и в двузначной логике, n -значные дизъюнкция и конъюнкция подчиняются коммутативному, ассоциативному и обоим дистрибу-

тивным законам. Поэтому вместе с инверсией эти операции превращают множество $\leq$ в булеву алгебру.

Кроме указанных функций двух переменных вводятся еще такие элементарные функции:

$\Gamma_1 \cdot \Gamma_2$, если $0 < x < y < k - 1$,

- 6) усеченная разность $_, _ \Gamma \dots / \dots \dots \dots _$
 (к — 1, если $0 < x < y < k - 1$,
 7) импликация $_ \Gamma$
 $l(fc\ 1)\ x + y$, если $0 < y < x < fc-l$;
 8) функция Вебба $\blacksquare \setminus .v\ y\ j - 1\ j < j\ \blacksquare$
 $t(x\ y)$, если $0 < y < x < fc-l$;
 9) разность по модулю
 $\{k(y\ x)$, если $0 < x < y < fc-l$.

Построение моделей многозначной логики осуществляется по аналогии с построением двузначной логики. Так, индивидуальные высказывания логики, разбитые на классы с одним и тем же значением истинности, приводят к понятию множества $\langle$ констант модели, которые фактически отождествляют все индивидуальные высказывания, заменяя их соответствующими значениями истинности. Переменные

высказывания приводят к переменным величинам; ■, которые в качестве значений принимают значения из множества-.

Логические связки приводят к множеству элементарных функций (операций), которые, как и аргументы, принимают значения из множества $\mathcal{A}$.

Сложные высказывания, построенные из индивидуальных и переменных высказываний с использованием логических связей, приводят к множеству $\mathcal{M}$ над множеством $\mathbf{A}$. Значение истинности из множества $\mathcal{M}$ сложного высказывания является функцией от соответствующих значений истинности высказываний, входящих в данное сложное высказывание. В модели эта функция приписывается формуле, соответствующей данному сложному высказыванию. Говорят также, что формула реализует эту функцию.

Множество всех функций n -значной логики обозначают символом

-■. Как и для двузначных логик, вводится понятие замыкания | множества функций.
 Задание конкретной модели многозначной логики -■< считается эквивалентным указанию
 множеств -■-:, -, ::

и $\mathcal{M}$, при этом говорят, что модель порождается множеством Σ . Если множество Σ конечно, то модель является конечно порожденной и называется формульной моделью.

Многозначная логика обладает определенной спецификой, состоящей в рассмотрении задач и подходов, возникающих при исследовании объектов с позиций математической логики, математической кибернетики и алгебры. Так, с позиции математической кибернетики элементарные функции --значной логики интерпретируются как схемы, построенные из элементов, и осуществляющие переработку входной информации в выходную. Модели многозначной логики рассматриваются как языки, описывающие функционирование сложных управляющих систем, например, автоматов с памятью, компоненты которых могут находиться в некотором числе различных состояний. Вместе с тем существует ряд задач логики и кибернетики, которые связаны с изучением соответствий между множествами --и $[\wedge]$. При этом роль множества $\vdash$ несколько затушевывается, сводясь к способу сведения второго множества к первому. В этом случае приходят к другой модели многозначной логики, представляющей собой алгебру, элементами которой являются функции, принимающие в качестве значений, как и их аргументы, элементы из множества $\leq$. В качестве операций в этих алгебрах используется специальный набор, эквивалентный в смысле соответствий --и $[M]$ множеству формул, построенных из функций множества $\blacksquare$, то есть получению сложных функций из заданных функций путем подстановки одних функций вместо аргументов других. Эти алгебры называются алгебрами --значных логик.

В качестве примера такой алгебры можно привести так называемую алгебру Поста [1]. Пусть $\mathcal{A} = \langle A, \wedge, \vee, \neg, \rightarrow \rangle$ — алгебра Поста, где $\wedge, \vee, \neg, \rightarrow$ — унарные операции, бинарные операции. Эти операции в предположении, что каждая функция $f: A^n \rightarrow A$ с учетом фиктивных переменных зависит от переменных, определяются так:

$$= K^{x_2 > x_3} > \dots, x_p,$$

$$ML(x_1 \wedge x_2 \rightarrow x_{p+1}) = f(x_1, x_2, \dots, x_p) \text{ Если } p > 1, \text{ и } -7/ = T. \Gamma =$$

$$\blacksquare \text{ из } v. \blacksquare \blacksquare \blacksquare \blacksquare = . \bullet$$

$$(Y/)(x_1, x_2, \dots, x_p, x_{p+1}) = f(x_2, x_3, \dots, x_p, x_{p+1})$$

$$\text{а для} \quad \text{и} _ \blacksquare \text{ и } \blacksquare \text{ и, } - \text{И} \dots < 0$$

$$\blacksquare \dots - < \dots$$

Основные задачи многозначной логики:

1. Задача об описании

Эта задача характерна для формульной модели многозначной логики. Смысл этой проблемы состоит в указании для заданного множества Γ всех формул из Γ , реализующих функции из множества $\mathcal{F}$. Частным случаем такой задачи является важный вопрос математической логики об указании всех формул, реализующих заданную константу, что, например, для двузначной логики эквивалентно построению всех тождественно истинных (или тождественно ложных) высказываний.

2. Задача о тождественных преобразованиях

Пограничным вопросом между математической логикой и алгеброй, примыкающим к задаче об описании, является задача о тождественных преобразованиях. В ней при заданном множестве M требуется выделить в некотором смысле простейшее подмножество пар формул из множества $\mathcal{F}$, позволяющих путем подстановки выделенных равных формул одной вместо другой формулы получить из любой формулы все формулы, равные ей, то есть получить полную систему тождеств.

3. Задача о сложности реализации функций

Характерный для теории управляющих систем вопрос о сложности этих систем естественно возникает и по отношению к формулам и функциям из многозначной логики. Типичной при таком подходе является задача о сложности реализации функций. На множестве всех элементарных формул некоторым способом вводится числовая мера (сложность формул), которая затем распространяется на множество всех формул, например, путем суммирования мер всех тех элементарных формул, которые участвуют в построении заданной формулы. Требуется для заданной функции указать ту простейшую формулу, которая реализует эту функцию и имеет наименьшую сложность, а также выяснить, как эта сложность зависит от некоторых свойств рассматриваемой функции.

4. Задача представления функций многозначной логики в СДНФ и СКНФ

Воспользовавшись понятием характеристических функций для двузначного случая можно представить СДНФ и СКНФ бу-

левой функции выражениями

$$f(x_1, x_2, \dots, x_n) = \bigvee f(a_1, a_2, \dots, a_n) \< p_{a\pm} (x_j \& (x_j) =$$

$$= \bigwedge f(a_1, a_2, \dots, (x_j) \vee \dots \vee \bigwedge_{a_n} (x_n).$$

Здесь дизъюнкция в первом выражении и конъюнкция во втором берутся по всем двоичным наборам $(a_1, a_2,$

значений аргументов $x_1, \dots, x_n$. Ясно, что оба выражения равны 1 только на тех наборах $\{0, 1\}^n$, на которых функция принимает единичные

значения, так как при этом $\bigvee \dots = 1$ и $\bigwedge \dots = 1$. На

тех наборах $\{0, 1\}^n$, на которых функция принимает нулевые значения, $\bigvee \dots = 0$ и $\bigwedge \dots = 0$, следовательно, оба выражения обращаются в нуль.

Таким образом, первое выражение представляет собой СДНФ. Ее члены — дизъюнкции, играют роль конъюнкты единицы. При этом в диз-

ъюнкцию входят только те из них, которые соответствуют $\sim x_i$ и x_i . Второе выражение представляет собой СКНФ.

Его члены — конъюнкции, называемые характеристиче-

скими дизъюнкциями, играют роль конъюнкты нуля. При этом в

конъюнкцию входят только те из них, которые соответствуют x_i и $\sim x_i$.

Приведенные в начале пункта 4 выражения распространяются на случай $n \geq 2$ и рассматриваются как n -значные СНФ. Возможность такого обобщения следует из того, что по определению характеристические конъюнкции равны 1 и характеристические дизъюнкции равны нулю только при условии $x_i = 2^{i-1}$ для всех $i = 1, \dots, n$. Из определения n -значных дизъюнкций и конъюнкций ясно, что СДНФ содержит только те члены, которые соответствуют неравенству

$\sum_{i=1}^n x_i \neq 2^n - 1$, а СКНФ содержит только те члены, которые соответствуют неравенству $\sum_{i=1}^n x_i \neq 0$. Кроме того,

для упрощения можно не выписывать значение в $\sum_{i=1}^n x_i$ в первом случае и значение

$\sum_{i=1}^n x_i$ во втором.

Например, пусть 4-значная функция задана таблицей:

	000	111	222	333
x_1	012	012	012	012
$f(x_1, x_2, x_3, x_4)$	203	031	310	320

В соответствии с изложенными выше правилами получим: $f(x_1, x_2, x_3, x_4) = (x_1 \neq 0 \vee x_2 \neq 0) \wedge (x_1 \neq 2 \vee x_2 \neq 2) \vee (x_1 \neq 4 \vee x_2 \neq 4) \vee (x_1 \neq 6 \vee x_2 \neq 6) \wedge (x_3 \neq 0 \vee x_4 \neq 0) \wedge (x_3 \neq 2 \vee x_4 \neq 2) \wedge (x_3 \neq 4 \vee x_4 \neq 4) \wedge (x_3 \neq 6 \vee x_4 \neq 6)$

$\wedge (x_1 \neq 0 \vee x_2 \neq 2 \vee x_3 \neq 4 \vee x_4 \neq 6) \wedge (x_1 \neq 2 \vee x_2 \neq 4 \vee x_3 \neq 6 \vee x_4 \neq 0) \wedge (x_1 \neq 4 \vee x_2 \neq 6 \vee x_3 \neq 0 \vee x_4 \neq 2) \wedge (x_1 \neq 6 \vee x_2 \neq 0 \vee x_3 \neq 2 \vee x_4 \neq 4)$

5. Задача о функционально полных системах функций в многозначных логиках

Проблема полноты состоит в указании таких подмножеств заданного замкнутого, то есть совпадающего со своим замыканием, множества, для которых выполняется равенство $[M_2] = M_0$ то есть имеет место свойство функциональной полноты множества функций M_1 во множестве M_0 .

Выделяют два подхода к решению задачи о полноте алгоритмический и алгебраический. В первом случае ставится вопрос о существовании алгоритма, устанавливающего полноту или неполноту систем функций, описанных на некотором языке. Во втором случае переходят к изучению свойств решетки подалгебр данной алгебры k -значной логики и решают задачу о полноте, используя эти свойства. Здесь большое значение имеет понятие критериальной системы подалгебр, каждая из которых содержит все максимальные подалгебры алгебры M_0 (то есть предполные классы). В связи со вторым подходом к решению задачи о полноте возникает глобальная задача описания решетки замкнутых классов данной модели многозначной логики.

Возможность представления любой функции многозначной логики в СКНФ служит доказательством полноты системы, включающей дизъюнкцию, конъюнкцию, характеристические функции и константы (система Россера-Тьюкетта). Таким образом, эта система k -значной логики порождается алгеброй $\langle M, \vee, \wedge, f_i, c_i \rangle$,

где $c_i = i$, а f_i характеристические функции,

$S : S \rightarrow S$. В системе Россера-Тьюкетта конъюнкция и дизъюнкция определяются соответственно формулами

$x \vee y = \min(x, y)$, $x \wedge y = \max(x, y)$, а характеристические функции имеют вид $f_i(x) = 1$ при $x = i$, $f_i(x) = 0$ в противном случае.

Было предложено много других функционально полных систем в k -значной логике. Например, полную систему образуют дизъюнкция и циклическое отрицание (система Поста). Эта логика определяется алгеброй

$A_n = \langle M, \vee, \sim \rangle$; $M = \{0, 1, \dots, k-1\}$; $x \vee y = \min(x, y)$; $\sim x = (x+1) \bmod k$; $x, y \in M$

Логика Поста обобщает логику Буля $k=2$, где « $\sim$ »

символ отрицания, а $\sim i = k-i$.

Для доказательства полноты системы Поста достаточно выразить константы,

характеристические функции и конъюнкцию через дизъюнкцию и циклическое отрицание. Так как $\mathbf{a} \cdot \mathbf{b} = \langle \mathbf{a}'\mathbf{b}' \rangle \text{ — } \langle \mathbf{a}\mathbf{b} \rangle = \langle \text{—} \rangle$ (сложение следует проводить по $\langle \rangle$), то все константы можно

получить с помощью функции циклического отрицания и дизъюнкции. Можно также показать, что $\langle \mathbf{a}; \mathbf{b} \rangle = \langle \mathbf{a} \rangle \vee \langle \mathbf{b} \rangle$, где

$3 < \mathbf{a}$ и $\mathbf{b} \text{ — } 1$ и $2 = \dots$ Действительно, если $\langle \mathbf{a} \rangle = \langle \mathbf{b} \rangle$, то

$\langle \mathbf{a} \rangle = \langle \mathbf{b} \rangle$ — и $\mathbf{a} \text{ — } \mathbf{b}$; л: $\mathbf{a}'\mathbf{b}' \text{ — } \mathbf{a}\mathbf{b} = \langle \text{—} \rangle$ 1, то есть $\langle \mathbf{a} \rangle = \langle \mathbf{b} \rangle$. Конъюнкция

выражается через дизъюнкцию и инверсию: $\langle \mathbf{a}\mathbf{b} \rangle = \langle \mathbf{a}'\mathbf{b}' \rangle$.

В свою очередь $\langle \mathbf{a} \rangle = \langle \mathbf{b} \rangle$ $\langle \mathbf{a}\mathbf{b} \rangle = \langle \mathbf{a}'\mathbf{b}' \rangle$.

в чем можно убедиться, полагая $\langle \mathbf{a} \rangle$ и $\langle \mathbf{b} \rangle$. Функции $\langle \mathbf{a} \rangle$ были определены в п.4.4.1 как функции одной переменной более общего вида. Приведенная цепочка зависимостей и служит доказательством полноты системы Поста.

Система, состоящая из единственной функции $\langle \mathbf{a}\mathbf{b} \rangle$; (система Вебба) также является полной. Циклическое отрицание и дизъюнкция выражаются через функцию Шеффера-Вебба следующим образом:

и $\langle \mathbf{a} \rangle$ и $\langle \mathbf{b} \rangle$ и $\langle \mathbf{a}\mathbf{b} \rangle$; $\langle \mathbf{a} \rangle = \langle \mathbf{a}\mathbf{b} \rangle \vee \langle \mathbf{b} \rangle$ $\langle \mathbf{a} \rangle \sim \langle \mathbf{b} \rangle = \langle \mathbf{a}\mathbf{b} \rangle$

$= x_1 x_2 + (k - 1)$. Поэтому из полноты системы Поста следует

полнота системы Вебба.

Если найдены все полные во множестве $\mathbf{p}$ подмножества $\langle \mathbf{a}\mathbf{b} \rangle$, никакое собственное подмножество которых уже неполно во множестве $\mathbf{p}$, то подмножества $\langle \mathbf{a}\mathbf{b} \rangle$ являются базисами, то есть полными независимыми системами функций. Доказано, что при $k > 2$ существуют

$\langle \mathbf{a}\mathbf{b} \rangle$ -значные логики с конечным базисом, и в то же время не имеющие конечной полной системы тождеств, тогда как в двузначной логике каждый замкнутый класс имеет конечную полную систему тождеств. При $k = 2$ во множестве $\langle \mathbf{a}\mathbf{b} \rangle$ имеется континуум замкнутых классов.

Существуют замкнутые классы с базисами заданной конечной или счетной мощности, а также классы, которые не имеют базисов, при этом сами семейства классов без базисов или со счетным базисом континуальны.

Пример 4.3. Рассмотрим класс функций $\langle \mathbf{a}\mathbf{b} \rangle$, где $\langle \mathbf{a}\mathbf{b} \rangle = \langle \mathbf{a} \rangle \vee \langle \mathbf{b} \rangle$, а при $\langle \mathbf{a}\mathbf{b} \rangle$ значение функции $\langle \mathbf{a}\mathbf{b} \rangle$ равно 0 на всех наборах, кроме $\langle 2, 2, \dots, 2 \rangle$, на котором $\langle \mathbf{a}\mathbf{b} \rangle = 1$. Это пример класса без базиса.

Пример 4.4. Рассмотрим трехзначную функцию Вебба, заданную таблицей истинности:

$\mathbf{y}$ $\mathbf{x}$	0	1	2
0	1	2	0
1	2	2	0
2	0	0	0

Из этой таблицы видно, что функция Вебба задается формулой $f(x,y) = x \vee y = \max((x,y) + 1) \pmod 3$.

В общем виде $\langle \mathbf{a}\mathbf{b} \rangle = \langle \mathbf{a} \rangle \vee \langle \mathbf{b} \rangle$ $\langle \mathbf{a}\mathbf{b} \rangle = \langle \mathbf{a} \rangle \vee \langle \mathbf{b} \rangle$ $\langle \mathbf{a}\mathbf{b} \rangle = \langle \mathbf{a} \rangle \vee \langle \mathbf{b} \rangle$

Функция Вебба является полной в конечнозначной логике, а соответствующая алгебра Вебба $\langle \mathbf{a}\mathbf{b} \rangle$ порождает-значную логику, в частности при $\langle \mathbf{a}\mathbf{b} \rangle$.

Существует ряд признаков полноты, в которых рассматриваются множества функций, содержащих некоторые совокупности функций от одной переменной и еще только одну функцию, существенно зависящую не менее чем от двух переменных (критерий Слупецкого, критерий Яблонского, критерий Саломеа, теорема Пикара).

6. Задача о полиномиальном представлении функций в k -значной логике

Подобно каноническим полиномам в алгебре Жегалкина можно рассматривать задачу о полиномиальном представлении функций и в $\langle \mathbf{a}\mathbf{b} \rangle$ -значной логике. При решении

этой задачи будем исходить из возможности выражения любой функции в так называемой - П (сигма-пи) форме:

где суммирование ведется по всем наборам значений переменных, и используются операции сложения и умножения по модулю-. Действительно, так как $\sum_{i=1}^n x_i = 1$ при $x_i = 1$ и $\sum_{i=1}^n x_i = 0$ при $x_i = 0$, то произведение $\prod_{i=1}^n x_i$ отлично от нуля только на наборе $x_i = 1$ и равно 1. Поэтому на каждом наборе

единственный ненулевой член суммы всегда равен $\prod_{i=1}^n x_i$, то есть значению функции на данном наборе. Теперь необходимо найти полиномиальные представления функций $f(x_1, \dots, x_n)$.

В соответствии с малой теоремой Ферма $a^{p-1} \equiv 1 \pmod{p}$. Если простое число, то множество классов вычетов по модулю p образует поле и при $V = p$ обе части последнего сравнения можно разделить на a^{p-1} . Тогда $a \equiv 1 \pmod{p}$, что после умножения обеих частей на a и прибавления к ним 1 дает

$1 - a \equiv 1 - a^p \pmod{p}$. Полученное выражение справедливо при $a \equiv 1 \pmod{p}$, а при $a \not\equiv 1 \pmod{p}$ его левая часть равна 1, то есть для всех значений

левая часть совпадает с функцией $\chi(a)$.

Обобщая этот результат, получим $\chi(a) = \frac{1}{p-1} \sum_{k=0}^{p-2} a^{k(p-1)}$, где $a \equiv 1 \pmod{p}$.

Подставляя эти значения в формулу, и производя умножения по модулю p получим искомый многочлен, представляющий данную функцию. Все коэффициенты этого многочлена выражаются целыми числами, заключенными между 1 и $p-1$, а степени входящих в него переменных не превышают $p-1$.

Пример 4.5. Пусть 3-значная функция $f(x_1, x_2)$ двух переменных

	000	111	222
x_2	012	012	012
$f(x_1, x_2)$	200	120	010

и задана таблицей истинности:

Представим функцию в полиномиальной форме:

$$f(x_1, x_2) = 2(1 + x_1)(1 + 2x_2) + (1 + 2(x_1 - 1)^2)(1 + 2x_2) + 2(1 + 2(x_1 - 1)^2)(1 + 2(x_2 - 1)^2) - 4(1 + 2(x_1 - 1)^2)(1 + 2(x_2 - 1)^2),$$

откуда после выполнения арифметических операций по модулю 3 получим:

$$f(x_1, x_2) = 2 - x_1 + x_2 + x_1 x_2 - x_1^2 x_2.$$

Из изложенного видно, что система функций n -значной логики, состоящая из операций суммы и произведения по модулю n и констант, является полной, если простое число. В случае составного числа эта система неполна, что ограничивает область ее применения. Таким образом, имеет место следующая теорема: любая функция из множества $\mathcal{F}_n$ представима полиномом по модулю n тогда и только тогда, когда n — простое число.

7. Задача минимизации функций n -значной логики

Процесс минимизации многозначных функций зависит от выбранной полной системы функций и значительно сложнее, чем в двузначной логике. Для этой цели можно использовать тождественные преобразования логических формул, стремясь получить выражение, содержащее по возможности наименьшее число входящих в него характеристических функций и аргументов. Развита также общие методы приведения канонических форм представления функций к минимальным формам.

Рассмотрим, например, систему Россера-Тьюкетта. Тождественные преобразования в этой системе основаны на соотношениях:

$x \cdot x = x$ (идемпотентность);

$x \cdot 0 = 0$; $x \cdot 1 = x$; $0 \cdot x = 0$; $1 \cdot x = x$; $x \cdot x = x$; $x \cdot x = x$ (свойства констант);

$x \cdot 1 = x$ при условии $x \neq 0$;

!■ $qP_1(x)V_2 \rightarrow p_2(x) \vee \dots \vee V(kl)_{k-1}(x) = x$ (свойства характеристических функций). С

Для этого элементы множества значений-значной логики объединяются в попарно пересекающиеся подмножества. В соответствии с одним из способов общим элементом всех таких подмножеств принимается 0, а остальные элементы 1, 2, ..., – области значений-

, φ —, то есть φ — функций вида:

$\dots | 1$ при $\dots | 2$ при $\dots$;
 $\dots$ (и при $\dots$; $\dots$ $\dots$;
 $\dots$ ($\dots = 1$ при $\dots = i$,

$$22. < 2 \therefore$$

нимает значение $\frac{1}{\alpha_j}$, имеет вид

$$e_{\alpha_j}(x_1) \cdot e_{\alpha_j}(x_2) \cdot \dots \cdot e_{\alpha_j}(x_n)$$

Минимальная форма для n -значной функции совпадает с дизъюнкцией минимальных форм двужначных логических функций f_{ij} , принимающих значения на двух элементарных множествах.

Пример 4.6. Для функции примера 4.4.5 имеем:

$$/(^*p^*_2) = (e_{11}(x_1)e_{21}(x_2)V e_{21}(x_1)e_{12}(x_2))v(e_{02}(x_1)e_{02}(x_2) \vee e_{1r}\{x^{\wedge}e_{3r}\{x_r\} \vee v_{e_{32}(x_1)}e_{12}(x_2))^{y(e_{03}(x_1)e_{23}(x_2))^{y_{e_{13}(x_1)}e_{13}(x_2))^{y_{e_{23}(x_1)}e_{03}(x_2)}} \vee \vee_{e_{33}(x_1)e_{13}(x_2)}V_{e_{33}}(\%>_{33}(\wedge_2)).$$

157

4.6. Продукционные системы

При логическом подходе утверждения представляются в виде формул некоторой логики традиционной, модальной, многозначной, нечеткой и т.п. Логический подход позволяет применить развитый аппарат вывода новых утверждений (фактов) на основе имеющихся в базе знаний (БЗ). Результат легко интерпретируется на естественном языке. Здесь гарантируется правильность вывода, а в случае его невозможности установление причины затруднения. Последнее свойство используется при поддержании непротиворечивости баз данных (знаний).

Чаще всего используются продукционные системы с конечным набором правил вывода вида «если $A_1, A_2, \dots, A_n$, то B », то есть если выполняются все условия $A_1, A_2, \dots, A_n$, то следует выполнить действие B . Часть правила после слова «если» называется посылкой, а часть правила после слова «то» выводом (действием, заключением). Посылки $A_1, A_2, \dots, A_n$ обычно являются фактами. С помощью фактов описывается текущее состояние предметной области. Факты бывают истинными, ложными либо правдоподобными. В последнем случае истинность факта допускается с некоторой степенью уверенности. Итоговое условие можно рассматривать как образец ситуации, в которой следует выполнять указанное действие. Действие B одновременно трактуется как добавление нового истинного факта и описание текущего состояния процесса вывода. Правила описывают причинно-следственные связи между фактами (при необходимости и правилами). Имея такое описание, его не трудно ввести в ЭВМ. Для этого достаточно организовать базу фактов и базу правил и построить интерпретатор базы правил, который по описанию текущего состояния по предъявленным ему фактам выводил бы заключения из этих фактов. На этой общей простой идее и построены продукционные системы, которые имеют следующую структуру (рис. 4.1).

