Ю. И. Журавлёв, Ю. А. Флёров, М. Н. Вялый

ДИСКРЕТНЫЙ АНАЛИЗ. ФОРМАЛЬНЫЕ СИСТЕМЫ И АЛГОРИТМЫ

Рекомендовано
Учебно-методическим объединением
высших учебных заведений Российской Федерации
по образованию в области
прикладных математики и физики
в качестве учебного пособия по направлению
«Прикладные математика и физика»

Москва 2010

УДК 510.51
ББК 22.12
Ж 91

Серия «Естественные науки. Математика. Информатика»

Ответственный секретарь серии Лобанов А.И.

Рецензенты:
Докт. физ.-мат. наук, проф. *В. К. Леонтьев* (ВЦ РАН)
Кафедра высшей математики МГУТУ

Ю. И. Журавлёв и др.
Ж 91 Дискретный анализ. Формальные системы и алгоритмы: Учебное пособие./ Ю. И. Журавлёв, Ю. А. Флёров, М. Н. Вялый – М: ООО Контакт Плюс, 2010. – 336 с.: ил.

ISBN 978–5–86567–092–1

Эта книга является учебным пособием по математической логике и теории алгоритмов. Она написана на основе материалов курса «Дискретный анализ», читаемого многие годы для студентов факультета управления и прикладной математики Московского физико-технического института.

Для студентов, специализирующихся на прикладной математике.

УДК 510.51
ББК 22.14

ISBN 978–5–86567–092–1

Оглавление

Предисловие

Эта книга написана на основе курса лекций Ю. И. Журавлёва для факультета управления и прикладной математики Московского физико-технического института. Она посвящена введению в математическую логику и теорию алгоритмов и предназначена для студентов младших курсов. Наиболее полезной она окажется для тех, кто специализируется на прикладной математике.

Авторы благодарят всех, кто помогал в подготовке этой книги. Особая благодарность — Д. Саянкину, который взял на себя нелегкий труд записи и расшифровки лекций. Предварительный вариант этой книги использовался на занятиях в МФТИ и многие ценные изменения были сделаны благодаря замечаниям студентов. Кроме того неоценимую роль в подготовке этого издания сыграла О. С. Федько. Без её вдохновляющего энтузиазма работа над книгой вряд ли была бы закончена.

Введение

В этой книге излагаются основы теории формальных систем и теории алгоритмов.

Одной из отправных точек в развитии математической логики была задача формализации математического рассуждения. Формальное представление математического рассуждения нужно, чтобы зафиксировать предельно чёткие и объективные критерии корректности рассуждения. Формализация не только позволяет доказывать средствами математики корректность рассуждения, но также строго формулировать утверждения о невозможности доказательства.

Зачем нужна такая строгая проверка корректности математических рассуждений? Ведь общеупотребительный в математике способ разобраться с рассуждением состоит в том, чтобы *понять* его. В большинстве случаев этого достаточно. Однако время от времени в математике возникают правдоподобные рассуждения, которые противоречат друг другу. Именно так и обстояли дела во второй половине 19 века, когда свободное применение разработанных Кантором средств работы с произвольными совокупностями (множествами) привело к противоречиям.

Пример В.1 (*парадокс Рассела*). Всем очевидно, что из совокупности разноцветных шаров всегда можно выбрать совокупность шаров красного цвета.

В общем случае этот приём формулируется так. Если есть множество M и некоторое свойство φ его элементов, то совокупность тех элементов множества M, которые удовлетворяют этому свойству, также образует множество. Этот способ построения новых множеств применяется очень часто. Есть даже стандартное обозначение $\{x \in M : \varphi(M)\}$ для построенного таким образом множества.

И тем не менее неограниченное применение такой конструкции приводит к противоречию.

Рассмотрим свойство *самовходимости*: «множество является своим элементом». Элементами множеств могут быть другие множества, так что это свойство не лишено смысла.

Рассмотрим множество всех множеств. Выделим в нём подмножество N, которое состоит в точности из всех несамовходимых множеств.

Если бы множество N было несамовходимым, то его нужно было бы включить в элементы N. Значит, множество N самовходимое. Но это означает, что оно не является элементом N. Получили противоречие с определением самовходимого множества.

Пример В.2 (*парадокс Берри*). Фраз русского языка, содержащих не более десяти слов, конечное количество, а натуральных чисел бесконечно много. Поэтому найдутся такие натуральные числа, которые можно задать лишь фразами, содержащими по крайней мере одиннадцать слов.

Теперь определим натуральное число B как наименьшее натуральное число, которое нельзя задать менее чем одиннадцатью словами.

Несложным подсчётом убеждаемся, что число B задано всего десятью словами. Приходим к противо-

речию с тем фактом, что в каждом подмножестве натуральных чисел существует наименьшее.

Эти и другие парадоксы стали одной из основных причин пристального внимания к возможности формализации рассуждений.

Опишем в общих чертах один из стандартных вариантов формализации.

Прежде всего нужно описать *синтаксис*, т. е. правила формальной записи рассуждений. Особые способы записи рассуждений стали развиваться в европейской математике начиная с Виета (он придумал основную идею знакомых всем алгебраических формул). Завершился этот процесс уже в 20 веке изобретением понятия *формальной системы*. Значительная часть этой книги посвящена анализу формальных систем. Поэтому сразу дадим общее определение.

Определение В.3. *Формальная система* $\mathcal{F}$ задаётся *алфавитом* $\mathcal{L}$, *формулами* $\mathcal{F}$, *аксиомами* $\mathcal{A}$ и *правилами вывода* $\{\mathcal{R}_1, \ldots, \mathcal{R}_n\}$.

Алфавит $\mathcal{L}$ — это счётное множество. Формулы $\mathcal{F}$ — это подмножество слов в алфавите $\mathcal{L}$ (т. е. конечных последовательностей, состоящих из элементов $\mathcal{L}$). Аксиомы $\mathcal{A}$ — это подмножество формул. Каждое правило вывода $\mathcal{R}_i$ является *отношением* на наборах формул, т. е. подмножеством декартова произведения $\mathcal{F}^k$. Если $(F_1, \ldots, F_k) \in \mathcal{R}$, то формула F_k *выводится* из формул $F_1, \ldots, F_{k-1}$ по правилу вывода $\mathcal{R}_i$. *Выводом* называется такая конечная последовательность формул F_1, $\ldots$, F_n, что всякая формула в ней либо является аксиомой, либо выводится из предыдущих формул по одному из правил вывода. Формула называется *выводимой* (или формальной теоремой), если она встречается в одном из выводов.

Видно, что свобода в построении формальной системы очень велика. Общих свойств у всей совокупности формальных систем немного, поэтому общая теория формальных систем оказывается достаточно бедной.

Можно заметить соответствие между данным выше определением и математическими рассуждениями. Формулам соответствуют математические утверждения. Правила вывода соответствуют элементарным шагам математического рассуждения. При таком понимании выводимые формулы соответствуют тем утверждениям, которые можно доказать (в обычном математическом смысле), исходя из аксиом.

Это соответствие подсказывает, какие формальные системы интересны для изучения.

Какого рода вопросы возникают при изучении формальной теории? Одним из свойств формальной теории является *непротиворечивость*. Формальная теория называется непротиворечивой, если в ней выводимы не все формулы. Проверка непротиворечивости формальной теории — одна из типичных задач математической логики. В самом деле, в противоречивой теории выводимы все формулы и поэтому она в какомто смысле тривиальна. Во многих интересных случаях условие непротиворечивости можно ослабить. В частности, в большинстве формальных теорий, рассматриваемых в математической логике, непротиворечивость равносильна тому, что нельзя вывести одновременно формулу и её отрицание (конечно, в таких теориях должно быть формально определено отрицание формулы).

Независимость аксиом является другим примером свойства формальной системы, которое заслуживает изучения. Аксиомы формальной теории называются

независимыми, если никакую из них нельзя вывести из остальных.

Ещё одним свойством формальных систем является существование алгоритма, который по заданной формуле устанавливает её выводимость. Такие системы называются *разрешимыми*. Для доказательства неразрешимости формальной системы требуется формальное определение алгоритма. Эта связь между математической логикой (изучением средствами математики возможностей математических рассуждений) и теорией алгоритмов (изучением возможностей вычисления) оказалась весьма важной и плодотворной для развития обеих областей.

До сих пор мы говорили лишь о структуре математического рассуждения. Для формализации понятия истины необходима ещё некоторая *семантика*, т. е. правила соответствия между формулами формальной системы и утверждениями о некоторых математических объектах. После того, как задана семантика формальной теории, можно говорить об истинных и ложных формулах, имея в виду соответствующие им утверждения. Одними из самых важных свойств, которые характеризуют формальную систему с заданной семантикой, являются корректность и полнота. Корректность означает, что в формальной теории выводимы только истинные (в данной семантике) формулы. Полнота означает, что все истинные (в данной семантике) формулы выводимы в формальной теории. Выполнение свойств корректности и полноты указывает, что данная формальная теория является адекватной формализацией тех содержательных утверждений, которые записываются формулами этой теории.

Изучение полноты формальных систем также оказывается связанным с теорией алгоритмов. Здесь

важно свойство *перечислимости*. Перечислимые множества — это такие множества, список элементов которых может быть получен алгоритмом. В большинстве случаев множество формальных теорем формальной системы оказывается перечислимым. Если при этом множество истинных в данной семантике формул оказывается неперечислимым, то отсюда следует неполнота формальной системы. Именно так обстоит дело с одной из самых известных формальных систем — формальной арифметикой.

Книга состоит из четырёх глав.

Первые две главы содержат примеры формальных систем — классического и интуиционистского исчисления высказываний гильбертовского типа. Изучение этих систем даёт представление о методах и приёмах, используемых в математической логике.

Кроме того, классическое исчисление высказываний является частью более сложной формальной системы — исчисления предикатов. Это исчисление позволяет формализовать значительную часть математики и даёт возможность разделить «логические законы» (технически эти утверждения называются общезначимыми формулами) и содержательные факты, которые справедливы в конкретных математических теориях.

Исчисление предикатов рассматривается в третьей главе. Оказывается, что для «логических законов» возможна полная и корректная формализация, задаваемая исчислением предикатов. Наиболее трудная часть в доказательстве этого утверждения состоит в доказательстве полноты исчисления предикатов.

В четвёртой главе рассмотрены начала теории алгоритмов. Мы выбираем в качестве базовой модели алгоритма машины Тьюринга. Используя это опре-

деление, мы устанавливаем алгоритмическую неразрешимость некоторых задач, например, проверки равенства слов в полугруппе, заданной образующими и соотношениями.

Более сложные (но и более интересные!) примеры — алгоритмическая неразрешимость проверки существования решения у диофантова уравнения и алгоритмическая неразрешимость проблемы равенства слов в группе — не рассматриваются здесь из-за недостатка места и трудности доказательств для начинающих. Об этих проблемах можно прочитать, например, в книгах Ю. В. Матиясевича [13] и Ю. И. Манина [12].

В этой же главе более подробно рассмотрена намеченная выше связь между неполнотой и неперечислимостью на примере формальной арифметики. В результате такого анализа мы получаем одну из версий знаменитой теоремы Гёделя о неполноте.

Книга написана на основе материалов курса «Дискретный анализ», читаемого студентам факультета управления и прикладной математики МФТИ. Обычно этот курс включает в себя классическое исчисление предикатов, основы формальной логики первого порядка и основы теории алгоритмов. Это примерно соответствует главе 1, разделам 1–6 главы 3, и разделам 1–3 главы 4 данной книги. Остальные разделы книги содержат важный дополнительный материал, которого учебный курс лишён из-за нехватки времени.

Глава 1

Исчисление высказываний

Неформально под высказыванием понимается утверждение, которое может быть либо истинным, либо ложным. Из высказываний можно строить более сложные высказывания, используя пропозициональные связки (И; ИЛИ; НЕ; ЕСЛИ..., ТО...). Истинность составного высказывания определяется истинностью составляющих его простых высказываний. Отрицание НЕ А высказывания А истинно тогда и только тогда, когда высказывание А ложно. Конъюнкция А И Б двух высказываний истинна лишь тогда, когда оба высказывания истинны, а в остальных случаях ложна. Дизъюнкция А ИЛИ Б двух высказываний ложна лишь тогда, когда оба высказывания ложны, а в остальных случаях истинна. Импликация ЕСЛИ А, ТО Б двух высказываний ложна лишь тогда, когда *заключение* Б ложно, а *посылка* А истинна, в остальных случаях импликация истинна.

Некоторые составные высказывания истинны при любых значениях истинности входящих в них простых высказываний. Такие высказывания называются *тавтологиями*. (Пример тавтологии: «целое число x чётно ИЛИ целое число x нечётно».)

Наша основная цель в этой главе — построить формальную теорию, теоремами которой являлись бы тавтологии (свойство корректности) и только они (свойство полноты). Таких формальных теорий можно построить много. Мы подробно изучим одну из них, которую будем называть исчислением высказываний и обозначать ИВ.

1.1. Определение формальной системы ИВ

Общее определение формальной системы дано во введении на с. 9. В соответствии с ним для определения конкретной формальной системы (в данном случае — ИВ) нужно указать алфавит, формулы, аксиомы и правила вывода. Мы сделаем это для ИВ в нескольких следующих разделах.

1.1.1. Алфавит

Алфавит ИВ состоит из

- счётного множества L *пропозициональных букв* или, для краткости, *переменных* (далее мы будем обозначать пропозициональные буквы строчными латинскими буквами, возможно с индексом: x, y, x_1 и т. д.);
- *пропозициональных связок*:

$\to$	символ импликации,
$\neg$	символ отрицания;

- символов изменения приоритета — скобок:

(	открывающая скобка,
)	закрывающая скобка.

Пример 1.1. Вот несколько слов в алфавите L:

y	1 символ,
$)x_1{\to}($	4 символа,
$(\neg(x_1{\to}(\neg x_2)))$	11 символов.

Замечание 1.2. Знак равенства не входит в алфавит ИВ. Всюду мы используем знак равенства в обычном математическом смысле для обозначения равенства двух объектов (слов, функций, чисел и т. д.).

1.1.2. Формулы

Среди слов в алфавите ИВ (т. е. конечных последовательностей символов из алфавита L) выделяются *формулы*.

Определение 1.3. Слово A в алфавите ИВ является *формулой* тогда и только тогда, когда выполняется одно их трёх условий

1: A является *переменной* (более строго: словом длины 1, состоящим из пропозициональной буквы);

2: A является *отрицанием* некоторой формулы B, т. е. имеет вид $(\neg B)$;

3: A является *импликацией*, т. е. имеет вид $(B{\to}C)$, где B, C — формулы (B называется *посылкой* импликации, а C — *заключением*).

Данное определение формулы *рекурсивно*: чтобы проверить, является ли слово формулой, мы должны выяснить, являются ли формулами некоторые его части.

Замечание 1.4. Использование рекурсивных определений требует особой аккуратности.

Рассмотрим пример: назовём целое число *удивительным*, если оно равно 1 или является суммой удивительных чисел. Читатель без труда проверит, что все положительные целые числа оказываются по такому определению удивительными. Но зададимся вопросом: является ли удивительным число 0? Данному выше определению удовлетворяют оба возможных ответа.

«Да, 0 — удивительное число. Все целые числа удивительны.»

«Нет, удивительны только положительные целые числа. Поэтому 0 не является удивительным числом.»

Из этих двух примеров видно, что данное выше определение удивительного числа некорректно — ему удовлетворяют два разных множества целых чисел.

Есть общий рецепт устранения этой трудности. В данном случае добавим к определению, что удивительные числа образуют наименьшее по включению множество целых чисел, которое содержит 1 и замкнуто относительно сложения. Другими словами, возьмём пересечение всех множеств целых чисел, удовлетворяющих указанным свойствам.

Все рекурсивные определения, которыми мы будем пользоваться, будут корректными, т. е. описанная здесь трудность вообще у нас не встретится (см. следующий раздел, в котором подробно анализируется случай определения формул ИВ). Однако читатель может добавлять мысленно ко всем рекурсивным определениям в книге сформулированное выше свойство минимальности — это не помешает пониманию прочитанного.

Задача 1.1. Длиной формулы назовём количество символов в ней. Верно ли, что формула ИВ длины 400 содержит по крайней мере 100 пропозициональных букв?

1.1.3. О структуре формул

В определении формулы ИВ требуется, чтобы каждая формула, отличная от переменной, представлялась в виде $(\neg A)$ или $(A{\to}B)$, где A, B — формулы. Оказывается, что такое представление единственно. Это свойство формул называется *однозначностью разбора формулы*.

Утверждение об однозначности разбора формулы выглядит почти очевидным. Мы тем не менее приведём его доказательство, использующее важный приём рассуждения, которым мы будем часто пользоваться в дальнейшем.

Лемма 1.5. *В любой формуле количество открывающих скобок равно количеству закрывающих скобок.*

Доказательство. Индукция по длине формулы (количеству символов). Для формул, которые состоят из одной переменной, утверждение леммы справедливо: $0-0=0$. Других формул длины 1 нет. Это доказывает базу индукции.

Индукционный переход: предположим, что утверждение леммы справедливо для формул длины меньше, чем n ($n>1$). Рассмотрим формулу A длины n. По определению она имеет вид $A=(\neg B)$ или $A=(B{\to}C)$, где B, C — более короткие формулы, в которых одинаково количество открывающих и закрывающих скобок по предположению индукции. Но тогда и в формуле A количество открывающих и закрывающих скобок одинаково. □

Замечание 1.6. Приём, использованный в доказательстве леммы, называется *индукцией по построению формулы*. Заметим, что применение индукции по построению формулы не использует однозначности разбора. Свойство однозначности разбора будет важно при построении соответствия между формулами и булевыми функциями, см. раздел 1.2.

Определение 1.7. *Скобочный итог* входящего в формулу символа равен разности между числом открывающих и закрывающих скобок, предшествующих этому символу в формуле.

Лемма 1.8. *Для любой формулы скобочный итог любого символа пропозициональной связки положителен, а если формула содержит хотя бы один символ пропозициональной связки, то в ней есть ровно один символ пропозициональной связки, для которого скобочный итог равен* $+1$.

Доказательство. Индукция по построению формулы. База индукции (формула состоит из одной переменной) очевидна.

Индукционный переход: пусть формула имеет вид $(\neg B)$. Для первого символа пропозициональной связки $\neg$ скобочный итог равен $+1$, а для любого символа пропозициональной связки, входящего в формулу B, скобочный итог по предположению индукции больше $+1$. Аналогичное рассуждение справедливо и в случае формулы вида $(B{\to}C)$: скобочный итог любого символа пропозициональной связки в подформулах B, C больше $+1$, а для символа $\to$ между этими подформулами скобочный итог равен $+1$. Здесь использована лемма 1.5. □

Следствие 1.9. *Для любой формулы справедливо свойство однозначности разбора.*

В частности, для каждой формулы, отличной от переменной, однозначно определена *внешняя связка*, скобочный итог которой равен +1, а для формул, у которых внешняя связка — импликация, однозначно определены *левая* и *правая* подформулы (соответственно, A и B для формулы $(A{\to}B)$).

Из свойства однозначности разбора формулы вытекает, что каждой формуле можно однозначно сопоставить некоторое *плоское корневое дерево*, вершины которого помечены символами из алфавита ИВ.

Определения, относящиеся к графам, деревьям и плоским корневым деревьям, приведены в справочном приложении (с. 330). Здесь мы напомним, что *деревом* называется связный неориентированный граф без циклов. У корневого дерева выделена одна из вершин, она называется *корнем*. *Плоское корневое дерево* можно нарисовать на плоскости так, чтобы путь из корня в любую другую вершину шёл в одном направлении (вниз на рис 1.1). Заметим, что такой рисунок задаёт упорядочение тех рёбер, которые идут из вершины вниз (*нижних рёбер*), мы используем порядок слева направо, который также указан на рис. 1.1.

Каждой формуле ИВ сопоставим помеченное плоское корневое дерево следующим образом. Формуле, состоящей из одной переменной, сопоставим дерево, состоящее из одного корня, которой помечен этой

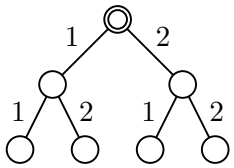


Рис. 1.1.

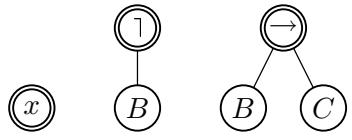


Рис. 1.2.

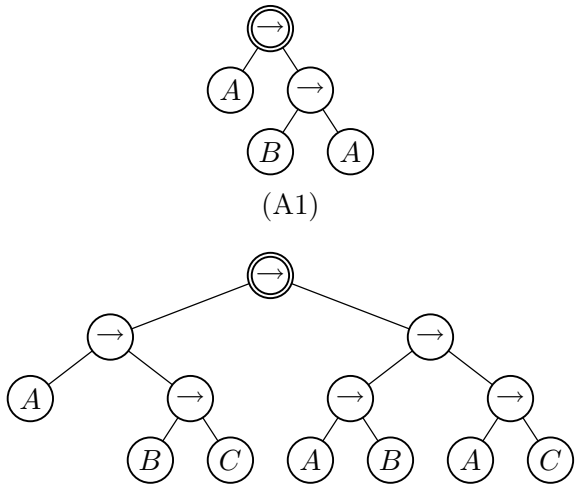
переменной (рис. 1.2(а)). Далее определяем соответствие индуктивно. Пусть для всех формул длины меньше, чем n ($n > 1$), соответствие уже определено. Рассмотрим формулу A длины n. Если $A = (\neg B)$, то из корня дерева, соответствующего формуле B, проведём ребро в новую вершину, пометим эту вершину символом $\neg$ и объявим новую вершину корнем полученного дерева (рис. 1.2(б)). Построенное таким образом дерево сопоставим формуле A. Если $A = (B{\to}C)$, то дерево, соответствующее формуле A (см. рис. 1.2(в)), состоит из корня, помеченного символом $\to$, и двух поддеревьев, соответствующих формулам B и C. Из корня дерева выходит два ребра, левое ведёт в корень поддерева, соответствующего B, а правое — в корень поддерева, соответствующего C.

Определение 1.10. *Подформулой* формулы называется такое её подслово (т. е. последовательность подряд идущих символов), которое также является формулой.

Задача 1.2. Докажите, что построенное выше соответствие порождает взаимно однозначное соответствие между подформулами и вершинами дерева формулы.

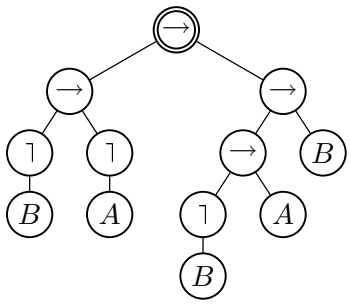
Указание: используйте индукцию по длине формулы.

На рис. 1.3 изображены деревья, соответствующие аксиомам ИВ (см. далее, с. 24).



(A1)

(A2)



(A3)

Рис. 1.3.

1.1.4. Сокращения при записи формул

Мы определили формулы ИВ, используя всего две пропозициональных связки. Это удобно при доказательствах утверждений о формальной системе (нужно разбирать меньше случаев в доказательствах индукцией по построению формулы), но неудобно при записи формул, отвечающих интересным высказываниям. Помимо отсутствия символов, отвечающих конъюнкции и дизъюнкции, формулы в ИВ содержат большое количество скобок. В дальнейшем мы будем использовать разнообразные варианты сокращённой записи формул. Обратите внимание, что формулы в сокращённой записи не являются формулами ИВ, они служат для более компактного и наглядного представления формул ИВ.

Мы будем использовать при сокращённой записи формул дополнительные связки $\wedge$ (конъюнкция), $\vee$ (дизъюнкция), $\sim$ (эквивалентность). Если A, B — формулы, то сокращение $(A \wedge B)$ означает формулу $(\neg(A{\to}(\neg B)))$, сокращение $(A \vee B)$ означает формулу $((\neg A){\to}B)$, а сокращение $(A{\sim}B)$ означает формулу $((A{\to}B) \wedge (B{\to}A))$ (здесь мы использовали ранее введённое сокращение).

Для сокращения количества скобок мы будем применять такие же правила записи формул, которые используются в алгебре. Для пропозициональных связок введём отношение старшинства, перечислив их в порядке убывания старшинства:

$$\neg,\ \wedge,\ \vee,\ \to,\ \sim.$$

Восстановление полной записи формулы по записи, в которой опущена часть скобок, производится по правилу: если скобками не указано иное, то связки применяются по старшинству, а одинаковые связки

применяются в порядке слева направо. Внешняя пара скобок будет опускаться. Вот примеры записи формул с пропущенными скобками: $A{\to}B{\to}A{\to}A$ означает формулу $(((A{\to}B){\to}A){\to}A)$, а $A{\wedge}B{\to}C{\vee}D$ — формулу $((\neg(A{\to}(\neg B))){\to}((\neg C){\to}D))$ (в последнем случае мы не только восстановили скобки, но и заменили $\wedge$ и $\vee$ на их определения).

1.1.5. Аксиомы

Аксиом в ИВ бесконечно много. Чтобы задать бесконечное множество формул, мы используем *схемы аксиом*. Схема аксиом — это формула, переменные которой понимаются как произвольные формулы. Другими словами, мы сопоставляем каждой формуле Φ множество формул, которые получаются подстановкой в формулу Φ произвольных формул вместо переменных, входящих в формулу Φ.

В ИВ есть три схемы аксиом.

Определение 1.11. Формула ИВ является *аксиомой*, если её можно представить одним из следующих способов:

$$(A{\to}(B{\to}A)), \tag{A1}$$
$$((A{\to}(B{\to}C)){\to}((A{\to}B){\to}(A{\to}C))), \tag{A2}$$
$$(((\neg B){\to}(\neg A)){\to}(((\neg B){\to}A){\to}B)), \tag{A3}$$

где A, B, C — произвольные формулы.

1.1.6. Правило вывода

Правило вывода в исчислении высказываний ровно одно. Оно называется modus ponens (правило отделения) и может быть формально записано как

$$\frac{A,\ A{\to}B}{B}\,. \tag{modus ponens}$$

Неформально эта запись означает утверждение «из выводимости формул A, $A{\to}B$ следует выводимость формулы B».

1.1.7. Выводы и выводимые формулы

Теперь можно дать определение выводимой формулы исчисления высказываний, следуя общему определению из введения.

Определение 1.12. Формула A называется *выводимой* в исчислении высказываний (или *формальной теоремой исчисления высказываний*), если существует последовательность формул (*вывод*)

$$A_1, \ldots, A_n = A,$$

в которой для любой формулы A_i выполнено хотя бы одно из двух условий:

- A_i является аксиомой;
- найдутся такие $j, k < i$, что $A_k = A_j \to A_i$ (т. е. ранее в последовательности формул встречались и формула A_j, и формула $A_j{\to}A_i$).

Для выводимости мы будем использовать специальное обозначение $\vdash A$, означающее выводимость формулы A.

Мы будем записывать выводы не в строку, а в столбец, нумеровать формулы в выводе, а также указывать для каждой формулы её обоснование (является ли формула аксиомой или из каких предыдущих формул она получена правилом вывода). Такой способ записи вывода облегчает сопоставление написанной последовательности формул и определения вывода.

Пример вывода даёт следующая лемма.

Лемма 1.13. $\vdash (A{\to}A)$.

Доказательство. Приведём последовательность формул и проверим, что она является выводом:

1. $(A{\to}((A{\to}A){\to}A))$ (A1)
2. $((A{\to}((A{\to}A){\to}A)){\to}((A{\to}(A{\to}A)){\to}(A{\to}A)))$ (A2)
3. $((A{\to}(A{\to}A)){\to}(A{\to}A))$ (MP 1,2)
4. $(A{\to}(A{\to}A))$ (A1)
5. $(A{\to}A)$ (MP 4,3)

Чтобы получить первую формулу из первой схемы аксиом, нужно подставить вместо B формулу $(A{\to}A)$. Вторая формула получается из второй схемы аксиом подстановкой вместо B формулу $(A{\to}A)$, а вместо C — формулы A. Третья и пятая формулы в выводе получаются применением правила modus ponens, что отмечено в правом столбце. Например, запись (MP 1,2) указывает, что третья формула вывода получена из первой и второй применением правила modus ponens. Наконец, четвёртая формула получается из первой схемы аксиом подстановкой формулы A вместо формулы B. □

Задача 1.3. Существует ли формула исчисления высказываний, которая встречается ровно в двух выводах?

Кроме выводимости мы будем использовать выводимость из множества формул Γ (по традиции формулы из этого множества называются *гипотезами*). Определение формулы, выводимой из множества Γ, почти дословно повторяет определение выводимой формулы.