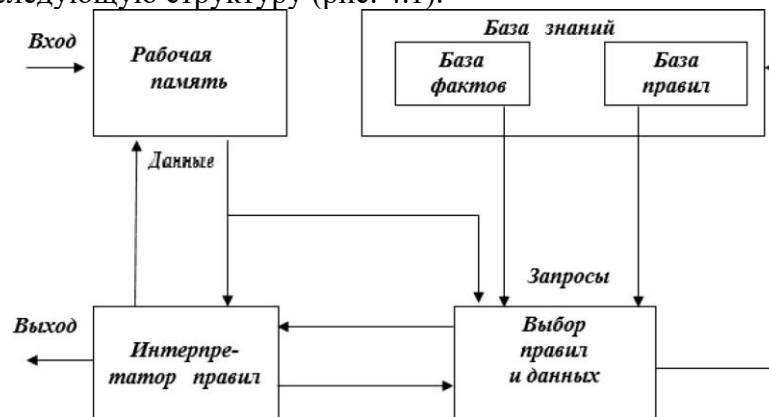


Рисунок 4.1. Структура продукционной системы

В продукционных системах используются два основных механизма вывода: прямой и обратный (вывод от данных и вывод от цели соответственно). В прямом выводе идут от известных фактов и на каждом шаге вывода применяют все допустимые, то есть с истинными посылками, правила, которые порождают новые факты до тех пор, пока не будет порожден факт-цель. При обратном выводе рассуждение ведется в обратном направлении: если цель согласуется с заключением правила, то посылку правила принимают за подцель или гипотезу, и этот процесс повторяется до тех пор, пока не будет получено совпадение подцели с известными фактами. Чисто прямая цепочка рассуждений ведет к неуправляемому режиму задания вопросов в диалоге. Обратная цепочка рассуждений будет, как правило, приводить к настойчивому продвижению к цели, но требует предварительного определения цели. Наиболее удачные продукционные системы используют комбинированные варианты.

При работе с неполными (неточными) данными система должна быть способной делать предположения, а при получении новой информации по необходимости отказываться от сделанных предположений (целей) и последующих умозаключений. В

некоторых случаях допустимо считать, что предположение X имеет место, если, используя ограниченные ресурсы, нельзя доказать противоположное.

При неудаче вывода на очередном шаге выполняется откат (возврат), легко реализуемый с помощью стека. Однако возвращаться надо не на один шаг назад, а в точку, где было сделано ошибочное предположение.

В общем случае продукционная система на очередном шаге своей работы:

- выбирает правило (продукцию) с истинным условием;
- выполняет указанное в нем действие;
- меняет разрешающее условие некоторого другого правила.

Правила некоторым образом упорядочены. После отработки шага проверка применимости возобновляется, начиная с наиболее приоритетного правила. Этот цикл продолжается до обнаружения тупика или до применения правила, помеченного как заключительного.

Продукционные системы могут управляться данными (предусловиями правил) и целями (действиями правил). В наиболее развитых системах механизм вывода может иметь иерархию правил, метаправила или сложные схемы управления, подобные сетям Петри.

В традиционном программировании все места ветвления указываются явно. Подобный способ удобен, если ветвление скорее исключение, чем норма. В задачах искусственного интеллекта, решаемых в продукционных системах, программу лучше рассматривать как совокупность независимых модулей, управляемых образцами. На каждом шаге работы такая программа анализирует текущую ситуацию и по результату анализа определяет ее обработчика. Особенности такого подхода с точки зрения представления знаний являются:

- 1) Разделение постоянных знаний, хранимых в БЗ, и временных знаний, хранимых в рабочей памяти.
- 2) Структурная независимость модулей (облегчается их разработка и модификация).
- 3) Отделение схемы управления от модулей.

Преимущества продукционных систем от других систем искусственного интеллекта состоят в следующем:

- Модульность: каждое правило описывает небольшой относительно независимый фрагмент знаний.
- Инкрементность: возможность добавления новых правил относительно независимо от других правил.
- Удобство модификации как следствие модульности и инкрементности.
- Прозрачность системы (легкость прослеживания логики и объяснения вывода).

Впервые метод продукций был предложен Е. Постом в 1936 г. и в 1963 г. Хомским был использован для определения языков и формальных грамматик. С 1974 г. он находит широкое применение в многочисленных программах, которые получили название эксперт-ных систем (ЭС).

Экспертные системы составляют наиболее разработанное направление систем искусственного интеллекта. ЭС обычно определяют как программы для ЭВМ, моделирующие действия эксперта-человека при решении задач в узкой предметной области на основе накопленных знаний, составляющих базу знаний. Существует большое количество профессиональных ЭС разных типов, предназначенных для решения задач в разных областях. В частности, продукционные ЭС находят широкое применение в военных приложениях: идентификация самолетов на основании визуальных наблюдений (идентификация); помощь при запуске самолетов, предупреждение аварийных ситуаций (диагностика, управление); анализ поля боя, классификация целей в реальном масштабе времени (управление); рекомендация по размещению орудий на поле боя (контроль);

оценка ситуации по разведанным о средствах связи (интерпретация); предсказание возможных конфликтов (прогнозирование); определение целей для атаки в воздухе, выбор наиболее эффективного оружия (управление) и др.

Экспертные системы включают знания специалистов (экспертов) в некоторой слабо структурированной и трудно формализуемой узкой предметной области и способные в этой области предлагать и объяснять разумные решения.

Их основным назначением являются:

- интерпретация данных (например, определение структуры химических соединений, минералов, насекомых и растений; классификация типа личности по результатам тестирования);
- диагностика состояния технического устройства или больного;
- мониторинг (наблюдение за процессом и сигнализация об аномалиях);
- прогнозирование вероятного развития событий (погоды, урожая, экономической конъюнктуры);
- планирование последовательности действий для достижения цели;
- обучение будущих специалистов.

Все эти задачи по существу сводятся к распознаванию образов.

С помощью ЭС решаются задачи, требующие значительных объемов эмпирических ассоциативных знаний, субъективного опыта, связанных с изменяющейся обстановкой. ЭС наилучшим образом отвечают ситуациям, где

- критическим фактором является не только диагноз, но и время;
- необходимо постоянное и длительное во времени пребывание в экстремальных условиях;
- имеется дефицит квалифицированных специалистов;
- нет специалистов, достаточно компетентных по всему кругу проблем предметной области;
- необходимо гибкое средство обучения и контроля подготовленности персонала.

Примеры некоторых экспертных систем были приведены в начале второй главы. Более подробно рассмотрим ЭС MYCIN, предназначенную для определения диагностики бактериальных заболеваний. Сложность проблемы состоит в том, что приступать к лечению надо как можно скорее, а точный диагноз связан с выращиванием культур и требует не менее двух суток. Врач оказывается перед выбором: либо прописать средство с широким спектром действия (но малоэффективное и с большой вероятностью побочного эффекта), либо назначить специфическое лекарство со значительным риском промахнуться. Обычно выбирается первый путь. В системе MYCIN около 400 правил. Предпосылки комбинируются по правилам многозначной логики, в которой операции $\& = \min$, $\vee = \max$. В системе имеется 24 предикатных функции (в том числе равен, узнать, определить, ...); используется 80 атрибутов (идентичность, место, чувствительность, ...); 11 объектов (микроорганизм, культура, лекарство,...). Нечеткие рассуждения допускают одновременное существование взаимоисключающих значений атрибута. Система ищет путь максимальной определенности. Хотя система «знает» меньше, чем человек, тщательность и неуклонность ею проверок более чем компенсирует этот недостаток.

С другой стороны, ЭС лишены интуиции. Поэтому для них неудобны широкие предметные области, творческие рассуждения, учет пространственных и временных отношений. Исключаются также ситуации, где знания экспертов относятся к области чувств (подготовка чайных смесей, дегустация вин) или не могут быть выражены словами (если в роли эксперта выступает опытный автомеханик без специального образования, использующий в основном жесты и «ненормативную» лексику).

Стратегии вывода в ЭС реализуют метазнания о том, как и когда использовать различные источники знаний объектного уровня. Метазнания могут выражать знания о разбиении задачи на подзадачи, кооперации источников знания, различных вариантах

поиска. Чтобы избежать лишних сопоставлений, образцы правил организуются в сеть.

ЭС должна быть способна к метарассуждениям, то есть рассуждениям о собственной работе и структуре. Тогда метазнания можно применить для

- обнаружения синтаксических и смысловых ошибок в предметных правилах;
- обоснования целесообразности применения правил;
- адаптации системы к изменившейся обстановке;
- явного указания возможностей и ограничений ЭС, неизвестных разработчикам.

Достоинством большинства ЭС является наличие в их структуре компонента объяснения. Этот компонент опирается на диагностическое дерево, построенное при выводе решения. Каждая вершина дерева, которой соответствует некоторое правило, снабжается поясняющим текстом. Планировщик при поиске решения запоминает тот путь, по которому он движется. Каждый шаг на этом пути альтернативный выбор, справедливость которого может быть подкреплена текстом объяснения. При запросе на объяснение можно выдать все эти тексты, двигаясь по диагностическому дереву снизу вверх и повторяя в обратном направлении путь, пройденный планировщиком (как трассировка программы на Прологе). Если же пользователь запросит, почему не годится некоторая другая гипотеза, подсистема объяснения должна пройти снизу вверх путь до встречи с основным путем, зафиксировать разделяющее правило и комментарий к ней.

Иногда в ЭС включается также подсистема доверия, которая демонстрирует пользователю компетентность ЭС.

Основным сдерживающим фактором в разработке ЭС считается нехватка инженеров по знаниям. Труден также процесс приобретения знаний у экспертов и поддержание непротиворечивости приобретенных знаний. При разработке ЭС приходится решать проблемы выбора языка, создания управляющего механизма, накопления фактов и правил. Из них первые две проблемы слабо связаны с предметной областью в отличие от третьей проблемы, которая имеет предметную ориентацию. Обычно ЭС пишутся на Лиспе или на Прологе. В Прологе порядок следования предложений влияет на результат; многие встроенные предикаты дают побочные эффекты. Есть и другие недостатки Пролога при программировании ЭС. Однако можно применить любой известный разработчику ЭС язык программирования, а затем написать вариант с использованием правил на эффективных и компактных языках Си, Фортран, Паскаль, Форт и др.

Значительно удобнее воспользоваться «пустой ЭС» (оболочкой), включающей в себя «машину вывода», схему построения объяснений и интерфейс, и заполнить ее правилами из новой области. Разумеется, выбор этого варианта автоматически определит и язык программирования. Приведем пример такой оболочковой продукционной ЭС с обратной цепочкой рассуждений. Оболочка состоит из четырех функционально законченных блоков:

- набора правил вывода (продукций) в формате «если предпосылка, то заключение»;
- фактов (обычно пользователь их не задает, они появляются во время работы механизма вывода);
- вопросов (если ЭС не будет хватать фактов, то она будет задавать пользователю соответствующие вопросы для рассматриваемых объектов);
- списка разрешенных значений для каждого объекта, то есть всех возможных значений, которые может принимать объект.

Для удобства данные можно хранить в виде разных файлов на диске, а затем загружать их. Остановимся более подробно на каждом из редакторов. Редактор правил позволяет создать новое правило, загрузить правила из файла, сохранить правила в текущий файл, сохранить правила в другой файл, редактировать и удалять правило.

Оболочка снабжена интерактивными редакторами для каждого модуля. Общий вид графической интерфейсной части данной оболоч

ки представлен на обложке книги. Вкладки «Правила», «Факты», «Вопросы», «Значения» позволяют переключаться соответственно между редакторами правил, фактов, вопросов и списка допустимых значений объектов. После задания всех правил, добавления вопросов и ввода разрешенных значений можно приступить к процессу достижения цели. Для этого нужно в поле «Цель» ввести объект-цель и нажать кнопку «Достижение цели». Нажатие кнопки «Достижение цели» запустит механизм вывода ЭС, а нажатие кнопки «Просмотр объяснения» позволяет просмотреть «рассуждения» ЭС при достижении цели. После этого можно просмотреть факты, которые появились в ходе достижения цели. При повторном нажатии кнопки «Достижение цели» эти факты будут использоваться как известные, поэтому, если нужно достичь другую цель или ту же самую цель, но с другими фактами, нужно очистить базу фактов. Для этого достаточно переключиться на вкладку «Факты» и нажать кнопку «Очистить базу фактов», которая служит для удаления из памяти оболочки ЭС всех правил, объектов и их допустимых значений, фактов и вопросов.

Правило состоит из двух частей: предпосылок и заключений. Как предпосылки, так и заключения являются фактами базы знаний, выраженными парами «объект значение». В верхнем поле для ввода задается имя нового правила. В среднем поле список предпосылок в формате «имя объекта = значение». В нижнем поле список заключений правила в том же формате. Если при этом хотя бы одна ячейка в поле заключений останется незаполненной, то это часть заключения не будет учтена, поэтому процесс заполнения правил требует особой внимательности. Все предпосылки любого правила (если их несколько) соединены по умолчанию логическим «И» (конъюнкцией).

Редактор фактов позволяет создать новый факт, загрузить факты из файла, сохранить факты в текущий или в другой файл, добавить новое значение объекта, удалить выделенное значение объекта или все значения для выбранного объекта, удалить все факты базы знаний.

Редактор вопросов позволяет загрузить вопросы из файла, сохранить вопросы в текущий или в другой файл. Для добавления /изменения вопроса для выбранного объекта достаточно ввести вопрос в соответствующее поле ввода и нажать кнопку «It!».

С помощью редактора списка допустимых значений объекта можно загрузить список допустимых значений из файла, сохранить список допустимых значений в текущий или в другой файл, удалить выделенное значение для выбранного объекта. Для добавления допустимого значения для выбранного объекта надо ввести его в поле с названием «Значение» и нажать на кнопку «Добавить».

Рассмотрим два примера конкретных экспертных систем.

Пример 4.7. Обнаружение неисправности автомобиля. Обнаружив, что анализируемая система (в данном случае автомобиль) неисправна, специалист пытается определить ту ее подсистему, в которой могут быть неполадки (подсистема электропитания, охлаждения, подачи топлива и т.д.). После выявления такой подсистемы он приступает к поиску неисправных компонентов (рис. 4.2).

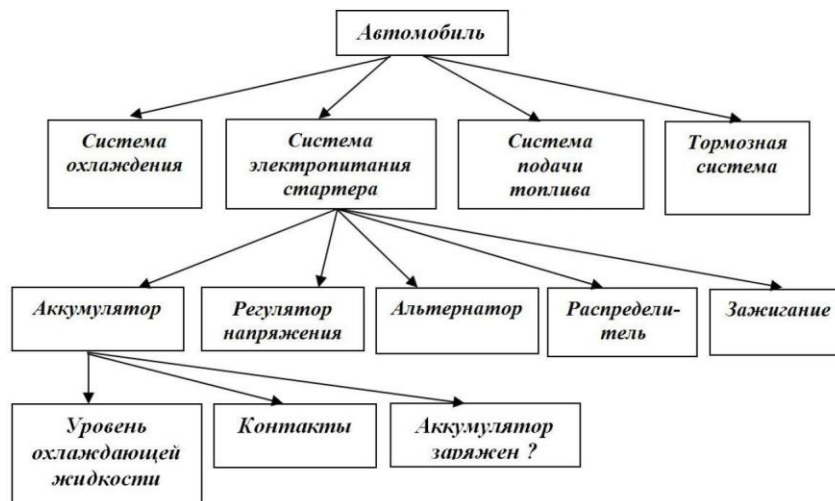


Рисунок 4.2. Иерархическая схема поиска неисправностей в автомобиле

Этот процесс выглядит так, как будто специалист постоянно обращается к неявно заданному множеству правил, получая промежуточные заключения при выполнении каких-либо условий. Промежуточные заключения в свою очередь становятся условиями для вывода следующих заключений.

Каждое правило в этом простом примере содержит атрибуты и значения. Так, в первом правиле выражение «не заводится» является значением, а слово «двигатель» атрибутом. Любое правило состоит из одной или нескольких пар «атрибут значение» и заключения.

Например, в рассматриваемом случае (с упрощенными правилами) цепь рассуждений будет выглядеть приблизительно так:

Правило 1:

Если двигатель не заводится и стартер двигателя не работает, то неполадки в системе электропитания стартера.

Правило 2:

Если двигатель не заводится и стартер двигателя не работает, то неполадки в системе подачи топлива.

Правило 3:

Если неполадки в системе топлива и показатель уровня топлива находится на нуле, то газовая камера пуста.

Правило 4:

Если неполадки в системе питания стартера и нарушены контакты аккумулятора, то плохо присоединен аккумулятор.

Продукционную модель, послужившую нам для построения базы правил, по сути можно рассматривать как вариант модели «стимул реакция», которая была предложена в психологии и теории автоматов задолго до появления ЭС. Многие ученые полагают, что эта модель вполне адекватно описывает поведение человека.

Пример 4.8. Приведем более сложную базу знаний, которая по некоторым медицинским показателям определяет продолжительность жизни человека. Разумеется, эта база знаний носит учебный характер и не претендует на серьезные выводы. Она состоит из трех частей: разрешенных значений атрибутов, вопросов к пациенту и правил.

разрешзн(вес)=55_или_меньше,55-85,85_или_больше

разрешзн(сложение)=мелкое, крупное

разрешзн(жиры)=норма, избышек

разрешзн(холестерин)=норма, избышек разрешзн(соль)=норма, избышек

разрешзн(потребление_алкоголя)= =не_употребляет, умеренное, чрезмерно

разрешзн(пол)=м,ж

разрешзн(возраст)=25_или_меньше,25-55,55_или_больше

разрешзн(характер)=агрессивный, мягкий
 разрешзн(курение)=да, нет
 вопрос(имя)=Какое имя у пациента?
 вопрос(вес)=Каков вес пациента?
 вопрос(сложение)=Телосложение:
 вопрос(жиры)=Потребление ненасыщенных жиров:
 вопрос(холестерин)=Потребление животных жиров: вопрос(соль)=Потребление соли:
 вопрос(потребление_алкоголя)=Потребление алкоголя: вопрос(пол)=Какого пола пациент?
 вопрос(возраст)=Сколько пациенту лет?
 вопрос(курение)=Пациент курит?
 вопрос(характер)=Характер пациента:
 Правила
 Правило 1: если относительный_вес=нормальный, то зн=да.
 Правило 2: если относительный_вес=недостаточный, то зн=да.
 Правило 3: если относительный_вес=излишний, то зн=да.
 Правило 4: если коронарный_риск=ниже_среднего, то сердзаб=низкий.
 Правило 5: если коронарный_риск=средний, то сердзаб=низкий.
 Правило 6: если коронарный_риск=выше_среднего, то сердзаб=высокий.
 Правило 7: если возраст=25_или_меньше и пол=ж,
 то основная_продолжительность=72.
 Правило 8: если возраст=25_или_меньше и пол=м,
 то основная_продолжительность=67.
 Правило 9: если возраст=25-55 и пол=ж,
 то основная_продолжительность=67.
 Правило 10: если возраст=25-55 и пол=м,
 то основная_продолжительность=62.
 Правило 11: если возраст=55_или_больше и пол=ж,
 то основная_продолжительность=64.
 Правило 12: если возраст=55_или_больше и пол=м, то основная_продолжительность=60.
 Правило 13: если вес=55_или_меньше и сложение=мелкое и пол=ж, то относительный_вес=нормальный.
 Правило 14: если вес=85_или_больше и сложение=мелкое и пол=ж, то относительный_вес=излишний.
 Правило 15: если вес=55-85 и сложение=мелкое и пол=ж, то относительный_вес=излишний.
 Правило 16: если вес=55_или_меньше и сложение=крупное и пол=ж, то относительный_вес=недостаточный.
 Правило 17: если вес=55_или_меньше и сложение=крупное и пол=м, то относительный_вес=недостаточный.
 Правило 18: если вес=55-85 и сложение=крупное и пол=ж, то относительный_вес=нормальный.
 Правило 19: если вес=55-85 и сложение=мелкое и пол=м, то относительный_вес=нормальный.
 Правило 20: если вес=55-85 и сложение=крупное и пол=м, то относительный_вес=недостаточный.

Правило 21: если вес=85_или_больше и пол=ж, то относительный_вес=излишний.

Правило 22: если вес=85_или_больше и пол=м и сложение=мелкое, то относительный_вес=излишний.

Правило 23: если вес=85_или_больше и пол=м и сложение=крупное, то относительный_вес=нормальный.

Правило 24: если вес=55_или_меньше и пол=м и сложение=мелкое, то относительный_вес=нормальный.

Правило 25: если холестерин=норма и жиры=излишек, то коронарный_риск=ниже_среднего.

Правило 26: если холестерин=норма и жиры=норма, то коронарный_риск=средний.

Правило 27: если холестерин=излишек, то коронарный_риск=выше_среднего.

Правило 28: если холестерин=норма, то коронарный_риск=средний.

Правило 29: если соль=излишек, то кровяное_давление=выше_среднего.

Правило 30: если соль=норма, то кровяное_давление=среднее.

Правило 31: если относительный_вес=излишний и коронарный_риск=выше_среднего и кровяное_давление=выше_среднего и курение=да, то перспектива=унылая.

Правило 32: если относительный_вес=излишний и коронарный_риск=выше_среднего и кровяное_давление=выше_среднего и курение=нет, то перспектива=плохая.

Правило 33: если относительный_вес=излишний и коронарный_риск=выше_среднего и кровяное_давление=среднее, то перспектива=унылая.

Правило 34: если относительный_вес=излишний и сердзаб=низкий и 392кровяное_давление=среднее и курение=нет, то перспектива=хорошая.

Правило 35: если зн=да и сердзаб=низкий и кровяное_давление=среднее и курение=нет, то перспектива=отличная.

Правило 36: если зн=да и коронарный_риск=выше_среднего и курение=да, то перспектива=посредственная.

Правило 37: если зн=да и сердзаб=низкий и кровяное_давление=среднее и курение=да, то перспектива=посредственная.

Правило 38: если зн=да и сердзаб=низкий и кровяное_давление=выше_среднего, то перспектива=посредственная.

Правило 39: если перспектива=унылая и характер=агрессивный и потребление_алкоголя=чрезмерно, то фактор=минус_12.

Правило 40: если перспектива=отличная и характер=мягкий и потребление_алкоголя=умеренное, то фактор=плюс_12.

Правило 41: если перспектива=отличная и характер=агрессивный и потребление_алкоголя=умеренное, то фактор=ноль.

Правило 42: если перспектива=хорошая и характер=агрессивный и потребление_алкоголя=умеренное, то фактор=ноль.

Правило 43: если перспектива=хорошая и характер=агрессивный, то фактор=минус_12.

Правило 44: если перспектива=посредственная и характер=агрессивный и потребление_алкоголя=чрезмерно, то фактор=минус_12.

Правило 45: если перспектива=посредственная, то фактор=ноль.

Правило 46: если перспектива=отличная и потребление_алкоголя=не_употребляет, то фактор=плюс_12.

Правило 47: если перспектива=отличная и характер=мягкий и потребление_алкоголя=чрезмерно, то фактор=плюс_12.

Правило 47: если перспектива=отличная и характер=агрессивный и потребление_алкоголя=чрезмерно, то фактор=ноль.

Правило 48: если перспектива=хорошая и характер=мягкий, то фактор=плюс_12.

Правило 49: если перспектива=унылая и характер=мягкий, то фактор=ноль.

Правило 50: если перспектива=унылая и характер=агрессивный, то фактор=минус_12.

Правило 51: если перспектива=плохая и потребление_алкоголя=чрезмерно, то фактор=минус_12.

Правило 52: если перспектива=плохая, то фактор=ноль

Правило 53: если основная_продолжительность=72 и фактор=ноль, то продолжительность=72_года.

Правило 54: если основная_продолжительность=67 и фактор=ноль, то продолжительность=67_лет.

Правило 55: если основная_продолжительность=64 и фактор=ноль, то продолжительность=64_года.

Правило 56: если основная_продолжительность=62 и фактор=ноль, то продолжительность=62_года.

Правило 57: если основная_продолжительность=60 и фактор=ноль, то продолжительность=60_лет.

Правило 58: если основная_продолжительность=72 и фактор=плюс_12, то продолжительность=84_года.

Правило 59: если основная_продолжительность=67 и фактор=плюс_12, то продолжительность=79_лет.

Правило 60: если основная_продолжительность=64 и фактор=плюс_12, то продолжительность=76_лет.

Правило 61: если основная_продолжительность=62 и фактор=плюс_12, то продолжительность=74_года.

Правило 62: если основная_продолжительность=60 и фактор=плюс_12, то продолжительность=72_года.

Правило 63: если основная_продолжительность=72 и фактор=минус_12, то продолжительность=60_лет.

Правило 64: если основная_продолжительность=67 и фактор=минус_12, то продолжительность=55_лет.

Правило 65: если основная_продолжительность=64 и фактор=минус_12, то продолжительность=52_года.

Правило 66: если основная_продолжительность=62 и фактор=минус_12, то продолжительность=50_лет.

Правило 67: если основная_продолжительность=60 и фактор=минус_12, то продолжительность=48_лет.

В этой базе знаний помимо правил фиксируются разрешенные значения для объектов при ответах на указанные вопросы. Эту базу знаний можно реализовать в оболочковой ЭС, в которую эту базу достаточно ввести в виде обычного текстового файла. Для каждой оболочковой ЭС системы определен свой синтаксис записи правил продукций.

4.7. Механизмы вывода в экспертных системах

Одним из важных вопросов, возникающих при проектировании управляющего компонента систем, основанных на знаниях, в частности, ЭС, является выбор метода поиска решения, то есть стратегии вывода. От выбранного метода поиска будет зависеть порядок применения и срабатывания правил. Процедура выбора сводится к определению направления поиска и способа его осуществления. Процедуры, реализующие поиск, обычно «защиты» в механизм вывода, поэтому в большинстве систем инженерии знаний не имеют к ним доступа и, следовательно, не могут в них ничего изменять по своему желанию. При разработке стратегии управления выводом необходимо ответить на два вопроса:

1. Какую точку в пространстве состояний принять в качестве исходной? Дело в том, что еще до начала поиска решения система, основанная на знаниях, должна каким-то образом выбрать исходную точку поиска. От выбора этой точки зависит и метод осуществления поиска в прямом или обратном направлении.

2. Как повысить эффективность поиска решения? Чтобы добиться повышения эффективности поиска решения, необходимо найти эвристики разрешения конфликтов, связанных с существованием нескольких возможных путей для продолжения поиска в пространстве состояний, поскольку требуется отбросить те из них, которые заведомо не ведут к искомому решению.

В системах с обратным выводом вначале выдвигается некоторая гипотеза, а затем механизм вывода в процессе работы как бы возвращается назад, переходя от нее к фактам, и пытается найти эту гипотезу. Если она оказалась правильной, то выбирается следующая гипотеза, детализирующая первую и являющаяся по отношению к ней подцелью. Далее отыскиваются факты, подтверждающие истинность подчиненной гипотезы. Вывод такого типа называется управляемым целями, или управляемым консеквентами. Обратный поиск применяется в тех случаях, когда цели известны и их сравнительно немного.

В системах с прямым выводом по известным фактам отыскивается заключение, которое из этих фактов следует. Если такое заключение удастся найти, то оно заносится в рабочую память. Прямой вывод часто называют выводом, управляемым данными или выводом, управляемым антецедентами.

В качестве примера запишем несколько фактов и правил, которые могли бы составить простую логическую систему:

Факты

r .

s . t . u . v . w .

Все эти факты представляют, безусловно, истинные утверждения и могут означать, что угодно, например, R = «Иван сын Петра», S = «Леонид внук Марии» и т.д.

Теперь запишем несколько правил:

u и $w \wedge x$.

и $z \wedge y$. $r \wedge z$.

Прямая цепочка рассуждений, как отмечалось выше, предполагает использование правил для логического вывода новых фактов, которые неявно (в импликациях) существовали и ранее, но могут быть сделаны явными посредством применения правил.

Вот какова типичная модель операции прямой цепочки рассуждений. Механизм вывода ЭС циклически просматривает все правила. Он исследует каждое из них по очереди, выясняя, является ли информация в левой части правила истинной. Если она истинна, механизм вывода добавляет факт из правой части правила к хранимым истинным фактам. Затем он переходит к следующему правилу, и процесс повторяется. Проверив все правила, механизм вывода начинает работу заново. Проследим этот процесс на нашем примере.

Сначала имеется только одно действующее правило:

$r \wedge z$.

Факт, который мы обозначили через r , находится среди хранимых истинных фактов, и так как r подразумевает z , механизм вывода добавляет z к хранимому набору истинных фактов. Затем механизм вывода ищет следующее правило, годное для выполнения. Факт u был дан как истинный в самом начале. Мы только что получили новый факт z . Имеется другое правило, которое подразумевает, что и u и $z \wedge y$. Механизм вывода находит и использует его. Таким образом, мы добавили новый факт y к хранимым истинным фактам и т.д. Это и есть процедура прямой цепочки рассуждений: механизм вывода работает для пополнения изначального запаса истинных фактов фактами, которые подразумеваются посредством набора правил. ЭС на базе прямой цепочки рассуждений чаще всего применяются в планировании или проектировании.

Предположим теперь, что нам необходимо установить истинность факта x . Мы можем сделать это, если будем знать, что y и w являются истинными фактами. Здесь w представляет собой один из изначально имеющих фактов, поэтому нужно установить истинность только факта y , который не является исходным фактом, для чего потребуются установить истинность фактов i и z . Факт i дан как истинный, а z нет. Мы сможем завершить весь процесс, если установим истинность факта z . Посмотрим на правила. Если факт g истинен, то и z , как следствие, будет истинным. Но g один из изначально данных фактов, и, значит, мы доказали, что x является истинным фактом. Цель достигнута.

Механизм вывода в Прологе основан на обратной цепочке рассуждений. Процесс выполнения программы на Прологе сводится к установлению истинности определенного предложения (и обычно в определении величин определенных переменных в процессе) посредством обратной цепочки рассуждений и продолжается до тех пор, пока не будут найдены некоторые базовые истинные факты, известные системе. ЭС, основанные на обратной цепочке рассуждений, как правило, применяются при диагностике или классификации. В некоторых системах вывод основывается на сочетании обратного и ограниченнопрямого выводов. Такой комбинированный вывод получил название циклического вывода.

В системах, база знаний которых насчитывает сотни правил, весьма желательным является использование какой-либо стратегии управления выводом, позволяющей минимизировать время поиска решения и тем самым повысить эффективность вывода. К числу таких стратегий относятся поиск в глубину, поиск в ширину, разбиение задачи на подзадачи, и так называемый альфа-бета алгоритм.

Продукционная система, или система, основанная на правилах, работает циклически. В каждом цикле правила из базы знаний просматриваются интерпретатором правил в определенном порядке, который устанавливается его управляющим компонентом. Если обнаруживается правило, посылка которого при сопоставлении совпала с некоторыми фактами из рабочей памяти, то правило срабатывает и его заключение добавляется в рабочую память. Затем цикл повторяется. Цикл имеет четыре фазы: сопоставление с образцом, выбор (разрешение конфликта), срабатывание и выполнение действия (изменение состояния рабочей памяти).

Любое правило представляет собой управляемый данными модуль, который является автономным, не зависимым от других правил элементом базы знаний. Это дает возможность легко наращивать и отлаживать программу, состоящую из продукций, поскольку при добавлении нового правила или редактировании имеющегося не требуется вносить изменения в остальные правила. Каждому правилу присваивается собственный номер, что позволяет весьма просто вызывать для просмотра цепочку сработавших правил. Система, основанная на правилах, содержит совокупность управляемых данными модулей, состоящих из двух частей посылки и заключения. Механизм вывода может быть назван системой вывода, использующей сопоставление по образцу. Продукционная система это система, основанная на правилах, механизм вывода которой включает в себя компоненты собственно вывода и управления им.

Выводы

На лекции введено понятие нечетких знаний и описано, каким образом они обрабатываются в инженерии знаний. Вряд ли можно достаточно строго упорядочить методы обработки нечетких знаний. Например, как указать пользователю, в каком направлении искать методы их обработки, когда возникла проблема с «нечеткостью». Адекватность того или другого метода можно оценить по конкретным примерам или благодаря эвристическим знаниям. Еще недостаточно проработаны методы обработки в случае, когда существует взаимосвязь или совокупность разновидностей нечеткости. Пока неизвестно, существуют ли унифицированные методы обработки нечетких знаний различного типа. Очевидно, в целях проектирования будущих интеллектуальных систем необходимо искать новые подходы к обработке таких знаний в дедуктивных выводах.

Также рассмотрены некоторые элементы модальной логики, которая позволяет по возможности полно и точно выражать, и учитывать динамику изменений изучаемой среды. Язык временной модальной логики основан на языке логики предикатов первого порядка и использует множество индивидуальных (объектных) и пропозициональных переменных и констант, функциональных и предикатных символов.

Экспертные системы доказали возможность совершенно нового подхода к развитию информатики. Он заключается в формализации нашего знания с помощью независимых декларативных понятий, в организации четко определенных структур таких, продукционные правила, фреймы, скрипты и семантические сети, с помощью которых можно получить изящное решение поставленных задач. Для таких систем характерны простота использования, эффективное представление знаний и отсутствие трудностей, присущих программированию на классических языках, что позволяет взаимодействовать с такими системами с помощью обычных слов естественного языка.

Вопросы и упражнения для самопроверки

4.1. Что такое модальность? Зачем вводятся модальные операторы во временной модальной логике? Каков смысл модальных операторов?

4.2. Из каких конструкций состоит язык временной модальной логики?

4.3. Чем временная модальная логика удобна для описания и проверки правильного выполнения программы?

4.4. К какому типу языков программирования относятся языки логического программирования?

4.5. Какие из перечисленных языков относятся к языкам логического программирования: а) Рефал; б) С++; в) Пролог; г) Паскаль; д) Visual Prolog; е) SQL.

4.6. Как называются объекты данных в языке Пролог?

4.7. Какая существует связь между исчислением предикатов первого порядка и системами искусственного интеллекта, основанными на знаниях?

4.8. Верно ли, что правило в языке Пролог это хорновская клауза? А набор фактов и цель это правило в Прологе ?

4.9. Доказать, что всякая функция к-значной логики единственным образом представляется в виде

4.10. Перечислить основные методы (модели) представления знаний.

4.11. Какие компоненты продукционной ЭС являются необязательными: а) механизм вывода; б) подсистема объяснения; в) база знаний; д) рабочая память?

4.12. Объяснить сущность механизмов вывода в ЭС.

4.13. В чем отличие в системах, основанных на знаниях, прямой цепочки рассуждений от обратной цепочки рассуждений?

4.14. Как представляются и используются в системах ИИ нечеткие знания?

4.15. Начертить «И-ИЛИ» дерево поиска по типу приведенного в пункте 3.2 на рисунке 3.1 для следующего набора правил:

Правило 1: если топливо = есть и

выделение_тепла = есть и кислород = есть то действие = пожар.

Правило 2: если вещество = жидкость и

вещество = воспламеняемое то вещество = топливо.

Правило 3: если вещество = твердое и

вещество = горючее

то вещество = топливо.

Правило 4: если вещество = газ и

вещество = горючее

то вещество = топливо.

Правило 5: если вещество = пламя и

вещество = открытое

то действие = выделение _тепла.

Правило 6: если вещество = электричество

то действие = выделение_тепла.

Правило 7: если действие = трение

то действие = выделение_тепла.

4.16. При разработке стратегии вывода в продукционной модели на каком правиле мы должны сделать свой выбор: на том, при котором у нас потерпеть неудачу, или на том, при котором вероятность неудачи наименьшая? Почему?

4.17. Сопоставить эффективность традиционных компьютерных программ, использующих для решения задач процедуры, логику, деревья решений, и систем, основанных на знаниях, которые ориентированы на решение этого же класса задач. Перечислить достоинства и недостатки систем, основанных на знаниях. Как изменится объем базы знаний и что с ней произойдет, если некоторые фрагменты знаний, используемые для решения задач, окажутся неоднозначными?

4.18. За счет чего применение инструментальных средств упрощает процесс построения систем, основанных на знаниях?

4.19. Какие доводы можно привести при выборе инструментальных средств для решения конкретной задачи? Какие критерии при этом можно использовать?

4.20. Как способ представления знаний в системе зависит от ее типа (система диагностики, планирования или интерпретации)? Может быть такая зависимость вообще отсутствует?

4.21. Как оценить мощность имеющихся знаний? Возможно ли это сделать в принципе?

4.22. Привести примеры метазнаний, которые могут быть использованы в ЭС, предназначенной для поиска неисправностей в автомобиле, автомате, компьютере.

4.23. Предположим, что мы решаем прикладную задачу поиска неисправностей в автомобиле. Какую стратегию следует применить эксперту для приобретения знаний, необходимых для решения этой задачи? Какие способы следует применить инженеру по знаниям для приобретения необходимых сведений и разработки экспертной системы?

4.24. В чем способ построения ЭС совпадает со способом разработки обычных программ? Чем они различаются?

4.25. Как можно развивать возможности ЭС? Определить пути развития ЭС, предназначенной для поиска неисправностей в автомобиле. Что в данной задаче поддается измерению, а что нет?

4.26. Привести примеры прикладных задач, для решения которых в ЭС достаточно заложить лишь поверхностные знания.

Придумать какую-нибудь задачу принятия решений и сформулировать соответствующие знания в форме правил «если..., то...». Можно рассмотреть, например, планирование отпуска, предсказание погоды, простой медицинский диагноз и лечение, устранение неисправностей в компьютере или другом техническом устройстве, спортивные состязания и вероятность выигрыша и т. д. База знаний должна содержать 10-15 взаимосвязанных правил, а решение должно быть получено с какой-то степенью достоверности (коэффициентом доверия в 100-балльной шкале).

ЛИТЕРАТУРА К ГЛАВЕ 2

1. Адаменко А.Н., Кучуков А.М. Логическое программирование и Visual Prolog. СПб.: БХВ-Петербург, 2003.

2. Братко И. Алгоритмы искусственного интеллекта на языке PROLOG. 3-е изд.: Пер. с англ. М.: Издательский дом «Вильямс», 2004.

3. Братко И. Программирование на языке Пролог для искусственного интеллекта: Пер. с англ. М.: Мир, 1990.

4. Гаврилова И. В., Масленникова О.Е. Основы искусственного интеллекта: учеб. пособие М.: ФЛИНТА, 2013.

5. Давыдова Н.А., Боровская Е.В. Основы искусственного интеллекта. М.:

БИНОМ. Лаборатория знаний, 2010.

6. Девятков В. В. Системы искусственного интеллекта: Учеб. пособие для вузов. М.: Изд-во МГТУ им. Н.Э. Баумана, 2001.

7. Джексон Питер. Введение в экспертные системы: Пер. с англ.: Уч. пос. М.: Издательский дом «Вильямс», 2001.

8. Джонс М.Т. Программирование искусственного интеллекта в приложениях. М.: ДМК Пресс, 2011.

9. Жданов А.А. Автономный искусственный интеллект. М.: БИНОМ. Лаборатория знаний, 2012.

10. Крисевич В.С., Кузьмич Л., Шиф А.М. и др. Экспертные системы для персональных компьютеров: методы, средства, реализации: Справ. пособие. Мн.: Выш. шк., 1990.

11. Лорьер Ж.-Л. Системы искусственного интеллекта. М.: Мир, 1991.

12. Макаллистер Дж. Искусственный интеллект и Пролог на микроЭВМ: Пер. с англ. М.: Машиностроение, 1990.

13. Нейлор К. Как построить свою экспертную систему: Пер.с англ. М.: Энергоатомиздат, 1991.

14. Сойер Б., Фостер Д. Л. Программирование экспертных систем на Паскале: Пер. с англ. М.: Финансы и статистика, 1990.

15. Стерлинг Л., Шапиро Э. Искусство программирования на языке Пролог: Пер. с англ. М.: Мир, 1990.

16. Таунсенд К., Фохт Д. Проектирование и программная реализация экспертных систем для персональных ЭВМ: Пер. с англ. М.: Финансы и статистика, 1990

17. Тей А., Грибомон П., Луи Ж. и др. Логический подход к искусственному интеллекту: от классической логики к логическому программированию. М.: Мир, 1990.

18. Уэно Х., Кояма Т. и др. Представление и использование знаний. М.: Мир, 1989.

19. Чень Ч., Ли Р. Математическая логика и автоматическое доказательство теорем. М.: Наука, 1983.

20. Шрайнер П. А. Основы программирования на языке Пролог: курс лекций: учеб. пособие для студентов, обучающихся по специальностям в обл. информ. технологий. М.: Интернет Ун-т. Информ. Технологий, 2005.

21. Янсон А. Турбо-Пролог в сжатом изложении: Пер. с нем. М.: Мир, 1991.

11. УТОЧНЕНИЕ ПОНЯТИЯ АЛГОРИТМА НА ОСНОВЕ ЧАСТИЧНО РЕКУРСИВНЫХ ФУНКЦИЙ

План занятия

1. Интуитивное понятие алгоритма. Общие свойства алгоритмов 2. Эффективно вычислимые и примитивно рекурсивные функции

3. Оператор минимизации и частично рекурсивные функции. Тезис Черча

1.1. Интуитивное понятие алгоритма. Общие свойства алгоритмов

В настоящее время понятие алгоритма активно используется в таких областях, как вычислительная математика, математическая кибернетика, проектирование ЭВМ и других автоматов, системное и программное программирование на ЭВМ. Каждая из этих областей вносит свой существенный вклад в изучение понятия алгоритма. Так, например, в математической кибернетике сформировался подход к понятию алгоритма как к сложной управляющей системе. Различные программы для ЭВМ также являются конкретным воплощением разнообразных алгоритмов. В связи с этим не существует единого подхода к интуитивному описанию алгоритма. Это понятие в различных книгах описывается поразному, тем более что возникло прикладное направление в теории алгоритмов, заставившее взглянуть на понятие алгоритма с точки зрения его оптимизации, синтеза, верификации. Вот как описывается понятие алгоритма в «Математической энциклопедии»:

Алгоритм это точное предписание исполнителю, которое задает вычислительный процесс (называемый в этом случае алгоритмическим), начинающийся с произвольного исходного данного (или из некоторой совокупности возможных для данного алгоритма исходных данных) и направленный на получение полностью определяемого этим исходным данным (данными) результата.

Понятие алгоритма тесно связано с понятием исполнителя и систем его команд.

Пример 1.1. Обычные правила сложения, вычитания, умножения и деления натуральных чисел. В этих алгоритмах возможными результатами служат натуральные числа, записанные в десятичной (или в другой позиционной) системе счисления, а возможными исходными данными упорядоченные пары таких чисел.

Замечание. Вообще говоря, не предполагается, что результат будет обязательно получен. Процесс применения алгоритма к конкретным возможным данным может и оборваться безрезультатно или не заканчиваться вовсе.

Пример 1.2. Алгоритм вычисления корней уравнений данного типа.

Понятие «вычислительный процесс» не следует трактовать в узком смысле слова, например, как только числовые вычисления. Над данными могут выполняться буквенные (символьные) вычисления, когда алгоритм оперирует с произвольными символами и их комбинациями (словами, строками, матрицами, векторами, графами и другими объектами).

В ходе работы алгоритма появляются промежуточные результаты, которые используются в дальнейшем. Таким образом, каждый алгоритм имеет дело с входными, промежуточными и выходными данными.

Поскольку мы собираемся уточнять понятие алгоритма, нужно уточнить и понятие данных, то есть указать, каким требованиям должны удовлетворять данные (объекты), чтобы алгоритмы могли с ними работать. Ясно, что эти объекты должны быть четко определены и должны отличаться как друг от друга, так и от других классов объектов. Во многих важных случаях понятно, что это значит: к таким алгоритмическим объектам относятся, как уже отмечалось, числа, векторы, матрицы, формулы, выражения и т.д. Графические изображения представляются уже менее естественными в качестве алгоритмических объектов. Наконец, с такими объектами, как «хорошая погода», «осмысленное утверждение», «разумный поступок», легко оперирует любой человек (но каждый посвоему!), алгоритм работать откажется, пока эти объекты не будут описаны как данные с помощью других, более подходящих объектов. В этом и состоит смысл

концепции типов данных в языках программирования типа Турбо Паскаль.

В общем случае в теории алгоритмов фиксируют конкретные конечные наборы исходных объектов (называемых элементарными) и конечный набор средств построения других объектов из элементарных объектов. Набор элементарных объектов образует конечный алфавит исходных символов (цифр, букв и т.д.), из которых строятся другие объекты.

Типичным средством построения являются индуктивные определения, указывающие, как строить новые объекты из уже построенных объектов. Такими определениями являются, например, слова конечной длины над данным алфавитом, формулы, термы и т.д.

Другим важным обстоятельством при задании алгоритма является объем необходимой памяти для размещения данных. Память обычно считается однородной и дискретной, то есть состоит из одинаковых ячеек, причем каждая ячейка может содержать один символ алфавита данных. Тем самым становятся согласованными единицы измерения объема данных и объема памяти, при этом память может быть бесконечной.

При изучении алгоритмов особую роль играют следующие критерии качества алгоритма.

10. Связанность количество промежуточных результатов, которые должны одновременно храниться в памяти. Алгоритм тем лучше, чем меньше его связанность

20. Объем алгоритма количество операций (шагов), которые необходимо выполнить для решения задачи.

30. Длительность выполнения алгоритма количество шагов алгоритма и сложность этих шагов.

40. Логическая сложность алгоритма количество ветвлений и циклов.

Из курса информатики известны способы задания алгоритмов, поэтому мы ограничимся лишь их перечислением.

Словесная запись алгоритма на естественном языке.

Запись алгоритма на формальном языке (математическом, алгоритмическом языке высокого уровня и т.д.).

Табличный способ.

Графическое описание, то есть представление алгоритма в виде блок-схемы.

Наиболее существенными являются следующие свойства алгоритмов:

10. Дискретность. Алгоритм состоит из отдельных элементарных шагов или действий, причем множество различных шагов, из которых составлен алгоритм, конечно.

Типичный пример множества элементарных действий система команд микропроцессора ПЭВМ.

20. Детерминированность. После каждого шага алгоритма либо указывается, какой шаг делать дальше, либо дается команда остановки, после чего работа алгоритма считается законченной. При этом система величин, получаемых в какой-то промежуточный (не начальный) момент выполнения алгоритма, однозначно определяется системой величин, полученных в предшествующие моменты времени.

30. Определенность (однозначность). Последовательность действий не допускает произвольного ее толкования.

40. Массовость. Это свойство означает пригодность данного алгоритма для решения многих или даже всех задач из заданного класса при различных значениях исходных величин из множества допустимых значений для данного алгоритма.

50. Результативность. Остановка работы алгоритма после конечного числа шагов (зависящего от данных) с указанием того, что считать результатом.

Будем предполагать, что описание алгоритма (инструкция или программа для ЭВМ) и механизм его реализации (например, ЧПУ, робот, автомат) конечны. Требования конечности процесса реализации совпадает со свойством результативности.

В дополнение к свойству 20 отметим использование общепринятого соглашения:

если некоторый шаг алгоритма не содержит указаний о дальнейшем переходе, то выполняется шаг, следующий за ним в описании.

Коснемся вопроса об уточнении интуитивного понятия алгоритма на конкретном примере. Пусть дана последовательность P (одномерный массив), состоящая из n положительных вещественных чисел. Требуется произвести сортировку последовательности P по неубыванию.

Для решения этой задачи можно привести следующий простой алгоритм. Обозначим через R неотсортированную часть данной последовательности чисел. Тогда шаги алгоритма можно записать так.

Шаг 1. Найти в последовательности P наименьшее число.

Шаг 2. Найденное число приписать справа к неотсортированной части R (в начальный момент R пустая последовательность) и вычеркиваем это число из последовательности P .

Шаг 3. Если в последовательности P нет чисел, перейти к шагу 4. В противном случае перейти к шагу 1.

Шаг 4. Конец. Результатом следует считать последовательность R , построенную к данному моменту.

Казалось бы, все четко и ясно в этом алгоритме, например, очевидна результативность алгоритма, так как после выполнения ровно n шагов 1 и 2 из последовательности P будут вычеркнуты все числа и алгоритм остановится, а множество R будет результатом. Не вызывает сомнения и выполнение свойства детерминированности. Однако возникает вопрос: что означает фраза «дана последовательность действительных чисел»? Например, является ли таковой последовательность $33, 9^{2 \cdot (12)}$? Очевидно, да, однако, в нашем описании ничего не сказано, как найти наименьшее число среди таких чисел. В нем вообще не говорится о том, как искать наименьшие числа. Видимо, предполагается, что речь идет о числах, представленных в виде десятичных дробей. Опять-таки, ничего не говорится о точности представления данных чисел (то есть о количестве разрядов), так как от этого будет зависеть объем необходимой памяти для хранения массивов чисел. И снова возникает вопрос: в какой форме следует представлять числа с фиксированной или плавающей точкой, чтобы их сравнивать? Есть и другие неясные места в приведенном алгоритме. И, прежде всего, что требует уточнения это форма представления данных.

Таким образом, даже в этом простом примере несложный анализ показывает, что описанию, которое выглядит вполне ясным, еще далеко до алгоритма. Мы столкнулись здесь с необходимостью уточнить многие характеристики алгоритма: алфавит данных, форму их представления, размещение в памяти элементов последовательностей P и R в случае исполнения алгоритма на ЭВМ, элементарные шаги (поскольку шаг 1 явно не элементарен). Непонятен также механизм реализации алгоритма: одно дело человек, другое ЭВМ. Поэтому в дальнейшем мы будем строить такие модели алгоритмов, которые будут считаться формализацией интуитивного понятия алгоритма, попутно уточняя те или иные его свойства.

В разное время были предложены следующие определения понятия алгоритма:

1. Рекурсивность (частично рекурсивные функции, Черч, 1936 г.)
2. Общерекурсивные функции (исчисление равенств Геделя Хербранда Клини, 1936 г.; Клини, 1952 г.; Мендельсон, 1964 г.).
3. λ -исчисления (Черч, 1936 г.).
4. Машины Тьюринга (Тьюринг, 1936 г.).
5. Система подстановок Поста (Пост, 1943 г.).
6. Алгоритмы Маркова (Марков, 1951 г.).
7. Регистровые машины (Шепердсон Штургис, 1963 г.).