Определение 1.14. Формула A называется *выводимой* в исчислении высказываний из множества гипотез Γ, если существует последовательность формул

$$A_1, \ldots, A_n = A,$$

в которой для любой формулы A_i выполнено хотя бы одно из трёх условий:

- A_i является гипотезой (принадлежит множеству Γ);
- A_i является аксиомой;
- найдутся такие $j, k < i$, что $A_k = A_j \to A_i$.

Для выводимости из множества Γ используется обозначение $\Gamma \vdash A$. Например, $\{A, B\} \vdash C$ или $A, B \vdash C$ (второе обозначение мы будем обычно использовать при записи выводимости из конечных множеств).

Выводимость из множества формул называется также *условной* выводимостью. Такое называние имеет понятный смысл: из $\Gamma \vdash A$ следует выводимость формулы A при условии, что любая формула из множества Γ выводима.

Приведём простой пример условной выводимости.

Лемма 1.15. $A \vdash (B \to A)$.

Доказательство.

1.	$(A \to (B \to A))$	(A1)
2.	A	(гипотеза)
3.	$(B \to A)$	(MP 2,1)

(в условном выводе одним из обоснований формулы является принадлежность к множеству гипотез). □

Из леммы 1.15 следует, в частности, что после того, как доказана выводимость формулы A можно утверждать выводимость любой формулы вида $(B{\to}A)$. Поэтому, в частности, из *противоречия* (т. е. формул A и $\neg A$) можно вывести любую формулу.

Лемма 1.16. $A, \neg A \vdash B$.

Доказательство. Применяем дважды лемму 1.15 и схему аксиом (A3):

1.	$(\neg B \to A)$	(лемма 1.15)
2.	$(\neg B \to \neg A)$	(лемма 1.15)
3.	$((\neg B \to \neg A) \to ((\neg B \to A) \to B))$	(A3)
4.	$((\neg B \to A) \to B)$	(MP 2,3)
5.	B	(MP 1,4)

Мы привели сокращённую запись вывода. Для получения полной записи нужно вставить вместо ссылок на лемму 1.15 вывод из этой леммы. Нумерация строк служит для удобства и не указывает на место формулы в полном выводе. Такую сокращённую запись вывода мы будем использовать и дальше. □

1.2. Исчисление высказываний и булевы функции

Для ИВ есть естественная семантика — формулы ИВ реализуют булевы функции.

Напомним, что аргументы и значение *булевой функции* $f(x_1, \ldots, x_n)$ принадлежат множеству $\{0, 1\}$. Поскольку область определения булевой функции конечна, её можно задать таблицей значений. Приведём таблицы значений некоторых булевых функций:

x	y	$\neg x$	$x \wedge y$	$x \vee y$	$x \to y$	$x \sim y$	$x \oplus y$
0	0	1	0	0	1	1	0
0	1	1	0	1	1	0	1
1	0	0	0	1	0	0	1
1	1	0	1	1	1	1	0

Если понимать под символом 0 ложь, а под символом 1 истину, то приведённые выше функции правильно отражают смысл пропозициональных связок (соответственно, отрицания, конъюнкции, дизъюнкции, импликации, эквивалентности; последняя функция часто обозначается XOR и имеет смысл исключающего ИЛИ — истинно ровно одно из двух высказываний).

Каждой формуле ИВ сопоставим булеву функцию, определив *значение формулы* при заданных значениях входящих в неё переменных. Определение даётся индукцией по построению формулы.

1: Значение формулы, которая является переменной, совпадает со значением этой переменной.

2: Значение формулы $(\neg B)$ является отрицанием значения формулы B.

3: Значение формулы $(A \to B)$ является импликацией значений формул A, B.

Таким образом, каждая формула A в ИВ *реализует* булеву функцию $A(x)$. При этом тавтологии реализуют тождественно равные 1 функции.

Имея некоторый набор булевых функций $\mathcal{F}$ (*систему функций*), можно строить новые функции с помощью операций замены переменных и суперпозиции (подстановки одной функции в аргумент другой функции).

Напомним, что система функций $\mathcal{F}$ называется *полной*, если любую булеву функцию можно представить с помощью замены переменных и суперпозиций функций из системы $\mathcal{F}$. Критерий полноты системы булевых функций даёт теорема Поста.

Теорема 1.17 (теорема Поста). Система функций полна тогда и только тогда, когда она не содержится целиком в одном из классов T_0, T_1, L, M, S, т. е. когда она содержит

1: функцию, не сохраняющую 0 ($f(\tilde{0}) = 1$);

2: функцию, не сохраняющую 1 ($f(\tilde{1}) = 0$);

3: нелинейную функцию (т. е. функцию, которая не представима в виде

$$\alpha_0 \oplus \alpha_1 x_1 \oplus \ldots \oplus \alpha_n x_n$$

при каких-то значениях $\alpha_i \in \{0, 1\}$);

4: немонотонную функцию (т. е. для некоторых наборов аргументов $\tilde{\alpha}$, $\tilde{\beta}$ выполнено $\tilde{\alpha} \leqslant \tilde{\beta}$, $f(\tilde{\alpha}) = 1$, $f(\tilde{\beta}) = 0$, сравнение наборов значений покомпонентное);

5: несамодвойственную функцию (неравенство

$$f(x_1, \ldots, x_n) \neq f(\neg x_1, \ldots \neg x_n)$$

выполнимо на некотором наборе значений аргументов $(x_1, \ldots, x_n)$).

Применим теорему Поста к системе функций $\{\neg, \to\}$.

Эта система содержит функцию $\neg$, которая не сохраняет 0, не сохраняет 1 и не является монотонной. Также в ней есть функция $\to$, которая не является

самодвойственной и не является линейной. Таким образом, по теореме Поста эта система функций полна, т. е. любую булеву функцию можно выразить через отрицание и импликацию. Например, можно проверить по таблице значений, что

$$x \vee y = (\neg x) \to y, \tag{1.1}$$

$$x \wedge y = \neg(x \to (\neg y)). \tag{1.2}$$

Из полноты системы функций $\{\neg, \to\}$ следует, что для любой булевой функции f, найдётся формула ИВ, которая реализует функцию f.

Задачи

1.4. Формулы $\neg F_1$ и F_2 выводимы в исчислении высказываний. Верно ли, что выводима формула $F_1 \to F_2$?

1.5. Множество формул Γ состоит из формул, в которые входят лишь переменные x, y. Верно ли, что для переменной z выполнено $\Gamma \vdash z$?

1.3. Непротиворечивость ИВ

Лемма 1.18. *Аксиомы* (A1), (A2), (A3) *являются тавтологиями.*

Доказательство. Простейший способ проверить тавтологичность формулы состоит в том, чтобы составить таблицу истинности для функции, реализуемой этой формулой.

Пример такой проверки для аксиомы (A1) приведен в таблице 1.1.

Хотя аналогичные таблицы можно составить и для остальных аксиом, мы проверим их тавтологичность иначе. Заметим, что ложность импликации означает

Таблица 1.1. Таблица истинности для аксиомы (A1)

A	B	$B{\to}A$	$A{\to}(B{\to}A)$
0	0	1	1
0	1	0	1
1	0	1	1
1	1	1	1

ложность заключения и истинность посылки. Поэтому значение формулы, удовлетворящей второй схеме аксиом, ложно лишь тогда, когда значение переменной C ложно, а значение переменной A истинно. При этом значение $A{\to}B$ обязано быть истинным (это посылка в ложной импликации), так что значение B истинно. Теперь осталось убедиться, что при таком наборе значений переменных, значение формулы

$$((A{\to}(B{\to}C)){\to}((A{\to}B){\to}(A{\to}C)))$$

истинно. Это действительно так, поскольку ложна посылка внешней импликации.

Проверим тавтологичность третьей схемы аксиом. Если ложна формула

$$(((\neg B){\to}(\neg A)){\to}(((\neg B){\to}A){\to}B)), \qquad \text{(A3)}$$

то значение переменной B ложно, а значение импликации $(\neg B){\to}A$ истинно. Это означает, что значение переменной A истинно. Но при таком наборе значений переменных ложна посылка внешней импликации. Поэтому формула (A3) является тавтологией. □

Теорема 1.19 (*корректность* ИВ). *В исчислении высказываний выводимы только тавтологии.*

Доказательство. Докажем, что любой вывод A_1, $\ldots$, A_n в ИВ состоит только из тавтологий. Применим индукцию по длине вывода. Если вывод состоит из одной формулы, то это формула является аксиомой. Аксиомы являются тавтологиями в силу леммы 1.18. Предположим теперь, что наше утверждение верно для всех выводов, длина которых меньше n. Рассмотрим вывод длины n и последнюю формулу A_n в нём. По определению вывода есть две возможности: (1) A_n — аксиома (и тогда она тавтология по лемме 1.18); (2) A_n получена по правилу вывода из формул A_i, $A_j = A_i \to A_n$, $i, j < n$. В последнем случае по предположению индукции формулы A_i и A_j являются тавтологиями. Но если A_i и $A_i \to A_n$ истинны, то истинной является и A_n (см. таблицу импликации). $\square$

Следствие 1.20. *Исчисление высказываний непротиворечиво, причём из формул A и $\neg A$ выводима не более, чем одна.*

Доказательство. Если одна из формул A и $\neg A$ является тавтологией, то значение другой на любом наборе значений пропозициональных переменных ложно. $\square$

Замечание 1.21. Из сказанного может сложиться ложное представление о простоте формальной системы ИВ. В самом деле, тождественно равные 1 булевы функции трудно не назвать тривиальным классом булевых функций, а выводятся в ИВ лишь тавтологии.

На самом деле формальная система ИВ очень сложна. Эта сложность обусловлена большим разнообразием способов реализации функций, тождественно равных 1. Отличить тавтологии от прочих формул в некотором смысле столь же трудно, как разобраться с произвольным утверждением о конечных объектах.

Задачи

1.6. Существует ли формула исчисления высказываний, которая не встречается ни в одном выводе?

1.7. Формулы F_1 и $\neg F_2$ выводимы в исчислении высказываний. Верно ли, что выводима формула $F_1 \to F_2$?

1.8. Формулы F_1 и F_2 невыводимы в исчислении высказываний. Верно ли, что невыводима формула $F_1 \to F_2$?

1.9. Существуют ли такие формулы F_1, F_2, что F_1, F_2 — не тавтологии, а $F_1 \to (x \to F_2)$ и $F_1 \to (\neg x \to F_2)$ — тавтологии?

1.10. Докажите, что если все переменные в формуле исчисления высказываний различны, то её нельзя вывести в исчислении высказываний.

1.4. Теорема дедукции

Доказательства выводимости сильно упрощает следующая теорема.

Теорема 1.22 (теорема дедукции). *Для произвольного множества формул* Γ *и произвольных формул* A, B *выводимость* $\Gamma \cup \{A\} \vdash B$ *равносильна выводимости* $\Gamma \vdash A \to B$.

Поскольку эта теорема очень важна для дальнейшего, приведём более развёрнутую формулировку: формула B выводима из множества гипотез $\Gamma \cup \{A\}$ тогда и только тогда, когда формула $A \to B$ выводима из множества гипотез Γ.

Доказательство. В одну сторону теорема дедукции очевидна: если существует вывод формулы $A \to B$ из

множества формул Γ, то после добавления к этому выводу формул

$$\begin{aligned} A &\qquad \text{(гипотеза)} \\ B &\qquad \text{(MP из } A,\ A{\to}B) \end{aligned}$$

получаем вывод формулы B из множества формул $\Gamma \cup \{A\}$.

В другую сторону будем доказывать индукцией по длине вывода. Пусть имеется вывод длины n формулы B из множества формул $\Gamma \cup \{A\}$: $B_1, \ldots, B_n = B$.

Для первой формулы в выводе есть три возможности:

- B_1 является аксиомой;
- B_1 принадлежит множеству Γ;
- $B_1 = A$.

В первых двух случаях для построения вывода $A{\to}B_1$ из множества Γ используем лемму 1.15. В третьем случае нужно построить вывод формулы $A{\to}A$, что сделано в лемме 1.13. База индукции доказана.

Предположим, что построен вывод $B'_1, \ldots, B'_N$ формул $A{\to}B_1$, $A{\to}B_2$, $\ldots$, $A{\to}B_k$ из множества формул Γ. Для формулы B_{k+1} есть четыре возможности. Первые три из совпадают с описанными выше для формулы B_1. В этих случаях мы поступаем так же, как и в случае B_1. Остаётся рассмотреть лишь применение правила вывода. Тогда $B_j = B_i{\to}B_{k+1}$, $i, j < k{+}1$. Добавим к выводу B' следующие формулы:

$$N+1. \quad (A{\to}(B_i{\to}B_{k+1})){\to}((A{\to}B_i){\to}(A{\to}B_{k+1})) \qquad \text{(A2)}$$

$N+2$. $(A{\to}B_i){\to}(A{\to}B_{k+1})$ (MP N', $N+1$)
$N+3$. $A{\to}B_{k+1}$ (MP N'', $N+2$)

Здесь через N' обозначен номер формулы

$$A{\to}(B_i{\to}B_{k+1}) = A{\to}B_j$$

в выводе B', а через N'' — номер формулы $A{\to}B_i$ в том же выводе. Теперь получен вывод B'' формул $A{\to}B_1$, $A{\to}B_2$, …, $A{\to}B_{k+1}$. Шаг индукции доказан, что завершает доказательство теоремы. □

Приведём примеры использования теоремы дедукции.

Лемма 1.23 (транзитивность).

$$\{A{\to}B,\ B{\to}C\} \vdash A{\to}C.$$

Доказательство. Построим вывод формулы C из формул A, $A{\to}B$, $B{\to}C$

1. A (гипотеза)
2. $A{\to}B$ (гипотеза)
3. B (MP 1,2)
4. $B{\to}C$ (гипотеза)
5. C (MP 3,4)

и применим теорему дедукции. □

Лемма 1.24 (снятие двойного отрицания).

(а) $\vdash \neg\neg A{\to}A$; (б) $\vdash A{\to}\neg\neg A$.

Доказательство. Для (а) построим вывод $\neg\neg A \vdash A$

1. $(\neg A{\to}\neg\neg A){\to}((\neg A{\to}\neg A){\to}A)$ (A3)
2. $\neg\neg A$ (гипотеза)

3. $\neg A \to \neg\neg A$ (лемма 1.15)
4. $(\neg A \to \neg A) \to A$ (MP 3,1)
5. $\neg A \to \neg A$ (лемма 1.13)
6. A (MP 4,5)

и применим теорему дедукции.

Для (б) поступаем аналогично. Строим вывод $A \vdash \neg\neg A$

1. $(\neg\neg\neg A \to \neg A) \to ((\neg\neg\neg A \to A) \to \neg\neg A)$ (A3)
2. $\neg\neg\neg A \to \neg A$ (a)
3. $(\neg\neg\neg A \to A) \to \neg\neg A$ (MP 2,1)
4. A (гипотеза)
5. $\neg\neg\neg A \to A$ (лемма 1.15)
6. $\neg\neg A$ (MP 5,3)

и применяем теорему дедукции. □

Лемма 1.25 (правило контрапозиции).

(а) $\vdash (\neg B \to \neg A) \to (A \to B)$; (б) $\vdash (A \to B) \to (\neg B \to \neg A)$.

Доказательство. Для (а) построим вывод формулы B из формул A, $\neg B \to \neg A$

1. A (гипотеза)
2. $A \to (\neg B \to A)$ (A1)
3. $(\neg B \to A)$ (MP 1,2)
4. $\neg B \to \neg A$ (гипотеза)
5. $(\neg B \to \neg A) \to ((\neg B \to A) \to B)$ (A3)
6. $(\neg B \to A) \to B$ (MP 4,5)
7. B (MP 3,6)

и дважды применяем теорему дедукции.

Для (б) поступаем аналогично. Строим вывод формулы $\neg A$ из формул $A \to B, \neg B$

1. $\neg B$ (гипотеза)
2. $\neg\neg A \to \neg B$ (лемма 1.15)
3. $(\neg\neg A \to \neg B) \to ((\neg\neg A \to B) \to \neg A)$ (A3)
4. $(\neg\neg A \to B) \to \neg A$ (MP 2,3)
5. $\neg\neg A \to A$ (лемма 1.24 (a))
6. $A \to B$ (гипотеза)
7. $\neg\neg A \to B$ (лемма 1.23 к 5,6)
8. $\neg A$ (MP 7,4)

и дважды применяем теорему дедукции. □

Лемма 1.26. *Если* $\neg T \vdash A$, $\neg T \vdash \neg A$, *то* $\vdash T$.

Доказательство. По теореме дедукции выводимы формулы $\neg T \to A$, $\neg T \to \neg A$. Вывод T получается добавлением к их выводам аксиомы

$$(\neg T \to \neg A) \to ((\neg T \to A) \to T),$$

за которой следует два применения modus ponens. □

Задачи

1.11. Докажите, что в исчислении высказываний $\neg a \nvdash b$.

1.12. Пусть для любой формулы B в исчислении высказываний выполнено $B \vdash A$. Докажите, что формула A — тавтология.

1.13. Пусть для любой формулы B в исчислении высказываний выполнено $A \vdash B$. Верно ли, что формула A невыполнима?

1.14. Для формул A, B исчисления высказываний выполняются условия $A \vdash B$ и $B \vdash \neg A$. Докажите, что $\neg A$ — тавтология.

1.5. Теорема о полноте

Теорема 1.27 (*полнота* ИВ). *Если формула является тавтологией, то она выводима в ИВ.*

Мы приведём два доказательства теоремы о полноте. Первое является «конструктивным», т. е. в нём описывается рецепт построения вывода для каждой тавтологии. Второе доказательство такого рецепта не даёт. Зато оно может применяться и для других формальных систем, когда идея первого доказательства не работает.

В обоих доказательствах используется выводимость некоторых конкретных формул. Часть из них уже встречалась выше. Остальные приводятся в следующей лемме.

Лемма 1.28. *В* ИВ *выполняются выводимости*:

$$\neg B \vdash B \to C \tag{1.3}$$

$$\vdash \neg B \to (B \to C) \tag{1.4}$$

$$B, \neg C \vdash \neg(B \to C) \tag{1.5}$$

$$\neg(B \to C) \vdash B \tag{1.6}$$

$$\vdash (\neg B \to A) \to ((B \to A) \to A) \tag{1.7}$$

Доказательство. По теореме дедукции из леммы 1.16 следует утверждение (1.3), а из (1.3) следует (1.4).

Покажем, как построить вывод для (1.5). Поскольку $B, B \to C \vdash C$ (применение modus ponens к гипотезам), то $B \vdash (B \to C) \to C$ (теорема дедукции). Из леммы 1.25 (б) следует, что $(B \to C) \to C \vdash \neg C \to \neg(B \to C)$. Значит, из транзитивности вывода (лемма 1.23) получаем $B \vdash \neg C \to \neg(B \to C)$. Теперь для получения искомого вывода формулы $\neg(B \to C)$ из гипотез B, $\neg C$ осталось применить правило modus ponens к гипотезе $\neg C$ и формуле $\neg C \to \neg(B \to C)$.

Приведём вывод для (1.6).

1. $\neg(B\to C)$ (гипотеза)
2. $\neg B\to\neg(B\to C)$ (лемма 1.15)
3. $(\neg B\to\neg(B\to C))\to((\neg B\to(B\to C))\to B)$ (A3)
4. $(\neg B\to(B\to C))\to B$ (MP 2,3)
5. $\neg B\to(B\to C)$ (формула (1.4))
6. B (MP 5,4)

Теперь докажем (1.7). Достаточно построить вывод $\neg B\to A\,, B\to A\vdash A$

1. $(\neg B\to A)\to(\neg A\to\neg\neg B)$ (лемма 1.25 (б))
2. $(\neg A\to\neg\neg B)\to((\neg A\to\neg B)\to A)$ (A3)
3. $(\neg B\to A)\to((\neg A\to\neg B)\to A)$ (лемма 1.23)
4. $\neg B\to A$ (гипотеза)
5. $(\neg A\to\neg B)\to A$ (MP 4,3)
6. $(B\to A)$ (гипотеза)
7. $(B\to A)\to(\neg A\to\neg B)$ (лемма 1.25 (б))
8. $(B\to A)\to A$ (лемма 1.23 к 7, 5)
9. A (MP 6,8)

и дважды применить теорему дедукции. □

1.5.1. Первое доказательство

Очевидный способ проверить тавтологичность состоит в том, чтобы вычислить значения формулы при всех возможных значениях входящих в неё переменных и проверить, что все они равны 1. Оказывается, вычисление значения формулы в некотором смысле реализуемо в ИВ.

Пусть α — некоторый набор значений переменных. Сопоставим каждой формуле A её α-версию A^α. По

определению, если A истинна на наборе значений α, то $A^\alpha = A$, а если A ложна на наборе значений α, то $A^\alpha = \neg A$.

Лемма 1.29. *Пусть в формулу A входят только переменные $x_1, x_2, \ldots, x_n$. Тогда для любого набора значений переменных α*

$$x_1^\alpha, x_2^\alpha, \ldots, x_n^\alpha \vdash A^\alpha.$$

Например, пусть $A = a \to (b \to \neg a)$. Формула A истинна, при $a = 0$, $b = 1$. Поэтому из формул $\neg a$, b можно вывести формулу A.

Сначала докажем теорему о полноте, опираясь на лемму 1.29.

Пусть T — некоторая тавтология, в которую входят только переменные $x_1, \ldots, x_n$.

По определению T истинна на любом наборе значений переменных α, входящих в эту формулу. Из леммы 1.29 получаем, что для любого α справедлива выводимость

$$x_1^\alpha, x_2^\alpha, \ldots, x_n^\alpha \vdash T.$$

Переменная x_n может принимать два значения: истина и ложь. Подставляя их в предыдущую формулу, получаем, что для любого α справедливы следующие выводимости:

$$x_1^\alpha, x_2^\alpha, \ldots, x_{n-1}^\alpha, x_n \vdash T, \tag{1.8}$$

$$x_1^\alpha, x_2^\alpha, \ldots, x_{n-1}^\alpha, \neg x_n \vdash T. \tag{1.9}$$

Из теоремы дедукции заключаем, что для любого α справедливы выводимости

$$x_1^\alpha, x_2^\alpha, \ldots, x_{n-1}^\alpha \vdash x_n \to T, \tag{1.10}$$

$$x_1^\alpha, x_2^\alpha, \ldots, x_{n-1}^\alpha \vdash \neg x_n \to T. \tag{1.11}$$

Добавим вывод формулы (1.7) из леммы 1.28. Используя формулы (1.7), (1.10), (1.11) легко вывести формулу T:

$N.$	$x_n \to T$	(формула (1.10))
$N+1.$	$\neg x_n \to T$	(формула (1.11))
$N+2.$	$(\neg x_n \to T) \to ((x_n \to T) \to T)$	(формула (1.7))
$N+3.$	$(x_n \to T) \to T$	(MP $N+1$, $N+2$)
$N+4.$	T	(MP N, $N+3$)

Значит, для любого набора значений переменных α

$$x_1^\alpha, x_2^\alpha, \ldots, x_{n-1}^\alpha \vdash T.$$

Теперь повторим процесс сокращения, применив его к переменной x_{n-1}, затем к переменной x_{n-2} и т. д. Итак, при любом i и любом α справедливы выводимости

$$x_1^\alpha, x_2^\alpha, \ldots, x_i^\alpha \vdash T.$$

Но при $i = 0$ это означает выводимость формулы T, что и требовалось доказать.

Для завершения доказательства нам необходимо доказать лемму 1.29.

Доказательство леммы 1.29. Обозначим через Γ множество литералов $\{x_1^\alpha, x_2^\alpha, \ldots, x_n^\alpha\}$. Докажем лемму индукцией по построению формулы.

База индукции. Пусть формула A является переменной. В этом случае утверждение леммы $x_i^\alpha \vdash x_i^\alpha$ очевидно.

Индукционный переход. Предположим, что лемма верна для формул длины меньше n, где $n > 1$. Рассмотрим формулу A длины n.

(а) Пусть $A = \neg B$. Рассмотрим возможные значения формулы B на наборе α.

- $B(\alpha) = 0$, $B^\alpha = \neg B$, $A(\alpha) = 1$, $A^\alpha = \neg B$.

  Так как длина формулы B меньше n, по предположению индукции выполнено $\Gamma \vdash \neg B$, т. е. $\Gamma \vdash A^\alpha$.

- $B(\alpha) = 1$, $B^\alpha = B$, $A(\alpha) = 0$, $A^\alpha = \neg A = \neg\neg B$.

  По предположению индукции $\Gamma \vdash B$, поэтому достаточно доказать, что $B \vdash \neg\neg B$. Для этого используем формулу (б) из леммы 1.24 и правило modus ponens.

(б) Пусть $A = (B \to C)$. В зависимости от значений $B(\alpha)$, $C(\alpha)$ возможны такие случаи.

- $B(\alpha) = 0$, $B^\alpha = \neg B$, $A(\alpha) = 1$, $A^\alpha = A = B \to C$.

  По предположению индукции $\Gamma \vdash \neg B$, поэтому достаточно доказать, что $\neg B \vdash B \to C$. Это формула (1.3) из леммы 1.28.

- $B(\alpha) = 1$, $B^\alpha = B$, $C(\alpha) = 1$, $C^\alpha = C$, $A(\alpha) = 1$, $A^\alpha = A = B \to C$.

  По предположению индукции $\Gamma \vdash C$, поэтому достаточно доказать, что что $C \vdash B \to C$. Это следует из леммы 1.15.

- $B(\alpha) = 1$, $B^\alpha = B$, $C(\alpha) = 0$, $C^\alpha = \neg C$, $A(\alpha) = 0$, $A^\alpha = \neg A = \neg(B \to C)$.

  По предположению индукции $\Gamma \vdash B$, $\Gamma \vdash \neg C$, поэтому достаточно доказать, что $B, \neg C \vdash \neg(B \to C)$. Это формула (1.5) из леммы 1.28.

Лемма доказана. □

1.5.2. Второе доказательство

Приводимое ниже доказательство полноты исчисления высказываний можно обобщить на более интересный случай исчисления предикатов. Наше изложение близко следует книге [4].

Назовём множество формул Γ *противоречивым*, если из него можно вывести некоторую формулу A и её отрицание $\neg A$. В противном случае будем называть множество формул *непротиворечивым*.

Пример 1.30. Множество формул $\{x \to y, x, \neg y\}$ противоречиво. Оно содержит формулу $\neg y$, а формула y выводится применением modus ponens к двум гипотезам.

Из противоречивого множества можно вывести любую формулу.

Лемма 1.31. *Если* $\Gamma \vdash A$, $\Gamma \vdash \neg A$, *то для любой формулы* B *выполнено* $\Gamma \vdash B$.

Доказательство. Вывод формулы B получается добавлением к выводам формул A, $\neg A$ условного вывода из леммы 1.16. □

Множество формул Γ называется *полным*, если для любой формулы A из Γ выводима одна из формул A или $\neg A$. Если множество формул полное и непротиворечивое, то из него можно вывести в точности одну из формул A, $\neg A$.

План доказательства теоремы полноты такой. Для отрицания тавтологии $\neg T$ возможны два случая: (а) множество $\{\neg T\}$ противоречиво; (б) множество $\{\neg T\}$ непротиворечиво. В первом случае формула T выводима по лемме 1.26. Осталось доказать, что второй

случай невозможен. Для этого мы докажем существование такого набора значений переменных, на котором формула $\neg T$ истинна (см. ниже теорему 1.33), что противоречит тавтологичности T.

Определение 1.32. Множество формул Γ называется *совместным*, если существует такой набор α значений переменных, на котором истинны все формулы из Γ.

Теорема 1.33. *Любое непротиворечивое множество формул ИВ совместно.*

Доказательство этой теоремы проведём в два шага. Вначале ограничимся непротиворечивыми полными множествами формул.

Лемма 1.34. *Любое непротиворечивое и полное множество формул* ИВ *совместно.*

Доказательство. Для непротиворечивого полного множества формул Γ построим набор значений переменных по следующему правилу: если $\Gamma \vdash x$, то $x(\alpha)$ истинно, а если $\Gamma \vdash \neg x$, то $x(\alpha)$ ложно. Далее докажем индукцией по построению формулы, что все формулы, выводимые из множества Γ, истинны на наборе значений переменных α, а все формулы, невыводимые из множества Γ, ложны. Этого, разумеется, достаточно, так как из $A \in \Gamma$ следует, что $\Gamma \vdash A$.

База индукции: формулы, являющиеся переменными. В этом случае утверждение справедливо по определению набора α.

Индукционный переход. Предположим, что для формул длины меньше n доказываемое утверждение верно. Рассмотрим формулу A длины n.