Все эти формализации понятия алгоритма эквивалентны между собой и ведут к понятию вычислимой функции.

1.2. Эффективно вычислимые и примитивно рекурсивные функции

Для алгоритмических проблем типичным является то обстоятельство, что требуется найти алгоритм вычисления значений числовой функции $f(x_1, x_2, \dots, x_n)$ от конечного набора целочисленных аргументов $x_1, x_2, \dots, x_n$. При этом результат должен быть также целым числом. Спрашивается: для каких функций алгоритм вычисления существует? Если такой алгоритм существует, то функция $f(x_1, x_2, \dots, x_n)$ называется эффективно вычислимой. Так как в определении эффективно вычислимой функции алгоритм понимается в интуитивном смысле, то и понятие эффективно вычислимой функции является интуитивным. Однако переход от алгоритма к эффективно вычислимой функции дает определенные преимущества, поскольку рассмотренные нами свойства алгоритмов выполняются и для совокупности всех вычислимых функций. Но тогда возникает проблема описания класса всех эффективно вычислимых функций?

Исчерпывающей формализацией понятия вычислимой функции оказалось понятие частично рекурсивной функции. Оно и является одним из математических уточнений интуитивного понятия вычислимой функции и, следовательно, уточнением интуитивного понятия алгоритма, если мы алгоритмом назовем всякую частично рекурсивную функцию.

В теории рекурсивных функций, как и вообще в теории алгоритмов, принят конструктивный, финитный подход, основной чертой которого является то, что все множество исследуемых функций строится из конечного числа исходных функций базиса с помощью простых операций, эффективная вычислимость которых достаточно очевидна. Эти простые операции обычно называют простейшими функциями или простейшими операторами. Они были введены американским математиком Черчем в 1936 г. К ним относятся следующие операторы:

1. $X(x) = x + 1$ (или $X(x) = x^2$) оператор сдвига, или оператор следования для натурального ряда, начиная с нуля. Этот оператор позволяет построить весь натуральный ряд $0, 0' = 0+1=1, 1' = 1+1=2, \dots$, и т. д.;
2. $O(x) = 0$ оператор аннулирования;
3. $I_n(x_1, x_2, \dots, x_n) = x_m, 1 < m < n$ оператор проектирования. Сефункцией

мейство $\{I_m^n\}$ называется также семейством функций *тождества*, или введения фиктивных переменных.

Очевидно, что операторы 1)-3) всюду определены и эффективно вычислимы. На их основе мы введем новые, необходимые для дальнейшего, операторы оператор суперпозиции функций, оператор примитивной рекурсии и оператор минимизации.

Оператор суперпозиции (подстановки) функций

Определение 1.1. Оператором суперпозиции функций S_m^n называется подстановка в функцию от m переменных m функций от одних и тех же n переменных.

Эта подстановка дает новую функцию (оператор) от n переменных. Так, если задана система m функций $f_1(x_1, x_2, \dots, x_n), f_2(x_1, x_2, \dots, x_n), \dots, f_m(x_1, x_2, \dots, x_n)$ от n переменных $x_1, x_2, \dots, x_n$, и функция $h(y_1, y_2, \dots, y_m)$, то функция $\phi(x_1, x_2, \dots, x_n) = h(f_1(x_1, x_2, \dots, x_n), f_2(x_1, x_2, \dots, x_n), \dots, f_m(x_1, x_2, \dots, x_n))$ является суперпозицией, полученной из функций $h, f_1, f_2, \dots, f_m$.

Если мы каким-либо образом умеем вычислять значения функций $f_1, f_2, \dots, f_m$ и h , то функция ϕ может быть вычислена так: дадим переменным $x_1, x_2, \dots, x_n$ некоторые значения $a_1, a_2, \dots, a_n$. Вычисляя все значения $f_i(a_1, a_2, \dots, a_n)$, найдем $b_i = f_i(a_1, a_2, \dots, a_n)$. Вычисляя теперь

$h(b_1, b_2, \dots, b_m)$, найдем $c = \phi(a_1, a_2, \dots, a_n)$.

Ясно, что если все функции $f_1, f_2, \dots, f_m$ и h всюду определены, то функция ϕ также будет всюду определена. Функция ϕ не будет всюду определенной, если хотя бы одна из

функций $f_1, f_2, \dots, f_m$ не всюду определена, или если можно найти такие значения аргументов $a_1, a_2, \dots, a_n$, что $b_i = f_i(a_1, a_2, \dots, a_n)$, $i = 1, m$, но значение функции $h(b_1, b_2, \dots, b_m)$ неопределено. Таким образом, если функции $f_1, f_2, \dots, f_m$ и h эффективно (интуитивно) вычислимы, то будет эффективно (интуитивно) вычислима и функция f .

Отметим, что возможны случаи, когда не все функции $f_1, f_2, \dots, f_m$ зависят от всех n аргументов $x_1, x_2, \dots, x_n$. В этих случаях для получения суперпозиции используются фиктивные аргументы и операторы проектирования $I^n(x_1, x_2, \dots, x_n)$. Например, функция

$$\phi(x, y, z) = h(f_1(x), f_2(x, y, z), y, x)$$
 получается суперпозицией функций $h(x_1, x_2, x_3, x_4)$ и $F_1(x, y, z) = f_1(x)$, $F_2(x, y, z) = f_2(x, y, z)$, $F_3(x, y, z) = I^3(x, y, z)$, $F_4(x, y, z) = I^3(x, y, z)$.

Используя подстановку и оператор проектирования, можно переставлять и отождествлять аргументы в данной функции, например: $f(x_2, x_1, x_3, \dots, x_n) = f(I^2_2(x_1, x_2), I^2_1(x_1, x_2), x_3, \dots, x_n)$;

$$f(x_1, x_2, x_3, \dots, x_n) = f(I^2_1(x_1, x_2), I^2_2(x_1, x_2), x_3, \dots, x_n).$$

Таким образом, если заданы функции f и операторы S^n , то I^n можно считать заданными всевозможные операторы подстановки функций в функции, а также переименования, перестановки и отождествления переменных.

Оператор примитивной рекурсии и примитивно рекурсивные функции

Определение 1.2. Оператор примитивной рекурсии R_n определяет $(n+1)$ -местную функцию f через n -местную функцию g и $(n+2)$ -местную функцию h следующим образом:

$$\begin{aligned} f(x_1, x_2, \dots, x_n, 0) &= g(x_1, x_2, \dots, x_n); \\ f(x_1, x_2, \dots, x_n, y+1) &= h(x_1, x_2, \dots, x_n, y, f(x_1, x_2, \dots, x_n, y)). \end{aligned} \quad (1.1)$$

Пара уравнений (1.1) называется схемой примитивной рекурсии. Тот факт, что функция f определена схемой (1.1), выражается равенством $f(x_1, x_2, \dots, x_n, y) = R_n(g, h)$.

В случае, когда $n = 0$, то есть когда функция f является одноместной, схема (1.1) принимает более простой вид:

$$f^f(y+1) = h(y, f(y)), \quad (1.2)$$

где c константа. Схемы (1.1) и (1.2) определяют функцию f рекурсивно не только через другие функции g и h , но и через значения функции f в предшествующих точках: значение функции f в точке $y+1$ зависит от значения этой функции в точке y . Для вычисления значения $f(x_1, x_2, \dots, x_n, k)$ понадобится $k+1$ вычисление по схеме (1.2) для всех $y = 0, k$.

Существенным в операторе примитивной рекурсии является тот факт, что, независимо от числа переменных в функции f , рекурсия ведется только по одной переменной y . Остальные n переменных $x_1, x_2, \dots, x_n$ на момент применения схемы (1.1) или схемы (1.2) зафиксированы и играют роль параметров.

Определение 1.3. Функция называется примитивно рекурсивной, если она может быть получена из операторов сдвига, операторов аннулирования и операторов проектирования (то есть из константы O , функций x' и функций I^n_m) с помощью применения конечного числа операторов суперпозиции функций и примитивной рекурсии.

Иначе говоря, функция $f(x_1, x_2, \dots, x_n)$ называется примитивно рекурсивной, если существует такая конечная последовательность функций $f_1, f_2, \dots, f_k$, что $f_k = f$ и для $\forall i$ ($0 < i < k$), функция f_i либо

простейшая, либо может быть получена из некоторых функций, предшествовавших ей в последовательности $f_1, f_2, \dots, f_k$ с помощью применения операторов суперпозиции и примитивной рекурсии.

Этому определению можно придать более формальный индуктивный вид.

1. Функции O , x' и I_m^n для всех натуральных n и m ($m < n$), являются примитивно рекурсивными.

2. Если $g_1(x_1, x_2, \dots, x_n)$, $g_2(x_1, x_2, \dots, x_n)$, $\dots$, $g_m(x_1, x_2, \dots, x_n)$, $h(y_1, y_2, \dots, y_t)$ — примитивно рекурсивные функции, то $S^n(h, g_1, g_2, \dots, g_m)$ примитивно рекурсивные функции для всех натуральных чисел m и n .

3. Если $g(x_1, x_2, \dots, x_n)$ и $h(x_1, x_2, \dots, x_n, y, z)$ примитивно рекурсивные функции, то $R_n(g, h)$ примитивно рекурсивная функция.

4. Других примитивно рекурсивных функций нет.

Из такого индуктивного описания нетрудно построить процедуру, порождающую все примитивно рекурсивные функции.

Примерами примитивно рекурсивных функций могут служить сложение и умножение натуральных чисел; возведение натурального числа, большего 1, в натуральную степень; нахождение остатка от деления одного целого числа на другое; все логические функции и др. Покажем, например, что сложение целых неотрицательных чисел рекурсивно, то есть рассмотрим функцию $f_+(x, y) = x + y$. Действительно,

$$f_+(V, 0) = x = I^1(x) \\ f_+(x, y+1) = f_+(x, y) + 1 \quad (\text{т.е. } g = x)$$

$$f_+(x, y+1) = f_+(x, y) + 1 = f_+(x, y) + 1$$

Таким образом, $f_+(x, y) = R_1(I^1(x), h(x, y, z))$, где $h(x, y, z) = z' = z + 1$.

Так, вычисляя значение функции $f_+(x, y)$ при $x = 5$, $y = 2$, получим: $f_+(5, 0) = 5$ (первое равенство); из второго равенства последовательно получим $f_+(5, 1) = f_+(5, 0) + 1 = 5 + 1 = 6$; $f_+(5, 2) = f_+(5, 1) + 1 = 6 + 1 = 7$.

Предлагаем самостоятельно доказать, что умножение натуральных чисел примитивно рекурсивно, если его задать так:

$$f \cdot (x, 0) = 0, \\ f \cdot (x, y+1) = f \cdot (x, y) + x.$$

Исходя из этой схемы примитивной рекурсии, вычислить 7·5.

Докажем теперь, что введение фиктивных переменных, а также перестановка и отождествление переменных не выводит за пределы класса примитивно рекурсивных функций.

Теорема 1.1. Пусть $g(y_1, y_2, \dots, y_k)$ примитивно рекурсивная функция и пусть $x_1, x_2, \dots, x_n$ различные переменные. Тогда, если при каждом i , $1 < i < k$, z_i есть одна из переменных $x_1, x_2, \dots, x_n$, то функция $f(x_1, x_2, \dots, x_n) = g(z_1, z_2, \dots, z_k)$ тоже примитивно рекурсивна.

Доказательство. Пусть $z_i = x_{j_i}$, где $1 < j_i < n$. Тогда

$$f(x_1, x_2, \dots, x_n) = g(I^{j_1}(x_1, x_2, \dots, x_n), \dots, I^{j_k}(x_1, x_2, \dots, x_n)).$$

Таким образом, функция f может быть получена из функций $g, I^{j_1}, \dots, I^{j_k}$,

примитивно рекурсивная функция.

Пример 1.3. Применить схему примитивной рекурсии к функциям $g(x) = x$ и $h(x, y, z) = z^x$ по переменным y и z . Функцию $f(x, y) = R_2(g, h)$ записать в аналитической форме.

Решение. Запишем схему примитивной рекурсии: $f(x, 0) = g(x) = x$;

$$f(x, y+1) = h(x, y, f(x, y)).$$

Тогда, поскольку $h(x, y, z) = z^x$, получим

$I^{j_1}, \dots, I^{j_k}$, с помощью подстановки (суперпозиции), то есть f есть

$$f(x, y+1) = [f(x, y)]x = [h(x, y-1, f(x, y-1))]x = [f(x, y-1)]x^x = \dots = [f(x, 0)]x^W = x^{y+1}.$$

Пример 1.4. Доказать, что если функция $f(x_1, x_2, \dots, x_n)$ примитивно рекурсивна, то следующие функции также примитивно рекурсивны:

а) $g_1(x_1, x_2, \dots, x_n) = f(x_2, x_3, \dots, x_n, x_1)$ (циклическая перестановка аргументов);

б) $g_2(x_1, x_2, \dots, x_n, x_{n+1}) = f(x_1, x_2, \dots, x_n)$ (введение фиктивного аргумента).

Решение. Функции g_1 и g_2 получаются из функций $f(x_1, x_2, \dots, x_n)$ и

$I^n(x, x, \dots, x)$ ($m < n$) суперпозициями: $m_1 \quad 2n$

а) $g(x, x, \dots, x) = f(I^n(x, x, \dots, x), I^n(x, x, \dots, x), \dots, I^n(x, x, \dots, x), I^n(x, x, \dots, x))$.

б) $g(x, x, \dots, x, x) = f(I^{n+1}(x, x, \dots, x, x), I^{n+1}(x, x, \dots, x, x), \dots, I^{n+1}(x, x, \dots, x, x), I^{n+1}(x, x, \dots, x, x))$.

$I^{n+1}(x, x, \dots, x, x) = f(x, x, \dots, x, x) = f(x, x, \dots, x)$.

По определению функции O , $X(x) = x'$ и S_m^n для всех n и m ($m < n$), являются примитивно рекурсивными функциями; применение операторов S_m^n и R_n к примитивно рекурсивной функции также приводит к примитивно рекурсивной функции. Других примитивно рекурсивных функций нет.

Таким образом, класс примитивно рекурсивных функций замкнут относительно введения фиктивных переменных, перестановки и отождествления переменных, применения конечного числа операторов O , $X(x) = x'$ и S_m^n , операторов суперпозиции функций и операторов R_n примитивной рекурсии.

Приведем еще ряд примеров примитивно рекурсивных функций, играющих существенную роль при построении теории целых неотрицательных чисел.

Пример 1.5. Следующие функции, заданные на множестве целых неотрицательных чисел, являются примитивно рекурсивными:

$$3. f(x, y) = x - y = J^x - y, \text{ если } x > y,$$

$$1. f(x, y) = x^y \quad (x > 1);$$

$$\text{зг/л } \Gamma, x-1, \text{ если } x > 0, \quad 2. 5(x) = \begin{cases} 0, & \text{если } x = 0; \\ \lfloor 0, & \text{если } x < y; \end{cases}$$

$$\lfloor 0, \text{ если } x < y;$$

$$4. f(x, y) = x - y;$$

$$5. f(X, y) = x + y;$$

$$6. f(x, y) = x \cdot y;$$

$$7. sg(x) = \begin{cases} 0, & \text{если } x=0, \\ 1, & \text{если } x \neq 0; \end{cases}$$

$$1. 1, \text{ если } x \neq 0;$$

$$8. f(x) = x!;$$

$$9. f(x_1, x_2, \dots, x_n) = \max(x_1, x_2, \dots, x_n);$$

$$10. y = r \bmod x \text{ остаток от деления числа } y \text{ на число } x > 0;$$

$$11. q = x \div y \text{ неполное частное от деления числа } x \text{ на число } y > 0.$$

В качестве примеров докажем примитивную рекурсивность функций 1, 3, 4 и 10.

1. В схеме примитивной рекурсии (1.1) положим

$$g(x) = k(O(x)) = k(0) = 1, h(x, y, z) = z \cdot x.$$

Тогда $f(x, 0) = x \cdot 0 = gx = 1; f(x, y+1) = h(x, y, f(x, y)) = f(x, y) \cdot x = (xy) \cdot x = xy^{+1}.$

$$3. \text{ Полагаем } f(x, 0) = x - 0 = g(x) = 1l(x) = x.$$

$$f(x, y+1) = x - (y+1) = h(x, y, f(x, y)) = (x - y) - 1 = f(x, y) - 1.$$

$$\text{Например, } f(9, 6) = f(9, 5+1) = h(9, 5, f(9, 5)) = (9 - 5) - 1 = (4) - 1 = 3;$$

$$f(9, 11) = f(9, 10+1) = h(9, 10, f(9, 10)) = (9 - 10) - 1 = (0) - 1 = 0.$$

Таким образом, $x \rightarrow 0 = x = I^1_2(x, y)$; $x \rightarrow (y+1) = (x \rightarrow y) \rightarrow 1$.

4. Так как $|x \rightarrow y| = (x \rightarrow y)$, если $x > y$, то
положим $|x \rightarrow y| =$
1 $[y \rightarrow x, \text{ если } x < y,$ 1 1

$(x \rightarrow y) + (y \rightarrow x)$.

В данном случае мы имеем суперпозицию функций: $f_1(x, y) = x \rightarrow y$, $f_2(x, y) = y \rightarrow x$.

Тогда

$$f(x, y) = S^2(f(x, y), f(x, y)) = f(x, y) + f(x, y). \quad 2 \quad 1 \quad 2 \quad 1 \quad 2$$

10. Рассмотрим функцию $f(x, y) = r \bmod x$, значением которой является остаток r от деления числа y на число $x > 0$. По определению остатка от деления числа y на x должны выполняться условия: $y = x \cdot q + r$, где $0 < r < x$. Очевидно, что $f(x, 0) = 0$. Тогда $y+1 = (x \cdot q + r) + 1 = x \cdot q + (r+1)$. Здесь возможны два случая: $x = r+1$ и $x / r+1$. В первом случае $y+1 = x \cdot q + x = x(q+1)$, т.е. остаток от деления числа $y+1$ на число x равен 0. Во втором случае $y+1 = x \cdot q + (r+1) = (x \cdot q + r) + 1 = (f(x, y))'$.

В обоих случаях по схеме (1.1) примитивной рекурсии и определения 3 данного примера получим

$$f(x, y+1) = (f(x, y))' \text{ sg}(x \rightarrow f(x, y))'.$$

Например, пусть $y = 13$, $x = 5$. Тогда $f(x, y+1) = 4$, а $f(x, 15) = 0$.

Рекурсивность остальных функций из примера 1.5 предлагаем установить самостоятельно.

1.3. Оператор минимизации и частично рекурсивные функции. Тезис Черча

Пусть задана некоторая 2-местная функция $f(x, y)$. Зафиксируем значение x и выясним, при каком значении переменной y значение функции $f(x, y) = 0$. Эту задачу усложним: найти для функции $f(x, y)$ при фиксированном значении x наименьшее из тех значений y , при которых $f(x, y) = 0$. Так как результат решения задачи зависит от пе-

ременной x , то наименьшее значение переменной y , при котором $f(x, y) = 0$, является функцией от переменной x .

Для такой задачи принято следующее обозначение:

$$\mu(x) = \mu[y | f(x, y) = 0] \text{ (читается:}$$

наименьшее значение y такое, что

$f(x, y) = 0$). Аналогично определяется функция μ для нескольких

$$\Phi \text{ переменных: } \mu(x_1, x_2, \dots, x_n) = \mu[y | f(x_1, x_2, \dots, x_n, y) = 0]$$

Переход от функции $f(x_1, x_2, \dots, x_n, y)$ к функции $\mu(x_1, x_2, \dots, x_n)$ называется применением /л оператора минимизации.

Для вычисления значения функции Φ можно предложить следующий алгоритм.

1. Вычислим $f(x_1, x_2, \dots, x_n, 0)$. Если это значение равно нулю, то полагаем $\mu(x_1, x_2, \dots, x_n) = 0$. Если же $f(x_1, x_2, \dots, x_n, 0) \neq 0$, то переходим к следующему шагу.

2. Вычислим $f(x_1, x_2, \dots, x_n, 1)$. Если $f(x_1, x_2, \dots, x_n, 1) = 0$, то полагаем $\mu(x_1, x_2, \dots, x_n) = 1$. Если же $f(x_1, x_2, \dots, x_n, 1) \neq 0$, то полагаем $y = 2$ и т. д.

Если окажется, что $\forall y \in \mathbb{N}$ значение функции $f(x_1, x_2, \dots, x_n, y) \neq 0$, то функция $\mu(x_1, x_2, \dots, x_n)$ в этом случае считается неопределенной. Но, возможно, что существует такое значение y_0 , что $f(x_1, x_2, \dots, x_n, y_0) = 0$ и, значит, есть и наименьшее значение y , при котором $f(x_1, x_2, \dots, x_n, y) = 0$. В то же время может случиться, что при некотором значении переменной z ($0 < z < y_0$) значение функции $f(x_1, x_2, \dots, x_n, z)$ не определено. Очевидно, что в этом случае процесс вычисления наименьшего значения y , при котором $f(x_1, x_2, \dots, x_n, y) = 0$ не дойдет до y_0 . И в этом случае функция $\mu(x_1, x_2, \dots, x_n)$ считается неопределенной.

Иногда μ -оператор определяют с ограничением, полагая $\mu(x_1, x_2, \dots, x_n) = y$ $y | f(x_1, x_2, \dots, x_n, y) = 0$

для $y < z$, которое дает гарантию окончания вычислений. Это ограничение оценивает сверху количество шагов вычислений, и это является существенной особенностью примитивно рекурсивных функций.

Заметим, что у-оператор (как ограниченный, так и неограниченный) является удобным средством для построения обратных функций.

Действительно, функция $\phi(x) = y \text{ у}[f(x, y) = x]$ является обратной к функции fx), поэтому в применении к одноместным функциям у-оператор иногда называют оператором обращения.

Пример 1.6. Рассмотрим функцию $fx, y) = x \text{ у}$, которая может быть получена с помощью оператора минимизации

$$f(x, y) = y \text{ з}(y + z = x) = y \text{ з}[I^3_3(x, y, z) + I^3_1(x, y, z) = I^3(x, y, z)] .$$

Вычислим, например, $f(7, 2)$. Для этого положим $y = 2$ и будем придавать z последовательно значения 0, 1, 2 и т.д.: $z = 0, 2 + 0 = 2 / 7$; $z = 1, 2 + 1 = 3 \# 7$; $z = 2, 2 + 2 = 4 \# 7$; $z = 3, 2 + 3 = 5 \# 7$; $z = 4, 2 + 4 = 6 / 7$; $z = 5, 2 + 5 = 7$. Таким образом, $f(7, 2) = 5$.

Оператору минимизации можно придать привычную функциональную форму $f(x_1, x_2, \dots, x_n) = p \text{ у}[(g(x_1, x_2, \dots, x_n, y) = 0)]$. Будучи применен к эффективно вычислимой функции, ц-оператор снова дает эффективно вычислимую функцию (то есть сохраняет вычислимость).

Действительно, для вычисления значения функции $f(x_1, x_2, \dots, x_n) = y \text{ у}[g(x_1, x_2, \dots, x_n, y) = 0]$

на наборе $(x_1, x_2, \dots, x_n)$ существует следующая простая процедура: вычисляем g на наборах $(x_1, x_2, \dots, x_n, 0), (x_1, x_2, \dots, x_n, 1), \dots$, до тех пор,

пока не получим значение 0. Однако в отличие от рассмотренных ранее процедур она может не привести к результату. Это может произойти в том случае, когда на данном наборе $(x_1, x_2, \dots, x_n)$ уравнение $g(x_1, x_2, \dots, x_n, y) = 0$ не имеет решения. В таком случае функция

$f(x_1, x_2, \dots, x_n)$ считается неопределенной. Например, функция $x^{-1} = y \text{ у}[y + 1 = x]$, обратная к функции $y = x + 1$, не определена при $x = 0$, и в случае неопределенности процесс вычисления не останавливается (проблема остановки).

Таким образом, среди примитивно рекурсивных функций появляются не полностью определенные, то есть частичные функции. При этом операторы над частичными функциями порождают новые частичные функции.

Характер неопределенности может оказаться довольно сложным, а именно: для данного набора значений $(x_1, x_2, \dots, x_n)$ не найдется способа установить, определена ли функция f на этом наборе, и нам придется продолжить процесс вычисления неопределенное время, не зная, остановится ли он, или нет.

В случае, когда функция g , к которой применяется p оператор, сама является частичной, функция $f(x_1, x_2, \dots, x_n) = p \text{ у}[g(x_1, x_2, \dots, x_n, y) = 0]$

вычисляется с учетом следующего условия: если для $y = 0, 1, \dots, i-1$ $g(x_1, x_2, \dots, x_n, y) \# 0$, а $g(x_1, x_2, \dots, x_n, i)$ не определена, то и $f(x_1, x_2, \dots, x_n)$

не определена. Например, функция $p \text{ у}[y \text{ х} = 5]$ при $x=1$ не определена (хотя уравнение $y - 1 = 5$ имеет решение $x = 6$), так как функция $y - 1 = 5$ не определена при $y = 0$.

Использование неограниченного p -оператора, приводит к понятию частично рекурсивной функции.

Определение 1.4. Частичная (то есть не всюду определенная) функция f называется частично рекурсивной (или просто рекурсивной), если она может быть получена из простейших функций O , x' и S_m^n с помощью применения конечного числа операторов суперпозиции, примитивной рекурсии и минимизации.

Все рассмотренные в примере 1.5 примитивно рекурсивные функции, являются примерами и частично рекурсивных функций, так как там не использовался оператор минимизации.

Определение 1.5. Частично рекурсивная функция называется общерекурсивной, если она определена для всех значений аргументов.

Понятие общерекурсивной функции может быть определено и независимо от

понятия частично рекурсивной функции следующим образом.

Класс всех общерекурсивных функций это наименьший класс функций, содержащий все примитивно рекурсивные функции и замкнутый относительно композиции функций и оператора минимизации при условии, что последний применяется ко всюду определенной функции $g(x_1, x_2, \dots, x_n, y)$. Однако изучение общерекурсивных функций обычно ведется в классе всех частично рекурсивных функций.

Из сказанного выше о неопределенности видно, что не всегда частично рекурсивную функцию можно эффективно образом доопределить до общерекурсивной функции. Более точно, существует такая частично рекурсивная функция f , что никакая общерекурсивная функция g не является ее доопределением. Следовательно, существуют частичные алгоритмы, которые нельзя доопределить до всюду определенных алгоритмов.

Выше было дано определение эффективно вычислимой (или просто вычислимой) функции, как функции, для которой существует алгоритм (в интуитивном смысле) вычисления ее значений. Для описания вычислимых функций были введены простейшие вычислимые функции и были рассмотрены средства получения новых вычислимых функций, в том числе примитивно рекурсивных функций. Кроме того, было показано, что существует алгоритм (в интуитивном смысле) вычисления значений суперпозиции функций и функций, полученных с помощью оператора примитивной рекурсии, следовательно, всякая примитивно рекурсивная функция является вычислимой (в интуитивном смысле).

Приведенный пример 1.5 наводит на мысль, что все вычислимые функции являются примитивно рекурсивными, то есть возникает вопрос: совпадает ли класс всех эффективно вычислимых функций с классом примитивно рекурсивных функций? Оказывается, что ответ на этот вопрос отрицательный. Для обоснования этого ответа в общем случае можно провести следующее рассуждение. Как известно из теории множеств, множество всех функций типа N^n несчетно. С другой стороны, поскольку каждая примитивно рекурсивная функция задается конечным словом (то есть имеет конечное описание), а множество всех конечных слов счетно (их можно перенумеровать), то множество всех примитивно рекурсивных функций образует не более чем счетное подмножество несчетного множества функций типа N^n . Таким образом получаем, что класс вычислимых функций значительно шире класса примитивно рекурсивных функций.

В действительности, нас интересует более узкий вопрос: все ли вычислимые функции можно описать как примитивно рекурсивные?

Ответ и на этот вопрос отрицательный. Об этом свидетельствует классический пример, так называемой, вычислимой функции Аккермана, не являющейся примитивно рекурсивной. Идея примера по-строения вычислимой, но не примитивно рекурсивной функции состоит в том, что, построив некоторую последовательность функций, каждая из которых растет быстрее предыдущей, сконструировать с помощью этой последовательности функцию, которая растет быстрее любой примитивно рекурсивной функции. Указанную последовательность функций $f_0, f_1, f_2, \dots$ зададим следующим образом. Положим $f_0(a, y) = a + y$; $f_1(a, y) = ay$, $f_2(a, y) = a^y$. Эти три функции связаны между собой следующими рекурсивными соотношениями:

$$f_1(a, y+1) = a + ay = f_0(a, f_1(a, y));$$

$$f_1^i(a, 1) = a; f_1^i(a, y+1) = a \cdot a^{f_1^i(a, y)}; f_2^i(a, 1) = a.$$

Продолжим эту последовательность, положив по определению (при $n > 2$):

$$f_{n+1}(a, 0) = 1,$$

$$f_{n+1}(a, 1) = a,$$

(1.3)

$$f_{n+1}^{i,j}(a, Y^{j+1}) = f_n^i(a, f_{n+1}^j(a, Y)).$$

Схема (1.3) имеет вид схемы примитивной рекурсии, следовательно, все функции f_i примитивно рекурсивны. Растут они очень быстро, например,

$$f_3(2, 1) = 2; f_3(2, 2) = f_2(2, f_3(2, 1)) = f_2(2, 2) = 2^2 = 4;$$

$$f_3(2, 3) = f_2(2, f_3(2, 2)) = f_1(2, 4) = 2^4 = 16;.,;$$

2 1

.■

$\wedge k$ раз

$$f_3(2, k) = 2^{2^{k-1}}.$$

Зафиксируем теперь значение $a = 2$. Получим последовательность одноместных функций: $f_0(2, y)$, $f_1(2, y)$, $f_2(2, y)$.. Определим теперь функцию $V(x, y) = f_x(2, y)$, т.е. $V(0, y) = f_0(2, y) = 2 + y$; $V(1, y) = f_1(2, y) = 2^y$

и т.д., а также диагональную функцию $A(x) = V(x, x)$. Эти функции и называют функциями Аккермана. Из соотношений (1.3) следует, что

$$\begin{aligned} V(0, y) &= 2 + y, \\ V(x+1, 0) &= V(x, V(x+1, 0)) = \text{sg } x, \\ V(x+1, y+1) &= V(x, V(x+1, y)). \end{aligned} \quad (1.4)$$

Соотношения (1.4) позволяют вычислять значения функции $V(x, y)$ и, следовательно, функции $A(x)$, причем для вычисления функции в данной точке нужно обратиться к значению функции в предшествующей точке, как в схеме примитивной рекурсии. Однако здесь рекурсия ведется сразу по двум переменным, которую называют двойной, 2й степени или перекрестной. Это существенно усложняет характер упорядочения точек, а, следовательно, и понятие предшествования точек.

Например, $V(3, 3) = V(2, V(3, 2))$, а так как $V(3, 2) = f_3(2, 2) = 2^2 = 4$, то вычислению V в точке $(3, 3)$ по схеме (3.4) должно предшествовать вычисление V в точке $(2, 4)$. Важно отметить, что упорядочение не предопределено заранее, как в схеме примитивной рекурсии, где n всегда предшествует $n+1$, а выясняется в ходе вычислений и для каждой схемы вида (1.4), вообще говоря, различно. Поэтому схему двойной рекурсии не всегда удастся свести к схеме примитивной рекурсии.

С помощью довольно сложных рассуждений можно доказать, что функция Аккермана $A(x)$ растет быстрее, чем любая примитивно рекурсивная функция, и, следовательно, не является примитивно рекурсивной.

Таким образом, пример, построенный Аккерманом, позволяет устанавливать существование таких рекурсий, которые не сводятся к примитивным функциям. Поскольку функция Аккермана вычислима, но не может быть описана с помощью примитивной рекурсии, то классы примитивно рекурсивных и вычислимых функций не совпадают.

Исчерпывающей формализацией понятия вычислимой функции оказалось понятие частично рекурсивной функции. Оно и является одним из математических уточнений интуитивного понятия вычислимой функции и, следовательно, уточнением интуитивного понятия алгоритма, если мы алгоритмом назовем всякую частично рекурсивную функцию.

Если мы обозначим через КОР, КЭВ, КЧР и КПР соответственно классы общерекурсивных, эффективно вычислимых, частично рекурсивных и примитивно рекурсивных функций, то, в конечном счете, получим следующее соотношение между ними:

$$\text{КЭВ} = \text{КЧР} \supset \text{КОР} \supset \text{КПР}.$$

Тезис Черча. Любая интуитивно вычислимая функция является частично рекурсивной.

Исторически именно это утверждение было первым точным математическим определением понятия (алгоритмически) вычислимой функции. Тезис А. Черча, сформулированный им в 1936 г., математически недоказуем. Самое большее, что можно сделать это опровергнуть его. Поскольку определение понятия «интуитивно вычисляемая функция» в тезисе Черча используется неформально, невозможно провести никакого формального доказательства. До сих пор не опровергнутый и общепринятый тезис гласит: понятие вычислимости, определенное с помощью рекурсивных функций, ведет к самому мощному реалистическому понятию вычислимости, которое также эквивалентно другим

понятиям вычислимости.

Понятие вычислимости, лежащее в основе тезиса Черча, исходит из предположения о бесконечном множестве состояний вычислительного устройства (конечного автомата). Число состояний конечного автомата может быть неограниченно большим. Это конечно, нереально. Каждая конкретная ЭВМ имеет лишь конечное пространство (множество) состояний и поэтому соответствует конечному автомату. Тем не менее, обобщенное понятие состояния, которое исходит из неограниченного числа состояний и неограниченного множества данных, весьма целесообразно. Понятие алгоритма по Черчу позволяет рассматривать вычисления для множества всех конечных автоматов. Любое завершающееся вычисление может быть успешно проведено, если конечный автомат, на котором оно проводится, достаточно велик.

Тезис Черча дает алгоритмическое истолкование понятию частично рекурсивной функции. Несколько сложнее дело обстоит с истолкованием понятия частичной рекурсивности относительно заданной системы частичных функций Q . Такое истолкование было впервые указано А. Тьюрингом. Для простоты предположим, что система Q состоит лишь из одной функции $h(x)$. Если значения этой функции вычислимы посредством некоторого алгоритма и, значит, ввиду тезиса Черча, функция $h(x)$ частично рекурсивна, то каждая функция f , частично рекурсивная относительно h , будет просто частично рекурсивной. Если заданная функция h не частично рекурсивна, то единого алгоритма для вычисления произвольного значения функции h нет, и вычислить какое-нибудь значение функции $h(x)$ это математическая проблема. Пусть теперь нам задана какая-нибудь функция f , частично рекурсивная относительно h . Это значит, что функция f может быть представлена в виде значения операторного термина, содержащего символ функции h и символы простейших функций. Предположим, что этот терм a нам известен. В таком случае мы легко убеждаемся, что вместе с термом a определен алгоритм, который позволяет вычислять значения функции f при условии, что мы можем находить те значения функции h , на которые указывает алгоритм.

Говорят, что функция f алгоритмически вычислима относительно функции h , если существует алгоритм, позволяющий вычислять значения функции f при условии, что мы можем находить те значения функции h , которые указываются алгоритмом. При этом значения функции h , необходимые на более поздних шагах алгоритма, могут зависеть от значений h , требовавшихся на предыдущих шагах алгоритма.

Это определение относительной алгоритмической вычислимости, конечно, не полное. Смысл его зависит от того, какой смысл мы вкладываем в понятие обычного (не относительного) алгоритма, и других обстоятельств. Как и понятие алгоритма, понятие относительного алгоритма можно уточнять различными способами. Однако при всех фактически испробованных уточнениях оказывалось, что относительно вычисляемые функции являлись относительно частично рекурсивными. Поэтому по аналогии с тезисом Черча обычно принимается в качестве естественно научной гипотезы и тезис Тьюринга.

Т е з и с Т ь ю р и н г а. Класс функций, алгоритмически вычисляемых относительно какой-либо функции h (или класса функций Q), совпадает с классом частичных функций, частично рекурсивных относительно h (соответственно, относительно системы Q).

Из тезиса Тьюринга вытекает тезис Черча. Обратное, повидимому, нельзя считать верным. Конечно, здесь речь идет не о строгой логической зависимости, а скорее о зависимости в некотором философском смысле.

Из простейших функций константы 0 , функции $x' = x+1$ и функций проектирования I_m^n с помощью операторов суперпозиции и примитивной рекурсии можно получить большое многообразие функций из арифметики, алгебры и анализа (с поправкой на целочисленность). Тем самым выяснено, что эти функции имеют примитивно рекурсивное описание, которое однозначно определяет процедуру их вычисления. Следовательно, их естественно отнести к классу вычисляемых функций. Заметим, во-первых, что все примитивно рекурсивные функции всюду определены, и, во-

вторых, строго говоря, мы имеем дело не с функциями, а с их примитивно рекурсивными описаниями.

Отметим, что существование универсальных частично рекурсивных функций (то есть семейства всех n -местных функций, связанных с $n+1$ -местной частичной функцией) является следствием существования универсальных рекурсивно перечислимых предикатов.

Выводы

Теория алгоритмов находит широкое применение, как в недрах самой математики, так и в прикладных вопросах. С возникновением проблем оснований математики на базе теории множеств Кантора возникла проблема уточнения понятия алгоритма. Для этой цели строятся формальные и универсальные модели алгоритмов. Первоисточником для построения таких моделей является интуитивное описание алгоритма и перечисление его основных свойств.

При формулировке общего понятия алгоритма неизбежно возникает дилемма: либо определение алгоритма должно быть общим, чтобы в число объектов, удовлетворяющих этому определению, заведомо вошли все объекты, которые естественно считать алгоритмами, либо должно сохраняться требование об обязательной рекурсивности алгоритма. В первом случае этому определению будут удовлетворять частичные алгоритмы, и избавиться от них конструктивным методом нельзя. Во втором случае никакую процедуру нельзя назвать алгоритмом до тех пор, пока для нее не будет решена проблема остановки. А единого решения этой проблемы не существует. В общей теории алгоритмов используется первый вариант. До сих пор не опровергнутый и общепринятый тезис гласит:

Понятие вычислимости, определенное с помощью рекурсивных функций, ведет к самому мощному реалистическому понятию вычислимости, которое также эквивалентно другим понятиям вычислимости.

Вопросы и упражнения для самопроверки

- 1.1. Привести интуитивное описание алгоритма и перечислить основные свойства алгоритма.
- 1.2. Какие функции называются эффективно вычислимыми?
- 1.3. Почему возникла необходимость уточнения интуитивного понятия алгоритма?
- 1.4. Перечислить простейшие вычислимые функции и указать их область определения.
- 1.5. Дать определение оператора суперпозиции функций в аналитической форме.
- 1.6. Является ли суперпозиция функций всюду определенной?
- 1.7. Как может быть вычислена суперпозиция функций?
- 1.8. Дать аналитическое определение схемы примитивной рекурсии.
- 1.9. Доказать, что любая примитивно рекурсивная функция всюду определена на множестве целых неотрицательных чисел.

Указание. Утверждение следует из определения примитивно рекурсивной функции и того факта, что простейшие функции $z(x)$, $O(x)$ и $J_m(x, \dots, x)$ всюду определены, а применение к ним опе-

n 1 n раторов суперпозиции и примитивной рекурсии приводит снова к функциям, которые всюду определены.

- 1.10. Доказать, что следующие функции примитивно рекурсивны:
 - 1.10.1. $f(x, y) = x^y$ ($x > 0$).
 - 1.10.2. $f(x, y) = X y |$
- 1.11. Чем вызвано расширение средств построения вычислимых функций?
- 1.12. Дать определение оператора минимизации.
- 1.13. Дать определение частично рекурсивной функции.
- 1.14. Сформулировать тезис Черча.

Из задачи 1.18, в частности, следует, что из простейших операторов с помощью лишь суперпозиции можно получить функции вида 1.18.1 1.18.4.

1.19. Доказать, что из операторов $O(x)$ и $I^n(x, x, \dots, x)$ с помощью m 2 п

суперпозиции и схем примитивной рекурсии нельзя получить функции $X(x) = x + 1$ и $f(x) = 2x$.

1.20. Доказать, что любая примитивно рекурсивная функция всюду определена на множестве целых неотрицательных чисел.

Указание. Утверждение следует из определения примитивно рекурсивной функции и того факта, что простейшие функции $X(x), O(x)$ и $I^n(x, \dots, x)$ всюду определены, а применение к ним операторов суперпозиции и примитивной рекурсии приводит снова к функциям, которые всюду определены.

1.20.1. Доказать, что следующие функции примитивно рекурсивны: $f(x, y) = x^y$ ($x > 0$);

, ,, $\Gamma x-1$, если $x > 0$,

1.20.2. $3(x) = I$, ,

[0, если $x = 0$;

1.20.3. $f(x, y) = xy = |^x y$, если $x * y$,

[0, если $x < y$;

1.20.4. $f(x, y) = x y|$.

В качестве примера докажем, что функция 1.21.1 примитивно рекурсивна. Для этого в схеме примитивной рекурсии положим $g(x) = X(O(x)) = X(0) = 1$, $h(x, y, z) = z \cdot x$. Тогда $f(x, 0) = x^0 = g(x) = 1$; df

$f(x, y + 1) = h(x, y, f(x, y)) = f(x, y) \cdot x = (x^y) \cdot x = x^{y+1}$.

1.21. Доказать, что следующие функции частично рекурсивны:

1.21.1. $f(x, y) = \Gamma x^y$, если $x * y$,

[не определена в остальных случаях;

1.21.2. $f(x, y) = |xy$, если $y x$,

[не определена в остальных случаях.

1.22. Рассмотрим функцию $\phi(x, y) = x y$, определяющую разность двух целых неотрицательных чисел x и y . Получить эту функцию, применив оператор минимизации.

Решение. По определению, вычесть из числа x число y , значит найти такое число z , что $x = z + y$. Введем функцию $f(x, y, z) = y + z x$. Требуется найти такое наименьшее неотрицательное число z , чтобы $f(x, y, z) = 0$, т.е. чтобы $y + z = x$. Применяя к этой функции μ -оператор, получим:

$\phi(x, y) = x - y = \mu z[y + z = x] = \mu z[\overset{3}{I^3}(x, y, z) + \overset{1}{1^3}(x, y, z) = I(x, y, z)]$.

Ясно, что функция $\phi(x, y)$ определена только для $x * y$. Вычислим, например, $\phi(9, 4)$. Для этого положим $x = 9$, $y = 4$ и будем z придавать последовательно значения $0, 1, \dots$, пока не получим верное равенство $y + z = x$: $z = 0, 4 + 0 = 4 \wedge 9$; $z = 1, 4 + 1 = 5 \wedge 9$; $z = 2, 4 + 2 = 6 \wedge 9$; $z = 3, 4 + 3 = 7 \wedge 9$; $z = 4, 4 + 4 = 8 \wedge 9$; $z = 5, 4 + 5 = 9$. Таким образом, $\phi(9, 4) = 5$.

1.24. Применить оператор минимизации к следующим функциям f по переменной x_i .

1.24.1. $f(x_1) = 3$; $f(x_1) = [x_1/2]$, где символом $[x_1/2]$ обозначена целая часть числа $x_1/2$ и $[x_1/2] = m$, если $x_1 = 2m$ или $x_1 = 2m + 1$, $m > 0$;

1.24.2. $f(x_1, x_2) = I^2(x, x)$;

1 12

1.24.3. $f(x_i, x_2) = x_1 x_2$.

Указания. 1.24.1. Положим $y = x_1$. Тогда $\mu y[y - 3 = 0] = 3$, то есть $\phi(x_1) = 3$.

1.24.2. Снова полагая $x_1 = y$, запишем $\mu y[[y/2]] = 2x_1$, то есть $\phi(x_1) = 2x_1$.

1.24.3. $\mu y[I^2(x, y)] = \mu y[x_1] = 0$, то есть $\phi(x_1) = 0$.

- 1.4.4. 1) $\mu x (x \cdot x) = (x + x) \operatorname{sg} x$, где $\operatorname{sg} x = 0$, если $x = 0$,
 $\mu x (x \cdot x) = x + x$, если $x > 0$,
 2) $\mu x (x \cdot x + 1) = x + 1$.
- 1.25. Доказать, что следующие функции частично рекурсивны:
 1.25.1. Нигде не определенная функция $f(x)$.
 1.25.2. $f(x, y) = M_x y$, если $x > y$,
 не определена в остальных случаях.
 1.25.3. $f(x, y) = \Gamma_x / y$, если $y \mid x$,
 [не определена в остальных случаях.
 Указание. $f(x, y) = \mu z [x - zy = 0]$.
 1.25.4. $f(x, y) = z$, если $z \cdot y = x$,
 [не определена в остальных случаях.
 Указание. $f(x, y) = \mu z [x - z^y = 0]$.
- 1.26. Доказать, что всякая примитивно рекурсивная функция общерекурсивна.
- 1.27. Доказать, что следующие примитивно рекурсивные функции являются общерекурсивными:
 1.27.1. $f(x, y) = x^y$;
 1.27.2. $\delta(x) = M_x^{-1}$, если $x > 0$,
 $\delta(0) = 0$.
 Указание. Эта функция удовлетворяет соотношениям
 $\delta(0) = 0$, $\delta^{\circ}(0) = 0$,
 $\delta(y + 1) = y$ или $\delta(y + 1) = I^2(\delta(y), y)$.
 Изначит, функция $\delta(x)$ получается по схеме примитивной рекурсии из общерекурсивных функций и поэтому сама является общерекурсивной.
- 1.27.3. $x \dot{-} y = \begin{cases} x - y, & \text{если } x \geq y, \\ 0, & \text{если } x < y. \end{cases}$
 Указание. $x \dot{-} 0 = x$, $x \dot{-} (y + 1) = (x \dot{-} y) \dot{-} 1$.
 1.27.4. $|x - y|$.

12. УТОЧНЕНИЕ ПОНЯТИЯ АЛГОРИТМА НА ОСНОВЕ МАШИНЫ ТЬЮРИНГА

План занятия

1. Модель одноленточной машины Тьюринга.
2. Композиция, итерация и разветвление машин Тьюринга.
3. Методологическое значение моделей машин Тьюринга.

Еще одно направление уточнения понятия алгоритма связано с машинной математикой. Здесь сущность понятия алгоритма раскрывается путем рассмотрения процессов, осуществляемых в машине. Впервые это было сделано американским ученым Тьюрингом в 1936 г., который ввел еще один точно определенный класс эффективно вычислимых функций, который теперь называют функциями, вычислимыми по Тьюрингу. Относительно этого класса был высказан тезис (тезис Тьюринга): всякий алгоритм может быть реализован машиной Тьюринга.

Оказывается, что всякий алгоритм, описанный в терминах частично рекурсивных функций, можно реализовать машиной Тьюринга, и наоборот, то есть тезисы Черча и Тьюринга эквивалентны, поэтому их обычно объединяют в один тезис: тезис Черча Тьюринга.

Таким образом, можно утверждать, что любые предложения о существовании или отсутствии алгоритмов, сделанные в одной из этих теорий, верны и в другой. То же относится и к другим способам уточнения интуитивного понятия алгоритма. В сочетании с тезисом Черча-Тьюринга это означает, что такие утверждения можно сформулировать для алгоритмов вообще, не фиксируя конкретную модель и используя результаты всех подходов к уточнению понятия алгоритма. Таким образом, возможно изложение теории алгоритмов, инвариантное по отношению к способу формализации понятия алгоритма, так как при любой формализации основные свойства алгоритмов (кроме их сложности) остаются инвариантными. Основные понятия такой инвариантной (общей) теории алгоритмов это алгоритм (рекурсивное описание функции, система команд машины Тьюринга) и вычислимая функция.

Понятие машины Тьюринга возникает в результате прямой попытки разложить интуитивно известные нам вычислительные процедуры на элементарные операции. Тьюринг привел ряд доводов в пользу того, что повторение его элементарных операций достаточно для проведения любого возможного вычисления. Он описал некоторого рода гипотетическую вычислительную машину (раньше создания ЭВМ!).

От человека-вычислителя, выполняющего данные ему предписания, машина Тьюринга отличается в двух отношениях. Во-первых, машина Тьюринга не может ошибаться, то есть она без всяких отклонений выполняет правила, установленные для ее работы. Во-вторых, машина Тьюринга снабжена потенциально бесконечной памятью в отличие от ЭВМ. Это значит, что хотя в каждый момент времени количество накопленной ею информации конечно, для нее нет никакой верхней грани.

Таким образом, подход Тьюринга к уточнению понятия алгоритма основан на представлении об алгоритме как о некотором детерминированном устройстве, способном выполнять в каждый отдельный момент времени лишь весьма примитивные операции. Эвристика модели Тьюринга близка к ЭВМ и, следовательно, к инженерной интуиции.

Если для решения некоторой массовой проблемы известен алгоритм, то для его реализации необходимо лишь четкое выполнение предписаний этого алгоритма. Автоматизм, необходимый для реализации алгоритма, естественно приводит к мысли о передаче функции человека, реализующего алгоритм, машине. Рассмотрим одну из моделей машины Тьюринга.

2. 1. Модель одноленточной машины Тьюринга

Устройство одноленточной машины Тьюринга включает в себя:

1. Внешний алфавит, то есть конечное множество символов

$A = \{a_0, a_1, \dots, a_n\}$. В этом алфавите в виде слова кодируется та информация,

которая подается в машину. Машина преобразует информацию, поданную в виде слова, в новое слово.

2. Внутренний алфавит (внутренняя память) машины, состоящий из конечного множества состояний $Q = \{q_0, q_1, \dots, q_m\}$. Два состояния

имеют особое назначение: q_1 начальное состояние машины, q_0 заключительное состояние. Остальные состояния вместе с начальным состоянием называются рабочими состояниями.

3. Алфавит сдвигов $S = \{l, s, r\}$, где l, s, r соответственно означают сдвиг некоторого символа влево, оставить символ на месте, сдвиг символа вправо.