(а) Пусть $A = \neg B$. Длина формулы B меньше n, значит, к ней применимо предположение индукции. Если $\Gamma \vdash A$, то из непротиворечивости множества Γ следует, что B невыводима из Γ. Поэтому по предположению индукции $B(\alpha)$ ложно, а $A(\alpha) = (\neg B)(\alpha)$ истинно. И наоборот, если $\Gamma \vdash \neg A$, то $\Gamma \vdash B$ (к выводу $\neg A = \neg\neg B$ добавим вывод тавтологии $\neg\neg B \to B$, который существует по лемме 1.24 (а), и применим к формулам $\neg\neg B$, $\neg\neg B \to B$ правило modus ponens). Из предположения индукции вытекает, что $B(\alpha)$ истинно, а $A(\alpha) = (\neg B)(\alpha)$ ложно.

(б) Пусть $A = (B \to C)$. Длина формул B и C меньше n, поэтому к ним применимо предположение индукции.

Пусть $\Gamma \vdash A$. Рассмотрим возможные случаи выводимости формулы B:

- $\Gamma \vdash \neg B$. По предположению индукции $B(\alpha) = 0$, т. е. $A(\alpha) = 1$.
- $\Gamma \vdash B$. Тогда и $\Gamma \vdash C$ (применим modus ponens к формулам B и $A = B \to C$). По предположению индукции $B(\alpha) = C(\alpha) = 1$, поэтому $A(\alpha) = 1$.

Пусть $\Gamma \vdash \neg A$. Поскольку $\vdash \neg(B \to C) \to \neg C$ (применим modus ponens к аксиоме $C \to (B \to C)$ и формуле (б) леммы 1.25), то $\Gamma \vdash \neg C$. Из формулы (1.6) леммы 1.28 заключаем, что $\Gamma \vdash B$. Значит, по предположению индукции $B(\alpha) = 1$, $C(\alpha) = 0$, откуда получаем $A(\alpha) = 0$. □

Для завершения доказательства теоремы 1.33 докажем, что всякое непротиворечивое множество содержится в полном и непротиворечивом множестве. Нам потребуется следующее вспомогательное утверждение.

Лемма 1.35. *Множество формул* ИВ *счётно.*

Доказательство. Счётность множества X равносильна тому, что существует последовательность

$$x_1, \ldots, x_n, \ldots,$$

содержащая все его элементы. В частности, любое подмножество множества натуральных чисел счётно. Отсюда следует, что и любое подмножество любого счётного множества счётно.

Множество формул ИВ содержится в множестве слов (конечных последовательностей) в счётном алфавите. Занумеруем символы алфавита положительными целыми числами. Осталось доказать, что множество конечных последовательностей положительных целых чисел счётно. Для этого построим отображение

$$g\colon (a_1, a_2, \ldots, a_n) \mapsto p_1^{a_1} p_2^{a_2} \ldots p_n^{a_n},$$

где $p_1, \ldots, p_n, \ldots$ — последовательные простые числа. По основной теореме арифметики отображение g инъективно (различным последовательностям сопоставляет различные числа). Значит, множество последовательностей можно занумеровать, нумеруя те натуральные числа, которые принадлежат образу отображения g. □

Лемма 1.36. *Любое непротиворечивое множество формул* Γ *содержится в некотором полном и непротиворечивом множестве формул* Γ'.

Доказательство. К непротиворечивому и неполному множеству формул Γ можно добавить хотя бы одну формулу, сохраняя непротиворечивость множества. Действительно, если ни формула A, ни её отрицание $\neg A$ невыводимы из Γ, то $\Gamma \cup \{\neg A\}$ непротиворечиво, так как из $\Gamma \cup \{\neg A\} \vdash B$ и $\Gamma \cup \{\neg A\} \vdash \neg B$ и теоремы дедукции

следует, что $\Gamma \vdash \neg A \to B$ и $\Gamma \vdash \neg A \to \neg B$. Добавляя к этим двум выводам третью аксиому

$$(\neg A \to \neg B) \to ((\neg A \to B) \to A)$$

и применяя дважды modus ponens, получаем вывод формулы A из множества формул Γ.

Пусть последовательность $F_1, F_2, \ldots, F_n, \ldots$ содержит все формулы. Построим бесконечную возрастающую последовательность множеств формул

$$\Gamma = \Gamma_0 \subseteq \Gamma_1 \subseteq \cdots \subseteq \Gamma_n \subseteq \ldots,$$

обладающую таким свойством: все множества в этой последовательности непротиворечивы и для любого $n \geqslant 1$ из множества Γ_n можно вывести хотя бы одну из формул F_i, $\neg F_i$ для любого $i \leqslant n$. Из сказанного выше следует, что для перехода от Γ_n к Γ_{n+1} достаточно добавить разве что одну из формул F_{n+1}, $\neg F_{n+1}$.

Множество $\Gamma' = \bigcup_{n=0}^{\infty} \Gamma_n$ будет искомым.

Полнота следует из построения: любая формула A ИВ найдётся в последовательности F_i, и если её номер в этой последовательности равен n, то уже из множества Γ_n можно вывести либо A, либо $\neg A$.

Непротиворечивость множества Γ' следует из того, что любой вывод состоит из конечного набора формул. Поэтому любой вывод из множества формул Γ' является также выводом из некоторого множества Γ_n. Поскольку все Γ_n непротиворечивы, то и Γ' непротиворечиво. □

1.5.3. Критерий условной выводимости

Теорема 1.37. $\Gamma \vdash A$ *в* ИВ *тогда и только тогда, когда формула* A *истинна на любом наборе значений переменных, на котором истинны все формулы из множества* Γ.

Доказательство. В одну сторону доказательство повторяет доказательство корректности ИВ: аксиомы и гипотезы истинны на любом наборе значений переменных, на котором истинны все формулы из Γ, применение правила modus ponens сохраняет это свойство.

Чтобы доказать утверждение теоремы в обратную сторону, рассмотрим вначале случай конечного множества

$$\Gamma = \{B_1, \ldots, B_m\}.$$

Пусть формула A истинна на любом наборе значений, на котором истинны все B_i. Многократным применением теоремы дедукции устанавливаем, что $\Gamma \vdash A$ равносильно $\vdash C$, где

$$C = \Big(B_1 \to \big(B_2 \to \ldots \to (B_m \to A)\big)\Big).$$

Формула C является тавтологией: она истинна, если хотя бы одна из формул B_i ложна, а если все B_i истинны, то A также истинна по условию теоремы, так что формула C также истинна. Из теоремы полноты заключаем, что формула C выводима.

Случай бесконечного множества формул сведём к случаю конечного множества. Если формула A истинна на любом наборе значений, на котором истинны все формулы из множества Γ, то для каждого набора α значений входящих в формулу A переменных, на котором A ложна, можно указать формулу $B_\alpha \in \Gamma$, которая также ложна на α. Переменные, входящие в формулу A, могут принимать лишь конечное число различных значений, поэтому множество формул $\{B_\alpha\}$ конечно и по его построению формула A истинна на любом наборе значений, на котором истинны все формулы из множества $\{B_\alpha\}$. Значит, $\{B_\alpha\} \vdash A$. $\square$

Задачи

1.15. Проверьте выполнение следующих утверждений о выводимости в исчислении высказываний

а) $a \to \neg b \vdash \neg b \to a$;

б) $a \to b \vdash (a \to c) \to b$;

в) $\neg((x \to y) \to (y \to z)) \vdash x \to z$;

г) $(\neg x \to y) \to x \vdash x \to (\neg y \to \neg x)$;

д) $\neg((x \to y) \to (y \to z)) \vdash x \to z$.

1.16. Пусть Γ — множество всех формул исчисления высказываний, в которые не входят отрицания. Докажите, что

$$\Gamma \vdash (\neg x \to y) \to \neg(x \to \neg y).$$

1.17. Формулы F_1 и F_2 невыводимы в исчислении высказываний. Верно ли, что невыводима формула $\neg(F_1 \to \neg F_2)$?

1.18. Формулы A, B выводимы в исчислении высказываний. Докажите, что любая формула вида $((C \to A) \to \neg B) \to C$ также выводима в исчислении высказываний.

1.19. Множество формул Γ состоит из всех формул вида $x_1 \to \neg x_n$, $n = 2, 3, \ldots$. Верно ли, что

а) формула $x_2 \to \neg x_1$ выводима из Γ?

б) формула $\neg x_2 \to \neg x_1$ выводима из Γ?

1.20. Противоречиво ли множество формул Γ, которые имеют вид $\neg A \to B$, где A, B — произвольные формулы?

1.21. Для каких n существует такое противоречивое множество из n формул, что любое его подмножество из $n-1$ формулы непротиворечиво?

1.22. Проверьте полноту множества формул Γ, если

а) Γ включает в себя формулы $x_i \to y_i$, $y_i \to z_i$,

$z_i \to \neg x_i$, $\neg x_i \to z_i$, $z_i \to \neg y_i$, $\neg y_i \to x_i$? (x_i, y_i, z_i — переменные);

б) Γ включает в себя формулы вида $A \to \neg A$, где A — произвольная формула;

в) Γ включает в себя формулы вида $A \to B$, где A, B — произвольные формулы.

г) Γ включает в себя формулы вида $x \to A$, где x — некоторая переменная, а A — произвольная формула.

1.23. Верно ли, что существует полное и непротиворечивое множество формул, в каждую формулу которого входит более 2 переменных?

1.6. Независимость аксиом ИВ

Свойство независимости аксиом формальной системы означает, что никакую из аксиом нельзя вывести из других. Для выбранной нами формальной системы ИВ это свойство выполняется.

Если говорить более строго, то свойство независимости можно сформулировать так. Рассмотрим формальную систему ИВ $\setminus$ 3, которая отличается от рассмотренной нами формальной системы ИВ тем, что в ней есть только схемы аксиом (А1) и (А2). Независимость третьей схемы аксиом означает, что в ИВ $\setminus$ 3 не выводится формула

$$(\neg B \to \neg A) \to ((\neg B \to A) \to B). \qquad (1.12)$$

Аналогично формулируются и два других свойства независимости.

Доказательства независимости будут опираться на общий метод, который часто применяется для доказательства независимости аксиом. Основная идея состоит в том, чтобы найти некоторую *модель* (или интерпретацию), для которой верна данная аксиоматика,

и в данной модели найти какой-либо инвариант, присущий всем аксиомам, кроме испытуемой. Далее проверяется, что правила вывода не изменяют значения инварианта, а отсюда, как следствие, имеем, что интересующую нас аксиому нельзя вывести из остальных. Таким образом, к примеру, и была доказана независимость V постулата Евклида (аксиомы о параллельных прямых) от остальных аксиом геометрии.

В случае исчисления высказываний мы будем строить модели, используя функции, которые зависят от переменных, принимающих значения в некотором множестве M, и принимают значения в том же множестве M. Если заданы функции $f_\neg(x)$ и $f_\to(x,y)$, соответствующие отрицанию и импликации, то для каждой формулы ИВ определим функцию, которая реализуется этой формулой, так же, как это было сделано в разделе 1.2: переменной x_i соответствует тождественная функция $[x_i]\colon x_i \mapsto x_i$; если формула A имеет вид $\neg B$, то $[A] = f_\neg([B])$; а если $A = B{\to}C$, то $[A] = f_\to([B],[C])$. Мы здесь используем обозначение $[A]$ для функции, реализуемой формулой A.

Теорема 1.38. *Третья аксиома не зависит от первых двух.*

Доказательство. Зададим модель булевыми функциями $f_\neg(x) = x$, $f_\to(x,y) = (x{\to}y)$. Функции, которые получаются из первых двух схем аксиом подстановкой вместо отрицания и импликации функций $f_\neg$ и $f_\to$, являются тавтологиями (тождественно равны 1). Действительно, в эти аксиомы отрицание вообще не входит, а $f_\to$ совпадает с импликацией. Поэтому можно утверждать, что в формальной системе ИВ $\setminus$ 3 выводимы только тавтологии.

Теперь подставим функции $f_\neg$ и $f_\to$ в формулу (A3), соответствующую третьей аксиоме:

$$[(A3)] = f_\to(f_\to(f_\neg(B), f_\neg(A)), f_\to(f_\to(f_\neg(B), A), B)) = \\ = (B\to A)\to((B\to A)\to B),$$

где в последней формуле использована стандартная интерпретация формул ИВ. Вычисление

$$(0\to 0)\to((0\to 0)\to 0) = 1\to(1\to 0) = 1\to 0 = 0$$

показывает, что эта функция не является тавтологией. Значит, формула (A3) невыводима в системе ИВ $\setminus 3$. □

Теорема 1.39. *Первая аксиома не зависит от второй и третьей.*

Доказательство. Построим модель с множеством значений $\{0, 0', 1\}$. Такие странные обозначения объясняются выбором функций $f_\neg$, $f_\to$: при ограничении на множество $\{0, 1\}$ они принимают те же значения, что и обычные булевы отрицание и импликация (см. таблицу 1.2).

Проверим, что формула (A2)

$$(x\to(y\to z))\to((x\to y)\to(x\to z))$$

(вторая схема аксиом) реализует в такой модели функцию, тождественно равную 1. Значение $f_\to(x, y)$ никогда не равно $0'$, поэтому значение формулы (A2) в данной модели отлично от 1 только в том случае, когда $[x\to(y\to z)] = 1$, а $[(x\to y)\to(x\to z)] = 0$. Применяя ко второму равенству то же самое рассуждение, получаем условия:

$$\begin{cases} [x\to(y\to z)] = 1; \\ [x\to y] = 1; \\ [x\to z] = 0. \end{cases} \tag{1.13}$$

Таблица 1.2. Модель для независимости первой аксиомы

x	$f_{\neg}(x)$
0	1
$0'$	1
1	0

x	y	$f_{\to}(x,y)$
0	0	1
0	$0'$	1
0	1	1
$0'$	0	0
$0'$	$0'$	0
$0'$	1	1
1	0	0
1	$0'$	0
1	1	1

Если $x = 1$, то из второго условия в (1.13) получаем $y = 1$, а из первого $[y{\to}z] = [1{\to}z] = 1$, что противоречит третьему условию. Если $x = 0'$, то из второго условия $y = 1$, но тогда из третьего условия $z \neq 1$. Значит, $[y{\to}z] \neq 1$, $[x{\to}(y{\to}z)] = 0$, а это противоречит первому условию. Случай $x = 0$ противоречит третьему условию. Несовместность условий (1.13) показывает, что значение функции, реализуемой формулой (A2) тождественно равно 1.

Проверим то же свойство для формулы (A3)

$$(\neg x{\to}\neg y){\to}((\neg x{\to}y){\to}x).$$

Аналогично предыдущему рассуждению убеждаемся, что если значение формулы (A3) не равно 1, то $x \neq 1$, а $[\neg x] = 1$. Но $f_{\to}(x,0) = f_{\to}(x,0')$ при любом x, поэтому формула (A3) в этом случае принимает такие же значения, как и булева функция в стандартной модели (и, значит, равна 1).

Таблица 1.3. Модель для независимости второй аксиомы

x	$f_\neg(x)$
0	1
1	0
2	1

x	y	$f_\to(x,y)$
0	0	0
0	1	2
0	2	1
1	0	0
1	1	2
1	2	0
2	0	0
2	1	0
2	2	0

Кроме того, из таблицы 1.2 легко видеть, что если $x = 1$ и $f_\to(x,y) = 1$, то $y = 1$. Значит, применение правила вывода сохраняет формулы, реализующие тождественно равные 1 функции.

Поэтому из аксиом (A2) и (A3) можно вывести лишь формулы, реализующие тождественно равные 1 функции.

Для завершения доказательства заметим, что

$$[0'\to(1\to0')] = [0'\to0] = 0\,.$$

Поэтому формула $(x\to(y\to x))$ реализует функцию, которая не равна 1 тождественно $\square$

Теорема 1.40. *Вторая аксиома не зависит от первой и третьей.*

Доказательство. Здесь мы выберем множеством значений функций $\{0,1,2\}$, а функции $f_\neg(x)$ и $f_\to(x,y)$ зададим таблицей 1.3.

Докажем, что формула $x{\to}(y{\to}x)$ (первая схема аксиом) реализует функцию, тождественно равную 0. Из таблицы 1.3 видно, что $f_{\to}$ принимает отличное от 0 значение на трёх наборах переменных $(0,1)$, $(0,2)$ и $(1,1)$. Но из таблицы видно, что если $x = 0$, то $[y{\to}x] = [y{\to}0] = 0$; а если $x = 1$, то $[y{\to}x] = [y{\to}1] \neq 1$.

Аналогичным образом проверим, что формула

$$(\neg x{\to}\neg y){\to}((\neg x{\to}y){\to}x)$$

(третья схема аксиом) также реализует функцию, тождественно равную 0. Рассмотрим возможные варианты, когда внешняя импликация даёт отличное от 0 значение.

- Случай $(1,1)$: $[\neg x{\to}\neg y] = 1$ влечёт $[\neg x] = 0$, $[\neg y] = 2$, но последнее равенство невозможно.
- Случай $(0,1)$: если $[(\neg x{\to}y){\to}x] = 1$, $[\neg x{\to}\neg y] = 0$, то $x = 2$, $[\neg x] = 1$, $[\neg x{\to}y] = [1{\to}y] = 0$, $[\neg x{\to}\neg y] = [1{\to}\neg y] = 0$. Из последнего равенства заключаем, что $[\neg y] = 0$, $y = 1$. Пришли к противоречию, так как $[1{\to}1] \neq 0$.
- Случай $(0,2)$: если $[(\neg x{\to}y){\to}x] = 2$, то $x = 1$, а $[\neg x{\to}y]$ принимает значения 0 или 1. Значит, $[\neg x{\to}y] = [0{\to}y]$, а y принимает значения 0 или 2. Поэтому $[\neg y] = 1$. Но тогда $[\neg x{\to}\neg y] = [0{\to}1] = 2 \neq 0$.

Кроме того, из таблицы легко видеть, что если $x = 0$ и $f_{\to}(x,y) = 0$, то $y = 0$. Значит, применение правила вывода сохраняет формулы, реализующие тождественно равные 0 функции.

Поэтому из аксиом (A1) и (A3) можно вывести лишь такие формулы, которые задают функции, тождественно равные 0.

Проверим, что соответствующая второй схеме аксиом формула $(x{\to}(y{\to}z)){\to}((x{\to}y){\to}(x{\to}z))$ реализует функцию, которая не равна 0 тождественно:

$$[(0{\to}(0{\to}1)){\to}((0{\to}0){\to}(0{\to}1))] = [(0{\to}2){\to}(0{\to}2)] = \\ = [1{\to}1] = 2.$$

Независимость второй схемы аксиом доказана. □

1.7. Другие аксиоматизации исчисления высказываний

Выше мы построили формальную систему, в которой выводимы тавтологии и только они. Такая система, разумеется, не единственна. Можно изменять наборы пропозициональных связок, входящих в формулы, наборы аксиом или правила вывода.

Если, например, не заботиться о независимости аксиом, то можно включить в схемы аксиом все формулы, которые использовались для доказательства корректности и полноты ИВ (их конечное число). Получим формальную систему, в которой выводимы тавтологии и только они, причём доказательство полноты значительно упрощается, а в доказательстве корректности нужно будет проверить тавтологичность всех этих формул.

В рассмотренной нами формальной системе ИВ представлены только две пропозициональные связки $\neg$ и $\to$. Чтобы записать высказывание, которое является дизъюнкцией высказываний A и B, в такой формальной системе нужно написать формулу $\neg A{\to}B$. Поскольку $\neg$ и $\to$ образуют полную систему функций, то остальные связки также реализуются формулами ИВ. Такой подход несколько сокращает доказательства, но в других отношениях неудобен.

Наиболее естественным набором пропозициональных связок является набор $\{\neg, \wedge, \vee, \to\}$ (мы перечислили связки в порядке убывания старшинства). Приведём формальную систему КИВ (классическое исчисление высказываний), в которой используется этот набор, причём выводимы в точности тавтологии. Алфавит КИВ состоит из счётного множества переменных, пропозициональных связок $\neg, \wedge, \vee, \to$ и скобок. Определение формулы для КИВ почти дословно повторяет данное для ИВ, только добавляются ещё две возможности построения формул $(A \vee B)$ и $(A \wedge B)$. Правило вывода остаётся прежним: modus ponens. А вот список схем аксиом изменяется и становится намного длиннее:

$$(A{\to}(B{\to}A)), \tag{P1}$$
$$((A{\to}(B{\to}C)){\to}((A{\to}B){\to}(A{\to}C))), \tag{P2}$$
$$((A \wedge B){\to}A), \tag{P3}$$
$$((A \wedge B){\to}B), \tag{P4}$$
$$(A{\to}(B{\to}(A \wedge B))), \tag{P5}$$
$$(A{\to}(A \vee B)), \tag{P6}$$
$$(B{\to}(A \vee B)), \tag{P7}$$
$$((A{\to}C){\to}((B{\to}C){\to}((A \vee B){\to}C))), \tag{P8}$$
$$((\neg A){\to}(A{\to}B)), \tag{P9}$$
$$((B{\to}(\neg A)){\to}((B{\to}A){\to}(\neg B))), \tag{P10}$$
$$(A \vee (\neg A)). \tag{P11}$$

Несмотря на большое количество аксиом, проверка корректности КИВ не вызывает затруднений. Оба подхода к доказательству теоремы о полноте по прежнему применимы. Рассуждения становятся, конечно, длиннее, но их суть не меняется.

Вернёмся к формулам, в которые входят только связки $\neg$, $\to$. Можно ли построить формальную систему, в которой выводимы в точности тавтологии и в которую входит лишь конечное число аксиом? (В системе ИВ аксиом бесконечно много.) Легко понять, что это невозможно, если ограничиться только правилом вывода modus ponens: в конечное число аксиом будет входить лишь конечное число переменных, поэтому выводимыми будут лишь формулы, использующие эти переменные, а тавтология может содержать сколь угодно большое число переменных.

Добавим ещё одно правило вывода — правило подстановки. Оно состоит в том, что если выводима формула A, то выводимы также формулы, которые получаются из A подстановкой вместо переменных произвольных формул (разумеется, каждое вхождение переменной нужно заменять на одну и ту же формулу).

Выберем в качестве аксиом три формулы (А1–А3). Полученную формальную систему назовём исчислением высказываний с подстановкой. В этой системе по-прежнему выводимы в точности тавтологии.

Корректность исчисления высказываний с подстановкой вытекает из того, что подстановка сохраняет тавтологичность формул.

Любой вывод в ИВ можно преобразовать в вывод в исчислении высказываний с подстановкой, заменяя каждое использование схемы аксиом на использование соответствующей аксиомы и последующую подстановку. Отсюда следует полнота исчисления высказываний с подстановкой.

Непосредственный анализ исчисления высказываний с подстановкой оказывается менее удобным. В частности, здесь не выполняется теорема дедукции.

Действительно, из формулы x можно получить подстановкой любую формулу A, поэтому $x \vdash A$. Однако не любая формула вида $x \to A$ является тавтологией, например, $x \to y$ — не тавтология.

Задачи

1.24. Проверьте корректность (выводимы только тавтологии) и полноту (все тавтологии выводимы) формальных систем, в которых множество формул совпадает с множеством формул ИВ, правило вывода — modus ponens, а схемы аксиом имеют вид

а) (B1) $A \to A$, (B2) $(A \to B) \to (\neg B \to \neg A)$, (B3) $\neg\neg A \to A$, (B4) $A \to \neg\neg A$;

б) (A1), (A2), (C3) $A \to (\neg A \to B)$.

1.25. Найдите конечный набор $\mathcal{A}$ схем аксиом такой, что в формальной системе с тем же множеством формул, что и в ИВ, со схемами аксиом $\mathcal{A}$ и правилом вывода

$$\frac{A, \quad A \to (\neg B \to \neg A)}{B}$$

выводимы тавтологии и только они.

1.26. Изменится ли множество выводимых формул, если к ИВ добавить ещё одно правило вывода

$$\frac{A, \quad (B \to A) \to A}{A \to B}?$$

1.8. Метод резолюций

В силу корректности ИВ построение вывода даёт доказательство тавтологичности формулы. Однако вывод, который строится в соответствии с первым доказательством теоремы полноты, требует фактически перебора всех возможных значений переменных. Это

ничем не лучше прямого использования определения тавтологии.

В этом разделе мы обсудим другой способ доказательства тавтологичности формулы — метод резолюций. Этот способ во многих случаях позволяет доказывать тавтологичность быстрее, поэтому он (и его модификации для более сложных формальных систем) широко применяется в компьютерных системах построения доказательств.

Формула называется *выполнимой*, если существует хотя бы один набор значений переменных, на котором эта формула истинна. Метод резолюций проверяет выполнимость формул. Поскольку тавтологичность формулы F равносильна невыполнимости формулы $\neg F$, он также пригоден и для доказательства тавтологичности.

1.8.1. КНФ

Метод резолюций работает с формулами в конъюнктивной нормальной форме (КНФ). Напомним, что это такое. *Литерал* — это переменная или отрицание переменной. *Дизъюнкт* — это формула, которая является дизъюнкцией литералов. *Пустой дизъюнкт* по определению означает ложь. *Конъюнктивная нормальная форма* (КНФ) — это формула, которая является конъюнкцией дизъюнктов.

Заметим, что переход к КНФ не ограничивает общности метода.

В самом деле, по любой формуле A с переменными $x_1, \ldots, x_n$ можно построить равносильную ей КНФ. Для этого по каждому набору значений переменных $\tilde{\alpha} = (\alpha_1, \ldots, \alpha_n)$, на котором A ложна, построим дизъюнкт

$$D(\tilde{\alpha}) = \neg x_1^{\alpha_1} \vee \neg x_2^{\alpha_2} \vee \cdots \vee \neg x_n^{\alpha_n},$$

где, как обычно, $x^1 = x$, $x^0 = \neg x$. Так как $x^x = 1$, а $x^{\neg x} = 0$, дизъюнкт $D(\tilde{\alpha})$ обращается в 0 в точности на наборе $\tilde{\alpha}$. Поэтому конъюнкция $D(\tilde{\alpha})$ по всем наборам значений переменных, на которых формула A ложна, равносильна A:

$$A \sim \bigwedge_{\tilde{\alpha}:A(\tilde{\alpha})=0} D(\tilde{\alpha}). \tag{1.14}$$

Формула (1.14) отвечает на вопрос о равносильности произвольных формул и КНФ. Но КНФ из формулы (1.14) в общем случае очень длинная: нулей у булевой функции от n переменных может быть 2^n. Если нас интересует лишь выполнимость формулы A, то довольно просто написать более короткую КНФ, *выполнимость* которой равносильна *выполнимости* формулы A. Для каждой подформулы B формулы A (см. определение на с. 21) введём переменную z_B. Если подформула B имеет вид $\neg D$, то построим по (1.14) КНФ, равносильную формуле $z_B \sim \neg z_D$. Если подформула B имеет вид $D \to E$, то построим по (1.14) КНФ, равносильную формуле $z_B \sim (z_D \to z_E)$. Конъюнкция этих КНФ и дизъюнкта z_A образует КНФ $C(A)$, выполнимость которой равносильна выполнимости формулы A. По индукции легко проверить, что истинность КНФ $C(A)$ на наборе значений $z_X = \alpha_X$, где X пробегает все подформулы A, в том числе и переменные, входящие в формулу A, равносильна тому, что значение любой подформулы B на наборе значений переменных $x_i = \alpha_{x_i}$ равно z_B, причём значение z_A равно 1 (т. е. формула A выполнима на наборе $x_i = \alpha_{x_i}$).

Количество дизъюнктов в КНФ $C(A)$ легко оценить через количество связок m формулы A. Каждой связке в формуле A соответствует КНФ, равносильная

формуле из не более чем 3 переменных. У булевой функции от 3 переменных не более 8 нулей. Значит, общее количество дизъюнктов в $C(A)$ не превосходит $1 + 8m$.

1.8.2. Резолютивный вывод

Далее в этом разделе мы рассматриваем вопрос о выполнимости КНФ.

Поскольку $a \vee a \sim a \wedge a \sim a$, мы предполагаем в дальнейшем, что все дизъюнкты не содержат повторений литералов и попарно различны. Другими словами это означает, что мы представляем КНФ как совокупность различных дизъюнктов $D_1, \ldots, D_n$, каждый из которых в свою очередь является множеством различных литералов. Для удобства будем допускать и пустое множество литералов, которому соответствует тождественно ложный *пустой дизъюнкт*. Далее в этом разделе мы всюду говорим о дизъюнктах как о множествах литералов.

Определение 1.41. *Резольвентой* пары дизъюнктов вида $\{x\} \cup A$ и $\{\neg x\} \cup B$ относительно переменной x называется дизъюнкт $A \cup B$. Здесь $\{x^\alpha\}$ обозначает дизъюнкт, состоящий в точности из литерала x^α.

Заметим, что если A и B содержат пару противоположных литералов y и $\neg y$, то резольвента $A \cup B$ является тавтологией.