Кроме того, в конфигурации машины имеются:

а) бесконечная в обе стороны лента (внешняя память машины). Она разбита на клетки. В каждую клетку может быть записана только одна буква из алфавита A , включая пустой символ (пробел), который мы будем обозначать через a_0 или ϵ ;

б) управляющая головка (считывающе-записывающее устройство). Она обладает способностью передвигаться вдоль ленты и может останавливаться напротив какой-либо клетки, то есть обозревать одну ячейку ленты, считывать букву, записанную в эту ячейку, заносить на место считываемой буквы любую другую, в том числе пустой символ a_0 (то есть стирать в ячейке символ). В одном такте работы машины управляющая головка может сдвигаться влево или вправо только на одну клетку или может оставаться на месте;

в) устройство управления, которое управляет работой машины с помощью программы;

г) программа машины, определяющая переходы машины от одной конфигурации к другой, то есть от одного состояния q_i вместе с содержимым обозреваемой клетки на данном такте работы к другому состоянию q_j вместе с соответствующим содержимым обозреваемой клетки на следующем такте работы машины.

Хотя лента и бесконечна в обе стороны, в начальный момент времени только конечное число ячеек ленты заполнено непустыми символами; остальные ячейки пусты, то есть содержат пустой символ a_0 . Из характера работы машины Тьюринга следует, что и в любой последующий момент времени лишь конечный отрезок ленты будет заполнен символами алфавита A . Поэтому важна не фактическая (актуальная) бесконечность ленты, а ее неограниченность, то есть возможность писать на ней сколь угодно длинные, но конечные слова. Данные в машине Тьюринга это слова в алфавите A . На ленте записываются исходные данные, и окончательные результаты. Элементарные шаги машины это считывание и запись символов, сдвиг головки на одну ячейку влево или вправо, а также переход управляющего устройства в следующее состояние. Детерминированность машины, то есть последовательность ее шагов, определяется следующим образом: для любого внутреннего состояния q_i и символа a однозначно заданы: а) следующее состояние q_t ; б) буква a_v , которую нужно записать вместо a_j в ту же ячейку (стирание символа будем понимать как запись пустого символа a_0); в) направление сдвига головки d_k , обозначаемое одним из символов l, s, r . Это задание может описываться либо системой правил (команд), имеющих вид:

$$q_i a_j \rightarrow q_t a_v d_k, \quad (2.1)$$

либо таблицей, строкам которой соответствуют состояния, столбцам входные символы, а на пересечении строки q_i и столбца a_j записана тройка символов $q_t a_v d_k$, и, наконец, диаграммой переходов (оргра

фом). На этой диаграмме состояниям соответствуют вершины, а команде вида (2.1) дуга, ведущая из вершины q_i в вершину q_t , на которой написано $a_j \rightarrow a_v d_k$. Условие однозначности требует, чтобы для любого j и любого $i \in Q$ в системе команд имелась одна команда, аналогичная команде (2.1) с левой частью $q_i a_j$; состояние q_0 в левых частях команд не встречается. На диаграмме переходов это выражается условием, что из каждой

вершины, кроме q_0 , выходит ровно n дуг, причем на разных дугах левые части различны; в вершине q_0 нет выходящих дуг.

Из отдельных команд машины Тьюринга составляется программа этой машины.

Программа машины Тьюринга это отображение $\langle p: A^* \times Q \setminus \{q_0\} \rightarrow A^* \times Q \times S \rangle$, то есть правило, сопоставляющее любой паре (a^*, q_i) , $i \neq 0$, тройку (av, qt, dk) , как было описано выше.

В качестве начальной информации на ленту можно подать любую конечную систему знаков внешнего алфавита A (то есть любое слово в этом алфавите), расставленную произвольным образом по ячейкам. Но, в зависимости от того, какая была подана начальная информация, возможны два случая.

1) После конечного числа тактов машина останавливается (переходит в состояние q_0) и при этом на ленте оказывается изображенной выходная информация (результат). В таком случае говорят, что машина Тьюринга применима к начальной информации и преобразует ее в результирующую информацию.

2) Машина никогда не останавливается. Тогда говорят, что машина не применима к начальной информации (проблема остановки).

Теперь опишем кратко функционирование машины Тьюринга.

Машина работает дискретно (по шагам). На каждом шаге (такте) происходит переход от одной конфигурации к другой. Перед началом работы машина находится в начальной конфигурации: считывающе-записывающее устройство, то есть управляющая головка, обозревает первую букву слова, а машина находится в начальном состоянии q_1 . Головка считывает букву, находящуюся в обозреваемой ячейке. Устройство управления обращается к программе машины, находит в таблице клетку, соответствующую считанной букве и состоянию машины. Пусть в этой клетке находится тройка (a_y, qt, dk) . Тогда буква av заносится в обозреваемую ячейку, машина переводится в состояние qt , а головка совершает сдвиг в соответствии со значением dk . На этом завершается работа машины Тьюринга на первом такте, и она готова к выполнению следующего аналогичного шага в соответствии с программой

О существовании машины Тьюринга свидетельствуют простейшие (стандартные) машины: тождественная, копирующая, заменяющая машины! и машины! проекторы.

Эти машины определяются следующим образом.

1. Тождественная машина E применима к любому слову над алфавитом A и $\forall u \in A^* E(u) = u$, где через A^* обозначено множество всех входных слов над алфавитом A , а через $E(u)$ результат работы машины E над входным словом u .

2. Пусть A внешний алфавит, A^* множество всех входных слов над алфавитом A , $A \in A$, $n \in \mathbb{N}$. Машина K_n называется копирующей, если она применима к любому слову $u \in A^*$, причем $K_n(u) = uA^nA - A^n$

n раз

3. Пусть A алфавит и $a, v \in A$ ($a \neq v$). Машиной Z_a^v , заменяющей символ a символом v , называется машина, применяемая к любому слову $u \in A^*$ так, что $\forall i \in \mathbb{Z} (z_a^v(i) = u(i) \text{ если } u(i) \neq a, \text{ и } v \text{ если } u(i) = a)$

$f_i(i)$, если $u(i) \neq a$

$(i) = a$

$[v]$, если $u(i) = a$.

4. Пусть A внешний алфавит и $A \in A$. Машиной проектором $\Pi_1(A)$ ($\Pi_2(A)$) называют машину, применимую к любому слову $u \in A^*$, где $u, v \in A^*$, причем $\Pi_1(u) = u$; $\Pi_2(u) = v$.

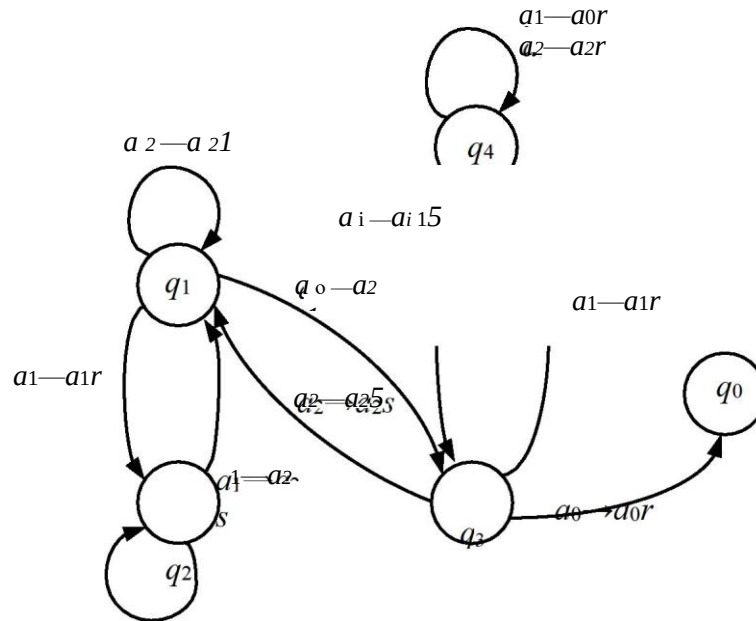
Пример 2.1. Пусть внешний алфавит машины Тьюринга $A = \{a_0, a_1, a_2\}$, множество ее состояний $Q = \{q_1, q_2, q_3, q_4, q_0\}$, а таблица 2.1 переходов имеет вид:

Таблица 2.1

Таблица переходов машины Тьюринга к примеру 2.1

a_i	a_o	a_i	a_2
$Q \setminus \{$			
q_1	$a_2 q_3$	$a_i q_2 r$	$a_2 q_1 l$
q_2	$a_o q_2$	$a_2 q_1 s$	$a_i q_2 s$
q_3	$a_o q_o$	$a_i q_4 r$	$a_2 q_1 s$
q_4	$a_i q_3$	$a_o q_4 r$	$a_2 q_4 r$

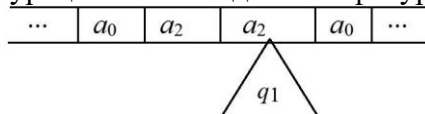
Диаграмма (граф) переходов для данной машины имеет вид:



$a_1 \rightarrow a_{25}$

Рисунок 2.1. Диаграмма (граф) переходов данной машины Тьюринга

Рассмотрим, как работает данная машина Тьюринга. а) Пусть начальная конфигурация имеет вид: 1^я конфигурация:



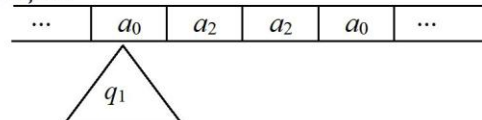
Так как управляющей головкой обзревается буква a_2 , а машина находится в состоянии q_1 , то вырабатывается команда $a_2 q_1 l$ и в результате получаем вторую конфигурацию:

Выполненная команда

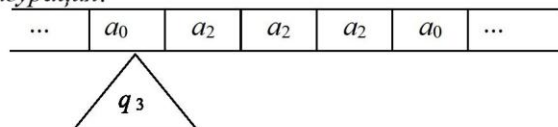
2^я конфигурация:

Следующие конфигурации имеют вид:

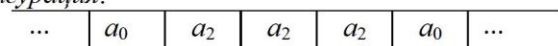
3^я конфигурация:



4^я конфигурация:

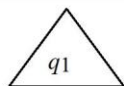


5^я конфигурация:

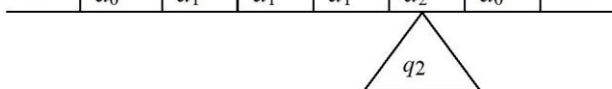
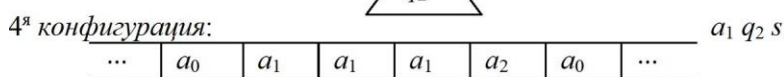
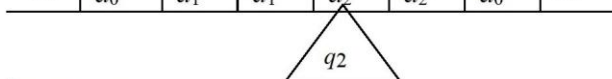
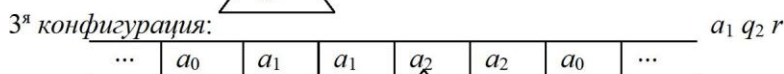
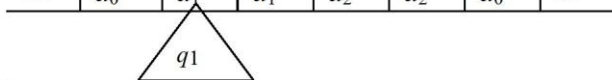
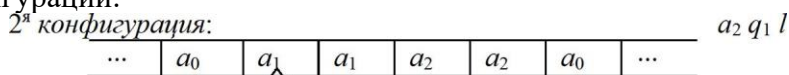




Так как в пятой конфигурации машина перешла в состояние q_0 , то слово $a_2 a_2 a_2$ является результатом и машина остановится. б) Пусть теперь начальная конфигурация имеет вид: 1^я конфигурация:



Выполняя последовательно программу как в случае а), получим следующие конфигурации:



5^a конфигурация: a2 q 1 5 ... a o a 1 a 1 a 1 a 1 a o ...
 I | Us Ls J qJ U

Начальная и пятая конфигурации показывают, что функционирование машины будет повторяться и, следовательно, машина никогда не остановится и поэтому не применима к данному слову.

Пример 2.1. Реализация на машине Тьюринга алгоритма перехода от числа n к $n+1$ в десятичной системе счисления.

Ясно, что внешний алфавит $A = \{a_0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, где a_0 символ пустой клетки. Число n будем записывать в виде его цифр по одной в каждой клетке подряд без пропусков.

Чтобы решить поставленную задачу, машина должна на первом такте работы стереть последнюю цифру числа n , заменить ее цифрой на единицу большей и перейти в состояние q_0 , если последняя цифра была меньше 9. Если же последняя цифра числа n была 9, то машина должна, заменив цифру 9 пустым символом, записать в освободившуюся клетку цифру 0 и произвести сдвиг влево к соседнему более высокому разряду, оставаясь в том же начальном состоянии. Здесь, на втором такте работы машина должна прибавить единицу к цифре более высокого разряда. Очевидно, что в случае сдвига влево управляющая головка машины может выйти на пустую клетку в случае, когда цифры более высокого разряда нет. При этом машина вписывает в пустую клетку цифру 1.

Из сказанного следует, что при реализации алгоритма вычисления функции $Z(n) = n+1$ машина может пребывать лишь в двух состояниях q_1 и q_0 . Таким образом, программу машины Тьюринга, реализующей алгоритм перехода от числа n к числу $n + 1$ в десятичной системе счисления, можно представить в виде следующей таблицы 2.2:

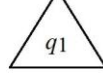
Таблица 2.2

Таблица переходов к примеру 2.1

[illegible]

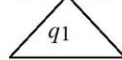
Начальная (1^я) конфигурация:

...	a_0	3	9	9	a_0	...
-----	-------	---	---	---	-------	-----



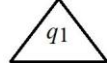
2^я конфигурация:

...	a_0	3	9	0	a_0	...
-----	-------	---	---	---	-------	-----



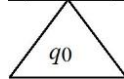
3^я конфигурация:

...	a_0	3	0	0	a_0	...
-----	-------	---	---	---	-------	-----



4^я конфигурация:

...	a_0	4	0	0	a_0	...
-----	-------	---	---	---	-------	-----



Пусть, например, $n = 399$. Тогда соответствующие конфигурации машины Тьюринга имеют вид:

2. 2. Композиция, итерация и разветвление машин Тьюринга

Определение 2.1. Пусть T_1 и T_2 две машины Тьюринга соответственно с внешними и внутренними алфавитами A_1, A_2, Q_1, Q_2 , причем $Q_1 \cap Q_2 = \emptyset$. Композицией машин T_1 и T_2 называется машина

$T_2 \circ T_1$ с внешним алфавитом $A = A_1 \cup A_2$, алфавитом состояний

$(Q_1 \cup Q_2) \setminus \{q_1\}$ (q_1 — заключительное состояние машины T_1) и работающая по правилу:

$$\forall u \in A^* \quad T_2 \circ T_1(u) = T_2(T_1(u)).$$

2 1 2 1

Пусть P_1 и P_2 программы соответственно машин T_1 и T_2 , q_1 начальное состояние машины T_2 . Заменяя всюду в программе P_1 состояние q_1 на состояние q_2 , и, объединяя полученную программу P_1 с программой P_2 , получим программу $P = P_1 \cup P_2$, являющуюся про-

граммой машины $T_2 \circ T_1$ по паре состояний (q_1, q_2) .

Определение 2.2. Пусть q_0 — заключительное состояние машины Тьюринга T , q_1 — какое-либо рабочее состояние. Заменяем всюду в программе P машины T состояние q_0 на q_1 . Получим программу P' , определяющую машину $T(q_0, q_1)$. Машина T' называется итерацией машины T по паре состояний (q_0, q_1) , при этом машина T может иметь несколько заключительных состояний. Определение 2.3. Пусть машины T_1, T_2 и T_3 задаются программами P_1, P_2 и P_3 соответственно. Предположим, что множества состояний Q_1, Q_2 и Q_3 этих машин попарно не пересекаются. Пусть q_1 и q_2 — какие-либо различные заключительные состояния машины T_1 . Заменяем всюду в программе P_1 состояние q_1 некоторым начальным состоянием q'_1 машины T_2 , а состояние q_2 некоторым начальным состоянием q'_2 машины T_3 . Затем новую программу объединим с программами P_2 и P_3 . Получим программу P , задающую машину Тьюринга $T = T(T_1, (q_1, q'_1), T_2, (q'_2, q'_3), T_3)$. Эта машина называется разветвлением машин T_2 и T_3 , управляемым машиной T_1 .

Пример 2.2. Построить композицию $T_2 \circ T_1$ машин Тьюринга T_1 и T_2 по паре состояний (q_1, q_2) , где q_2 — заключительное состояние машины T_2 . Машины T_1 и T_2 заданы своими программами соответственно P_1 и P_2 в виде таблиц:

Таблица 2.3

Таблица 2.4

Программа машины T_1

Программа машины T_2

1:	Δa_i			2:	$\Delta 4 a_i \Delta$		

По определению композиции машин Тьюринга и в соответствии с алгоритмом составления программы этой композиции, получим программу композиции данных машин Тьюринга в виде таблицы 2.5.

Таблица 2.5

Программа композиции			
машин T1 и T2		0	1
Р T T • T 2	q		
	q 11	q 12 0 r	q 11 1 r
	q 12	q 13 0 r	q 11 1 r
	q 13	q 21 1 l	-
	q 21	q 22 0 l	q 21 1 l
	q 22	q 23 0 l	q 22 1 l
	q 23	q 20 0 r	q 23 1 l

Пример 2.3. Составить программу итерации машины Тьюринга Т по паре состояний (q0, q1) (заключительными состояниями являются q0 и q0), если эта машина задана программой Р в виде таблицы 2.6:

Таблица 2.6

Программа машины Т

A		0	1
q"-A : :	q 1	q 0 0	q 2 0
	q 2	q 0 0	q 3 0
	q 3	q 4 0	q 3 1
	q 4	q 5 1	q 4 1
	q 5	q 6 0	q 5 1
	q 6	q 0	q 6 1

Заменяя в программе Р состояние q0 на состояние q1, получим программу Р итерации машины! Тв виде таблицы 2.7:

Таблица 2.7

Программа итерации машины Т

^qT^		0	1
^	q 1	q '0	q 2 0
	q 2	q0'0	q 3 0
	q 3	q 4 0	q 3 1
	q 4	q 5 1	q 4 1
	q 5	q 6 0	q 5 1
	q 6	q 10	q 6 1

Пример 2.4. Пусть машины T1, T2 и T3 задаются программами P1, P2 и P3 соответственно в таблицах 2.8, 2.9 и 2.10:

Программа машины T1

Таблица 2.8

$q_{ij} \backslash A$	0	1
q_{11}	$q_{12} 0 r$	$q_{11} 1 r$
q_{12}	$q_{10}' 0 l$	$q_{13} 1 r$
q_{13}	$q_{10}'' 0 l$	$q_{13} 1 r$

Таблица 2.9

Программа машины T_2

$q_{ij} \backslash A$	0	1
q_{21}	$q_{22} 0 l$	$q_{21} 1 l$
q_{22}	$q_{20} 0 r$	$q_{22} 1 l$

Таблица 2.10

Программа машины T_3

q_{ij}	0	1
q	$q_{32} 0 r$	$q_{31} 1 r$
q	$q_{30} 1 \wedge$	$q_{31} 1 r$

Построить программу P , задающую разветвление машин T_2 и T_3 , управляемое машиной T_1 , где q_{20} заключительное состояние машины T_2 , а q_{30} заключительное состояние машины T_3 .

Согласно определению 2.3 разветвления машин Тьюринга программа P имеет вид таблицы 2.11:

Таблица 2.11

Программа разветвления

$q_{ij} \backslash A$	0	1
q_{11}	q	q_{11}
q_{12}	q	q_{13}
$P \quad q_{13}$	q	q_{13}
q_{21}	q_{22}	q_{21}
q_{22}	q	q_{22}
q_{31}	q_{32}	q_{32}

$q_{30} \wedge Q \wedge q_{30} T \wedge I Z Z q_{ii} Z Z$. 3. Методологическое значение моделей машин Тьюринга

Машина Тьюринга дает один из путей уточнения понятия алгоритма. В связи с этим возникают вопросы: насколько общим является понятие машины Тьюринга? Можно ли считать, что способ задания алгоритмов с помощью машины Тьюринга является универсальным? Может ли всякий алгоритм задаваться таким образом? На эти вопросы современная теория алгоритмов предлагает ответ в виде следующей гипотезы:

Всякий алгоритм может быть задан посредством функциональной схемы Тьюринга (таблицы) и реализован в соответствующей машине Тьюринга (тезис Тьюринга).

Эта гипотеза может быть опровергнута, если удастся привести пример алгоритма, который не может быть реализован с помощью функциональной схемы Тьюринга.

Однако все известные до сих пор алгоритмы могут быть заданы посредством функциональных схем Тьюринга. Более того, тезис Тьюринга оказался равносильным тезису Черча и другим способам уточнения интуитивного понятия алгоритма. Машина Тьюринга является важным инструментом исследований в современной теории алгоритмов и вычислимости. Именно машина Тьюринга в абстрактной форме предвосхитила многие основные принципы современных ЭВМ, которые, вероятно,

оказали существенное влияние и на работы Джон фон Неймана по созданию ЭВМ и его идеи в этом направлении.

Доказано, что существуют универсальные машины Тьюринга, способные моделировать любую конкретную машины Тьюринга с внешним алфавитом S и внутренним алфавитом Q . Более того, доказано существование универсальных машин Тьюринга с бинарным внешним алфавитом $S = \{0, 1\}$ и машин, чья минимальная сложность $SL = s \times q$ определяется наилучшим на сегодняшний день значением $SL = 4 \times 28$ (включая заключительное состояние q_0). Таким образом, столь просто устроенный и функционирующий абстрактный автомат, как машина Тьюринга, способен реализовать сколь угодно сложный алгоритм в нашем современном его понимании.

Являясь абстрактной моделью реальных ЭВМ, машины Тьюринга определяют их вычислительные возможности вычислимость частично рекурсивных функций, тем самым, давая ответ на вопрос о возможностях современной цифровой вычислительной техники. Иными словами, на сегодняшний день теоретически не существует алгоритма в нашем понимании, который не мог бы быть реализован средствами современной вычислительной техники. С практической же точки зрения данное утверждение не выглядит столь убедительным. Например, существуют целочисленные функции, вычисление значений которых требует сколь угодно большой виртуальной памяти (оперативной и внешней) и времени вычислений. Типичным примером таких задач является вычисление n -го члена последовательности однозначно определенных сумм для сколь угодно большого значения n . Единственным алгоритмом для получения n -го члена такой суммы является наличие информации обо всех предыдущих членах последовательности. Поэтому для каждой конкретной ЭВМ существует такое целое число $n > 0$, для которого вычисление n -го члена суммы требует увеличения основных ресурсов ЭВМ. В этом смысле ни одна ЭВМ не может считаться универсальной в смысле вычислимости произвольной частично рекурсивной функции, то есть для нее существует класс нерешаемых задач при условии неизменности ее ресурсов (в первую очередь памяти). Универсальность ЭВМ носит сугубо потенциальный характер, ибо любая ЭВМ является конечным автоматом (хотя и с очень большим числом внутренних состояний).

Таким образом, бесконечность внешней памяти машины Тьюринга принципиально расширяет возможности абстрактных машин Тьюринга как вычислителей.

Каковы же общие черты формальной машины Тьюринга и реальной современной ЭВМ?

1. Подобно модели машины Тьюринга в ЭВМ можно выделить элементарные неделимые единицы информации соответственно символы алфавита S и двоичный алфавит.
2. Подобно модели машины Тьюринга ЭВМ располагает множеством команд, лежащих в основе реализации и выполнения каждого алгоритма.
3. Подобно модели машины Тьюринга ЭВМ функционирует дискретно (по тактам) под управлением программы, хранящейся в оперативной памяти.
4. Устройство управления модели машины Тьюринга по назначению в общих чертах аналогично устройству управления ЭВМ.

Однако ЭВМ имеют относительно моделей машин Тьюринга существенно более сложную организацию и широкий набор более крупных команд, что позволяет эффективно и выразительно представлять разнообразные алгоритмы решаемых задач. Более того, в предположении о возможности наращивания памяти в необходимых объемах каждая ЭВМ может моделировать любую машину Тьюринга с алфавитами S и Q , являясь потенциально универсальной. В этом смысле термин «универсальная ЭВМ» теряет определенный идентифицирующий смысл и его использование при классификации ЭВМ в строгом понимании некорректно. Более приемлемым и обоснованным следует признать классификацию ЭВМ по сферам и видам применения.

Несложно погрузить имитатор любой модели машины Тьюринга с алфавитами S и

Q в программную среду современной ЭВМ, моделируя ее работу и универсальную вычислимость ЭВМ при условии возможности неограниченного наращивания ресурсов памяти.

Выводы

Машина Тьюринга была предложена как простая гипотетическая модель вычислительного устройства, а вычисления этой машины могут быть описаны, как мы видели, через последовательность конфигураций. Имеются и другие варианты машин Тьюринга, например, машины с лентой, бесконечной только в одну сторону; многоленточные машины Тьюринга. Однако все они приводят к тому же самому понятию функции, вычислимой по Тьюрингу.