Определение 1.42. Пусть $\mathcal{D}$ — множество дизъюнктов. *Резолютивный вывод* дизъюнкта D из $\mathcal{D}$ — это такая последовательность дизъюнктов $R_1, \ldots, R_n$, что $R_n = D$ и каждый член R_i этой последовательности либо принадлежит $\mathcal{D}$, либо является резольвентой каких-то предыдущих дизъюнктов R_j, R_k, $j, k < i$.

Резолютивный вывод пустого дизъюнкта из $\mathcal{D}$ называется опровержением $\mathcal{D}$. (Как будет показано ниже, опровержение является фактически доказательством невыполнимости $\mathcal{D}$.)

Теорема 1.43 (корректность метода резолюций). *Если существует резолютивный вывод пустого дизъюнкта из дизъюнктов КНФ C, то КНФ невыполнима.*

Доказательство. Пусть КНФ $\mathcal{D}'$ получена из КНФ $\mathcal{D}$ добавлением резольвенты пары дизъюнктов $\{x\} \cup A$ и $\{\neg x\} \cup B$, которые входят в $\mathcal{D}$.

Выполнимость $\mathcal{D}'$ равносильна выполнимости $\mathcal{D}$, так как формулы

$$(x \vee A) \wedge (\neg x \vee B) \text{ и } (x \vee A) \wedge (\neg x \vee B) \wedge (A \vee B)$$

эквивалентны: чтобы первая формула была истинной, должна быть истинна одна из формул A или B. Это означает, что добавление к КНФ дизъюнктов любого резолютивного вывода из множества дизъюнктов этой КНФ не меняет выполнимости КНФ. Поэтому из резолютивного вывода пустого дизъюнкта следует невыполнимость КНФ. □

Назовём множество дизъюнктов $\mathcal{D}$ *полным непротиворечивым*, если $\mathcal{D}$ не содержит пустого дизъюнкта и любая резольвента дизъюнктов из $\mathcal{D}$ либо тавтологична, либо содержится в $\mathcal{D}$.

Лемма 1.44. *КНФ с полным непротиворечивым множеством дизъюнктов выполнима.*

Доказательство. Конъюнкция ложна, если ложен хотя бы один её член. Поэтому знания значений части переменных КНФ бывает достаточно, чтобы сделать

вывод о ложности КНФ на таком частично определённом наборе значений. Точнее говоря, если известно, что ложны литералы $\ell_1, \ldots, \ell_k$ и в КНФ входит дизъюнкт $D \subseteq \{\ell_1, \ldots, \ell_k\}$, то КНФ ложна.

Построим выполняющий набор для полного непротиворечивого множества дизъюнктов $\mathcal{D}$. Будем присваивать значения переменным по очереди. Для $i = 1, \ldots, n$ полагаем $x_i = 1$, если на частично определённом наборе значений $x_1 = \alpha_1, \ldots, x_{i-1} = \alpha_{i-1}$, $x_i = 0$ КНФ ложна; в противном случае полагаем $x_i = 0$. Заметим, что это правило применимо и к первой переменной.

Предположим, что КНФ ложна на построенном наборе значений α. Выберем наименьшее k, для которого КНФ ложна на (частичном) наборе $x_i = \alpha_i$, $i = 1, \ldots, k$. По построению это означает, что КНФ ложна на обоих наборах $\alpha_1, \ldots, \alpha_{k-1}, 0$ и $\alpha_1, \ldots, \alpha_{k-1}, 1$. Поэтому найдутся такие дизъюнкты D_1, D_2 из $\mathcal{D}$, что

$$D_1 \subseteq \{x_1^{\neg\alpha_1}, \ldots, x_{k-1}^{\neg\alpha_{k-1}}\} \cup \{x_k\},$$
$$D_2 \subseteq \{x_1^{\neg\alpha_1}, \ldots, x_{k-1}^{\neg\alpha_{k-1}}\} \cup \{\neg x_k\}.$$

В силу выбора k ни один дизъюнкт из $\mathcal{D}$ не содержится в множестве $L = \{x_1^{\neg\alpha_1}, \ldots, x_{k-1}^{\neg\alpha_{k-1}}\}$ (иначе КНФ была бы ложна на более коротком частичном наборе значений переменных). С другой стороны, D_1 содержит x_k, D_2 содержит $\neg x_k$, а их резольвента относительно x_k содержится в L. Эта резольвента нетавтологична в силу выбора D_1, D_2 и не принадлежит $\mathcal{D}$. Получили противоречие с полнотой $\mathcal{D}$, которое и доказывает истинность КНФ на построенном наборе значений α. □

Теорема 1.45 (полнота метода резолюций). *Если КНФ C невыполнима, то существует резолютивный вывод пустого дизъюнкта из дизъюнктов КНФ C.*

Доказательство. Рассуждаем следующим образом. Выводим из дизъюнктов C все возможные резольвенты, пока это возможно и не выведен пустой дизъюнкт. Если процесс завершится выводом пустого дизъюнкта, то мы получили искомый резолютивный вывод. В противном случае процесс остановится на полном непротиворечивом множестве дизъюнктов, которое выполнимо в силу предыдущей леммы. □

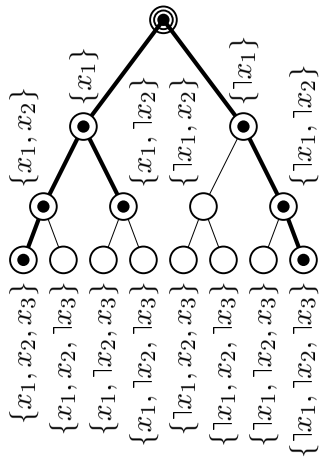
1.8.3. Синтаксическое дерево и допустимое дерево

Приведённое выше доказательство полноты метода резолюций становится нагляднее, если рассмотреть *синтаксическое дерево*.

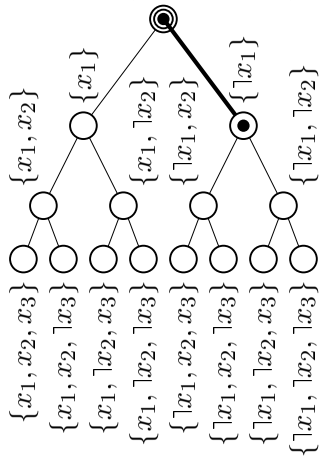
Пусть в КНФ C входит n переменных $x_1, \ldots, x_n$. Рассмотрим дизъюнкты вида $\{\ell_1, \ldots, \ell_k\}$, где $k \geqslant 0$, а литерал ℓ_i — это переменная x_i или её отрицание. Совокупность таких дизъюнктов образует *полное корневое бинарное дерево* глубины n, которое и называется синтаксическим деревом. Корнем является пустой дизъюнкт, а у вершины $\{\ell_1, \ldots, \ell_k\}$ есть два потомка: $\{\ell_1, \ldots, \ell_k, x_{k+1}\}$ и $\{\ell_1, \ldots, \ell_k, \neg x_{k+1}\}$.

Вершинами *допустимого дерева* для КНФ $\mathcal{D}$ являются те дизъюнкты указанного выше вида, которые не включают как множество ни одного дизъюнкта из $\mathcal{D}$.

Чтобы наглядно представить себе эту конструкцию, нужно считать, что в каждой вершине ложен соответствующий ей дизъюнкт. Тогда недопустимые вершины — это те вершины, в которых мы можем утверждать ложность КНФ. На рис. 1.4 показаны примеры. Тонкими линиями на этих рисунках нарисованы рёбра полного бинарного дерева, а толстыми — рёбра



(а) $(\neg x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_3 \vee x_1)$



(б) $x_1 \wedge (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$

Рис. 1.4. Примеры допустимых деревьев

допустимых деревьев для указанных КНФ, вершины допустимых деревьев помечены точками.

Эти наводящие соображения помогают понять доказательство следующей простой леммы.

Лемма 1.46. *КНФ выполнима тогда и только тогда, когда допустимое дерево имеет глубину n.*

Доказательство. Пусть КНФ выполнима на наборе $(\alpha_1, \alpha_2, \ldots, \alpha_n)$. Тогда вершина, помеченная

$$\{x_1^{\neg\alpha_1}, \ldots, x_n^{\neg\alpha_n}\}, \tag{1.15}$$

допустима, так как в каждом дизъюнкте КНФ должен найтись литерал $x_i^{\alpha_i}$. И в обратную сторону: если вершина

$$\{x_1^{\alpha_1}, \ldots, x_n^{\alpha_n}\}, \tag{1.16}$$

допустима, то ни один дизъюнкт КНФ не содержится в множестве (1.15), следовательно, содержит хотя бы один литерал $x_i^{\neg\alpha_i}$. Это и означает выполнимость КНФ на наборе $(\neg\alpha_1, \ldots, \neg\alpha_n)$. □

В доказательстве леммы 1.44 мы фактически строили самый левый путь глубины n для полного непротиворечивого множества дизъюнктов.

1.8.4. Сравнение метода резолюций и ИВ

Мы привели конструктивные доказательства полноты исчисления высказываний и полноты метода резолюций. Оба они используют перебор по полному бинарному дереву. Но доказательство полноты для ИВ перебирает дерево в направлении от листьев к корню и всегда даёт экспоненциально длинный вывод, а в построении резолютивного вывода используется лишь допустимое дерево, которое может быть намного меньше.

Есть и другие причины, по которым метод резолюций может давать достаточно короткие опровержения.

Пример 1.47 (2-*КНФ*). Если каждый дизъюнкт содержит не более двух литералов, то такая КНФ называется 2-КНФ. Резольвента дизъюнктов, в каждый из которых входит не более двух литералов, также содержит не более двух литералов. Поэтому резолютивный вывод для 2-КНФ не может быть очень длинным — из n переменных можно образовать всего $O(n^2)$ дизъюнктов, содержащих не более двух литералов.

Однако если сравнивать возможности двух систем вывода: исчисления высказываний и метода резолюций, то картина получится обратная.

Если для формулы существует короткое опровержение методом резолюций, то для отрицания этой формулы существует не слишком длинный вывод в ИВ.

С другой стороны, известны примеры формул, для которых не существует коротких опровержений методом резолюций, но существуют короткие выводы их отрицаний в исчислении высказываний.

Отметим, что неизвестен эффективный алгоритм поиска кратчайшего вывода в исчислении высказываний, причём скорее всего такого алгоритма и вовсе не существует.

Задачи

1.27. Докажите выполнимость КНФ, которая состоит из 200 дизъюнктов, каждый из которых содержит 8 различных литералов.

1.28. Назовём КНФ C замкнутой по длине, если она не содержит пустого дизъюнкта, а резольвента любых

её двух дизъюнктов (1) либо тавтологична; (2) либо содержится в КНФ; (3) либо содержит больше литералов, чем любой дизъюнкт из C. Приведите пример замкнутой по длине невыполнимой КНФ с $2n-1$ переменными, все дизъюнкты которой содержат не более n литералов ($n \geqslant 3$).

1.29. Проверьте методом резолюций выполнимость КНФ

$$(\neg a \vee b) \wedge (a \vee \neg b);$$
$$(\neg a \vee \neg b \vee c) \wedge (\neg a \vee b) \wedge a \wedge \neg c;$$
$$(\neg c \vee \neg b \vee a) \wedge (b \vee c) \wedge (\neg a \vee c) \wedge$$
$$\wedge (\neg d \vee \neg a \vee c) \wedge (a \vee d) \wedge (\neg c \vee d);$$
$$(\neg a \vee b) \wedge (c \vee \neg a) \wedge (\neg b \vee \neg c) \wedge (a \vee c) \wedge$$
$$\wedge (a \vee b) \wedge (\neg c \vee \neg b).$$

Глава 2

Интуиционистское исчисление высказываний

В этой главе мы рассмотрим формальные системы, которые внешне похожи на формализации исчисления высказываний (системы ИВ и КИВ), но тем не менее значительно от них отличаются. Эти системы являются формальными аналогами *интуиционистской логики высказываний*.

Интуиционизм — это такое направление в основаниях математики, которое запрещает использование некоторых способов рассуждения, применяемых в классической математике. Наиболее ярким примером является запрет на использование принципа исключённого третьего. Основанные на этом принципе доказательства неконструктивны. Приведём один из самых известных примеров такого рода.

Задача 2.1. Докажите, что существуют два таких иррациональных числа a, b, что a^b рационально.

Решение. Рассмотрим число $a = (\sqrt{2})^{\sqrt{2}}$. Если оно рационально, то утверждение доказано. В противном случае $a^{\sqrt{2}} = (\sqrt{2})^2 = 2$ рационально.

Хотя мы доказали утверждение, мы не указали никакой пары искомых чисел. Рассуждения такого типа

интуиционисты отвергают. В частности, они считают дизъюнкцию двух утверждений доказанной лишь в том случае, когда предъявлено доказательство одного из них.

Другой яркий пример ограничений в рассуждениях интуиционистов — запрет на использование закона двойного отрицания. Предположим, что доказано утверждение «неверно, что не существует алгоритма для решения задачи A». Классическая логика позволяет сделать вывод, что такой алгоритм существует. Интуиционисты отказываются признавать этот вывод.

Мы не будем рассматривать интуиционистскую математику и её связи с формальными системами, приводимыми ниже. Наша цель состоит в построении подходящей семантики для таких формальных систем.

Для построения семантики нужно придумать модель и описать в ней множество выводимых формул. Ниже мы приводим найденное Крипке решение этой задачи. Оно применимо ко многим формальным системам (модальная логика и т. п.), важным в приложениях.

2.1. Ограниченное исчисление ИИВ$_{\to}$

Рассмотрим вначале ограниченное интуиционистское исчисление высказываний ИИВ$_{\to}$. Эта формальная система является аналогом формальной системы ИВ. Её определение даётся по описанному во введении шаблону.

Алфавит ИИВ$_{\to}$ состоит из

- счётного множества L пропозициональных букв;
- пропозициональной связки $\to$ (импликация);
- символа лжи $\bot$;
- скобок — открывающей (и закрывающей).

Формулы ИИВ$_{\to}$ определяются аналогично формулам ИВ: формула является либо переменной, либо $\bot$, либо имеет вид $(A{\to}B)$. Более подробное определение формулы ИИВ$_{\to}$ выглядит так.

Слово A в алфавите ИИВ$_{\to}$ является формулой тогда и только тогда, когда выполняется одно их трёх условий

1: A является переменной;

2: A является символом лжи;

3: A является импликацией, т. е. имеет вид $(B{\to}C)$.

Замечание 2.1. Наличие символа $\bot$ позволяет обойтись без отрицания. В дальнейшем мы используем $\neg A$ как сокращение для $A{\to}\bot$. Аналогично можно было бы поступить и в случае классического исчисления высказываний.

Аксиом в формальной системе ИИВ$_{\to}$ бесконечно много. Они задаются, как в случае ИВ, тремя схемами.

Формула является аксиомой ИИВ$_{\to}$, если её можно представить одним из следующих способов (используем сокращённую запись формул):

$$A{\to}(B{\to}A), \tag{A1}$$

$$(A{\to}(B{\to}C)){\to}((A{\to}B){\to}(A{\to}C)), \tag{A2}$$

$$\bot{\to}A, \tag{I}$$

где A, B, C — некоторые формулы.

Первые две схемы аксиом совпадают с аксиомами ИВ, поэтому мы их обозначили так же, как и раньше.

Правило вывода в ИИВ$_{\to}$ остаётся тем же самым — modus ponens.

Если рассмотреть стандартную интерпретацию формул как булевых функций, полагая, что $\bot$ соответствует тождественному 0, то легко видеть, что схема

аксиом (I) является тавтологией. Таким образом, в ИИВ$_{\to}$ выводимы только тавтологии. Из-за отсутствия в аксиоматике схемы аксиом (A3) не все тавтологии выводимы в ИИВ$_{\to}$. Примером такой тавтологии может служить сама аксиома (A3). Для доказательства невыводимости (A3) в ИИВ$_{\to}$ нам потребуется семантика моделей Крипке.

Замечание 2.2. В аксиоматике ИИВ$_{\to}$ неявно присутствует ослабленный вариант аксиомы (A3), который в КИВ был обозначен как (P10) (см. с. 58):

$$(B{\to}\neg A){\to}((B{\to}A){\to}\neg B).$$

Действительно, после раскрытия сокращения $\neg$ эта формула запишется как

$$(B{\to}(A{\to}\bot)){\to}((B{\to}A){\to}(B{\to}\bot)),$$

что является частным случаем аксиомы (A2).

2.2. Примеры выводов в ИИВ$_{\to}$

Поскольку ИИВ$_{\to}$ содержит схемы аксиом (A1), (A2) ИВ, в ИИВ$_{\to}$ справедливы леммы 1.13, 1.15 и теорема дедукции (в её доказательстве использовались только аксиомы (A1), (A2) и лемма 1.13). В доказательстве леммы 1.23 (транзитивность импликации) также не использовалась третья схема аксиом, поэтому эта лемма справедлива и для ИИВ$_{\to}$. В доказательстве леммы 1.16 в предыдущей главе применялась третья схема аксиом. Тем не менее эта лемма справедлива и в ИИВ$_{\to}$.

Лемма 2.3. *В* ИИВ$_{\to}$ *выполнено*:

$$\{\neg A, A\} \vdash B \tag{2.1}$$

$$\{\neg A\} \vdash A{\to}B \tag{2.2}$$

Доказательство. Вывод формулы (2.1):

1.	$A \to \bot$	(гипотеза)
2.	A	(гипотеза)
3.	$\bot$	(MP 2,1)
4.	$\bot \to B$	(I)
5.	B	(MP 3,4)

Формула (2.2) получается из формулы (2.1) применением теоремы дедукции. □

2.3. Модели Крипке

Перейдём теперь к построению семантики для ИИВ$_{\to}$. Прежде всего определим *шкалу Крипке*. Это *частично упорядоченное множество* $(W, \leqslant)$. Элементы этого множества называются *мирами Крипке*. Порядок на мирах задаёт отношение *достижимости*.

Напомним, что частично упорядоченное множество — это множество W и заданное на нём бинарное отношение $\leqslant$ (отношение порядка), удовлетворяющее свойствам

– *рефлексивности* (для любого элемента $w \in W$ выполнено $w \leqslant w$);

– *антисимметричности* (если $u \leqslant v$ и $v \leqslant u$, то $u = v$);

– *транзитивности* (для любых элементов $u, v, w \in W$ из $u \leqslant v$ и $v \leqslant w$ следует $u \leqslant w$).

Примерами частично упорядоченных множеств являются множество целых чисел с отношением порядка, которое задаётся обычным сравнением чисел;

семейство подмножеств множества X (отношение порядка задаётся включением множеств); множество целых положительных чисел с отношением порядка, которое задаётся отношением делимости.

В построении моделей Крипке мы будем использовать, как правило, конечные частично упорядоченные множества. В этом случае отношение порядка удобно задавать *диаграммой Хассе*.

Диаграмма Хассе частично упорядоченного множества $(W, \leqslant)$ является ориентированным графом с множеством вершин W и множеством рёбер, включающем в себя все пары (u, v) вершин, которые удовлетворяют условиям *непосредственного накрытия*: (1) $u \leqslant v$, (2) $u \neq v$, (3) если $u \leqslant w$, то $v \leqslant w$ (в интервале между u и v нет других элементов).

Отношение порядка однозначно восстанавливается по диаграмме Хассе: $u \leqslant v$ тогда и только тогда, когда вершина v достижима из вершины u в диаграмме Хассе. Примеры диаграмм Хассе приведены на рис. 2.1.

Возвращаясь к шкалам Крипке, отметим, что неформально шкалу Крипке нужно понимать как сценарий развития некоторого процесса. Миры Крипке в таком понимании отвечают возможным состояниям процесса, а отношение достижимости фиксирует, какие состояния могут получиться из данного.

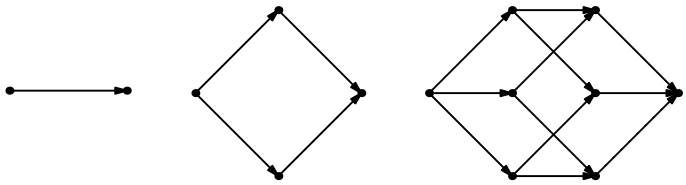


Рис. 2.1. Булевы кубы размерностей 1, 2, 3

Для построения модели Крипке нужно задать некоторую шкалу Крипке и каждой переменной сопоставить монотонную булеву функцию $[x]\colon W \to \{0,1\}$. Монотонность означает, что если $u \leqslant w$, то $[x](u) \leqslant [x](w)$. Продолжая описанное выше неформальное объяснение, можно сказать, что переменной соответствует некоторое событие. В каждом мире событие, отвечающее переменной, либо произошло, либо не произошло. Но если событие уже произошло в мире w, то следует считать, что оно произошло и во всех мирах, которые могут получиться из мира w.

Если шкала состоит из одного мира, мы приходим к классической интерпретации переменной: она принимает одно из двух возможных значений.

Чтобы закончить определение модели Крипке, нужно объяснить, как интерпретируются формулы в этой модели. Каждой формуле A также соответствует булева функция $[A]\colon W \to \{0,1\}$, которая определяется индуктивно. Формуле $\bot$ соответствует функция, тождественно равная 0 (неформально это отвечает событию, которое никогда не происходит). Если формула A имеет вид $B{\to}C$, то A истинна в мире w (т. е. $[A](w) = 1$) тогда и только тогда, когда в любом мире u, достижимом из мира w (т. е. $w \leqslant u$), в котором истинна формула B, истинна также и формула C.

Пример 2.4. Рассмотрим шкалу Крипке, изображённую на рис. 2.2. Функция $[x]$, соответствующая переменной, также изображена на рис. 2.2: у тех миров, в которых x истинна, поставлен знак x. На том же рисунке отмечен мир, в котором истинно отрицание $\neg x$, т. е. $x{\to}\bot$. Действительно, истинность отрицания формулы A в некотором мире w означает, что в этом мире и в достижимых из него мирах формула A ложна.

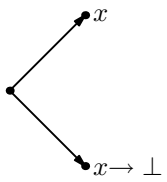


Рис. 2.2.

Лемма 2.5. *Любой формуле* ИИВ$_{\to}$ *соответствует в модели Крипке монотонная булева функция.*

Доказательство. Индукция по построению формулы. Переменным и константе $\bot$ по определению соответствуют монотонные булевы функции, это база индукции.

Индукционный переход: пусть утверждение леммы справедливо для формул длины меньше n, где $n > 1$. Рассмотрим формулу A длины n. Она имеет вид $A = B{\to}C$. Пусть $[A](w) = 1$. Рассмотрим мир u, достижимый из мира w. Любой мир v, достижимый из u, достижим также и из w (транзитивность отношения порядка). По определению функции $[A]$, если в мире v истинна формула B, то истинна также и формула C. Но это означает, что $[A](u) = 1$. □

Если шкала состоит из одного мира, то данные выше определения превращаются в стандартные определения семантики для формул — формулам сопоставляются булевы функции. В частности, тавтологии — это формулы, которые истинны в любой модели Крипке, состоящей из одного мира. В других моделях Крипке некоторые тавтологии могут быть ложными в некоторых мирах.

Определение 2.6. Формулы ИИВ$_{\to}$, истинные в любом мире любой модели Крипке, будем называть

интуиционистскими тавтологиями (сокращённо, И-тавтологиями).

Теорема 2.7 (корректность ИИВ$_{\to}$). *В* ИИВ$_{\to}$ *выводимы только И-тавтологии.*

Доказательство. Проверим, что из И-тавтологий с помощью правила modus ponens выводятся только И-тавтологии. Пусть A и $A{\to}B$ — И-тавтологии. Это означает, что они истинны в любом мире любой модели Крипке. Но тогда и формула B истинна в любом мире, так как истинность A и $A{\to}B$ в мире w влечёт истинность B в мире w по определению модели Крипке.

Осталось проверить, что все аксиомы ИИВ$_{\to}$ являются И-тавтологиями.

Случай аксиомы (I) тривиален: поскольку $\bot$ ложно во всех мирах, $\bot{\to}A$ истинно во всех мирах.

Пусть A истинна в мире w. В силу монотонности это означает, что A истинна и в любом мире u, достижимом из w. Но тогда для любой формулы B можно утверждать, что $B{\to}A$ истинна в мире w (здесь использована транзитивность отношения порядка). Это показывает, что (A1) является И-тавтологией.

Теперь рассмотрим (A2). Будем рассуждать от противного. Пусть в некотором мире u формула $A{\to}(B{\to}C)$ истинна, а формула $(A{\to}B){\to}(A{\to}C)$ ложна. Это означает, что в некотором мире v, $u \leqslant v$, формула $A{\to}B$ истинна, а формула $A{\to}C$ ложна. Значит, найдётся такой мир w, что $u \leqslant v \leqslant w$ и $[A](w) = 1$, $[B](w) = 1$, $[C](w) = 0$. Но тогда $[B{\to}C](w) = 0$, откуда следует ложность формулы $A{\to}(B{\to}C)$ в мире u. Полученное противоречие доказывает, что (A2) является И-тавтологией. $\square$

Следствие 2.8. ИИВ$_{\to}$ *непротиворечиво*: *из аксиом нельзя вывести* $\bot$.

Рассмотрим ещё несколько примеров.

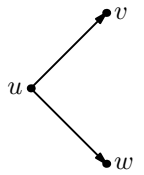
Пример 2.9. Рассмотрим формулу $x \to \neg\neg x$. Докажем, что она является И-тавтологией. Пусть x истинна в мире u, а $\neg\neg x$ ложна в этом мире. Это означает (см. пример 2.4), что в одном из более поздних миров, скажем v, истинна формула $\neg x$. Но тогда формула x ложна в мире v, что противоречит монотонности.

С другой стороны, формула $\neg\neg x \to x$ не является И-тавтологией. Рассмотрим шкалу Крипке из двух миров $u < v$. Пусть x ложно в мире u и истинно в мире v. Тогда $\neg x$ ложна в обоих мирах шкалы, $\neg\neg x$ истинна в обоих мирах, так что $\neg\neg x \to x$ ложна в мире u.

Этот пример показывает, что, в отличие от классического исчисления высказываний, формулы A и $\neg\neg A$ неэквивалентны.

Пример 2.10. Формула $A = (\neg x \to y) \to ((\neg\neg x \to y) \to y)$ является, как легко проверить, классической тавтологией.

Однако формула A не является И-тавтологией. Рассмотрим шкалу Крипке, изображённую на рисунке 2.3, миры которой обозначены u, v, w. На этой шкале определим модель, присвоив значения переменным как указано на рисунке 2.3. Нетрудно проверить, что формула A ложна в мире u этой модели.



	u	v	w
$[x]$	0	1	0
$[y]$	0	1	1

Рис. 2.3.

Задачи

2.2. Постройте модель Крипке, в одном из миров которых ложна аксиома (А3).

2.3. Из двух импликаций

$$(x{\to}y){\to}(\neg y{\to}\neg x) \quad \text{и} \quad (\neg y{\to}\neg x){\to}(x{\to}y),$$

выражающих принцип контрапозиции, И-тавтологией является ровно одна. Определите, какая именно.

2.4. Докажите, что любой вывод формулы

$$((a \to b) \to a) \to a$$

в ИВ содержит аксиому из третьей схемы аксиом.

2.5. а) Завершите рассуждение из примера 2.10: докажите, что формула A ложна в мире u.

б) Покажите, что формула A истинна в каждом мире всякой модели на *линейной* шкале Крипке. Шкала называется линейной, если любые два мира в ней сравнимы.

2.4. Теорема о полноте ИИВ$_{\to}$

Теорема 2.11 (полнота ИИВ$_{\to}$). *Любая И-тавтология выводима в* ИИВ$_{\to}$.

Идея доказательства этой теоремы близка ко второму доказательству полноты исчисления высказываний. Мы будем доказывать, что для любой невыводимой формулы A существует такая модель Крипке, что A ложна в некотором мире этой модели (это и значит по определению, что A не является И-тавтологией).

Поскольку в ИИВ$_{\to}$ формулы A и $\neg\neg A$ не равносильны, необходимо модифицировать определение непротиворечивого множества формул. Кроме того для упрощения рассуждений мы будем рассматривать только некоторые множества формул.

Обозначим через Φ множество всех подформул формулы A. Очевидно, что это множество замкнуто относительно взятия подформул, т. е. если $B \in \Phi$ и C — подформула формулы B, то $C \in \Phi$. Мы ограничимся рассмотрением именно этого множества. Читателю следует иметь в виду, что на протяжении всего дальнейшего рассуждения мы фиксируем формулу A и, тем самым, множество Φ.

В классическом исчислении высказываний мы называли непротиворечивым множество формул, из которого нельзя вывести противоречие. Для интуиционистской логики этого оказывается недостаточно. Мы будем рассматривать пары множеств формул (Γ, Δ), где $\Gamma \subseteq \Phi$, $\Delta \subseteq \Phi$, и накладывать запреты на выводимости из множества формул Γ с учётом множества формул Δ. Нам потребуются следующие определения.

Определение 2.12. Пара (Γ, Δ), $\Delta \neq \varnothing$, множеств формул называется *противоречивой*, если из множества формул Γ выводится формула $B \in \Delta$. Пара $(\Gamma, \varnothing)$ называется противоречивой, если $\Gamma \vdash \bot$ (это объясняет использование термина «противоречивая пара»).

Определение 2.13. Пара (Γ, Δ) множеств формул называется *совместной*, если существуют такие модель Крипке $(W, \leqslant)$ и мир $w \in W$, что все формулы из множества Γ истинны в мире w, а все формулы из множества Δ — ложны в мире w.

Определение 2.14. Непротиворечивую пару (Γ, Δ) назовём *полной*, если $\Gamma \cup \Delta = \Phi$.

Прямо из определения следует, что совместная пара непротиворечива: из формул, истинных в мире w, можно вывести только формулы, истинные в этом мире. Обратное утверждение доказать труднее, в этом

и состоит основная часть доказательства теоремы о полноте.

Теорема 2.15. *Всякая непротиворечивая пара* (Γ, Δ) *совместна.*

Теорема 2.11 о полноте ИИВ$_{\to}$ легко следует из теоремы 2.15. Действительно, рассмотрим пару $(\varnothing, \{A\})$. Поскольку A невыводима, эта пара непротиворечива, т. е. по теореме 2.15 совместна. По определению 2.13 это означает, что в некотором мире некоторой модели Крипке формула A ложна и потому не является И-тавтологией.

Для доказательства совместности непротиворечивой пары (Γ, Δ) мы применим тот же приём, что и в случае классического исчисления высказываний: расширение непротиворечивой пары до полной непротиворечивой пары (Γ', Δ').

Возможность такого расширения обеспечивают следующие леммы.

Лемма 2.16. *Пусть* $\Gamma \subset \Phi$, $\Delta \subset \Phi$, $B \in \Phi \setminus (\Gamma \cup \Delta)$ *и пара* (Γ, Δ) *непротиворечива. Тогда одна из двух пар* $(\Gamma \cup \{B\}, \Delta)$, $(\Gamma, \Delta \cup \{B\})$ *также непротиворечива.*

Доказательство. От противного. Предположим, что пары $(\Gamma, \Delta \cup \{B\})$ и $(\Gamma \cup \{B\}, \Delta)$ противоречивы и докажем, что тогда пара (Γ, Δ) также противоречива.

Если $\Gamma \vdash C$ для некоторой формулы $C \in \Delta$, то пара (Γ, Δ) противоречива. Поскольку пара $(\Gamma, \Delta \cup \{B\})$ противоречива, то $\Gamma \vdash B$. Противоречивость пары $(\Gamma \cup \{B\}, \Delta)$ означает, что $\Gamma \cup \{B\} \vdash C$ для некоторой формулы $C \in \Delta$. По теореме дедукции отсюда следует, что $\Gamma \vdash B \to C$. Так как и B, и $B \to C$ выводятся из Γ, получаем, применяя правило modus ponens, вывод $C \in \Delta$ из множества формул Γ. $\square$

Лемма 2.17. *Всякая непротиворечивая пара* (Γ, Δ) *может быть дополнена до полной непротиворечивой пары* (Γ', Δ'), *где* $\Gamma \subseteq \Gamma'$, $\Delta \subseteq \Delta'$.

Доказательство. Применяя лемму 2.16, мы можем последовательно добавлять формулы из множества Φ либо в первое множество, либо во второе, сохраняя непротиворечивость расширенной пары. Через конечное число шагов получим искомую полную непротиворечивую пару. □

Теперь построим нужную для доказательства теоремы 2.15 модель Крипке. Шкалой $(W_\Phi, \leqslant_\Phi)$ этой модели будут полные непротиворечивые пары (Γ, Δ) множеств формул из Φ. Отношение порядка на этих парах зададим следующим образом: если $\Gamma_1 \subseteq \Gamma_2$, то $(\Gamma_1, \Delta_1) \leqslant_\Phi (\Gamma_2, \Delta_2)$. Функцию истинности для переменных зададим следующим образом: если $x \in \Gamma$, то x истинна в мире (Γ, Δ), в противном случае x ложна в этом мире. Определение порядка на парах гарантирует монотонность определённой таким образом функции.

Лемма 2.18. *В любом мире* (Γ, Δ) *модели Крипке* $(W_\Phi, \leqslant_\Phi)$ *все формулы из множества* Γ *истинны, а все формулы из множества* Δ *— ложны.*

Доказательство. Индукция по построению формулы. Для переменных утверждение леммы справедливо по построению модели Крипке $(W_\Phi, \leqslant_\Phi)$. Ложь $\bot$ не может принадлежать Γ, так как по аксиоме (I) из неё выводима любая формула, а пара (Γ, Δ) непротиворечива. База индукции доказана.

Индукционный переход. Предположим, что утверждение справедливо для формул длины меньше n, где $n > 1$. Рассмотрим формулу B длины n. Такая формула имеет вид $C \to D$, причём для более коротких

формул C, D справедливо предположение индукции. Рассмотрим два возможных случая.

Пусть $B \in \Gamma$. Рассмотрим мир (Γ', Δ'), достижимый из (Γ, Δ), т. е. $(\Gamma, \Delta) \leqslant_\Phi (\Gamma', \Delta')$, причём C истинна в мире (Γ', Δ'). По предположению индукции это означает, что $C \in \Gamma'$. Так как $\Gamma \subseteq \Gamma'$, то по определению порядка в данной модели Крипке $B \in \Gamma'$. Но $C, C{\to}D \vdash D$, значит, $\Gamma' \vdash D$. Из непротиворечивости пары (Γ', Δ') заключаем, что $D \in \Gamma'$, откуда по предположению индукции следует истинность D в мире (Γ', Δ'). Поэтому B истинна в мире Γ.

Пусть $B \in \Delta$. Заметим, что пара $(\Gamma \cup \{C\}, \{D\})$ непротиворечива, так как из $\Gamma \cup \{C\} \vdash D$ по теореме дедукции следует $\Gamma \vdash C{\to}D = B \in \Delta$. Дополним эту пару до полной (Γ', Δ'), $\Gamma' \supset \Gamma$. В мире (Γ', Δ'), достижимом из мира (Γ, Δ), по предположению индукции формула C истинна, а формула D ложна. Но это означает, что в мире (Γ, Δ) формула $C{\to}D = B$ ложна. □

Доказательство теоремы 2.15. По лемме 2.17 дополним непротиворечивую пару (Γ, Δ) до полной (Γ', Δ'), $\Gamma' \supset \Gamma$, $\Delta' \supset \Delta$. По лемме 2.18 в мире $(\Gamma', \Delta') \in W_\Phi$ все формулы из множества Γ' истинны, а все формулы из множества Δ' ложны, что и означает совместность пары (Γ, Δ). □

2.5. Интуиционистское исчисление высказываний ИИВ

Мы рассмотрели формальную систему ИИВ$_\to$, которая использует только связку $\to$. В классическом случае это не мешало выражать в формальной системе ИВ конъюнкцию и дизъюнкцию высказываний: для

этого использовались сокращения $\neg(A{\to}\neg B)$ и $\neg A{\to}B$ соответственно.

Аналогичные сокращения можно использовать и в интуиционистском исчислении. Однако результат при этом получается не слишком удовлетворительный.

Чтобы пояснить суть возникающей проблемы, нужно вернуться к неформальному обсуждению интуиционистской математики. Как уже говорилось, интуиционисты считают доказанной дизъюнкцию двух утверждений лишь в том случае, когда приведено доказательство хотя бы одного из этих утверждений. Из этого принципа следует, что если доказаны утверждения «из A следует C», «из B следует C», «A или B», то существует доказательство C (возьмём доказательство одного из членов дизъюнкции, добавим к нему доказательство соответствующей импликации и применим правило modus ponens).

В формальной системе ИИВ$_{\to}$ указанное свойство соответствует условной выводимости

$$A{\to}C, B{\to}C, A \vee B \vdash C,$$

которая в силу теоремы дедукции равносильна аксиоме (P8) на с. 58:

$$(A{\to}C){\to}((B{\to}C){\to}(A \vee B{\to}C)).$$

Если переписать эту формулу, заменяя дизъюнкцию $A \vee B$ на $\neg A{\to}B$, то получим

$$(A{\to}C){\to}((B{\to}C){\to}(((A{\to}\bot){\to}B){\to}C)). \qquad (2.3)$$

Но эта формула не является И-тавтологией и поэтому невыводима в ИИВ$_{\to}$. Таким образом, возможности формального доказательства в системе ИИВ$_{\to}$ у́же, чем хотелось бы.

Задача 2.6. Постройте модель Крипке, демонстрирующую, что формула (2.3) не является И-тавтологией.

Итак, желательно формализовать интуиционистское исчисление высказываний такой формальной системой, в которой присутствуют пропозициональные связки для конъюнкции и дизъюнкции. Другими словами, нужен аналог формальной системы КИВ.

Мы построим формальную систему ИИВ (интуиционистское исчисление высказываний), которая удовлетворяет этим требованиям. Алфавит этой системы расширяется по сравнению с алфавитом ИИВ$_{\to}$ добавлением пропозициональных связок $\vee$, $\wedge$. Формулы определяются аналогично формальной системе КИВ. Правило вывода не меняется — это по-прежнему modus ponens. К схемам аксиом (A1), (A2), (I) добавляются схемы аксиом (P3)–(P8) (с. 58), описывающие свойства конъюнкции и дизъюнкции.

Замечание 2.19. Если вместо символа $\bot$ использовать отрицание $\neg$, то можно определить ИИВ как усечённый вариант КИВ, из которого удалена схема аксиом (P11) (закон исключённого третьего).

Поскольку ИИВ содержит ИИВ$_{\to}$ (формулы и аксиомы ИИВ$_{\to}$ являются, соответственно, формулами и аксиомами ИИВ, а правило вывода одно и то же), все выводимые формулы в ИИВ$_{\to}$ формулы выводимы также и в ИИВ. Дополнительно нам понадобится выводимость следующих формул.

Лемма 2.20. *В* ИИВ *имеют место выводы*

$$\{\neg A, A \vee B\} \vdash B \tag{2.4}$$

$$\{A \vee B, A{\to}B\} \vdash B \tag{2.5}$$

$$\{A \vee B, B{\to}A\} \vdash A \tag{2.6}$$

Доказательство. Вывод формулы (2.4):

1. $A{\to}B$ (формула (2.2))

2. $(A{\to}B){\to}((B{\to}B){\to}((A \vee B){\to}B))$ (P8)
3. $(B{\to}B){\to}((A \vee B){\to}B)$ (MP 1,2)
4. $B{\to}B$ (лемма 1.13)
5. $(A \vee B){\to}B$ (MP 4,3)
6. $A \vee B$ (гипотеза)
7. B (MP 6,5)

Вывод формулы (2.5):

1. $A \vee B$ (гипотеза)
2. $A{\to}B$ (гипотеза)
3. $(A{\to}B){\to}((B{\to}B){\to}((A \vee B){\to}B))$ (P8)
4. $(B{\to}B){\to}((A \vee B){\to}B)$ (MP 2,3)
5. $B{\to}B$ (лемма 1.13)
6. $(A \vee B){\to}B$ (MP 5,4)
7. B (MP 1,6)

Вывод формулы (2.6) получается аналогично выводу формулы (2.5). Вместо третьей формулы в нём нужно использовать формулу

$$(A{\to}A){\to}((B{\to}A){\to}((A \vee B){\to}A)),$$

остальные изменения очевидны. □

2.5.1. Модели Крипке для ИИВ

Модели Крипке могут быть расширены для описания семантики системы ИИВ. Само определение модели Крипке сохраняется. Нужно расширить правила интерпретации формул, добавив правила интерпретации дизъюнкции и конъюнкции.

Если формула A являются дизъюнкцией, т. е. имеет вид $B \vee C$, то $[A] = \max([B], [C])$. Другими словами,

формула A истинна в мире w, если и только если в мире w истинна хотя бы одна из формул B, C (функция $[A]$ является дизъюнкцией функций $[B]$, $[C]$).

Аналогично, если формула A является конъюнкцией, т. е. имеет вид $B \wedge C$, то $[A] = \min([B], [C])$. Другими словами, формула A истинна в мире w, если и только если в мире w истинны обе формулы B и C (функция $[A]$ является конъюнкцией функций $[B]$, $[C]$).

Из этих правил легко следует справедливость обобщения леммы 2.5.

Лемма 2.21. *Любой формуле* ИИВ *соответствует в модели Крипке монотонная булева функция.*

Доказательство. Аналогично доказательству леммы 2.5. При индукционном переходе нужно рассмотреть ещё две возможности $A = B \vee C$, $A = B \wedge C$. Если по индукционному предположению функции $[B]$ и $[C]$ монотонны, то и $[A]$ будет монотонной, так как дизъюнкция и конъюнкция двух монотонных функций является монотонной функцией. □

Определение 2.6 И-тавтологии остаётся для ИИВ тем же самым.

Пример 2.22. Аксиома (P11) $x \vee \neg x$ не является И-тавтологией. Рассмотрим шкалу Крипке из двух миров $u < v$. Пусть x ложна в мире u и истинна в мире v. Тогда по данному выше определению $\neg x = x \to \bot$ ложна в обоих мирах шкалы, а $x \vee \neg x$ ложна в мире u.

Пример 2.23. Формула $\neg x \vee \neg\neg x$ также не является И-тавтологией. Контрмоделью является модель, описанная в примере 2.10. В этой модели $\neg x$ истинна

только в мире w, а $\neg\neg x$ — только в мире v. В мире u обе эти формулы ложны.

Эта формула обладает отмеченной в задаче 2.5 (с. 81) особенностью: она истинна в любом мире любой модели на линейной шкале Крипке. Действительно, в любой линейной модели отрицание любой формулы либо истинно во всех мирах (если сама формула ложна во всех мирах), либо ложно во всех мирах (если сама формула хотя бы в одном мире истинна). Поэтому в линейных моделях формула $\neg x \vee \neg\neg x$ всюду истинна.

Теорема 2.24 (корректность ИИВ). *В* ИИВ *выводимы только И-тавтологии.*

Доказательство. В дополнение к теореме 2.7 нужно проверить, что аксиомы (P3)–(P8) являются И-тавтологиями.

Истинность $A \wedge B$ в мире w влечёт по определению истинность формул A и B в мире w (аксиомы (P3), (P4)). Истинность A в мире w влечёт по определению истинность формулы $A \vee B$ в мире w (аксиома (P6)). Точно так же проверяется, что (P7) является И-тавтологией.

Рассмотрим теперь аксиому (P5). Пусть $[A](w) = 1$. Тогда $[A](u) = 1$ при $w \leqslant u$. Если $[B](u) = 1$, то $[B](v) = 1$ при $u \leqslant v$. Но это означает, что в любом мире v таком, что $w \leqslant u \leqslant v$, истинны формулы A и B, т. е. $[A \wedge B](v) = 1$. Из определения истинности импликации имеем

$$[B \to A \wedge B](u) = 1 \text{ и } [A \to (B \to A \wedge B)](w) = 1.$$

В случае аксиомы (P8) рассуждаем от противного. Пусть в некотором мире u истинна формула $A \to C$, а формула $(B \to C) \to (A \vee B \to C)$ ложна. Значит, в

некотором мире v, достижимом из u, формула $B{\to}C$ истинна, а формула $A\vee B{\to}C$ ложна. Последнее означает, что найдётся такой мир w, достижимый из v, что $[A\vee B](w)=1$, $[C](w)=0$. Но тогда $[B](w)=0$ в силу истинности $B{\to}C$ в мире v, поэтому $[A](w)=1$. С другой стороны, из истинности формулы $A{\to}C$ в мире u следует её истинность в мире w, т. е. $[C](w)=1$. Полученное противоречие доказывает, что (P8) является И-тавтологией. □

Следствие 2.25. ИИВ *непротиворечиво*: *из аксиом нельзя вывести* $\bot$.

2.5.2. Полнота ИИВ

Теорема 2.26 (полнота ИИВ). *Любая И-тавтология выводима в* ИИВ.

Доказательство теоремы 2.26 принципиально не отличается от доказательства теоремы 2.11 полноты системы ИИВ$_{\to}$. Однако возникают технические трудности при доказательстве леммы 2.18 применительно к построенной выше модели Крипке $(W_\Phi,\leqslant_\Phi)$. Поэтому мы модифицируем эту модель.

Замечание 2.27. Другой способ доказательства теоремы полноты состоит в том, чтобы модифицировать определение 2.12 противоречивой пары множеств формул для удобства доказательства леммы 2.18, см., например, [4].

Итак, проследим за доказательством теоремы 2.11 и обобщим его для формальной системы ИИВ.

Как и раньше, фиксируем некоторую невыводимую формулу A. Определения непротиворечивой, совместной и полной пары множеств подформул

формулы A остаются теми же самыми. Наша цель состоит в доказательстве совместности пары $(\varnothing, \{A\})$.

Леммы 2.16 о расширении непротиворечивой пары одной формулой и 2.17 о дополнении непротиворечивой пары до полной выполняются и для ИИВ, причём их доказательства остаются теми же самыми.

Для доказательства совместности пары $(\varnothing, \{A\})$ мы не сможем использовать ту же самую модель Крипке $(W_\Phi, \leqslant_\Phi)$, что и раньше. Для построения искомой модели нужно ограничиться лишь *нормальными* полными непротиворечивыми парами.

Определение 2.28. Непротиворечивую полную пару (Γ, Δ) назовём *нормальной*, если для любой дизъюнкции $B = C \vee D$, $B \in \Phi$, из $B \in \Gamma$ следует $C \in \Gamma$ или $D \in \Gamma$.

Лемма 2.29. *Пусть (Γ, Δ) — непротиворечивая полная пара. Тогда существует нормальная непротиворечивая полная пара (Γ', Δ') такая, что $\Gamma \subseteq \Gamma'$. Если при этом $A \in \Delta$, то существует такая нормальная непротиворечивая полная пара (Γ', Δ'), что $\Gamma \subseteq \Gamma'$, $A \in \Delta'$.*

Замечание 2.30. В формулировке леммы A обозначает ту самую невыводимую формулу, которую мы фиксировали в начале всего рассуждения.

Доказательство леммы 2.29. Построим по ненормальной непротиворечивой полной паре (Γ_0, Δ_0) её *нормализацию* (Γ', Δ'). Построение будет происходить по шагам.

Пусть $B \in \Gamma_0$, $B = C \vee D$, $C, D \in \Delta_0$. Докажем, что пара $(\Gamma_0 \cup \{C\}, \{D\})$ непротиворечива. Действительно, если $\Gamma_0 \cup \{C\} \vdash D$, то по теореме дедукции $\Gamma_0 \vdash C \to D$.

Но $C \vee D, C{\to}D \vdash D$ (формула (2.5) леммы 2.3) и получаем противоречивость пары (Γ_0, Δ_0).

Дополняя непротиворечивую пару $(\Gamma_0 \cup \{C\}, \{D\})$ до полной, получаем пару (Γ_1, Δ_1).

Повторим описанные действия для пары (Γ_1, Δ_1), а затем для следующей пары и т. д., пока это возможно. Из-за конечности множества формул Φ через конечное число шагов процесс остановится. Полученная в итоге пара (Γ', Δ') и будет искомой нормальной полной непротиворечивой парой.

Осталось доказать вторую часть леммы, которая говорит о случае $A \in \Delta_0$. Рассуждаем аналогично. Прежде всего заметим, что A не является подформулой никакой формулы из Φ, так что $A \neq C$, $A \neq D$. Докажем, что хотя бы одна из пар $(\Gamma_0 \cup \{C\}, \{A, D\})$ или $(\Gamma_0 \cup \{D\}, \{A, C\})$ непротиворечива. Выводимости

$$\Gamma_0 \cup \{C\} \vdash A, \quad \Gamma_0 \cup \{D\} \vdash A \tag{2.7}$$

не могут выполняться одновременно. Действительно, если допустить противное, то по теореме дедукции получаем выводимости $\Gamma_0 \vdash C{\to}A$, $\Gamma_0 \vdash D{\to}A$. Но тогда можно вывести A из Γ_0: к аксиоме (P8)

$$(C{\to}A){\to}((D{\to}A){\to}((C \vee D){\to}A))$$

добавим выводы $C{\to}A$, $D{\to}A$ и применим трижды правило modus ponens (вспомним, что $C \vee D = B \in \Gamma_0$). Полученное противоречие доказывает, что хотя бы одна из выводимостей (2.7) не выполняется.

Пусть $\Gamma_0 \cup \{C\} \nvdash A$. Тогда пара $(\Gamma_0 \cup \{C\}, \{A, D\})$ непротиворечива, так выше было доказано, что $\Gamma_0 \cup \{C\} \nvdash D$. Дополняя пару $(\Gamma_0 \cup \{C\}, \{A, D\})$ до полной, получаем пару (Γ_1, Δ_1), $A \in \Delta_1$.

Пусть $\Gamma_0 \cup \{D\} \nvdash A$. Тогда пара $(\Gamma_0 \cup \{D\}, \{A, C\})$ непротиворечива. Рассуждаем аналогично первому случаю. При доказательстве утверждения $\Gamma_0 \cup \{D\} \nvdash C$

нужно применять формулу (2.6) леммы 2.3 вместо формулы (2.5).

Продолжая построение как и выше, придём к нормальной полной непротиворечивой паре (Γ', Δ'), причём по построению $A \in \Delta'$. $\square$

Теперь построим модифицированную модель Крипке. Шкалой $(N_\Phi, \leqslant_\Phi)$ этой модели будут нормальные полные непротиворечивые пары (Γ, Δ) множеств формул из Φ. Отношение порядка остаётся тем же самым: если $\Gamma_1 \subseteq \Gamma_2$, то $(\Gamma_1, \Delta_1) \leqslant_\Phi (\Gamma_2, \Delta_2)$. Функцию истинности для переменных зададим аналогично модели $(W_\Phi, \leqslant_\Phi)$: если $x \in \Gamma$, то x истинна в мире (Γ, Δ), в противном случае x ложна в этом мире.

Докажем лемму, аналогичную лемме 2.18.

Лемма 2.31. *В любом мире (Γ, Δ) модели Крипке $(N_\Phi, \leqslant_\Phi)$ все формулы из множества Γ истинны, а все формулы из множества Δ — ложны.*

Доказательство. Индукция по построению формулы.

База индукции доказывается в точности так же, как в лемме 2.18.

Индукционный переход. Предположим, что утверждение справедливо для формул длины меньше n, где $n > 1$. Рассмотрим формулу B длины n и возможные варианты её внешней связки и принадлежности множеству Γ. По предположению индукции для подформул формулы B, как более коротких, утверждение леммы справедливо.

Пусть $B = C \to D$. В этом случае можно дословно повторить рассуждения из доказательства леммы 2.18.

Пусть $B = C \wedge D$, $B \in \Gamma$. Так как $C \wedge D \vdash C$, $C \wedge D \vdash D$, формулы C и D также принадлежат Γ.

По предположению индукции они истинны. Значит, истинна и формула B.

Пусть $B = C \wedge D$, $B \in \Delta$. Пусть $C, D \in \Gamma$. Так как $C, D \vdash C \wedge D$ (аксиома (P5) и два modus ponens), приходим к противоречию с непротиворечивостью пары (Γ, Δ). Таким образом, хотя бы одна из формул C или D входит в Δ. По предположению индукции эта формула ложна. Значит, ложна и формула B.

Пусть $B = C \vee D$, $B \in \Gamma$. В этом случае мы используем условие нормальности пары (Γ, Δ). Хотя бы одна из формул C или D входит в Γ. По предположению индукции эта формула истинна. Значит, истинна и формула B.

Пусть $B = C \vee D$, $B \in \Delta$. Если хотя бы одна из формул C, D принадлежит Γ, пара (Γ, Δ) противоречива: $C{\to}(C \vee D)$, $D{\to}(C \vee D)$ — аксиомы (P6) и (P7) соответственно, добавляя к ним гипотезу C или D и применяя modus ponens, выводим формулу $B = C \vee D$. Значит, обе формулы C, D принадлежат Δ. По предположению индукции эти формулы ложны. Значит, ложна и формула B. □

Чтобы завершить доказательство теоремы о полноте заметим, что по лемме (2.31) формула A ложна в мире, соответствующем нормальной полной непротиворечивой паре, построенной применением лемм 2.17 и 2.29 из пары $(\varnothing, \{A\})$.

В качестве одного из следствий теоремы о полноте ИИВ предлагаем читателю убедиться в том, что из выводимости в ИИВ дизъюнкции следует выводимость одного из её членов.

Задача 2.7. Если в ИИВ выполнено $\vdash A \vee B$, то справедлива хотя бы одна из выводимостей $\vdash A$ или $\vdash B$.

Глава 3

Предикаты и логика первого порядка

Как уже отмечалось во введении, одной из целей математической логики является формализация математического рассуждения. Оказалось, что значительную часть математических рассуждений можно формализовать на основе логики *первого порядка*, которая описана в этой главе.

Во введении также объяснялось, что формализовать нужно и синтаксис (структуру записи утверждений и доказательств), и семантику (смысл утверждений анализируемой теории).

В этой главе мы рассмотрим формальную систему, которая называется исчислением предикатов. Она описывает структуру математического рассуждения: утверждения и доказательства. Утверждениям соответствуют формулы исчисления предикатов, а доказательствам — выводы.

Семантикой исчисления предикатов являются алгебры предикатов на произвольных множествах.

Формулами, выводимыми в исчислении предикатов, являются так называемые общезначимые формулы. Эти формулы обобщают тавтологии исчисления высказываний. Неформально можно сказать, что

общезначимые формулы отвечают «общим законам логики».

3.1. Предикаты, кванторы, функции и константы

Проанализируем утверждение

«если целое число делится на 4, то оно чётно». (3.1)

Его структура отличается от высказываний, которые мы изучили раньше. В данном утверждении речь идёт о некоторых свойствах целых чисел: делимости на 4 и делимости на 2. Каждое целое число либо обладает свойством делимости на 4, либо нет. Такие свойства называют *предикатами*[1)]. Предикат может быть определён не для одного числа, а для набора чисел. Скажем, предикат «число x делится на число y» есть свойство (упорядоченных) пар целых чисел (x, y). На паре $(4, 2)$ этот предикат истинен (свойство выполняется), а на паре $(2, 4)$ — ложен (свойство не выполняется). Разумеется, предикаты можно определять не только для чисел, а для произвольных объектов.

Данное выше описание предиката неформально. Но с любым предикатом можно связать две математические конструкции, которые его полностью характеризуют.

Пусть объекты, для которых определяется предикат, образуют множество Z. Сопоставим предикату от k объектов множество P упорядоченных наборов из k объектов, для которых этот предикат выполняется

[1)] В логике термин «предикат» (лат. praedicatum — сказанное) означает то, что *высказывается* (утверждается или отрицается) *в суждении* о субъекте.

(т. е. эти объекты обладают указанным в предикате свойством). Ясно, что это множество полностью характеризует предикат. Множество P является подмножеством множества Z^k. Обычно оно называется *k-арным отношением*.

Пример 3.1. Рассмотрим первый квадрант координатной плоскости. Это подмножество пар (x, y) действительных чисел, которое по данному выше определению является бинарным отношением. Это отношение соответствует предикату «числа x, y неотрицательны».

Пример 3.2. Тернарному предикату «числа x, y, z являются длинами сторон треугольника» отвечает следующее отношение: множество таких троек (x, y, z), что

$$0 < x < y + z,$$
$$0 < y < z + x,$$
$$0 < z < x + y.$$

Отношению $P \subseteq Z^k$ можно сопоставить *характеристическую функцию* $\chi_P \colon Z^k \to \{0, 1\}$ по правилу $\chi_P(x) = 1$ тогда и только тогда, когда $x \in P$. Ясно, что это соответствие взаимно однозначно.

Итак, предикат P может быть описан подмножеством $P \subseteq Z^k$ (т. е. k-арным отношением) или характеристической функцией χ_P, которая зависит от k аргументов, принимающих значения из множества объектов, и принимает булевы значения (0 — ложь, 1 — истина). Далее мы будем считать синонимами утверждения «предикат P выполняется (истинен) на x», «для x выполнено свойство P», «$x \in P$», «$\chi_P(x) = 1$». Здесь x обозначает набор k объектов, где значение k зависит, разумеется, от предиката.

Вернёмся к утверждению (3.1). Оно говорит о двух предикатах от целых чисел и фактически определяет новый предикат с помощью импликации. Однако в соответствии с традиционным пониманием математических текстов эта фраза является утверждением, т. е. она либо истинна, либо ложна. Что же в ней утверждается? Содержательно это утверждение можно переписать подробнее так:

«если взять любое целое число, делящееся на 4, то это число окажется чётным». (3.1′)

Другими словами, утверждается истинность построенного составного предиката для любого целого числа.