Вопросы и упражнения для самопроверки

- 2.1. Какие алфавиты включает в себя машина Тьюринга?
- 2.2. Какие состояния выделяются особо во внутренней памяти?
- 2.3. Что еще входит в конструкцию машины Тьюринга?
- 2.4. Что понимается под конфигурацией машины Тьюринга?
- 2.5. Что представляет собой программа машины Тьюринга? Какими способами ее можно задать?
- 2.6. Что понимают под данными и результатами в машине Тьюринга?
- 2.7. Какие шаги машины являются элементарными?
- 2.8. В каких случаях считается, что машина Тьюринга применима к начальной информации, а в каких случаях не применима?
- 2.9. Охарактеризовать функционирование машины Тьюринга.
- 2.10. Сформулировать тезис Тьюринга.
- 2.11. Выяснить, применима ли машина Тьюринга Т, задаваемая программой Р, к слову u . Если она применима, то найти результат применения машины Т к слову u . Предполагается, что во внешнем алфавите $A=\{0,1\}$ символ 0 пустой, а в алфавите состояний $Q=\{q_0, q_1, q_2, q_3\}$ q_1 начальное состояние, q_0 заключительное и в начальной конфигурации управляющая головка обозревает самую левую единицу на ленте, если не

2.11.1.

$$P: \begin{array}{l|l} q_1 0 \wedge q_2 0 r, & \\ q_1 \wedge q_1 r, & a) u = 1^3 0^2 1^2 \\ q_2 0 \wedge q_3 0 r, & б) u = 1^3 0 1^3 \\ q_1 \wedge q_1, & в) u = 10[01] \end{array}$$

оговорено противное. Программу Р задать таблицей и диаграммой переходов.

$$q_3 0 \wedge q_0 0 5,$$

$$q_1 \wedge q_1 r.$$

2.11.2.

$$\Gamma q_1 0 \wedge q_1 1 r,$$

$$q_1 \wedge q_0 r,$$

$$| 1$$

$$P: \text{Й } 2 0 \wedge q_1 1 r,$$

$$^1 q_1 \wedge q_1 1,$$

$$| 2$$

$$q_3 0 \wedge q_1 1 1.$$

Замечание. Если в программе машины нет команды вида $q_i a j \wedge q_k a l dt$ для пары (q_i, a_j) , то машина прекращает работу, а результирующая конфигурация является заключительной.

Решение 2.11.1 а). Запишем программу Р в виде таблицы:

Таблица 2.12

Программа Р машины Тьюринга

A q _i	0	1
q 1	q 20r	q 11 r
q 2	q 30r	q 111
q 3	q 005	q 21 r

Запишем последовательно все переходы от одной конфигурации к другой согласно программе, начиная с начальной конфигурации:

.
 : 0 1
 .
 .
 .
 .
 q1
 10 0 1 10
 q 2 6: 0 1
 1 1
 q3 7: 0 1 0 0 1 1 0
 1 1 8: 0 1 0 0 1 1 0
 q2
 1 1 9: 0 1 0 0 1 1 0
 q1
 1 1 10: 0 1 0 0 1 1 0
 q1
 1 1 11: 0 1 0 0 1 1 0
 q1
 1 1 12: 0 1 0 0 1 1 0
 q2
 1 1 13: 0 0 1 1 0
 q3
 1 1 0 0 1 1 0
 q0

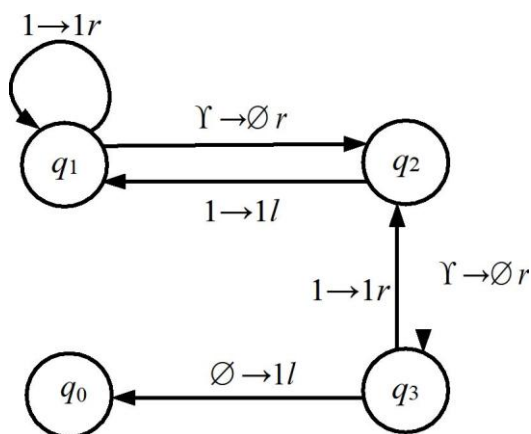


Рисунок 2.2. Граф переходов машины Т

Таким образом, $T(u) = 1110011 = 1^3 0^2 1^2 = u$. Изобразим программу машины Т диаграммой переходов (рис.2.2):

Легко проследить по этой диаграмме переходов процесс преобразования вектора u в вектор $T(u)$.

Решение 2.11.1 б). Запишем последовательность конфигурации для вектора $u=1^3 0 1^3$.

1:	0	1	1	1	0	1	1	1	0
		q^1							
2:	0	1	1	1	0	1	1	1	0
			q^1						
3:	0	1	1	1	0	1	1	1	0
				q^1					
4:				1	0	1	1	1	0
					q^1				
5:				1	0	1	1	1	0
						q^2			
6:				1	0	1	1	1	0

q^1
Конфигурация 6 совпадает с конфигурацией 4, следовательно, машина Тьюринга не применима к входному слову $u=1^3 0 1^3$.

Ответ к задаче 2.11.1 в): $10 [01]^2 1$.

2.12. По заданной начальной конфигурации $K1$ найти заключительную конфигурацию.

а) $K1 = \dots 0111111\dots$

q^1

б) $K1 = \dots 01011110\dots$

q^1

Таблица 2.13

Программа Р машины Тьюринга

A q_i	0	1
q^1	$q^0 0s$	$q^2 1$
q^2	$q^0 1l$	$q^3 0$
q^3	$q^1 1l$	$q^1 0$

Решение 2.12 б). Запишем программу Р в виде системы команд и построим диаграмму переходов:

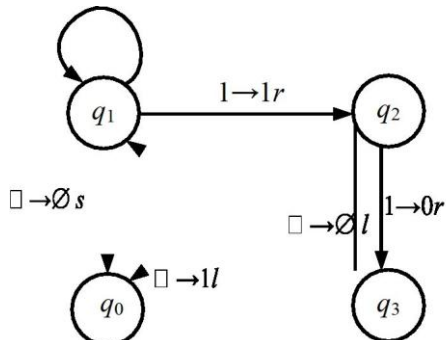
$| q^0 0 \wedge q^0 0 5, q^1 \wedge q^1 r,$
 I^1

$J^2 q^2 0 \wedge q^0 11, q^1 \wedge q^0 r,$
 23

$q^3 0 \wedge q^1 1l,$

$q^3 1 \wedge q^0 r.$

$1 \wedge 1 r$



Итак, последовательность конфигураций имеет вид: $K1: \dots 01011110$.

q^1

$K2: \dots 01011110$.

Кз: ...01011110...
 q 1
 К4: .01011110.
 q 1
 К5: .01011110.
 q 1
 Кs: .010111100.
 q2
 Кг. .0101111000.
 q 3
 К0: .0101111000.
 q0 stop

Таким образом, конечная конфигурация К0 является заключительной и результатом работы машины Т является то же самое слово $u=101^4$.

Отметим, что в случае задачи 2.12 а) результатом работы машины Т над словом u также является то же слово u .

2.13. Построить композицию $T_2 \circ T_1$ машин Тьюринга Т1 и Т2 (по паре состояний (q_{10}, q_{21})) и найти результат применения композиции $T_2 \circ T_1$ к слову u (q_{20} заключительное состояние машины Т2).

Таблица 2.14

Программа P_1			
	0	1	
q_{11}	$q_{12} 0 r$	$q_{11} 1 r$	
q_{12}	$q_{13} 0 r$	$q_{11} 1 r$	
q_{13}	$q_{10} 1 l$	-	

Таблица 2.15

Программа P_2			
$A \setminus A$ $q_{ij}''' \setminus$	0	1	
q_{21}	$q_{22} 0 l$	$q_{21} 1 l$	
q_{22}	$q_{23} 0 l$	$q_{22} 1 l$	
q_{23}	$q_{20} 0 l$	$q_{23} 1 l$	

а) $u = 120 1 3012$; б) $u = 12 0 12 02 12$.

Решение а). По определению композиции машин Тьюринга и согласно алгоритму составления программы этой композиции, получим:

Таблица 2.14

Программа композиции машин Т1 и Т2

Р Т 2 ° Т 1:	А	0	1
	q_{ij}		
	q	q_{12}	q_{11}
	q	q_{13}	q_{11}
	q	q_{21}	-
	q	q_{22}	q_{21}
	q	q_{23}	q_{22}
	q	q_{20}	q_{23}

Применяя эту программу к слову $u = 1^2 0 1^3 0 1^2$, получим заключительную конфигурацию $K_0 = \dots 01101110110\dots$, совпадающую с начальной конфигурацией (всего 18 конфигураций).

Ответ. $T_2 \circ T_1 (u) = u$.

2.14. Машины Тьюринга Т1 и Т2 заданы своими программами Р1 и Р2. Построить композицию $T_2 \circ T_1$ этих машин (по паре состояний (q_{10}, q_{21})) и найти результат применения композиции $T_2 \circ T_1$ к слову u (q_{20} заключительное состояние машины Т2).

Таблица 2.15

Программа P₁

	$\begin{matrix} \backslash A \\ q' / \sim \sim \sim \backslash \end{matrix}$	0	1
P ₁ :	q ₁₁	q ₁₂ 0 r	q ₁₁ 1 r
	q ₁₂	q ₁₃ 0 r	q ₁₁ 1 r
	q ₁₃	q ₁₀ 1 l	-

а) $u = 120\ 1\ 3012$; б) $u = 120\ 12\ 02\ 12$.

Таблица 2.16

Программа P₂

	$\begin{matrix} \backslash A' A \\ q' / \sim \sim \sim \backslash \end{matrix}$	0	1
P ₂ :	q ₂₁	q ₂₂ 0 l	q ₂₁ 1 l
	q ₂₂	q ₂₃ 0 l	q ₂₂ 1 l
	q ₂₃	q ₂₀ 0 l	q ₂₃ 1 l

2.15. Пусть машины T₁, T₂ и T₃ заданы соответственно программами P₁, P₂ и P₃. Предположим, что множества состояний Q₁, Q₂ и Q₃ этих машин попарно не пересекаются. Пусть q₁ 0 и q₁ 0 какие-либо различные заключительные состояния машины T₁. Пусть машина T

= T(T₁, (q'₀, q₂₁), T₂, (q'₀, q₃₁), T₃) — разветвление машин T₂ и T₃, управляемое машиной T₁. Найти результат применения машины T к слову u (q₂₀ заключительное состояние машины T₂, а q₃₀ заключительное состояние машины T₃).

Таблица 2.17

Таблица 2.18

	$\begin{matrix} \backslash A \\ q' / \sim \sim \sim \backslash \end{matrix}$	0	1
P ₁ :	q ₁₁	q ₁₂ 0 r	q ₁₁ 1 r
	q ₁₂	q ₁₀ 0 l	q ₁₃ 1 r
	q ₁₃	q ₁₀ 0 r	q ₁₃ 1 r

	$\begin{matrix} \backslash A \\ q' / \sim \sim \sim \backslash \end{matrix}$	0	1
P ₂ :	q ₂₁	q ₂₂ 0 l	q ₂₁ 1 l
	q ₂₂	q ₂₀ 0 r	q ₂₂ 1 l

Таблица 2. 19

	$\begin{matrix} \backslash A \\ q' / \sim \sim \sim \backslash \end{matrix}$	0	1
3:	q ₁	q ₃₂	q ₁
	q ₁	q ₃₀	q ₁

а) $u = 1\ x0^21\ (x > 1)$; б) $u = 1\ x01\ y0101\ z\ (x > 1, y > 1, z > 1)$.

Таблица 2. 20

Программа машины T

A	0	1
q ₀		
P: q ₁	q ₀ 0 r	а) $u = 1\ x, 1 = 1, x > 1$; а) $u = 1\ x, 1 = 1, x > 1$;
q ₂	q ₀ 0 r	q ₃₀ r
q ₃	q ₄ 0 r	q ₃₁ r
q ₄	q ₅ 1 l	q ₄₁ r
q ₅	q ₆ 0 l	q ₅₁ l
q ₆	q ₀ 0 r	q ₆₁ l

Найти результат применения итерации машины T (по паре состояний (q₀, q_i)) к слову u (заключительными состояниями являются q₀ и q₀). Программа P машины T задана таблицей 2.20.

Решение а). Заменяя в программе P q₀ на q₁, получим программу

P' итерации машины T:

Таблица 2. 21

Программа итерации машины T

A	0	1
q 1	q 0 0 r	q 20 r
q 2	q 0 ⁰ r	Q 30 r
q 3	q 40r	q 31 r
q 4	q 511	q 41 r
q 5	q 601	q 511
q 6	q 10 r	q 611

Опишем алгоритм, соответствующий программе P' . Головка, находясь в состояниях q1 или q2, двигаясь вправо, заменяет 1 символами 0, а машина переходит в состояние q 3. Далее, головка, двигаясь вправо при состоянии q3, сохраняет 1, а при достижении 0 переходит в состояние q4. При состоянии q4 0 заменяется единицей, и машина переходит в состояние q5. Головка при состоянии машины q5 перемещается влево, сохраняя 1, а при достижении 0 машина переходит в состояние q6. При состоянии q6 головка движется влево, сохраняя 1, а, дойдя до пустого символа, машина переходит в состояние q1. На этом завершается первая итерация. В результате этой итерации слева слова будут две 1 заменены пустыми символами, а справа слова будет добавлена 1. Затем этот процесс повторяется до тех пор, пока половина единиц не будет заменена пустыми символами.

Результатом работы машины T' будет слово 1 x, т.е. $T(12^x) = 1 x$.

Продemonстрируем работу этой программы при $x = 3$.

	...0111111	K	00111	
	q	1	K	q3
1·	00111110	8·	00111	
	q2	K	q4	
2·	..0011110..	9·	001111010	
	q3	K	q5	
4·	..0011110..	10·	..0111	
	q3	K	q6	
5·	..0011110..	11·	..0111	
	q3	K	q6	
6·	..0011110..	12·	..0111	
	q3		q6	
13·	..01111010..			
	q6			
14·	..0111101			
	q6			
15·	0111101			
	q1			

Далее процесс продолжается по аналогии до заключительной конфигурации q_0' :

$K_0 = \wedge 01110 \wedge q_0' \text{ stop}$

Заметим, что в случае б) $T'(u) = 1^x$. Итерация машины Тьюринга, как видно из решенных задач, предназначена для выполнения циклов.

2.16. Найти результат применения машины

$T = T(T_1(q_0', q_{21}), T_2(q_0', q_{31}), T_3)$ к слову u (q_{20} заключительное состояние

машины T_2 , а q_{30} заключительное машины T_3).

Таблица 2. 22

состояние	Программа машины T_1		2.24
	A		
qii			
q		Программа	
q		машины T_3	
q		Р	
3:			

Таблица 2. 23

Программа машины T_2	
A	q _{v'}

а) $u = 1 x021$

($x > 1$); б) $u = 1x01y0101z$ ($x > 1, y > 1, z > 1$).

Решение а). Согласно определению разветвления машин Тьюринга составим соответствующую программу Р:

Таблица 2. 25

Программа разветвления машин

A	0	1
q _{ii}		
q	q 12	q 11 1 r
q	q 21	q 13 1 r
q	q 31	q 13 1 r
q	q22	q 21 1 l
q	q20	q 22 1 l
q	q32	q 22 1 l
q	q 30	q 31 1 r

Нетрудно видеть, что $T(u) = u$.

В случае б) $T(u) = 1^x0101^y01^z01$.

2.17. Построить машину Тьюринга, реализующую сложение двух целых неотрицательных чисел, то есть вычисляет значение функции $f(x, y) = x + y$.

Решение. Будем натуральные числа изображать в так называемом унарном коде с помощью символа |, $n \in \mathbb{N}$, что не нарушает общности рассуждений.

Например, число 710 будет изображаться в виде |||||, при этом в каждой ячейке будет находиться только один символ|, причем данное натуральное число будет изображаться символами |, записанными подряд без пропусков ячеек.

Введем алфавит $A = \{a, 0, |, +\}$, где a обозначает пустой символ, а символ + знак сложения. Например, сумма чисел 4 и 3 на ленте будет выглядеть так, как показано на рис.2.3:

IIa o I | I | I | I | I + I | I | I I I a o I - П

q1

Рисунок 2.3. Изображение на ленте суммы чисел 4 и 3Идея решения задачи заключается в том, что управляющая головка, обозревая самый левый символ |, заменяет его пустым символом ао, перемещается вправо до тех пор, пока не достигнет символа +. Затем она заменяет символ + символом | и возвращается к первому символу | слева и переходит в заключительное состояние q0. Поэтому внутренний алфавит Q состоит из 4 состояний q0, q1, q2, и q3. Тогда один из вариантов программы Р искомой машины Тьюринга может быть таким:

Таблица 2. 26

Программа решения задачи 2.18

A q _i	a 0	+	
q 1	-	q 0 a 0	q 2 a 0 r
q 2	-	q 3 l	q 2 r
q 3	q 0 a 0	-	q 3 l

В системе команд программы Р перечислены не все сочетания состояний машины и символов ленты: опущены те из них, которые при стандартной начальной конфигурации никогда не встретятся. Кроме того, команда $q 1 + \rightarrow q 0a0r$ введена для случая, когда $x=0$ и исходное слово имеет вид $+|^y$ (начальное состояние q1 будет находиться под символом +).

Диаграмма переходов для данной машины имеет вид, показанный на рис. 2.4:

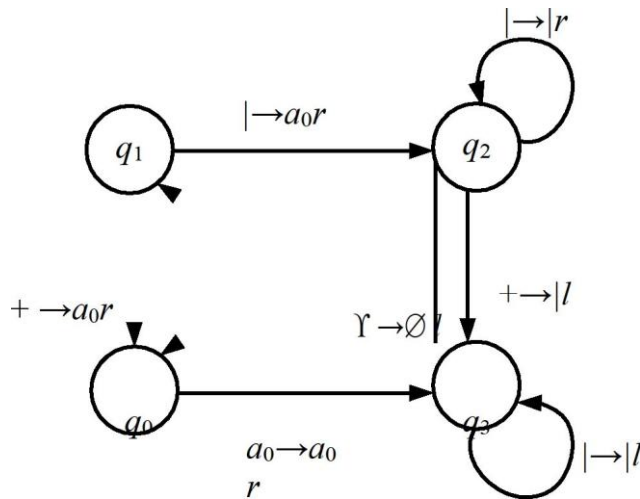


Рисунок 2.4. Диаграмма переходов машины к задаче 2.18. Запишем, например, последовательность конфигураций для тестового примера 4+3:

		...	a	0	I
		q			
2:	1	q	I	11	
3:	2	q	a		
4:	2	q	a	0.	
5:	2	q	a		
6:	5	q	a		
7:	3	q	a	0.	
8:	3	q	a	0.	
9:	3	q	a	0.	
0:	3	q	a		

q0

При решении задачи была построена «зацикленная» машина Тьюринга Т, в которой заключительное состояние совмещено с начальным состоянием машины Т. Цикл состоит из состояний q1, q2, q3. Сумма, полученная на очередной итерации, является первым слагаемым следующей итерации.

2.18. Построить машину Тьюринга для реализации алгоритма вычисления значения функции $f(x) = 2x$, $x \in \mathbb{N}$.

Решение. Для решения поставленной задачи работу машины Тьюринга можно представить следующим образом: во внешний алфавит машины А включим четыре символа {a0, I, a, v}. Символ I обозначает единицу, a0 пустой символ, а a и v будут играть роль промежуточных переменных. Число x записано на ленте, как и в предыдущей задаче 2.18, в виде набора x символов |. Пусть, например, x=4, тогда конфигурация

... a 0 I I I I a 0...

q1

будет начальной конфигурацией. На первом такте работы машины осуществляется сдвиг головки вправо, и машина остается в состоянии q1.

При этом машина вписывает в пустую клетку символ a, сдвигается влево и переходит в состояние q2. В состоянии q2 машина на каждом такте сдвигается влево, оставляя без изменений символы a и v. При достижении символа | в состоянии q2 символ | заменяется символом v, осуществляется сдвиг влево и переход в состояние q3. Если при

этом управляющая головка обозревает пустую ячейку, то машина переходит в состояние q4. Если же управляющая головка, находясь в состоянии q3, обозревает символ |, то, оставаясь на месте, машина переходит в состояние q1. В состоянии q1 головка сдвигается вправо до первой пустой клетки, оставляя при этом на месте символы |, а, в. Достигнув пустой ячейки в состоянии q1, машина вписывает в нее символ а и далее процесс повторяется. Машина будет работать до тех пор, пока все символы | не будут заменены символами а (столько же символов а будет вписано в пустые клетки). При этом машина перейдет в состояние q4. В состоянии q4 головка движется вправо, заменяя символы а и в на символ|. Достигнув пустой ячейки в состоянии q4, машина переходит в заключительное состояние q0.

Проиллюстрируем описанный алгоритм на примере $x=4$, то есть

выпише	игура	вва
м все кон	ции: а0.	K13: а0 аа0.
К	q1	K13K q1
2: .. а 0	а0.	18: а0 ввааа0.
К	q1	q1
3: .. а	а а	K19: а0 вваааа0.

Таблица 2.27

Программа машины T1		
$\backslash A$ qij^m	0	1
$P_1: q_{11}$	$q_{10} 0 l$	$q_{12} 1 r$
q_{12}	$q_{13} 0 r$	$q_{13} 1 r$
q_{13}	$q_{11} 0 r$	$q_{11} 0 r$
К		q2
4: а0	ваа0.	3
К		q1
5: а0	ваа0.	
К		q1
6K8: а0	ваа0.	
К		q1
9: а0	вааа0.	
К		q2
10K11: а0	вааа0.	
К		q2
12: а0	ввааа0.	

Таблица 2.28

Программа машины T2		
A q_{ij}^m	0	1
q_{21}	$q_{22} 1 l$	$q_{22} 1 l$
q_{22}	$q_{20} 0 r$	$q_{21} 0 l$
К20K		q2
23: а0	вваааа0.	q2
К24: а0	ввваааа0.	q3
К25: а0	ввваааа0.	q1
К26K		q1
32: а0	ввваааа0.	
К35: а0	ввваааа0.	q2
К0: а0	а0	q0
		stop

$x = 0,$

2.19. Построить машины Тьюринга для вычисления значений функций: $f(x, y) = x \cdot y$;

$f(x) = x-1$ I^0 , если

$|x-1$, если $x > 0$;

2.19.3. $f(x) = \text{sq}x = j^0$, если $x = 0$, $[1$, если $x \neq 0$;

2.19.4. $f(x, y) = x \cdot y$;

2.19.5. $f(x, y) = x^y$.

2.19.6. Построить композицию $T_2 \circ T_1$ машин Тьюринга T_1 и T_2 по паре состояний (q_{10}, q_{21}) (q_{20} заключительное состояние машины T_2). Программу машины $T_2 \circ T_1$ задать системой команд и диаграммой переходов. Применить композицию машин T_1 и T_2 к заданному слову u . Выписать последовательность конфигураций от начальной конфигурации до результирующей конфигурации. $u = 1^4 0^2 1^3 0^1 2^1$;

- 2.19.7. $u = 1^2 0 1^3$;
 2.19.8. $u = 10^2 1^2 0 1^4$;
 2.19.9. $u = 1^3 0 1^3 0 1^3$;
 2.19.10. $u = 10^3 1^3 0 1$;
 2.19.11. $u = 1^4 0 1^2 0 1^2$;
 2.19.12. $u = 10^4 10^3 1$;
 2.19.13. $u = 10 1^{20} 3 1^2$;
 2.21.14. $u = 1^{50} 1^2$;
 2.21.15. $u = 1^{20} 4 1^3 0 1$;
 2.21.16. $u = 1^5 1^2 0 1$;
 2.21.17. $u = 1^4 0^4 1^2$;
 2.21.18. $u = 1^{20} 3 1^4$;
 2.21.19. $u = 1^3 0^2 1^4$;
 2.21.20. $u = 1^6 0 1^2$;
 2.21.21. $u = 1^4 0^2 1$;
 2.21.22. $u = 1^{30} 2 1^2 0 1$;
 2.21.23. $u = 1^{30} 3 1^3 0 1$;
 2.21.24. $u = 1^3 0 10^3 1$;
 2.21.25. $u = 1^{20} 3 1^2 0 1$; $u = 10 1^{40} 2$
 2.21.26. $u = 1^{20} 2 1^2 0^2 1$;
 2.21.27. $u = 1^3 0 1^3 0 1$;
 2.21.28. $u = 1^{30} 3 1^2 0^2 1$;
 2.21.29. $u = 10 10^2 1^4$;

2.22. $u = 1^{60} 2 1$. Построить машину Тьюринга, которая на вход получает натуральное число m и добавляет к нему 1 в двоичной записи. Точнее, изначально на ленте стоит символ $\$$, за которым стоит число m в двоичном коде. Вначале головка в состоянии q_1 обозревает знак $\$$. Машина должна остановиться с двоичной записью числа $m+1$ на ленте, обозревая ее крайний слева символ и находясь в состоянии q_i . При необходимости можно удалить символ $\$$, например, $q_1 \$ 10011 - \$ q_i 10100 . I$

а) указать переходы машины Тьюринга и объяснить назначение каждого состояния;

б) указать последовательность конфигураций машины Тьюринга при обработке входа $\$ 111$.