Заметим, что истинностные значения «истина» и «ложь» также можно считать особым случаем предиката, который вообще ни от каких объектов не зависит. Такие предикаты будем называть 0-арными.

Теперь переход от фразы (3.1), которая в буквальном смысле определяет некоторый унарный предикат на множестве целых чисел, к утверждению (3.1′), которое является 0-арным предикатом, можно рассматривать как применение некоторой операции на предикатах. Как мы видим, такая операция изменяет арность предиката (понижает его на 1).

Пример 3.3. Рассмотрим структуру определения простого числа. В её основе лежит бинарный предикат (бинарное отношение D на парах (d, n) целых положительных чисел): «если n делится на d, то $d = 1$ или $d = n$».

Число p называется простым, если $D(d, p)$ истинно для всех d. Множество простых чисел является, как следует из определения, унарным предикатом. Мы опять видим пример применения операции перехода от k-арного предиката к $(k - 1)$-арному.

Оба рассмотренных примера выглядят похоже: в них к предикатам применяется одна и та же операция.

А именно, $(k+1)$-арному предикату

$$P(x_0, x_1, \ldots, x_k)$$

мы сопоставляем k-арный предикат

$$\forall\, x_0 P(x_0, x_1, \ldots, x_k),$$

который истинен тогда и только тогда, когда предикат $P(x_0, x_1, \ldots, x_k)$ истинен при любом значении x_0.

Такого рода операции с предикатами называют *навешиванием квантора*. Мы только что определили операцию навешивания *квантора всеобщности* $\forall$. Другой стандартный квантор — *квантор существования* $\exists$, с помощью которого строится утверждение о существовании объекта, на котором предикат истинен.

Например, утверждение

«существует нечётное число, которое делится на 4» (3.2)

построено из предиката «число нечётное и число делится на 4» с помощью квантора существования. (Это утверждение, разумеется, ложно.)

Более формальное определение звучит так: навешиванием квантора существования из $(k+1)$-арного предиката

$$P(x_0, x_1, \ldots, x_k)$$

получается k-арный предикат, обозначаемый

$$\exists\, x_0 P(x_1, \ldots, x_k).$$

Этот предикат истинен тогда и только тогда, когда предикат $P(x_0, x_1, \ldots, x_k)$ истинен хотя бы при одном значении x_0. Ещё одно название для получаемого таким образом предиката — *проекция* квантора P.

Кванторы всеобщности и существования являются основными кванторами для логики первого порядка. Заметим, что для описания квантора нужно задать переменную, на которую он навешивается. В логиках более высокого порядка используются и такие кванторы, которые навешиваются, скажем, на предикаты.

Пример 3.4. Рассмотрим принцип математической индукции в следующей формулировке: «в любом подмножестве натуральных чисел найдётся наименьший элемент». Слова «в любом» и «найдётся» (существует) явно указывают на кванторы всеобщности и существования. Однако, если квантор существования навешивается здесь на числа, то квантор всеобщности навешивается на предикаты (мы сейчас полагаем, что подмножество натуральных чисел — это предикат).

Далее мы будем обсуждать только логику первого порядка, в которой навешивание кванторов разрешено только на объекты, а не на предикаты.

В математических утверждениях встречаются ещё две важные составляющие. Например, аксиома существования единицы в группе звучит так:

существует такой элемент e группы G, что для любого элемента x группы G выполнено $xe = x = ex$. (3.3)

В структуре утверждения (3.3) видны кванторы существования и всеобщности, а также предикат равенства «$x = y$». Но единица и умножение в группе требуют дополнительного обсуждения.

Единица в утверждении (3.3) является примером *константы*, т. е. имени конкретного объекта. Умножение является примером функции на множестве объектов, в данном случае, функции из множества упорядоченных пар объектов во множество объектов.

Функции вида $Z^k \to Z$, т. е. функции со значениями во множестве Z от k аргументов, принимающие значения во множестве Z, называют *k-арными функциями*. Заметим, что константы можно считать частным случаем функций, а именно 0-арными функциями (аргументов нет, а значение есть; это и есть константа).

В принципе, константы и функции можно выражать через предикаты. Любой k-арной функции f сопоставляется $(k+1)$-арный предикат Γ_f, описывающий свойство «$x_0 = f(x_1, \ldots, x_k)$» (*график функции*). Соответствие между функцией и её графиком взаимно однозначно. Утверждение «x_0 является значением функции $f(x_1, \ldots, x_k)$» равносильно утверждению «набор $(x_0, x_1, \ldots, x_k)$ принадлежит графику функции f».

Тем не менее по традиции (и для удобства дальнейших рассуждений) и константы, и функции включаются в формальное описание математического утверждения. Удобство здесь заключается в том, что использование констант и функций позволяет включить в исчисление много имён конкретных объектов.

В следующем разделе 3.2 мы начнём построение формальной системы, формулы которой соответствуют описанным выше выразительным средствам. В разделе 3.3 мы опишем стандартную семантику для этих формул.

3.2. Формулы исчисления предикатов

Определение исчисления предикатов (ИП) мы дадим в два этапа. В этом разделе мы определим алфавит и формулы ИП, отложив определение аксиом и правил вывода до раздела 3.6.1.

3.2.1. Алфавит

Алфавит ИП состоит из

- счётного множества $\mathcal{X}$ *символов переменных*;
- счётного множества $\mathcal{P}$ *предикатных символов*;
- счётного множества $\mathcal{F}$ *функциональных символов*;
- символов пропозициональных связок $\to$ и $\neg$;
- кванторного символа $\forall$ (*квантор всеобщности*);
- служебных символов: скобок закрывающей) и открывающей (, а также символа запятой , .

При этом мы требуем, чтобы множества $\mathcal{X}$, $\mathcal{P}$, $\mathcal{F}$ не пересекались. Другими словами, роль каждого символа определена однозначно.

Придерживаясь традиции, мы будем обозначать предикатные символы прописными латинскими буквами, а переменные и функциональные символы (в том числе и константы) — строчными. Обычно роль символа будет указываться явно. Заметим, что обычно константы обозначаются символами из начала алфавита (a, b, c, ...), функциональные символы — символами из середины алфавита (f, g, ...), а переменные — символами из конца алфавита (..., x, y, z).

На множествах предикатных и функциональных символов задана также функция арности (неформально это означает, что о предикате или функции сказано, от какого числа аргументов он или она зависит). Мы предполагаем, что имеется достаточный запас (бесконечное количество) предикатных и функциональных символов каждой арности. Таким образом, множества

$\mathcal{P}$ и $\mathcal{F}$ разлагаются в объединение непересекающихся бесконечных множеств

$$\mathcal{P} = \bigcup_{k \geqslant 0} \mathcal{P}_k, \quad \mathcal{F} = \bigcup_{k \geqslant 0} \mathcal{F}_k, \tag{3.4}$$

где элементы из $\mathcal{F}_k$, $\mathcal{P}_k$ имеют арность k.

Замечание 3.5. Множество $\mathcal{F}_0$ называют также множеством символов констант.

3.2.2. Формулы

Среди слов (последовательностей символов) в алфавите исчисления предикатов выделяются *термы* и *формулы*. Неформально говоря, термы описывают объекты, а формулы — утверждения об объектах (предикаты от объектов). Формальные определения термов (и формул) даются рекурсивно с использованием определений терма (или формулы) меньшей длины.

Вначале определим термы.

Определение 3.6. Слово t в алфавите ИП является *термом* тогда и только тогда, когда оно удовлетворяет одному из двух условий

1: t является переменной или константой (т. е. 0-арным функциональным символом);

2: $t = f(t_1, t_2, \ldots, t_n)$, где в скобках записаны через запятую n термов, а f — символ из множества n-арных функциональных символов $\mathcal{F}_n$.

Пример 3.7. Пусть x, y — символы переменных, а f — символ бинарной функции. Тогда $f(f(x,y),y)$, $f(x,f(x,y))$ — термы, а $f(f,x)$ — нет.

Рассмотрим последний пример более подробно. Чтобы слово $f(f,x)$ являлось термом, символ f должен иметь арность 2, а слова f и x должны быть

термами. Но f термом не является: он не может удовлетворять первому условию, так как f не является переменной или 0-арным символом, однако и второму условию он удовлетворять не может хотя бы потому, что в слове f нет ни одной скобки.

Определение 3.8. Слово в алфавите ИП является *элементарной формулой* тогда и только тогда, когда оно имеет вид $A(t_1, \ldots, t_n)$, где в скобках записаны через запятую n термов, а A — символ из множества n-арных предикатных символов $\mathcal{P}_n$.

Пример 3.9. Слово $A(f(f(x,y),y), f(x, f(x,y)))$ является элементарной формулой тогда и только тогда, когда $x, y \in \mathcal{X}$, $f \in \mathcal{F}_2$, $A \in \mathcal{P}_2$.

А вот слова

$$A(f(A(x,y),y), f(x, f(x,y)))$$

и

$$f(f(f(x,y),y), f(x, f(x,y)))$$

не являются формулами. В первом случае A, а во втором f должны одновременно быть бинарным предикатным символом и бинарным функциональным символом, что невозможно по нашему определению алфавита ИП. (Проверьте это утверждение, следуя данным выше определениям!)

Теперь мы, наконец, дадим определение формулы исчисления предикатов.

Определение 3.10. Слово A в алфавите ИП является формулой тогда и только тогда, когда выполняется одно из четырёх условий

1: A является элементарной формулой;

2: $A = (\neg B)$, где B — формула;

3: $A = (B{\to}C)$, где B, C — формулы;

4: $A = (\forall\, xB)$, где B — формула, x — переменная.

При записи формул удобно применять и другие связки $\wedge, \vee, \sim$, которые обозначают такие же сокращения, как и в случае исчисления высказываний:

$(A \wedge B)$ это сокращение формулы $(\neg(A{\to}(\neg B)))$,

$(A \vee B)$ это сокращение формулы $((\neg A){\to}B)$,

$(A \sim B)$ это сокращение формулы $((A{\to}B) \wedge (B{\to}A))$.

Для сокращения записи формул используется также *квантор существования* $\exists$ по следующему соглашению:

$(\exists\, xA)$ сокращённо обозначает формулу $(\neg(\forall\, x(\neg A)))$.

Аналогично исчислению высказываний, для сокращённой записи формул используются правила опускания скобок. Кванторы по старшинству сильнее $\sim, \to$ и слабее $\vee, \wedge, \neg$. Кроме того, принято опускать скобки, в которые заключается кванторная формула $(\forall\, yA)$.

Задача 3.1. Докажите, что по сокращённой записи формулы в соответствии со сформулированными выше правилами однозначно восстанавливается её полная запись.

Как и формулы исчисления высказываний, формулы исчисления предикатов можно представлять в виде помеченных корневых плоских деревьев (см. с. 20). Вершины дерева, которое соответствует формуле, помечены переменными, символами пропозициональных связок $\to$, $\neg$, предикатными и функциональными символами и кванторными блоками $\forall\, x$. При этом выполняются следующие условия: переменными помечены только листья дерева; у каждой вершины, помеченной

символами $\neg$ или $\forall x$, есть один потомок; у вершины, помеченной символом $\to$, есть два потомка; у вершины, помеченной функциональным или предикатным символом, столько потомков, какова арность этого символа; потомками вершины, помеченной предикатным символом, являются вершины, помеченные переменными или функциональными символами.

Задача 3.2. Докажите, что по формуле ИП можно определить вид каждого входящего в неё символа (предикатный, функциональный, переменная), а также арность предикатных и функциональных символов.

3.3. Интерпретации

В исчислении высказываний формула принимает значение «истина» или «ложь», если присвоены истинностные значения всем входящим в неё переменным. В исчислении предикатов ситуация более сложная. Формула исчисления предикатов описывает некоторое утверждение о свойствах (предикатах) объектов. Чтобы *интерпретировать* это утверждение, нужно указать, о каких объектах и каких их свойствах идёт речь. Интерпретацией формулы в общем случае будет k-арный предикат, который, вообще говоря, истинен или ложен в зависимости от выбора значений входящих в формулу переменных. Если $k = 0$, то такой предикат является одним из истинностных значений — истиной или ложью — независимо от значений входящих в формулу переменных.

Пример 3.11. Рассмотрим формулу $A(x, y)$, $A \in \mathcal{P}_2$, $x, y \in \mathcal{X}$. Чтобы дать её возможную интерпретацию, будем, например, полагать, что x, y принимают значения во множестве целых чисел, а A — бинарный

предикат «x меньше y». Таким образом, интерпретацией нашей формулы является предикат сравнения чисел.

А что можно сказать в этой интерпретации про формулу $\forall\, x A(x, y)$? В соответствии с неформальным пониманием квантора всеобщности мы должны интерпретировать эту формулу как утверждение «для всех x выполнено, что x меньше y». Это утверждение задаёт унарный предикат: при фиксированном значении y оно истинно или ложно.

Наконец, если рассмотреть формулу $\exists\, y \forall\, x A(x, y)$, то она будет интерпретироваться как 0-арный предикат (истина или ложь) и означать утверждение «существует такое целое число, которое больше любого другого числа». (Это утверждение, конечно, ложно.)

Пример 3.11 показывает, как интерпретировать элементарные формулы. Кроме того, из этого примера видно, что применение кванторов понижает арность предиката. Однако из этого примера не вполне ясно, как интерпретировать формулу $\forall\, x \forall\, x A(x, y)$ (и какова арность предиката-интерпретации этой формулы). Все такие неясности будут устранены формальным определением, которое приводится ниже. А пока рассмотрим пример интерпретации формулы с термами общего вида.

Пример 3.12. Интерпретируем формулу

$$\forall\, x E(f(u, x), x)$$

на множестве целых чисел, считая, что $E \in \mathcal{P}_2$ — бинарный предикатный символ, интерпретируемый как предикат равенства, $f \in \mathcal{F}_2$ — бинарный функциональный символ, интерпретируемый как умножение аргументов, $x \in \mathcal{X}$ — символ переменной, $u \in \mathcal{F}_0$ —

константа, которую мы интерпретируем как число 1. Тогда написанная выше формула выражает следующее утверждение «для любого x выполнено равенство $1 \cdot x = x$». Это 0-арный предикат, который принимает по очевидным причинам значение «истина».

Аналогично интерпретируются и более сложные формулы. Рассмотрим ещё один пример.

Пример 3.13. Рассмотрим формулу

$$\forall x(E(f(x,x), f(x, f(x,x))) \to E(f(x,x), x))$$

в интерпретации примера 3.12.

Терм $f(x,x)$ означает (интерпретируется как) x^2, терм $f(x, f(x,x))$ — $x \cdot x^2 = x^3$. А вся формула означает утверждение: «для любого x из того, что куб целого числа x равен квадрату x, следует, что квадрат x числа равен x». Это утверждение истинно (посылка истинна лишь для 0 и 1, но для этих чисел справедливо равенство $x^2 = x$).

Для формального определения интерпретации оказывается полезным следующее техническое понятие.

Определение 3.14. Множество предикатных и функциональных символов с указанием арностей называется *сигнатурой*. Если в формулу входят только предикатные и функциональные символы из сигнатуры σ, то такая формула называется *формулой в сигнатуре* σ.

(Напомним, что константы — это функциональные символы арности 0, так что сигнатура может включать в себя и константы.)

Пример 3.15. Формула $\forall x E(f(u,x), x)$ из примера 3.12 является формулой в сигнатуре E, f, u, где E —

бинарный предикатный символ, f — бинарный функциональный символ, а u — 0-арный функциональный символ (константа).

Определение 3.16. *Интерпретацией* $\mathcal{M}$ сигнатуры σ назовём непустое множество M (*область интерпретации*) и *соответствие интерпретации*, которое каждому k-арному функциональному символу f из σ сопоставляет функцию $[\mathbf{f}]\colon M^k \to M$, отображающую M^k в M, а каждому k-арному предикатному символу A из σ — k-арное отношение $[\mathbf{A}]$ на M, т. е. $[\mathbf{A}] \subseteq M^k$.

Пример 3.17 (*продолжение примера* 3.15). Пусть областью интерпретации является множество действительных чисел $\mathbb{R}$, символу E соответствует отношение равенства

$$[\mathbf{E}] = \{(x, y) : x = y\},$$

символу f — операция сложения

$$[\mathbf{f}]\colon (x, y) \mapsto (x + y),$$

а константе u соответствует 0, т. е. $[\mathbf{u}] = 0$.

Этими условиями определяется некоторая интерпретация формул в сигнатуре E, f, u примера 3.15, в том числе и формулы из примера 3.12.

Интерпретация символов продолжается на все формулы и термы в сигнатуре σ. Термам соответствуют функции со значениями в области интерпретации M, а формулам — предикаты на M. Будем обозначать интерпретации формул и термов, заключая их в квадратные скобки и выделяя полужирным начертанием как это уже сделано в определении интерпретации.

Для формальных определений нам удобно будет представлять формулы и термы как деревья. Как уже

говорилось, листья такого дерева помечены переменными или константами (0-арными функциональными символами).

Определение 3.18 (*правила оценки*). Пусть каждой переменной x_i, которой помечен некоторый лист дерева, присвоено значение из области интерпретации — *оценка переменной*.

В таком случае можно определить *оценку* всех остальных вершин дерева по следующим рекурсивным правилам:

- оценка листа — это оценка переменной, которой помечен лист, или интерпретация константы, которой помечен лист;
- оценка вершины, помеченной k-арным функциональным символом f, равна $[\mathbf{f}](v_1, \ldots, v_k)$, где v_i — оценки потомков данной вершины;
- оценка вершины, помеченной символом A, который является k-арным предикатным символом, равна значению предиката $[\mathbf{A}]$ на наборе $(v_1, \ldots, v_k)$, где v_i — оценки потомков данной вершины;
- оценка вершины, помеченной символом $\neg$, является отрицанием оценки её потомка (у такой вершины ровно один потомок);
- оценка вершины, помеченной символом $\rightarrow$, равна импликации $v_1 \rightarrow v_2$ оценок v_1, v_2 её потомков;
- оценка вершины, помеченной символом $\forall\, x$, равна истине тогда и только тогда, когда оценка её потомка истинна при всех возможных оценках

переменных, отличающихся от данной лишь значением переменной x.

Оценкой формулы является оценка корня её дерева.

Во всех указанных случаях оценка вершины, не являющейся листом, однозначно определяется оценками вершин, которые лежат ниже (т. е. путь из которых в корень проходит через данную вершину), а оценка листа также определена однозначно. Поэтому данное определение корректно. Обратите внимание, что в последнем случае нужно оценивать потомка вершины, помеченной кванторным блоком $\forall x$, при разных, вообще говоря, оценках переменных.

Замечание 3.19. Напомним, что вершинам дерева формулы отвечают подформулы. Поэтому написанные выше правила оценки определяют оценки всех подформул формулы.

Пример 3.20. Рассмотрим формулу $\forall x E(f(u,x),x)$ в интерпретации, построенной в примере 3.17. Оценим эту формулу при значении переменной x, равном 1. Оценка терма $f(u,x)$ по второму правилу из данного выше списка равна $0+1=1$. Оценка формулы $E(f(u,x),x)$ по третьему правилу оценки равна значению предиката $[\mathbf{E}]$ на наборе $(1,1)$, т. е. равна истине.

Теперь осталось оценить значение самой формулы. Здесь нужно применить последнее правило оценки, т. е. вычислить оценку формулы $E(f(u,x),x)$ при всех возможных оценках переменных, отличающихся лишь значением x. Повторив все предыдущие рассуждения для оценки переменной x, равной a, мы получаем, что оценка $E(f(u,x),x)$ равна истине, так как $0+a=a$. Значит, оценка всей формулы $\forall x E(f(u,x),x)$ равна истине.

Пример 3.21. Оценим в той же интерпретации формулу $\forall x \forall x E(x, y)$ при оценке переменных $x = 1$, $y = 1$.

Оценка формулы $E(x, y)$ — истина. Однако при $x = 0$, $y = 1$ оценка формулы $E(x, y)$ — ложь, поэтому по последнему правилу оценки мы заключаем, что оценка формулы $\forall x E(x, y)$ — ложь.

Более того, по аналогичной причине оценка формулы $\forall x E(x, y)$ — ложь при любой оценке переменных x, y. В частности, это означает, что оценка исходной формулы $\forall x \forall x E(x, y)$ — ложь при оценке переменной $x = 1$, $y = 1$.

Пример 3.22. Оценим в той же интерпретации формулу $\forall x E(y, z)$ при оценке переменных $y = 1$, $z = 1$.

Оценка формулы $E(y, z)$ — истина. Для оценки формулы $\forall x E(x, z)$ по последнему правилу оценки нужно оценить $E(y, z)$ при всех оценках y и z, которые отличаются от данной значением переменной x. Но такая оценка единственна и совпадает с исходной. Таким образом, оценка всей формулы — истина.

Мы определим теперь интерпретации термов и формул с помощью введённой процедуры оценки.

Определение 3.23. Интерпретацией терма t является функция $[\mathbf{t}]$, которая отображает набор $(a_1, \ldots, a_k)$ значений переменных $x_1, \ldots, x_k$, входящих в терм, в оценку этого терма при оценке переменных $x_i = a_i$.

Пример 3.24 (*продолжение примера* 3.17). Несложно проверить, что

$$[\mathbf{f}(\mathbf{x}, \mathbf{f}(\mathbf{f}(\mathbf{x}, \mathbf{x}), \mathbf{x}))] : x \mapsto 4x.$$

Интерпретацией формулы является некоторый предикат. Проще всего определить его как оценку

формулы при некоторой оценке переменных. В этом случае получаем n-арный предикат, где n — количество входящих в формулу переменных.

Однако уже из самой структуры формулы можно заметить, что значение такого предиката не зависит от некоторых значений входящих в него переменных. Мы укажем эти переменные и избавимся от них в определении интерпретации формулы.

Определение 3.25. В формуле $\forall\, xA$ подформула A называется *областью действия квантора*. Вхождение переменной x называется *связанным*, если x входит в область действия некоторого квантора по x или в *кванторный блок* $\forall\, x$. Все остальные вхождения переменных называются *свободными*.

Переменные, имеющие свободные вхождения в формулу, будем называть *параметрами* формулы.

Если смотреть на формулы как на деревья, то область действия квантора — это дерево потомков той вершины, которая помечена данным квантором. Вхождение переменной x в формулу свободно, если на пути от листа, помеченного этим вхождением x, до корня дерева не встречается вершина с пометкой $\forall\, x$.

Пример 3.26. Рассмотрим формулы

$$A(x)\to A(y), \tag{3.5}$$

$$\forall\, xB(x,y,x)\to \forall\, yC(z,x), \tag{3.6}$$

$$\forall\, xA(x)\to A(x). \tag{3.7}$$

В первом случае кванторов вообще нет, поэтому все вхождения — свободные.

Во вторую формулу x входит 4 раза. Первые три вхождения — связанные, последнее — свободное. Переменная y первый раз входит в формулу свободно,

второй раз — связанно (как переменная квантора). Единственное вхождение z свободно. Заметим, что второй квантор берётся по переменной, которая ни разу не встречается в его области действия. Это допустимо по нашим правилам, хотя и не имеет особого смысла, как вскоре станет ясно.

В третьей формуле третье вхождение переменной — свободно (напомним, что приоритет импликации ниже, чем приоритет квантора), а остальные — связанны.

Задача 3.3. Укажите параметры формулы

$$\forall y(\forall y A(x,y) \to \forall x A(y,x)) \to \forall x(\forall y A(y,x) \to \forall x A(x,y))\,.$$

Лемма 3.27. *Оценка формулы A зависит только от значений параметров формулы A.*

Доказательство. Индукция по построению формулы. Для элементарных формул утверждение очевидно, так как все переменные входят в элементарную формулу свободно.

Если формула имеет вид $\neg B$, $B \to C$, то её параметры являются параметрами B (или C во втором случае). А по правилам оценки оценка формулы A зависит лишь от оценки формулы B (и оценки C во втором случае), которые по индуктивному предположению зависят только от параметров этих формул.

У формулы A вида $\forall x B$ параметры те же, что и у B, за исключением переменной x. По последнему правилу оценки оценка формулы A не зависит от оценки переменной x. □

Лемма 3.27 обеспечивает корректность следующего определения.

Определение 3.28. *Интерпретацией* [**A**] формулы A назовём k-арный предикат, где k — число параметров формулы A, которые мы обозначим $x_1, \ldots, x_k$. Этот предикат истинен при некотором наборе значений параметров $(a_1, \ldots, a_k)$ тогда и только тогда, когда истинна оценка формулы A при некоторой оценке переменных формулы, в которой $x_i = a_i$.

Мы также будем говорить, что формула A *выражает* предикат P в интерпретации $\mathcal{M}$, если $[\mathbf{A}] = P$ в интерпретации $\mathcal{M}$.

Замечание 3.29. Определение 3.28 может показаться избыточным. Как уже говорилось, сама оценка формулы является предикатом. Однако этот предикат зависит от всех переменных, входящих в формулу, а формальная зависимость от связанных переменных иногда бывает неудобна. Впрочем, принципиального значения эта разница не имеет.

Формула без свободных вхождений переменных называется *замкнутой*. Замкнутая формула интерпретируется как 0-арный предикат (истина или ложь). Таким образом, в заданной интерпретации замкнутые формулы становятся истинными или ложными.

Задачи

3.4. Проверьте истинность следующих утверждений.

1) Формула ИП имеет вид (на месте многоточий стоят какие-то символы)

$$\ldots \forall x \ldots x \ldots$$

и между квантором и вторым из указанных вхождений

переменной x нет других кванторов. Тогда оба указанных вхождения x связанные.

2) Формула ИП имеет вид (на месте многоточий стоят какие-то символы)

$$\dots \forall\, x \dots x \dots$$

и между квантором и вторым из указанных вхождений переменной x нет закрывающих скобок. Тогда оба указанных вхождения x связанные.

3) Формула ИП имеет вид (на месте многоточий стоят какие-то символы)

$$\dots A(x, \dots, x) \dots$$

где A — предикатный символ, а x — символ переменной. Тогда указанные вхождения x либо оба связанные, либо оба свободные.

4) Формула ИП имеет вид (на месте многоточий стоят какие-то символы)

$$\dots A(x, \dots, x) \dots$$

где A — предикатный символ, а x — символ переменной, причём между указанными вхождениями x нет предикатных символов. Тогда указанные вхождения x либо оба связанные, либо оба свободные.

3.5. Приведите пример формулы исчисления предикатов, которая в одной интерпретации задаёт предикат равенства «$x = y$», а в другой — предикат неравенства «$x \neq y$».

3.6. Существует ли такая формула ИП в сигнатуре из унарных предикатных и функциональных символов, что в некоторой интерпретации на бесконечной области она выражает предикат равенства «$x = y$»?

3.7. Пусть F — замкнутая формула ИП в сигнатуре из унарных предикатных и функциональных

символов. Докажите, что если F истинна в некоторой интерпретации на конечной области, то F также истинна в некоторой интерпретации на бесконечной области.

3.4. Модели и теории

Определения формул исчисления предикатов и их интерпретации весьма громоздки. Давайте посмотрим на их выразительную силу: какие утверждения можно записать с их помощью.

Пример 3.30 (*сигнатура теории групп*). Рассмотрим формулы в сигнатуре $e \in \mathcal{C}$, $E \in \mathcal{P}_2$, $m \in \mathcal{F}_2$, $i \in \mathcal{F}_1$.

Вот несколько замкнутых формул в этой сигнатуре.

$$\forall x \forall y \forall z E(m(x, m(y, z)), m(m(x, y), z)), \tag{3.8}$$

$$\forall x \forall y E(i(m(x, y)), m(i(y), i(x))), \tag{3.9}$$

$$\forall x (\neg E(x, e) \to \forall y \exists z E(x, m(z, m(y, i(z))))). \tag{3.10}$$

Будем интерпретировать эти формулы, считая что область интерпретации — группа G, e — единица этой группы, E — предикат равенства, m — функция умножения, i — функция взятия обратного элемента.

Формула (3.8) в такой интерпретации означает свойство ассоциативности умножения. В более привычной записи оно выглядит как

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z,$$

но наш синтаксис не позволяет такую запись. Во-первых, у нас нет *инфиксной* формы записи операции (функции), когда знак функции стоит между знаками аргументов. Во-вторых, мы заранее договорились не включать знак равенства в алфавиты формальных теорий, чтобы использовать его для собственных

нужд. Разумеется, в данном случае перевод из привычной записи в формулу исчисления предикатов очевиден.

Формула (3.9) записывает утверждение о справедливости известного равенства

$$(x \cdot y)^{-1} = y^{-1} \cdot x^{-1}.$$

Формулы (3.8) и (3.9) истинны для любой группы в указанной выше интерпретации. Формула (3.10) утверждает, что все неединичные элементы в группе сопряжены. Для некоторых групп это утверждение верно (скажем, для группы из 2 элементов), а для некоторых нет (например, оно ложно для группы целых чисел по сложению).

Какие ещё утверждения и предикаты в теории групп можно выразить нашими формулами? Достаточно ясно, что можно выразить свойство коммутативности группы, предикат «x имеет порядок 2», предикат сопряжённости «x сопряжён с y» и т. п. (Рекомендуем читателю в качестве упражнения выписать соответствующие формулы.)

Но попробуем задать формулу, которая была бы истинна в точности для циклических групп. Стандартное определение из теории групп говорит о том, что всякий элемент группы является степенью порождающего. И здесь возникает трудность: у нас нет способа записать x^n. Наш синтаксис логики первого порядка игнорирует существование целых или хотя бы натуральных чисел.

Эта проблема типична: очень часто возникает нужда в записи утверждений, в которые входят натуральные числа.

Что можно сказать о натуральных числах в логике первого порядка?

Пример 3.31 (*сигнатура арифметики*). Рассмотрим формулы в сигнатуре $0 \in \mathcal{F}_0$, $E \in \mathcal{P}_2$, $\nu \in \mathcal{F}_1$, $\times \in \mathcal{F}_2$, $+ \in \mathcal{F}_2$.

Будем интерпретировать эти формулы, считая, что область интерпретации — натуральные числа $\mathbb{N} = \{0, 1, \dots\}$, 0 так и будем интерпретировать как нуль во множестве натуральных чисел, E — по-прежнему предикат равенства, ν — функция следования (по определению, $\nu(x) = x + 1$), $\times$ — функция умножения, $+$ — функция сложения.

В такой интерпретации легко записать предикат «x не больше y»:

$$L(x, y) = \exists\, z E(+(z, x), y), \tag{3.11}$$

предикат делимости «x делит y»:

$$D(x, y) = \exists\, z E(\times(z, x), y), \tag{3.12}$$

предикат «r есть остаток от деления x на y»:

$$Q(r, x, y) = L(r, y) \wedge \neg E(r, y) \wedge \exists\, q E(x, +(\times(q, y), r)). \tag{3.13}$$

В последней формуле мы для краткости использовали предикат $L(x, y)$, чтобы переписать её в указанной выше сигнатуре, нужно подставить вместо $L(x, y)$ правую часть (3.11).

Хотя у нас есть имя только для 0, легко построить предикат $\mathrm{Is}_n(x)$, который будет истинен в точности для одного числа n. Вот примеры таких предикатов для чисел 1, 2, 3:

$$\mathrm{Is}_1(x) = E(x, \nu(0)); \tag{3.14}$$

$$\mathrm{Is}_2(x) = E(x, \nu(\nu(0))); \tag{3.15}$$

$$\mathrm{Is}_3(x) = E(x, \nu(\nu(\nu(0)))). \tag{3.16}$$

Для больших чисел n предикаты Is_n равенства числу n будут записываться длинными формулами, хотя и не

настолько длинными, как намекают написанные выше формулы.

Задача 3.8. Докажите, что предикат Is_n можно выразить формулой из $O(\log_2 n)$ символов в приведённой выше интерпретации.

А можно ли выразить в такой сигнатуре предикат «x равен y в степени z»? Это не очень простой вопрос. Но ответ на него положительный. Мы вернёмся к этому вопросу в заключительной главе (пример 4.79 на с. 261).

Хорошо известно, что для арифметики важны доказательства по индукции. Можно ли записать утверждение принципа математической индукции в данной сигнатуре? Для каждого свойства A, которое можно выразить формулой в данной интерпретации, мы можем записать утверждение принципа математической индукции:

$$A(0) \wedge \forall x(A(x) \to A(\nu(x))) \to \forall x A(x).$$

Обычно принцип математической индукции формулируется для произвольных свойств S натуральных чисел: «если 0 удовлетворяет свойству S и для каждого n справедливо, что если n удовлетворяет S, то $n+1$ также удовлетворяет S, то свойством S обладают все натуральные числа.» Записать такое утверждение в логике первого порядка нельзя — требуется квантор по предикатам.

Предыдущий пример показывает, что иногда бывает нужно делать утверждения о произвольных подмножествах бесконечных множеств. Такие утверждения — предмет теории множеств. Рассуждения о множествах требуют особой аккуратности, поскольку довольно легко прийти к противоречию, свободно

перенося интуицию, выработанную для конечных множеств, на бесконечные множества. Следующий пример заведомо некорректен, ибо не соответствует данному выше определению интерпретации (парадокс Рассела B.1 свидетельствует о невозможности построения множества всех множеств). Тем не менее, этот пример даёт представление о том, как записывать утверждения о множествах в логике первого порядка.

Пример 3.32 (*сигнатура теории множеств*). Рассмотрим формулы в сигнатуре $\{E, I\} \in \mathcal{P}_2$.

Будем интерпретировать эти формулы, считая «областью» интерпретации совокупность всех множеств. При этом E означает предикат равенства множеств (совпадение элементов), а I — предикат вхождения ($I(x, y)$ истинен тогда и только тогда, когда x является элементом y).

В такой интерпретации легко выражается предикат «x — пустое множество»:

$$\mathrm{Empty}(x) = \forall\, z \neg I(z, x), \tag{3.17}$$

предикат «x содержит ровно один элемент y»

$$\mathrm{Single}(x, y) = I(y, x) \wedge \forall\, z(I(z, x) \rightarrow E(z, y)), \tag{3.18}$$

предикат «x содержит только элементы y и z»

$$\begin{aligned}\mathrm{Double}(x, y, z) &= \\ &= I(y, x) \wedge I(z, x) \wedge \forall\, t(I(t, x) \rightarrow (E(t, x) \vee E(t, y))).\end{aligned} \tag{3.19}$$

(Обратите внимание, что при $y = z$ этот предикат задаёт одноэлементное множество.)

А можно ли выразить предикат «x — бесконечное множество» (или «x — бесконечное счётное множество»)? Это можно сделать, если определить предикат равномощности множеств. По определению два

множества равномощны, если существует взаимно однозначное отображение между их элементами. Трудность в выражении этого предиката заключается в том, что среди объектов у нас есть только множества и нет отображений.

Преодолеть эту трудность можно, если задавать отображения их графиками. График отображения $f\colon x \to y$ — это множество упорядоченных пар $(t, f(t))$ для всех $t \in x$. Графики можно выделить среди всех множеств следующим образом: если r — такое множество упорядоченных пар (t, u), что для любого t найдётся единственная пара (t, u), принадлежащая r, то r является графиком некоторого отображения.

Чтобы записать эти условия в виде формул в нашей сигнатуре, нужно научиться определять упорядоченную пару и декартово произведение множеств ($x \times y$ по определению состоит из всех упорядоченных пар (t, u), в которых $t \in x$, $u \in y$).

Предикат «z есть упорядоченная пара (x, y)» можно выразить так:

$$\mathrm{Pair}(z, x, y) = \exists\, t \mathrm{Double}(z, x, t) \wedge \mathrm{Double}(t, x, y). \quad (3.20)$$

Эта неуклюжая (особенно если заменить предикат Double на его определение (3.19)) формула выражает очень простую мысль: упорядоченная пара — это неупорядоченная пара с отмеченным первым элементом.

Используя предикат упорядоченной пары, довольно легко записать формулы, выражающие предикаты «быть декартовым произведением», «быть графиком отображения», «быть графиком биективного отображения».

Из этих предикатов можно уже составить предикаты «быть бесконечным множеством», поскольку для

бесконечных множеств, и только для них, существует биекция на собственное подмножество, и «быть счётным множеством», поскольку для счётных множеств, и только для них, верно, что любое бесконечное подмножество равномощно всему множеству.

Можно продолжать этот процесс и далее. В настоящее время более-менее известно как выражать на языке теории множеств все используемые в математике понятия. Именно поэтому считается, что логика первого порядка достаточна для формализации математики.

В рассмотренных примерах мы интерпретировали отдельные формулы. Формальный взгляд на построение математической теории состоит в том, что задаются аксиомы и изучаются следствия, получаемые из них по правилам логики, т. е. формулы, выводимые из аксиом в исчислении предикатов. Поэтому будем считать, что *теория* $\mathcal{T}$ — это просто набор замкнутых формул (аксиом теории). *Моделью* теории $\mathcal{T}$ называется такая интерпретация $\mathcal{M}$, в которой все аксиомы теории истинны. Забегая вперёд, заметим, что в модели истинны и все следствия из аксиом (см. ниже теорему 3.52 о корректности ИП и слабую форму дедукции — теорему 3.55).

Пример 3.33 (*продолжение примера* 3.31). *Формальной арифметикой* называется формальная теория в сигнатуре примера 3.31 с аксиомами

$$\forall x \forall y \forall z (E(x,y) \to (E(y,z) \to E(x,z))), \tag{Ar1}$$

$$\forall x \forall y (E(x,y) \to E(\nu(x), \nu(y))), \tag{Ar2}$$

$$\forall x \forall y (E(\nu(x), \nu(y)) \to E(x,y)), \tag{Ar3}$$

$$\forall x \neg E(0, \nu(x)), \tag{Ar4}$$

$$\forall x E(+(x,0), x), \tag{Ar5}$$

$$\forall\, x E(\times(x,0),0), \tag{Ar6}$$

$$\forall\, x \forall\, y E(+(x,\nu(y)),\nu(+(x,y))), \tag{Ar7}$$

$$\forall\, x \forall\, y E(\times(x,\nu(y)),+(\times(x,y),x)), \tag{Ar8}$$

$$A(0)\to(\forall\, x(A(x)\to A(\nu(x)))\to \forall\, x A(x)). \tag{Ar9}$$

Последняя формула — это схема аксиом, в ней A обозначает любую формулу в данной сигнатуре. Эта аксиоматика для формальной арифметики взята из книги [15].

Моделью формальной арифметики являются, например, натуральные числа, на которых выделена константа 0, и заданы функции следования, сложения и умножения, а также предикат равенства. Чтобы в этом убедиться, нужно проверить истинность приведённых девяти формул в интерпретации примера 3.31. (Перепишите их в общепринятых обозначениях и поймите, какие именно свойства натуральных чисел они утверждают.) Последняя формула (Ar9) выражает, как уже говорилось, ограниченный принцип математической индукции (только для предикатов, выразимых в формальной арифметике).

Задачи

3.9. Рассмотрим интерпретацию на множестве действительных чисел формул с предикатным символом $E(x,y)$ и функциональными символами $a(x,y)$, $m(x,y)$, при которой $E(x,y)$ интерпретируется как $x = y$, $m(x,y)$ — как xy, а $a(x,y)$ — как $x + y$. Существует ли формула, которая в такой интерпретации выражает предикат «x — простое десятизначное число»?

3.10. Докажите, что существует замкнутая формула ИП, которая ложна во всякой интерпретации на

конечной области, но истинна в некоторой интерпретации на бесконечной области.

3.11. Существует ли замкнутая формула ИП, которая истинна в некоторой интерпретации на конечной области, но ложна во всякой интерпретации на бесконечной области?

3.12. Существует ли такая формула исчисления предикатов, которая содержит один бинарный функциональный символ, один унарный предикатный символ, истинна в любых интерпретациях на областях из менее чем 11 элементов и ложна в некоторой интерпретации на области из 11 элементов?

3.13. Формулы в сигнатуре $B(x)$, $P(x, y)$ будем интерпретировать на непустых словах в алфавите $\{0, 1\}$ так: переменные формулы — это номера позиций в слове, $B(i)$ означает, что на i-м месте стоит 1, $P(i, j)$ означает, что $i < j$. Тогда замкнутая формула F определяет множество L_F тех слов, для которых её интерпретация истинна. Проверьте, можно ли определить следующие множества слов формулами первого порядка:

(а) слова, которые начинаются с 0;

(б) слова, которые заканчиваются 1;

(в) слова, содержащие подслово 111;

(г) слова вида $(01)^n$;

(д) слова вида $(001)^n$;

(е*) слова, содержащие чётное количество единиц;

(ж*) слова вида 0^n1^n.

3.5. Общезначимые формулы

Определение 3.34. *Замыкание формулы* A с параметрами $x_1, \ldots, x_n$ — это формула $\forall x_1 \ldots \forall x_n A$.

Определение 3.35. Формула ИП называется *общезначимой*, если её замыкание истинно в любой интерпретации.

Замечание 3.36. При проверке общезначимости формул важно помнить, что по параметрам формулы берутся кванторы всеобщности. Это соответствует принятой практике. Например, утверждение «квадрат действительного числа положителен» имеет смысл в «обиходном» математическом языке. Оно ложно, так как $0^2 = 0$ не является положительным числом.

Определение 3.37. Формула ИП называется *невыполнимой*, если замыкание её отрицания общезначимо. В противном случае формула называется *выполнимой*.

Легко видеть из определений интерпретации, что формула выполнима тогда и только тогда, когда у неё есть истинная оценка в какой-нибудь интерпретации при какой-нибудь оценке переменных.

Общезначимые формулы — это обобщение тавтологий в исчислении высказываний. Можно сформулировать это утверждение более формально.

Утверждение 3.38. *Пусть A — тавтология, x_1, $\ldots$, x_n — набор пропозициональных переменных, которые входят в A, а F_1, $\ldots$, F_n — формулы* ИП. *Тогда формула $A(F_1, \ldots, F_n)$ общезначима.*

Доказательство. По правилам оценки 3.18 оценка формулы $A(F_1, \ldots, F_n)$ зависит лишь от оценок формул F_i и определяется точно так же, как в исчислении высказываний (четвёртое и пятое правила оценки). Каковы бы ни были эти оценки, оценкой $A(F_1, \ldots, F_n)$ будет истина, так как тавтология истинна при любых значениях входящих в неё переменных. Значит,

в любой интерпретации оценкой замыкания формулы $A(F_1, \ldots, F_n)$ будет истина. □

Приведём несколько примеров общезначимых формул, не являющихся тавтологиями.

Пример 3.39. Докажем, что формула

$$\exists x \forall y A(x, y) \to \forall y \exists x A(x, y) \tag{3.21}$$

общезначима. Интерпретацию бинарного предикатного символа удобно представлять в виде ориентированного графа (быть может, бесконечного). Множество вершин графа совпадает с областью интерпретации, а упорядоченная пара (x, y) принадлежит множеству рёбер графа тогда и только тогда, когда пара (x, y) принадлежит отношению $[\mathbf{A}]$.

При таком понимании посылка написанной формулы означает, что в графе есть вершина x, из которой идут рёбра во все остальные вершины. Но тогда для любой вершины графа найдётся входящее в неё ребро (оно выходит из вершины x). Именно это и утверждает заключение формулы.

Пример 3.40. Докажем, что формула

$$\forall x A(x) \to (\exists x \neg B(x) \to \exists x \neg (A(x) \to B(x))) \tag{3.22}$$

общезначима. Пусть посылка истинна в некоторой интерпретации. Тогда при оценке заключения $A(x)$ истинна при любой оценке переменных, поэтому оценка $\neg(A(x) \to B(x))$ совпадает с оценкой $\neg B(x)$. Поэтому оценка заключения в рассматриваемой формуле совпадает с оценкой формулы

$$\exists x \neg B(x) \to \exists x \neg B(x),$$

которая является тавтологией.

На общезначимые формулы можно смотреть как на «законы формальной логики». При этом обращение с такими законами требует некоторой аккуратности. Рассмотрим несколько примеров.

Определение 3.41. Формулы A и B исчисления предикатов называются *эквивалентными*, если формула $A \sim B$ общезначима.

Определение истинностного значения формул таково, что оно зависит лишь от значений подформул. Отсюда следует такое утверждение.

Лемма 3.42 (лемма о замене). *Пусть формула B получается из формулы A заменой подформулы A' на эквивалентную ей формулу B'. Тогда A и B эквивалентны.*

Доказательство. Рассмотрим процесс вычисления оценки формулы $A \sim B$ при некоторой оценке переменных в некоторой интерпретации. Он состоит в последовательном вычислении оценок вершин дерева формулы от листьев к корню, причём в каждой вершине дерева вычисление оценки зависит лишь от значений оценок потомков этой вершины.

Значит, оценки вершин деревьев формул A и B могут различаться лишь в вершинах, соответствующих подформулам A' и B', и на путях из этих вершин к корню.

Формула $A' \sim B'$ общезначима. Поэтому оценки формул A' и B' всегда совпадают. (Напомним, что $A \sim B$ как булева функция равна $(A{\to}B){\wedge}(B{\to}A)$ или $\neg(A \oplus B)$.) Но тогда будут совпадать и остальные оценки на путях из вершин A', B' к корню. Таким образом, оценки формул A и B оказываются одинаковыми. Но это и означает, что формула $A \sim B$ истинна. $\square$

Фактически мы доказали больше: если формула B получается из формулы A заменой подформулы A' на формулу B', а замыкание $A' \sim B'$ истинно в некоторой интерпретации, то и замыкание $A \sim B$ истинно в этой интерпретации.

Но дальнейшее усиление оказывается ложным. Пусть формула B получается из формулы A заменой подформулы A' на формулу B', а оценка формулы $A' \sim B'$ в некоторой интерпретации при некоторой оценке переменных является истиной. Может так случиться, что при этом оценка формулы $A \sim B$ является ложью.

Пример 3.43. Рассмотрим формулы

$$A = \forall\, xE(x, f(x)) \quad \text{и} \quad B = \forall\, xE(x, x).$$

Формула B получается из A заменой подформулы $A' = E(x, f(x))$ на формулу $B' = E(x, x)$.

Рассмотрим интерпретацию на множестве действительных чисел, в которой E интерпретируется как отношения равенства, а $[\mathbf{f}] = x^2$. Тогда при оценке переменной $x = 1$ оценкой формулы $A' \sim B'$ является истина ($1 = 1^2$ истинна и $1 = 1$ истинна). Но $A \sim B$ в такой интерпретации ложна (B истинна, а A ложна, так как $2 \neq 2^2$).

Рассмотрим два важных примера, когда формальные правила оценки формул дают неожиданный результат.

Пример 3.44. Общезначима ли формула

$$\forall\, x(A(x) \to A(x)) \to (A(x) \to \forall\, xA(x))? \tag{3.23}$$

Посылка $\forall\, x(A(x) \to A(x))$ истинна, поскольку в ней под квантором записана тавтология, истинная при любом значении x. А вот заключение может быть ложно.

Чтобы унарный предикат $A(x) \to \forall\, x A(x)$ был ложен при некоторых x, достаточно, чтобы предикат $A(x)$ при каком-то x был истинен, а при каком-то — ложен. Тогда $\forall\, x A(x)$ ложна, и при тех x, для которых предикат A истинен, предикат $A(x) \to \forall\, x A(x)$ ложен. Таким образом, формула (3.23) не общезначима.

Утверждение 3.45. *Если в формуле A нет свободных вхождений переменной x, то формула*

$$\forall\, x(A \to B) \to (A \to \forall\, x B)$$

общезначима.

Доказательство. Предположим, что в некоторой интерпретации при некоторой оценке переменных π оценка указанной формулы — ложь. Это означает, что оценка $\forall\, x B$ — ложь, а оценка A — истина.

Таким образом, при некоторой оценке переменных π', отличающейся от оценки π лишь значением переменной x, оценка формулы B — ложь. Формула A не содержит свободных вхождений x, значит, оценка формулы A не зависит от оценки x. Поэтому оценка $A \to B$ при оценке переменных π' — ложь. Но тогда по последнему правилу оценки оценка формулы $\forall\, x(A \to B)$ при оценке переменных π — ложь.

Пришли к противоречию, так как при ложной посылке импликация истинна. □

Пример 3.44 показывает, что условие утверждения 3.45 (в формуле A нет свободных вхождений переменной x) существенно.

Рассмотрим ещё один пример, который показывает необходимость осторожного обращения с кванторами. Этот пример связан с заменой переменных в формуле и, более общó, с подстановкой в формулу терма вместо переменной.

Обозначение. Через $A(x{:=}t)$ будем обозначать формулу, которая получается из A подстановкой терма t вместо всех свободных вхождений переменной x.

Пример 3.46. Интуитивный смысл формулы $\forall\, xA$ заключается в том, что формула A истинна при любых значениях x. Это, казалось бы, означает, что можно заменить в A свободные вхождения x на любой терм: ведь терм обозначает некоторый объект, а формула A верна для всех объектов. Поэтому кажется правдоподобным, что любая формула вида $\forall\, xA{\to}A(x{:=}t)$ общезначима. Однако рассмотрим формулу $\neg\forall\, yA(x,y)$ и подставим в неё вместо x терм y. Формула

$$\forall\, x(\neg\forall\, yA(x,y)){\to}(\neg\forall\, yA(y,y)) \tag{3.24}$$

не общезначима. Рассмотрим интерпретацию на множестве натуральных чисел, в которой $A(x,y)$ означает равенство. Тогда заключение импликации ложно. Формула $\forall\, yA(x,y)$ в такой интерпретации тождественно ложна (для любого числа x найдётся не равное ему число). Значит, отрицание этой формулы тождественно истинно. Таким образом, $\forall\, x(\neg\forall\, yA(x,y))$ истинна в этой интерпретации, а формула (3.24) — ложна.

Патология в примере 3.46 возникает из-за того, что подстановка меняет «связанность» вхождений переменных.

Определение 3.47. Будем говорить, что *терм t свободен для переменной x в формуле A*, если ни одно свободное вхождение x в A не лежит в области действия квантора по переменной, входящей в терм t.

Возвращаясь к примеру 3.46, заметим, что в формуле $\neg\forall\, yA(x,y)$ терм y несвободен для переменной x.

Теперь можно уточнить неформальные соображения из примера 3.46.

Утверждение 3.48. *Если терм t свободен для переменной x в формуле A, то формула $\forall x A \to A(x{:=}t)$ общезначима.*

Доказательство. Если заключение импликации истинно, то и вся импликация истинна.

Теперь предположим, что в некоторой интерпретации при некоторой оценке переменных π формула $A(x{:=}t)$ имеет оценку ложь.

Обозначим через ξ оценку терма t при оценке переменных π. Рассмотрим оценку переменных π', которая отличается от оценки π разве что в переменной x, оценка π' которой равна ξ. Сравним оценку формулы A при оценке переменных π' и оценку формулы $A(x{:=}t)$ при оценке переменных π.

Так как терм t свободен для переменной x в формуле A, то свободны все вхождения переменных в экземпляры терма t, подставленные вместо свободных вхождений x. Поэтому при оценке формулы $A(x{:=}t)$ оценка любой вершины, которая лежит на пути от вершины t к корню, использует лишь оценку ξ терма t при оценке переменных π (последнее правило оценки не применяется, так как все вхождения переменных в терм t свободны).

В точности те же оценки получат эти вершины, когда вычисляется оценка формулы A при оценке переменных π'.

В остальных вершинах оценки формул A и $A(x{:=}t)$ и подавно совпадают (там записано одно и то же).

Итак, при оценке переменных π' оценкой формулы A является ложь. То же самое верно для формулы

$A(x{:=}t)$ при оценке π. Тогда по последнему правилу оценки формула $\forall\, xA$ имеет оценку ложь при оценке π. Значит, в этом случае оценка формулы $\forall\, xA{\to}A(x{:=}t)$ — истина.

Итак, при обеих возможных оценках формулы $A(x{:=}t)$ формула $\forall\, xA{\to}A(x{:=}t)$ имеет оценку истина. Поэтому она общезначимая. □

Оценки связанных переменных не влияют на оценку формулы. Поэтому замена имени связанной переменной даёт эквивалентную формулу. Сформулируем это утверждение строго.

Лемма 3.49. *Пусть формула A не содержит свободных вхождений переменной y. Тогда формулы $\forall\, xA$ и $\forall\, yA(x{:=}y)$ эквивалентны. Более того, $[\forall\, \mathbf{xA}]$ и $[\forall\, \mathbf{yA}(\mathbf{x}{:=}\mathbf{y})]$ равны как предикаты.*

Доказательство. Из условий леммы следует, что параметры формул $\forall\, xA$ и $\forall\, yA(x{:=}y)$ совпадают. Поэтому, в частности, из эквивалентности этих формул будет следовать совпадение предикатов $[\forall\, \mathbf{xA}]$ и $[\forall\, \mathbf{yA}(\mathbf{x}{:=}\mathbf{y})]$.

Рассмотрим некоторую интерпретацию и оценку π переменных формулы $\forall\, xA \sim \forall\, yA(x{:=}y)$. Пусть оценка переменных π' отличается от π разве что значением $x = \xi$, оценка переменных π'' отличается от π разве что значением $y = \xi$. Докажем, что оценка формулы A при оценке переменных π' совпадает с оценкой формулы $A(x{:=}y)$ при оценке переменных π''.

В самом деле оценки листьев для деревьев этих двух формул различаются лишь для связанных вхождений переменных x и y. Но по лемме 3.27 оценки тех вершин дерева формулы $A(x{:=}y)$, которые помечены кванторными блоками $\forall\, x$ и $\forall\, y$, будут такими же, как

и для формулы A при оценке π', так что эти различия не влияют на общую оценку формулы.

Итак, формула A при оценке π' ложна тогда и только тогда, когда $A(x{:=}y)$ ложна при оценке π''. По последнему правилу оценки это означает, что из ложности формулы $\forall\, xA$ при оценке π следует ложность формулы $\forall\, yA(x{:=}y)$ при той же самой оценке переменных и наоборот. Это означает истинность формулы $\forall\, xA \sim \forall\, yA(x{:=}y)$. □

Лемма 3.49 позволяет заменять переменные в формуле. Важный частный случай применения этой леммы — замена переменной на *новую переменную*, т. е. переменную, которая не встречается в формуле.

Определение 3.50. Формула A является *формулой с разделёнными переменными*, если каждая переменная x либо входит свободно в формулу A, либо входит в область действия ровно одного квантора.

Лемма 3.51. *Для каждой формулы существует эквивалентная ей формула с разделёнными переменными.*

Доказательство. Лемма о замене 3.42 означает, что замена на новую переменную в кванторной подформуле $\forall\, xB$ формулы A приводит к формуле, эквивалентной A. Применим замену на новые переменные к каждой кванторной подформуле формулы A. Получим эквивалентную A формулу с разделёнными переменными. □

Задачи

3.14. Проверьте общезначимость формул

а) $\forall\, x\, \exists\, yA(x,y) \to \exists\, x\, \forall\, yA(x,y)$;

б) $\exists\, x\, \forall\, y(A(x,y) \wedge \neg A(y,x) \to (A(x,x) \sim A(y,y)))$;

в) $\forall y \exists x \big((\exists x A(x,y)) \to (\forall y A(x,y))\big)$;

г) $\forall x \exists y \forall u \exists v (A(x,y) \to A(u,v))$;

д) $\forall x \exists y (A(x,y) \to A(x, f(x)))$.

3.15. Формула исчисления предикатов имеет вид

$$\mathsf{Q}x\, A(f(x, f(x, f(x,x))), f(f(x,x), f(x,x))),$$

где Q — один из кванторов $\forall$, $\exists$. Докажите, что эта формула не общезначима.

3.16. Проверьте выполнимость формул

а) $\forall x \exists y (A(f(y), y) \to \neg A(x, f(x)))$;

б) $\forall x \forall y (A(x,y) \to \neg A(y,x))$;

в) $\exists x \forall y (A(x,y) \wedge \neg A(x, f(x)))$.

3.17. Известно, что для формулы A с одним параметром x выполнимы и формула $\forall x A$, и формула $\exists x A$. Верно ли, что формула

$$\exists x \forall y (A(x) \wedge A(x := y))$$

также выполнима?

3.18. Существует ли такая формула A исчисления предикатов с параметрами x, y, что формула $\forall x \exists y A$ общезначима, а формула $\exists x \forall y A$ невыполнима?

3.19. Проверьте эквивалентность формул

$$\forall x ((\exists y A(x,y)) \to (\exists y B(x,y)))$$

и

$$\forall x \exists y (A(x,y) \to B(x,y)).$$

3.20. Проверьте истинность следующих утверждений.

1) Если переменная x входит свободно в формулу A ИП, то, заменив в формуле A все вхождения переменной x на y, мы получим эквивалентную формулу.

2) Если переменная y не входит в формулу A ИП, то, заменив в формуле A все связанные вхождения переменной x на y, мы получим эквивалентную формулу.

3) Если переменная x не имеет свободных вхождений в формулу A ИП, то, заменив в формуле A все вхождения переменной x на y, мы получим эквивалентную формулу.

4) Если переменная y не входит в формулу A ИП, то, заменив в формуле A все свободные вхождения переменной x на y, мы получим эквивалентную формулу.