2.23. Обычные операции в программах машин Тьюринга используют операцию сдвига головки вдоль ленты на одну позицию. Предположим, что в текущей позиции головки нужно создать дополнительную клетку, в которую можно было бы записать некоторый символ. Однако изменять ленту таким способом нельзя. Придется сдвинуть содержимое ленты справа от головки на одну клетку вправо и затем вернуться к текущей позиции. Показать, как выполнить данную операцию.

Указание. Следует зарезервировать специальный символ для отметки позиции, в которую нужно вернуться.

13. НОРМАЛЬНЫЕ АЛГОРИТМЫ МАРКОВА

План лекции

1. Основные понятия.
2. Композиция, итерация и разветвление нормальных алгоритмов.

Эквивалентность тезиса Черча и принципа нормализации Маркова.

Наряду с рекурсивными функциями и машинами Тьюринга нормальные алгоритмы, введенные А. А. Марковым в 1947 г., получили известность в качестве одного из наиболее удобных уточнений общего интуитивного представления об алгоритме.

Возникшие первоначально в связи с алгебраической проблематикой нормальные алгоритмы оказались удобным рабочим инструментом во многих исследованиях, требующих точного понятия алгоритма, особенно тогда, когда основные объекты рассмотрения имеют неарифметическую природу и допускают удобное представление в виде слов в некоторых алфавитах (такова, например, ситуация в конструктивном анализе).

Всякий нормальный алгоритм F , являясь алгоритмом в некотором алфавите A , порождает в нем детерминированный процесс переработки слов из множества A^* . Указание алфавита A входит в определение нормального алфавита F в качестве обязательной составной части, и в рассматриваемой ситуации про алгоритм F говорят, что он является нормальным алгоритмом в алфавите A . Любой нормальный алгоритм в фиксированном алфавите A вполне определяется указанием его схемы упорядоченного конечного списка формул подстановки в A . Каждая такая формула по существу представляет собой упорядоченную пару (u, v) слов из множества A^* .

Слово u называется левой частью этой формулы, а v ее правой частью. Среди формул данной схемы некоторые формулы выделяются специально и объявляются заключительными (ср. с машиной Тьюринга).

Имеет место эквивалентность понятий частичной вычислимости по Маркову и частичной рекурсивности по Черчу. Тезис Черча утверждает в свою очередь, что понятие вычислимости эквивалентно понятию рекурсивности (а в расширенной формулировке что понятие частично эффективной вычислимости эквивалентно понятию частичной рекурсивности). А.А. Марков в терминах алгоритмов сформулировал соответствующий принцип, названный им принципом нормализации: всякий алгоритм F в алфавите A эквивалентен относительно A некоторому нормальному алгоритму над алфавитом A .

Уточнение понятия алгоритма, осуществленное на основе понятия алгоритма Маркова, оказывается эквивалентным другим уточнениям рекурсивные функции, машина Тьюринга и др. Вследствие этого принцип нормализации оказывается равносильным тезису Черча-Тьюринга, предлагающему считать понятие частично рекурсивной функции адекватным уточнением понятия вычислимой арифметической функции.

3. 1. Основные понятия

При изучении нормальных алгоритмов Маркова, впрочем, как и при изучении других способов уточнения интуитивного понятия алгоритма, используется некоторый формальный язык, который задается так называемыми регулярными выражениями, и описывается с помощью своей грамматики. Всякий язык начинается с алфавита. Как и ранее, алфавитом будем называть всякое непустое конечное множество символов, а сами символы алфавита будем называть буквами. Словом в данном алфавите называется всякая конечная последовательность букв этого алфавита. В дальнейшем алгоритмы будем обозначать прописными буквами, выделенными полужирным шрифтом, алфавиты прописными обычными буквами, а символы алфавита строчными буквами с индексами или без индексов. Множество всех слов над алфавитом A , как и раньше, будем обозначать через A^* . Пустая последовательность букв называется пустым словом и обозначается символом Λ (по Мендельсону Э.), причем символ $\Lambda \in A^*$ воспринимается как равноправное слово из алфавита A^* .

Если заданы два слова $u = u_1 u_2 \dots u_k$ и $v = v_1 v_2 \dots v_l$, где

1 2 k 1 2 m

$u_1, u_2, \dots, u_j, v_1, v_2, \dots, v_j \in A$, то через uv обозначим объединение (конкатенацию) этих слов: $uv = u_1 u_2 \dots u_i v_1 v_2 \dots v_j$. В частности, $u1 = 1u = u$.

Кроме того, имеет место свойство ассоциативности для конкатенации трех (и более) слов $u, v, w \in A^*$: $(u v)w = u(v w)$.

Определение 3.1. Алфавит A называется расширением алфавита B , если $B \subset A$.

Очевидно, что в этом случае всякое слово в алфавите B является словом и в алфавите A .

Определение 3.2. Алгоритмом в алфавите A называется всякая эффективно вычислимая функция, областью определения и множеством значений которой служат какие-нибудь подмножества множества A^* (возможно совпадающие).

Определение 3.3. Пусть $u \in A^*$. Говорят, что алгоритм F применим к слову u , если u содержится в области определения алгоритма F . Если алфавит B является расширением алфавита A , то всякий алгоритм в алфавите B называется алгоритмом над алфавитом A .

Большинство известных алгоритмов можно разбить на простейшие шаги. Следуя А. А. Маркову, в качестве элементарной операции, на базе которой строятся алгоритмы, выделим подстановку одного слова вместо другого. Если $u, v \in A^*$, то выражения

$$u \wedge v \text{ и } u \wedge \bullet v$$

называются формулами подстановки в алфавите A . При этом предполагается, что символы « $\wedge$ » и « $\bullet$ » не являются буквами алфавита A , а каждое из слов u или v может быть и пустым словом. Формула подстановки $u \wedge v$ называется простой, а формула подстановки $u \wedge \bullet v$ — заключительной.

Определение 3.4. Пусть $u \wedge (0v)$ обозначает одну из формул подстановки $u \wedge v$ или $u \wedge \bullet v$. Конечный список формул подстановки в алфавите A

$$u_1 \wedge (\bullet) v_1$$

$$\dots u_2 \wedge (\bullet) v_2$$

$$\dots u_r \wedge (\bullet) v_r$$

называется схемой алгоритма и порождает нормальный алгоритм Маркова в алфавите A . Условимся считать, что слово t входит в слово v , если $\exists w \exists z ((w, z \in A^*) \& (v = wtz))$.

Пусть $u \in A^*$. Возможны два случая:

1. Ни одно из слов $u_1, u_2, \dots, u_r$ не входит в слово u (обозначается: $F: u \circ$).

2. Среди слов $u_1, u_2, \dots, u_r$ существуют такие слова, которые входят в слово u .

Пусть m — наименьший номер такой, что $1 < m < r$ и слово u_m входит в слово u , а p слово, которое получается, если самоелевое вхождение слова u_m в слово u заменить словом v_m . Тот факт, что u и p находятся в описанном отношении, коротко запишем в виде

$$F: u \rightarrow p, \quad (*)$$

если формула подстановки $u \wedge (\bullet) v_m$ простая, или в виде

$$F: u \rightarrow p \bullet, \quad (**)$$

если формула $u \wedge (\bullet)$ заключительная.

В случае $(*)$ говорят, что алгоритм F просто переводит слово u в слово p ; в случае $(**)$ говорят, что алгоритм F заключительно переводит слово u в слово p . Пусть далее, запись $F: u \rightarrow p$ означает, что существует такая последовательность $p_0, p_1, \dots, p_k$ слов из A^* , что $u = p_0$, $p = p_k$, $F: p_j \rightarrow p_{j+1}$ для $j = 0, 1, \dots, k-2$ и либо $F: p_{k-1} \rightarrow p_k$, либо $F: p_{k-1} \rightarrow p_k \bullet$ (в последнем случае вместо $F: u \rightarrow p$ пишут $F: u \rightarrow p \bullet$). Положим теперь $F(u) = p$ тогда и только тогда, когда-либо $F: u \rightarrow p$, либо $F: u \rightarrow p$ и $F: p \rightarrow \wedge$.

Алгоритм, определенный таким образом, называется нормальным алгоритмом или алгоритмом Маркова в алфавите A .

Работа алгоритма F может быть описана следующим образом. Пусть дано слово $u \in A^*$. Находим первую в схеме алгоритма F формулу подстановки $u \wedge (\bullet) v_m$ такую, что слово u_m входит в слово u . Совершаем подстановку слова v_m вместо самого левого

вхождения слова it в слово i . Пусть p_1 результат такой подстановки. Если $it \wedge (\bullet) V_m$ — заключительная формула подстановки, то работа алгоритма заканчивается и его значением является слово p_1 . Если формула подстановки $it \wedge (\bullet) V_m$ — простая, то к слову p_1 применим тот же поиск, который был только что применен к слову i и т.д. Если на конечном этапе будет получено такое слово p_i , что $F: p_i \wedge$, то есть ни одно из слов $i_1, i_2, \dots, i_n$ не входит в слово p_i , то работа алгоритма заканчивается и p_i будет его значением. Если описанный процесс на конечном этапе не заканчивается, то говорят, что алгоритм F не применим к слову i .

Пример 3.1. Пусть $A = \{b, c\}$. Рассмотрим схему

$b \wedge \bullet L$

$c \wedge c$.

Определяемый этой схемой нормальный алгоритм F перерабатывает всякое слово i SA^* , содержащее хотя бы одно вхождение буквы b в слово, которое получается вычеркиванием b и самого левого вхождения буквы b . Действительно, всякая буква c , находящаяся в слове левее самой левой буквы b , простой подстановкой $c \wedge c$ переводится в букву c , а самая левая буква b заключительной подстановкой переводится в пустое слово. Например, если $i = scbbcs$, то $i \wedge \bullet V$, где $V = scbcs$. Пустое слово алгоритм F перерабатывает в себя. Алгоритм F не применим к непустым словам, не содержащим вхождения буквы b . Действительно, если слово i содержит только буквы c , то простой подстановкой $c \wedge c$ оно будет перерабатываться в себя, но тогда всегда $i \wedge$ и i мы не приходим к заключительной подстановке, то есть процесс будет продолжаться бесконечно.

Пример 3.2. Пусть $A = \{a_0, a_1, \dots, a_n\}$. Рассмотрим схему

$V_i (a_i \wedge L) (a_i SA)$.

Эта схема определяет нормальный алгоритм F , перерабатывающий всякое слово из A^* в пустое слово. Например,

$F: a_1 a_2 a_1 a_3 a_0 a_1 a_2 a_1 a_3 a_2 a_1 a_3 a_2 a_3 a_3 a_3 L$, и, наконец, $F: L$ о. Следовательно, $F(a_1 a_2 a_1 a_3 a_0) = L$.

Пример 3.3. Пусть A — алфавит, содержащий букву a_1 , которую мы сокращенно обозначим цифрой 1. Для всякого натурального числа n определим по индукции $o = 1$ и $n + 1 = n1$, например, $1 = 11$, $2 = 111$ и т.д. Слова n называются цифрами. Определим теперь нормальный алгоритм F схемой $L \wedge \bullet 1$. Для любого слова i в алфавите A получим

$F(u) = 1u$, так как всякое слово u начинается с вхождения пустого слова L либо $u = L u$. В частности, $F(n) = n + 1$ для любого натурального числа n .

Замечание. Различие между понятием нормального алгоритма в алфавите A и понятием нормального алгоритма над алфавитом A является очень важным. Всякий нормальный алгоритм в алфавите A использует только буквы из A , тогда как нормальный алгоритм над алфавитом A может использовать и некоторые дополнительные буквы, не входящие в алфавит A . Всякий нормальный алгоритм в алфавите A является также нормальным алгоритмом над алфавитом A , но существуют, вообще говоря, алгоритмы в алфавите A , которые определяются нормальными алгоритмами над алфавитом A , но сами не являются нормальными алгоритмами в алфавите A .

3. 2. Композиция, итерация и разветвление нормальных алгоритмов. Эквивалентность тезиса Черча и принципа нормализации Маркова

Пусть A и B нормальные алгоритмы ir — слово. Запись $A(p) = B(p)$ будет означать, что алгоритмы A и B оба неприменимы к слову p , либо оба они к нему применимы и при этом $A(p) = B(p)$. Вообще, если C и D какие-нибудь выражения, то запись $C = D$ будет в дальнейшем означать, что либо оба эти выражения не определены, либо оба они определены и обозначают один и тот же объект.

Определение 3.5. Два алгоритма A и B над алфавитом A называются вполне эквивалентными относительно алфавита A , если для любого слова p в алфавите A $A(p) = B(p)$. Те же алгоритмы A и B называются эквивалентными относительно алфавита A , если

каждый раз, когда есть слово в алфавите A и хотя бы одно из слов $A(p)$ и $B(p)$ определено и тоже является словом в алфавите A .

Пусть $N = \{1\}$ и $M = \{1, *\}$ алфавиты, N множество всех натуральных чисел, а ϕ частично вычислимая арифметическая функция от n аргументов, то есть $\phi : N^n \rightarrow N$. Всякое натуральное число m можно представить цифрой $m = m_1$ (см. пример 3.3) как слово в алфавите N . Поставим в соответствие каждому вектору $(k_1, k_2, \dots, k_n)$,

где $k_1, k_2, \dots, k_n$ натуральные числа, слово $k_1 * k_2 * \dots * k_n$ в алфавите

M , которое обозначим через $(k_1, k_2, \dots, k_n)$. Например, запись $(4, 2, 3)$ означает слово $11111*111*1111$. Обозначим через B_ϕ соответствующий алгоритм в алфавите M , то есть такой алгоритм, что $B_\phi((k_1, k_2, \dots, k_n)) = \phi(k_1, k_2, \dots, k_n)$ каждый раз, когда хотя бы одна из частей этого равенства определена. Предположим также, что алгоритм B_ϕ не применим к словам, отличающимся от слов вида $(k_1, k_2, \dots, k_n)$.

Определение 3.6. Функция ϕ называется частично вычислимой по Маркову, если существует нормальный алгоритм A над алфавитом M , вполне эквивалентный алгоритму B_ϕ относительно алфавита M . Если функция ϕ всюду определена и является частично вычислимой по Маркову, то она называется вычислимой по Маркову.

Определение 3.7. Нормальный алгоритм называется замкнутым, если его схема содержит формулу подстановки вида $L \wedge \blacksquare$ и, где i — произвольное слово.

Ясно, что в работе замкнутого алгоритма возможен лишь заключительный обрыв, то есть обрыв в результате применения заключительной подстановки. Пусть A — произвольный алгоритм. Обозначим через $A \blacksquare$ нормальный алгоритм, схема которого получается из схемы алгоритма A добавлением новой формулы подстановки $L \wedge \blacksquare$ к ней в конце. Нормальный алгоритм $A \blacksquare$ замкнут и вполне эквивалентен алгоритму A относительно алфавита алгоритма A .

Пусть A и B — нормальные алгоритмы в алфавите A . Поставим в соответствие каждой букве $b \in A$, новую букву $\bar{b}$, которую назовем

двойником буквы b . Пусть $\bar{A}$ алфавит, состоящий из всех двойников букв алфавита A . Выберем еще какие-нибудь две буквы a и v , не принадлежащие объединению $A \cup \bar{A}$ алфавитов A и $\bar{A}$. Обозначим через SA схему, полученную из схемы нормального алгоритма $A \blacksquare$ заменой в ней точки в каждой заключительной формуле подстановки буквой a , и обозначим через SB схему, которая получается путем замены в схеме алгоритма $B \blacksquare$ всех букв алфавита A их двойниками, всех точек буквами v с последующей заменой всех формул подстановки вида $L \wedge i$ и $L \wedge \bar{i}$ и соответственно формулами подстановки

$a \rightarrow au$ и $a \rightarrow aftu$. Пусть $a, b \in A$. Рассмотрим схему:

	$aa \rightarrow a$
	$aa \rightarrow a$
	$ab \rightarrow$
,	
	$av \rightarrow va$
	$va \rightarrow va$
(2 1)	
	$ab \rightarrow$
	$av \rightarrow \blacksquare$
	S
	B
	^sA

Определение 3.8. Алгоритм F над алфавитом A , определяемый схемой (3.1), называется композицией нормальных алгоритмов A и B и обозначается символом $B \circ A$. В общем случае под композицией $A_1 \circ A_2 \circ \dots \circ A_n$ нормальных алгоритмов $A_1, A_2, \dots, A_n$ — $1, A_n$ понимают их последовательное выполнение: $A_n (\dots A_3 (A_2 (A_1)) \dots)$ (ср. с

Доказательство. Существует нормальный алгоритм Q_1 над алфавитом $A_0\{a\}$, перерабатывающий пустое слово L в a и всякое непустое слово в алфавите $A_0\{a\}$ в L . Такой алгоритм может быть задан, например, следующей схемой для $a \in A_0\{a\}$:

$$\begin{bmatrix} a \wedge^b M \rightarrow v \\ v \wedge \blacksquare L \end{bmatrix}$$

$$L \wedge \blacksquare a,$$

где v — буква, не принадлежащая алфавиту $A_0\{a\}$.

Пусть $Q_2 = Q_1 \circ C$. Для любого слова p в алфавите A , если $C(p) = L$, то $Q_2(p) = a$, а если $C(p) \neq L$, то $Q_2(p) = L$. Пусть I тождественный нормальный алгоритм в алфавите A со схемой $L \wedge \blacksquare L$ и пусть D есть соединение алгоритмов Q_2 и I . Тогда если $C(p) = L$, то $D(p) = a$, а если $C(p) \neq L$, то $D(p) = p$. Этим завершается доказательство теоремы 3.1.

Теорема 3.2. Пусть A и C нормальные алгоритмы, A объединение их алфавитов, и A_1 и C_1 формальные распространения соответственно алгоритмов A и C на алфавит A (см. определение 3.10). Тогда существует нормальный алгоритм B над алфавитом A , являющийся итерацией (см. определение 3.9) алгоритма A_1 , управляемой алгоритмом C_1 .

Доказательство. Теорему 3.2 достаточно доказать для случая, когда алфавиты алгоритмов A и C совпадают и когда, следовательно, $A_1 = A$ и $C_1 = C$. Пусть буква a не входит в алфавит A . По теореме 3.1 существует такой нормальный алгоритм D над алфавитом $B = A_0\{a\}$ такой, что $\Gamma_a p$ если P есть слово в алфавите A и $C(p) = L$, $D(P) = I p$ если P есть слово в алфавите A и $C(p) \neq L$.

$I p$, если

Пусть $F = D \circ A$. Алгоритм F является нормальным алгоритмом в некотором расширении F алфавита B . Пусть буква v не входит в алфавит F . Рассмотрим следующую схему ($\wedge^v F$):

$$\begin{bmatrix} (v \rightarrow v) a \rightarrow \blacksquare a h \rightarrow L \\ I Q F \end{bmatrix},$$

где Q схема, полученная из схемы для алгоритма F путем замены

в ней всех точек буквой v . Эта схема определяет некоторый нормальный алгоритм U , причем $U(p) = q$ тогда и только тогда, когда существует такая последовательность слов $p_0, p_1, \dots, p_n$, что $p = p_0$, $q = p_n$, $p_i = F(p_{i-1})$ ($0 < i < n$) и p_n единственное в этой последовательности слово, начинающееся с буквы a . Пусть L алгоритм, проектирующий алфавит F на алфавит $F \setminus \{a\}$ (то есть стирающий все вхождения буквы a ; см. определение 3.13). Теперь легко убедиться, что искомым является алгоритм $B = L \circ U$.

Определение 3.14. Пусть A, B, C нормальные алгоритмы и A объединение их алфавитов. Алгоритм F над алфавитом A называется разветвлением алгоритмов A и B , управляемым алгоритмом C , если

$$\Gamma, \sim \Gamma B(p), \text{ если } p \text{ слово в алфавите } A \text{ и } C(p) = L,$$

$$F(p) = I$$

$$[A(p), \text{ если } p \text{ слово в алфавите } A \text{ и } C(p) \neq L]$$

и алгоритм F применим к тем и только тем словам в алфавите A , к которым применим алгоритм C .

Можно доказать, что алгоритм F существует и является нормальным алгоритмом.

С помощью неэлементарных рассуждений можно установить эквивалентность тезиса Черча и принципа нормализации А.А. Маркова. Доказательство этого утверждения основывается на некоторых важных самих по себе утверждениях, которые мы сформулируем без доказательства.

Теорема 3.3. Если алгоритм A над алфавитом A таков, что функция f_A частично рекурсивна, то существует нормальный алгоритм над алфавитом A , вполне эквивалентный алгоритму A относительно алфавита A (см. определение 3.5).

Теорема 3.4. Если алгоритм A нормальный алгоритм над алфавитом A , то f_A частично рекурсивная функция. Если, кроме того, алгоритм A применим к любому слову в

алфавите A , то функция f_A рекурсивна.

Теорема 3.5. Если данная частичная функция f является частично вычислимой по Маркову, то она является частично рекурсивной по Черчу. Если же f функция, вычислимая по Маркову, то она рекурсивна по Черчу.

Теорема 3.6. Всякая частично рекурсивная функция по Черчу является частично вычислимой по Маркову функцией. Всякая рекурсивная функция по Черчу является вычислимой по Маркову функцией.

Теоремы 3.5 и 3.6 устанавливают эквивалентность понятий частичной вычислимости (вычислимости) по Маркову и частичной рекурсивности (рекурсивности) по Черчу. Теперь можно установить эквивалентность тезиса Черча и принципа нормализации А.А. Маркова.

Действительно, примем сначала тезис Черча. Пусть A алгоритм в алфавите A . Соответствующая функция f_A является частичной эффективно вычислимой функцией. Тогда, в силу тезиса Черча, f_A является частично рекурсивной функцией и, следовательно, алгоритм A на основании теоремы 3.3 вполне эквивалентен относительно алфавита A некоторому нормальному алгоритму над алфавитом A . Таким образом, если верен тезис Черча, то верен и принцип нормализации. Обратно, пусть верен принцип нормализации, и пусть f произвольная частичная эффективно вычислимая функция. Пусть, далее, B f соответствующий алгоритм в алфавите M . Согласно принципу нормализации алгоритм Bf вполне эквивалентен относительно алфавита M некоторому нормальному алгоритму над алфавитом M . Следовательно, функция f является частично вычислимой по Маркову. Но тогда по теореме 3.5 f является частично рекурсивной по Черчу, и мы вывели, тем самым, из принципа нормализации тезис Черча.

Существует также взаимосвязь между принципом нормализации и машинами Тьюринга.

Итак, подведем итоги. Три способа уточнения понятия эффективной вычислимости в терминах рекурсивных функций, машин Тьюринга и нормальных алгоритмов, по существу, эквивалентны. Машины Тьюринга можно рассматривать как некую абстрактную форму цифровых вычислительных машин. Совпадение классов функций, частично рекурсивных и вычислимых по Тьюрингу, интуитивно воспринимается как еще один факт, говорящий в пользу тезиса Черча. При этом класс вычислимых по Тьюрингу функций не изменится, если мы усложним определение машины Тьюринга, допуская, например, не одну, а несколько лент и читающих головок, или используя двумерную ленту.

ЛИТЕРАТУРА К ГЛАВЕ 3

1. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. М.: Мир, 1979.
2. Балюкевич Э.Л., Ковалева Л.Ф. Математическая логика и теория алгоритмов: учебно-практическое пособие. М.: Евразийский открытый институт, 2009. 189 с.
3. Гринченков, Д.В., Потоцкий С.И. Математическая логика и теория алгоритмов для программистов: учеб. пособие. М.: КНОРУС, 2010. 206 с.
4. Зайцева Е.В., Гурова Л.М. Математическая логика и теория алгоритмов: учебное пособие. М.: Московский государственный горный университет, 2006. 225 с.
5. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. М.: МЦНМО, 2000.
6. Крупский В.Н., Плиско В.Е. Математическая логика и теория алгоритмов: учеб. пособие для вузов. Москва: Академия, 2013. 415 с.
7. Лавров И.А., Максимова Л.Л. Задачи по теории множеств, математической логике и теории алгоритмов. М.: Физматлит, 2002. 258 с.
8. Макконнелл Дж. Анализ алгоритмов. Вводный курс. М.: Техносфера, 2002.
9. Макоха А. Н. Расширение понятия числа и измерение величин: учебное пособие. Ставрополь: СГПИ, 1991.
10. Макоха А.Н., Рябогин А.К. Математическая логика и теория алгоритмов: практикум. Ставрополь: Изд-во СГУ, 2009. 146 с.
11. Мальцев А. И. Алгоритмы и рекурсивные функции. М.: Наука, 1965.
12. Матрос Д.Ш. Поднебесова Г.Б. Теория алгоритмов: учебник. М.: БИНОМ. Лаборатория знаний, 2008. 202 с.
13. Новиков Ф. А. Дискретная математика для программистов. СПб.: Питер, 2000.
14. Судоплатов С.В., Овчинникова Е.В. Математическая логика и теория алгоритмов: учебник. Новосибирск: НГТУ, 2012. 254 с.
15. Тихомирова А.Н., Сафоненко Н.В. Практикум по теории алгоритмов: учебное пособие. М.: МИФИ, 2011. 132 с.
16. Хопкрофт Джон Э., Мотвани Раджив, Ульман Джеффри Д. Введение в теорию автоматов, языков и вычислений, 2-е изд.: М.: Издательский дом «Вильямс», 2002.