3.21. Является ли общезначимой формула F ИП в сигнатуре из четырёх унарных предикатных символов, если

а) F истинна в любой интерпретации на области из 17 элементов;

б) F истинна в любой интерпретации на области из 10 элементов.

3.6. Вывод в исчислении предикатов

Вернёмся к построению формальной системы ИП. Нам осталось определить аксиомы и правила вывода. При этом мы хотим, чтобы ИП давала адекватную формализацию описанной выше семантики (интерпретации формул). Другими словами, необходимо обеспечить корректность (все аксиомы должны быть истинными в любой интерпретации, т. е. быть общезначимыми; применение правил вывода к общезначимым формулам даёт общезначимую формулу) и полноту (общезначимые формулы должны быть выводимы).

Приводимые ниже аксиомы и правила вывода подобраны так, чтобы сделать доказательство теоремы о полноте как можно проще. В частности, они включают в себя аксиомы и правило вывода исчисления

высказываний. Это упрощает доказательства — можно ссылаться на теорему о полноте исчисления высказываний, когда речь идёт о тавтологиях.

3.6.1. Аксиомы и правила вывода

Вот один из возможных списков аксиом исчисления предикатов:

$$A\to(B\to A) \tag{A1}$$
$$(A\to(B\to C))\to((A\to B)\to(A\to C)) \tag{A2}$$
$$(\neg B\to\neg A)\to((\neg B\to A)\to B) \tag{A3}$$
$$\forall\, xA\to A(x:=t) \qquad \text{в условиях утв. 3.48} \tag{A4}$$
$$\forall\, x(A\to B)\to(A\to\forall\, xB) \qquad \text{в условиях утв. 3.45} \tag{A5}$$

Список состоит из схем аксиом исчисления высказываний и двух «кванторных» схем аксиом. Аксиомы исчисления высказываний общезначимы в силу утверждения 3.38. Некоторые формулы схем (A4), (A5) не являются общезначимыми, см. примеры 3.46, 3.44. Но при дополнительных условиях, которые сформулированы в утверждениях 3.48, 3.45, получаются только общезначимые формулы.

Правил вывода в ИП два. Во-первых, это правило modus ponens:

> если выводимы формулы A и $A\to B$, то выводима формула B.

Второе описывает работу с кванторами. Оно называется правилом обобщения (Gen):

> если выводима формула A, то выводима формула $\forall\, xA$.

Теперь построение формальной системы ИП закончено: мы определили все компоненты формальной системы — формулы, аксиомы, правила вывода.

Теорема 3.52 (корректность исчисления предикатов). *В* ИП *выводимы только общезначимые формулы.*

Доказательство. Нужно проверить общезначимость аксиом и то, что правила вывода сохраняют общезначимость.

Мы уже доказали общезначимость аксиом: аксиомы (А1–А3) общезначимы по утв. 3.38, а общезначимость аксиом (А4–А5) доказана в утверждениях 3.48, 3.45.

Если при некоторой оценке переменных формулы A, $A{\to}B$ истинны, то и формула B также истинна при этой оценке переменных. Поэтому правило modus ponens из общезначимых формул выводит только общезначимые.

Остаётся проверить, что правило обобщения сохраняет общезначимость. Действительно, из общезначимости A следует, что в любой интерпретации при любой оценке переменных она истинна. Но тогда по определению интерпретации квантора формула $\forall\, x A$ также истинна в любой интерпретации при любой оценке переменных. □

Как следствие получаем непротиворечивость исчисления предикатов. Поскольку в исчислении предикатов выводятся только общезначимые формулы, то из формул A и $\neg A$ выводимой может быть только одна.

Буквально так же, как в исчислении высказываний, определяется вывод из множества формул (гипотез). Формулы, выводимые из аксиом исчисления предикатов и аксиом некоторой теории $\mathcal{T}$, называются *выводимыми в* $\mathcal{T}$ или (формальными) теоремами теории $\mathcal{T}$.

Так как аксиомы исчисления предикатов содержат аксиомы исчисления высказываний, а правило modus ponens является одним из правил вывода, то любая тавтология выводима в исчислении предикатов. Мы сформулируем это утверждение в виде леммы.

Лемма 3.53. *Пусть A_1, A_2, ..., A_n — формулы исчисления предикатов, а $\Phi(x_1, \ldots, x_n)$ — тавтология, в которую входят переменные x_1, ..., x_n. Тогда формула $\Phi(A_1, \ldots, A_n)$ выводима в исчислении предикатов.*

Доказательство этой леммы очевидно — нужно подставить в вывод формулы Φ в исчислении высказываний вместо переменных x_i формулы A_i. Полученная последовательность формул будет выводом в исчислении предикатов.

3.6.2. Слабая теорема дедукции

В исчислении высказываний построение вывода существенно облегчалось теоремой дедукции. В исчислении предикатов теорема дедукции неверна. Вот простой пример: $A \vdash \forall\, x A$. Вывод:

1. A (гипотеза)
2. $\forall\, x A$. (Gen 1)

Среди формул вида $A \to \forall\, x A$ есть необщезначимые. Пусть у формулы A два параметра x, y. Рассмотрим интерпретацию на множестве натуральных чисел, в которой A интерпретируется как предикат равенства. Тогда предикат $A(x, y) \to \forall\, x A(x, y)$ ложен, если $x = y$. Действительно, при таких значениях параметров посылка импликации истинна, а заключение — ложно (не все числа равны).

Однако в исчислении предикатов верен ослабленный вариант теоремы дедукции.

Теорема 3.54 (ослабленная теорема дедукции). *Если* $\Gamma, A \vdash B$ *и существует вывод* B *из* Γ, A, *в котором не применяется правило обобщения к свободным переменным формулы* A, *то* $\Gamma \vdash A \to B$.

Доказательство. Следуем схеме доказательства теоремы дедукции для исчисления высказываний: вывод $\Phi_1, \ldots, \Phi_n$ формулы B из гипотез Γ, A, который существует по условию теоремы, заменяется на последовательность формул $A \to \Phi_1, \ldots, A \to \Phi_n$ и индукцией по n доказывается, что эту последовательность можно дополнить до вывода. Для этого надо разобрать два случая.

1. Формула Φ_i является аксиомой (A1–A5) или формула Φ_i получена по правилу modus ponens из предыдущих формул Φ_j, Φ_k, $j, k < i$. В этих случаях построение вывода $A \to \Phi_i$ в предположении, что уже выведены формулы $A \to \Phi_k$, $k < i$, в точности повторяет доказательство теоремы дедукции для исчисления высказываний.

2. Формула Φ_i является выводом по правилу Gen из формулы Φ_k, $k < i$. Это означает, что $\Phi_i = \forall x \Phi_k$. Добавим следующие формулы перед формулой $A \to \Phi_i$:

$$\forall x (A \to \Phi_k) \qquad \text{(Gen)}$$

$$\forall x (A \to \Phi_k) \to (A \to \forall x \Phi_k) \qquad \text{(A5)}$$

В первой строке правило обобщения применяется к формуле $A \to \Phi_k$, которая встречается раньше формулы $A \to \Phi_i$ в последовательности формул, которую мы пополняем. Вторая строка — это аксиома (A5). Условия утверждения 3.45 в данном случае выполнены,

потому что по условию x не является свободной переменной формулы A. $\square$

Следствие 3.55 (слабая теорема дедукции).
Если $\Gamma, A \vdash B$ *и формула* A *замкнутая, то* $\Gamma \vdash A \to B$.

Доказательство. Замкнутая формула вообще не содержит свободных переменных, поэтому любой вывод B из Γ, A удовлетворяет условию теоремы 3.54. $\square$

Приведём пример использования теоремы дедукции.

Пример 3.56. Докажем, что $\vdash \forall x \forall y A \to \forall y \forall x A$.

Построим вывод $\forall x \forall y A \vdash \forall y \forall x A$, удовлетворяющий условию ослабленной теоремы дедукции.

1.	$\forall x \forall y A$	(гипотеза)
2.	$\forall x \forall y A \to \forall y A$	(A4)
3.	$\forall y A$	(MP 1,2)
4.	$\forall y A \to A$	(A4)
5.	A	(MP 3,4)
6.	$\forall x A$	(Gen 5)
7.	$\forall y \forall x A$	(Gen 6)

В этом примере (строки 2,4) видно применение аксиомы (A4) в важном частном случае, когда вместо переменной подставляется она сама. Такая подстановка удовлетворяет условию корректности: свободное вхождение переменной по определению не попадает в область действия квантора по этой переменной. Правило обобщения в этом выводе применяется к переменным, которые не могут входить в $\forall x \forall y A$ свободно, поэтому условия ослабленной теоремы дедукции выполнены. Отсюда получаем $\vdash \forall x \forall y A \to \forall y \forall x A$.

3.6.3. Примеры выводов

Докажем ослабленный «синтаксический» вариант леммы 3.49.

Лемма 3.57. *Пусть формула A имеет единственный параметр x и не содержит вхождений переменной y. Тогда $\forall\, yA(x{:=}y) \vdash \forall\, xA$.*

Доказательство. Вот вывод формулы $\forall\, xA$.

1.	$\forall\, yA(x{:=}y)$	(гипотеза)
2.	$\forall\, yA(x{:=}y){\to}A$	(A4)
3.	A	(MP 1,2)
4.	$\forall\, xA$	(Gen 3)

Единственный шаг в этом выводе, который нуждается в объяснениях, это шаг 2. Формула A получается из $A(x{:=}y)$ подстановкой вместо переменной y терма x (состоящего из одной переменной). Так как формула $A(x{:=}y)$ получена заменой всех свободных вхождений x на y, терм x свободен для переменной y в формуле $A(x{:=}y)$. Таким образом, условия утверждения 3.48 выполнены и вторая строка действительно является частным случаем схемы аксиом (A4). □

Лемма 3.58. *Пусть формула A имеет единственный параметр x. Тогда $\neg\forall\, xA \vdash \exists\, x\neg A$.*

Доказательство. Напомним, что формула $\exists\, x\neg A$ — это сокращённая запись формулы $\neg\forall\, x\neg\neg A$. Докажем вначале, что $\forall\, x\neg\neg A \vdash \forall\, xA$:

1.	$\forall\, x\neg\neg A$	(гипотеза)
2.	$\forall\, x\neg\neg A{\to}\neg\neg A$	(A4)
3.	$\neg\neg A$	(MP 1,2)
4.	$\neg\neg A{\to}A$	(лемма 1.24a)

5. A (MP 3,4)
6. $\forall x A$ (Gen 5)

На втором шаге этого вывода мы применили аксиому (A4) с подстановкой терма x вместо переменной x (такая подстановка всегда корректна).

По слабой теореме дедукции получаем, что формула $(\forall x \neg\neg A) \to \forall x A$ выводима. Так как в исчислении высказываний выводима формула $(A \to B) \to (\neg B \to \neg A)$, то получаем выводимость формулы $\neg\forall x A \to \neg\forall x \neg\neg A$. По правилу modus ponens из этой формулы и гипотезы $\neg\forall x A$ выводится формула $\neg\forall x \neg\neg A$. $\square$

3.7. Полнота исчисления предикатов

Идея доказательства теоремы о полноте исчисления предикатов повторяет идею второго доказательства теоремы о полноте исчисления высказываний: нужно доказать, что всякая непротиворечивая теория имеет модель.

В исчислении высказываний нам было достаточно построить расширение непротиворечивой теории (множества формул) до непротиворечивой полной теории. После этого уже можно было указать те значения переменных, на которых истинны все формулы из нашей теории.

В случае исчисления предикатов дело обстоит сложнее. По-прежнему можно любую непротиворечивую теорию расширить до полной непротиворечивой. Доказательство повторяет рассуждение из исчисления высказываний.

Однако одного этого шага для построения модели оказывается недостаточно. Здесь применяется новая идея — расширение сигнатуры теории. К теории

добавляются новые константы. Требуется это для того, чтобы для выводимых формул вида $\exists\, x A$ можно было указать константу c, для которой выводима формула $A(x{:=}c)$. Понятно, почему такое требование помогает строить интерпретацию — у нас возникает явный «свидетель» истинности формулы $\exists\, x A$.

Пополнение константами, вообще говоря, делает теорию неполной. Поэтому нужно чередовать пополнение константами и расширение до полной теории счётное число раз. Тогда получится расширение исходной теории, для которого модель строится наиболее естественным образом: в качестве элементов области интерпретации берутся замкнутые термы (т. е. термы, построенные из констант) теории.

Подробно этот план реализуется ниже.

Выводимой в теории $\mathcal{T}$ формулой будем называть формулу, выводимую в ИП из множества формул $\mathcal{T}$. (Напомним, что теорией мы назвали множество замкнутых формул, см. с. 124.)

Лемма 3.59. *Если замкнутая формула A невыводима в теории $\mathcal{T}$, то теория $\mathcal{T} \cup \{\neg A\}$ непротиворечива.*

Доказательство. Пусть $\mathcal{T} \cup \{\neg A\}$ противоречива и для некоторой формулы B выполняются выводимости

$$\mathcal{T} \cup \{\neg A\} \vdash B, \quad \mathcal{T} \cup \{\neg A\} \vdash \neg B,$$

т. е. $\mathcal{T} \cup \{\neg A\} \vdash B \wedge \neg B$. Тогда по слабой теореме дедукции $\mathcal{T} \vdash \neg A{\to}(B \wedge \neg B)$. В исчислении высказываний существует вывод $\neg A{\to}(B \wedge \neg B) \vdash A$, так как $(\neg A{\to}(B \wedge \neg B)){\to}A$ — тавтология. Этот же вывод является выводом в исчислении предикатов. Отсюда получаем выводимость $\mathcal{T} \vdash A$ и приходим к противоречию. $\square$

Лемма 3.60 (лемма Линденбаума). *Если теория непротиворечива, то её можно расширить до непротиворечивой полной теории.*

Доказательство. Оно фактически повторяет доказательство леммы 1.36. Множество формул ИП счётно. Пусть последовательность

$$F_1, F_2, \ldots, F_n, \ldots$$

содержит все формулы. Построим по этой последовательности бесконечную возрастающую последовательность теорий (множеств формул)

$$\mathcal{T} = \mathcal{T}_0 \subseteq \mathcal{T}_1 \subseteq \cdots \subseteq \mathcal{T}_n \subseteq \ldots,$$

обладающую таким свойством: все теории в этой последовательности непротиворечивы и для любого $n \geqslant 1$ в теории $\mathcal{T}_n$ для любого $i \leqslant n$ выводима одна из формул F_i, $\neg F_i$. Для построения такой последовательности используем лемму 3.59.

Теория $\mathcal{T}' = \bigcup_{n \geqslant 0} \mathcal{T}_n$ будет искомой полной непротиворечивой теорией, содержащей теорию $\mathcal{T}$. Это доказывается буквально так же, как в лемме 1.36. □

Рассмотрим выводимые в теории $\mathcal{T}$ *экзистенциальные формулы*, т. е. формулы вида $\exists\, x A$, где A имеет единственный параметр x.

Расширим сигнатуру, добавив для каждой выводимой в теории $\mathcal{T}$ экзистенциальной формулы E константу c_E. Саму теорию $\mathcal{T}$ расширим формулами вида $A(x{:=}c_E)$ для всех выводимых в теории $\mathcal{T}$ экзистенциальных формул $E = \exists\, x A$. Полученную теорию назовём *экзистенциальным расширением* теории $\mathcal{T}$.

Лемма 3.61. *Экзистенциальное расширение непротиворечивой теории непротиворечиво.*

Доказательство. Обозначим экзистенциальное расширение теории $\mathcal{T}$ через $\mathcal{T}'$. Предположим, что $\mathcal{T}'$ противоречива. Тогда $\mathcal{T}' \vdash B \wedge \neg B$ для некоторой формулы B. В выводе $B \wedge \neg B$ содержится конечное число гипотез $A(x{:=}c_E)$, не принадлежащих теории $\mathcal{T}$. Мы модифицируем вывод, удаляя из него по очереди все эти гипотезы.

Рассмотрим один шаг этого процесса. Пусть в выводе противоречия встречается гипотеза $A(x{:=}c_E)$. Обозначим $\mathcal{T}'_1 = \mathcal{T}' \setminus \{A(x{:=}c_E)\}$. Поскольку

$$\mathcal{T}'_1 \cup \{A(x{:=}c_E)\} \vdash B \wedge \neg B,$$

то по слабой теореме дедукции

$$\mathcal{T}'_1 \vdash A(x{:=}c_E) \to B \wedge \neg B.$$

Заменим константу c_E на новую переменную y, которая ни разу не встречалась в этом выводе. Получим вывод

$$\mathcal{T}'_1 \vdash A(x{:=}y) \to (B \wedge \neg B)(c_E{:=}y).$$

Условия корректности для аксиом (A4–A5) выполняются в новой последовательности формул, так как все вхождения переменной y в формулы этой последовательности свободны.

По правилу контрапозиции (лемма 1.25), существует вывод

$$A(x{:=}y) \to (B \wedge \neg B)(c_E{:=}y) \vdash \neg(B \wedge \neg B)(c_E{:=}y) \to \neg A(x{:=}y).$$

Формула $\neg(B \wedge \neg B)$ является тавтологией и поэтому выводима. Значит, $\mathcal{T}'_1 \vdash \neg A(x{:=}y)$. Теперь применим правило Gen и выведем $\forall y \neg A(x{:=}y)$. По лемме 3.57 отсюда следует выводимость формулы $\forall x \neg A$ в теории $\mathcal{T}'_1$.

Однако в теории $\mathcal{T}$ выводима формула $\exists x A$, которая является сокращением для $\neg \forall x \neg A$. Поэтому

в теории $\mathcal{T}_1' \supseteq \mathcal{T}$ существует вывод противоречия $(\forall x \neg A) \wedge (\neg \forall x \neg A)$, в котором не используется гипотеза $A(x{:=}c_E)$.

Продолжая делать такие шаги, мы избавимся в выводе от всех гипотез, не принадлежащих теории $\mathcal{T}$. Получим вывод противоречия в теории $\mathcal{T}$. $\square$

Определение 3.62. Теория $\mathcal{T}$ называется *экзистенциально полной*, если для любой выводимой в $\mathcal{T}$ экзистенциальной формулы $\exists x A$ существует константа c, для которой в теории $\mathcal{T}$ выводима формула $A(x{:=}c)$.

Лемма 3.63. *Для любой непротиворечивой теории существует её непротиворечивое полное и экзистенциально полное расширение.*

Доказательство. Пусть $\mathcal{T}$ — непротиворечивая теория. Рассмотрим её экзистенциальное расширение $\mathcal{T}^{(1)}$. Оно непротиворечиво по лемме 3.61, но может быть неполно. Теперь обозначим через $\mathcal{T}^{(2)}$ какое-нибудь полное непротиворечивое расширение теории $\mathcal{T}^{(1)}$. Такое расширение существует в силу леммы Линденбаума 3.60. Теория $\mathcal{T}^{(2)}$ не обязательно экзистенциально полна. Поэтому повторим описанные действия счётное число раз и построим расширяющуюся последовательность теорий

$$\mathcal{T} = \mathcal{T}^{(0)} \subset \mathcal{T}^{(1)} \subset \mathcal{T}^{(2)} \subset \cdots \subset \mathcal{T}^{(n)} \subset \ldots, \qquad (3.25)$$

в которой $\mathcal{T}^{(2n+1)}$ является экзистенциальным расширением $\mathcal{T}^{(2n)}$, а $\mathcal{T}^{(2n)}$ является непротиворечивым полным расширением $\mathcal{T}^{(2n-1)}$.

Объединение всех $\mathcal{T}^{(n)}$ обозначим через $\mathcal{T}^{\infty}$.

Любой вывод из формул теории $\mathcal{T}^{\infty}$ использует лишь конечное число формул, так что является выводом в некоторой теории $\mathcal{T}^{(n)}$. Но все теории

в цепочке (3.25) непротиворечивы по построению. Значит, теория $\mathcal{T}^\infty$ непротиворечива.

Аналогично доказывается полнота $\mathcal{T}^\infty$. Каждая формула A в сигнатуре теории $\mathcal{T}^\infty$ является также формулой в сигнатуре некоторой полной теории $\mathcal{T}^{(2n)}$, так что либо A, либо $\neg A$ выводимы в $\mathcal{T}^{(2n)} \subseteq \mathcal{T}^\infty$.

Рассмотрим теперь некоторую экзистенциальную формулу $\exists\, x A$ теории $\mathcal{T}^\infty$. В эту формулу входит лишь конечное число констант, поэтому она является также формулой некоторой непротиворечивой полной теории $\mathcal{T}^{(2n)}$ из цепочки (3.25). Но тогда по построению в теории $\mathcal{T}^{(2n+1)}$ — экзистенциальном расширении теории $\mathcal{T}^{(2n)}$ — найдутся константа c и формула $A(x{:=}c)$. Значит, теория $\mathcal{T}^\infty$ экзистенциально полна. □

Лемма 3.64. *Непротиворечивая теория совместна.*

Доказательство. Рассмотрим непротиворечивую теорию $\mathcal{T}$. Для неё существует полное и экзистенциально полное непротиворечивое расширение $\mathcal{T}'$ (лемма 3.63). Мы будем строить модель $\mathcal{M}$ теории $\mathcal{T}'$, она же будет являться и моделью исходной теории $\mathcal{T}$. Область интерпретации состоит из констант теории $\mathcal{T}'$ и термов, построенных из этих констант. Такие термы будем называть *замкнутыми термами*.

Чтобы отличать замкнутые термы теории $\mathcal{T}'$ от элементов области интерпретации $\mathcal{M}$, будем записывать последние, заключая их в угловые скобки, и называть их *термами-объектами*.

Интерпретация функциональных символов очевидна:

$$[\mathbf{f}](\langle t_1\rangle, \ldots, \langle t_n\rangle) = \langle f(t_1, \ldots, t_n)\rangle, \tag{3.26}$$

т. е. функция $[\mathbf{f}]$ переводит набор термов-объектов $\langle t_1\rangle, \ldots, \langle t_n\rangle$ в терм-объект $\langle f(t_1, \ldots, t_n)\rangle$.

Интерпретация предикатных символов основывается на том, что для любой элементарной формулы $A(t_1, \ldots, t_n)$, где t_i — замкнутые термы, в $\mathcal{T}'$ выводима либо эта формула, либо её отрицание. В первом случае полагаем значение предиката $[\mathbf{A}]$ на наборе термов-объектов $\langle t_1 \rangle$, $\langle t_2 \rangle$, $\ldots$, $\langle t_n \rangle$ истинным, во втором случае — ложным.

Теперь индукцией по построению формулы докажем для любой замкнутой формулы A, что если A выводима в теории $\mathcal{T}'$, то A истинна в $\mathcal{M}$, а если A невыводима в теории $\mathcal{T}'$, то A ложна в $\mathcal{M}$. Для элементарных формул это верно по построению интерпретации.

С пропозициональными связками всё так же, как в исчислении высказываний. Если $\neg A$ или $A \to B$ — замкнутые, то замкнутыми являются и формулы A, B. Дальше нужно повторить рассуждение из второго доказательства теоремы о полноте исчисления высказываний (см. с. 45).

Рассмотрим формулу $\neg A$. Если A невыводима в $\mathcal{T}'$, то в силу полноты теории $\mathcal{T}'$ выводима $\neg A$. По предположению индукции A ложна в $\mathcal{M}$, значит, $\neg A$ — истинна. Если же A выводима в $\mathcal{T}'$, то в силу непротиворечивости теории $\mathcal{T}'$ невыводима $\neg A$. По предположению индукции A истинна, значит, $\neg A$ — ложна.

Аналогично разбирается и случай формулы $A \to B$. Если B выводима, то $A \to B$ тоже выводима. При этом B истинна по предположению индукции, так что $A \to B$ тоже истинна. Если A невыводима, то из полноты теории $\mathcal{T}'$ следует, что выводима $\neg A$. Так как $\neg A \to (A \to B)$ — тавтология, то $A \to B$ — выводима. По предположению индукции A ложна. Значит, $A \to B$ истинна. Наконец, если A выводима, а B — невыводима

(а $\neg B$ выводима), то в силу тавтологии

$$A\to(\neg B\to\neg(A\to B))$$

выводима формула $\neg(A\to B)$. Теория $\mathcal{T}'$ непротиворечива, так что формула $A\to B$ невыводима. При этом по предположению индукции A истинна, B ложна. Но это означает, что $A\to B$ ложна.

Теперь пусть $A = \forall\, xB$ — выводимая в теории $\mathcal{T}'$ замкнутая формула. Для любого замкнутого терма t в $\mathcal{T}'$ выводится $B(x{:=}t)$. К выводу формулы A нужно добавить формулу

$$\forall\, xB\to B(x{:=}t) \tag{3.27}$$

и применить правило modus ponens. Формула (3.27) является частным случаем аксиомы (A4). В самом деле, терм t замкнут, т. е. не содержит переменных. Значит, он свободен для переменной x в формуле B.

Из предположения индукции получаем, что на любом терме-объекте $\langle t\rangle$ предикат $[\mathbf{B}]$ истинен. Отсюда следует истинность A в интерпретации $\mathcal{M}$.

Осталось рассмотреть случай, когда замкнутая формула $A = \forall\, xB$ невыводима в теории $\mathcal{T}'$. В силу полноты $\mathcal{T}'$ это означает, что $\mathcal{T}' \vdash \neg A$ и по лемме 3.58 $\mathcal{T}' \vdash \exists\, x\neg B$. Так как теория $\mathcal{T}'$ экзистенциально полна, существует такая константа c, для которой выводима $\neg B(x{:=}c)$. Тогда $B(x{:=}c)$ невыводима в $\mathcal{T}'$ и по индуктивному предположению ложна в интерпретации $\mathcal{M}$. Значит, формула A ложна в интерпретации $\mathcal{M}$. □

Теорема 3.65 (о полноте исчисления предикатов, Гёдель, 1930). *Всякая общезначимая формула выводима в исчислении предикатов.*

Доказательство. Утверждение теоремы равносильно тому, что невыводимая формула необщезначима.

Пусть A — невыводимая в исчислении предикатов замкнутая формула. Тогда по лемме 3.59 теория $\{\neg A\}$ непротиворечива.

Значит, по лемме 3.64 у теории $\{\neg A\}$ есть модель. В этой модели формула $\neg A$ истинна. Но тогда A не является общезначимой. □

3.8. О проверке общезначимости формул

Определение общезначимости формулы ссылается на все возможные интерпретации, в том числе и бесконечные. Проверять истинность формулы в произвольной бесконечной интерпретации затруднительно уже хотя бы потому, что множество таких интерпретаций для самых простых формул (скажем, содержащих унарный предикатный символ) равномощно континууму.

Однако проверка общезначимости формул не настолько сложна, как может показаться из определения. В этом разделе мы рассмотрим задачу проверки общезначимости формул более подробно.

Прежде всего заметим, что общезначимость A равносильна невыполнимости $\neg A$. Поэтому далее будем говорить о проверке выполнимости формул.

Оказывается, что невыполнимость формулы можно установить, рассматривая лишь конечные «интерпретации». Кавычки здесь использованы потому, что без них это утверждение неверно, что демонстрируется задачей 3.10 на с. 125.

Чтобы ограничиться лишь конечными областями, нужно рассматривать частично определённые интерпретации. Даже если интерпретациями функциональных и предикатных символов являются частично определённые функции, для некоторых (конечно, не для

всех) формул возможно определить их истинностное значение. Далее мы покажем, как по любой формуле A построить такую формулу A', что выполнимость A' равносильна выполнимости A, а невыполнимость A' равносильна тому, что A' ложна на некотором семействе частично определённых интерпретаций на конечных областях.

Если в качестве A взять формулу из задачи 3.10, то истинностное значение соответствующей формулы A' для таких интерпретаций окажется неопределённым. Поэтому противоречия не возникает.

Замечание 3.66. Мы не будем буквально следовать намеченному плану. Вместо частичных интерпретаций мы будем использовать синтаксические и допустимые деревья, как в описании метода резолюций для исчисления высказываний.

Замечание 3.67. Размер областей частичных интерпретаций, необходимых для проверки невыполнимости данной формулы, может оказаться очень большим, причём затруднительно указать этот размер, глядя на запись формулы. Более точно это утверждение мы сформулируем в следующей главе, где будет объяснено, что задача проверки общезначимости формул алгоритмически неразрешима.

3.8.1. Предварённая нормальная форма

Прежде всего укажем для каждой формулы эквивалентную ей формулу специального вида.

Определение 3.68. Формула называется *предварённой нормальной формой*, если она имеет вид

$$Q_1 x_1 Q_2 x_2 \dots Q_k x_k A, \tag{3.28}$$

где $Q_i \in \{\forall, \exists\}$, а формула A не содержит кванторов.

Заметим, что для предварённой нормальной формы выполняется условие разделённости переменных.

Теорема 3.69. *Для любой формулы существует эквивалентная ей предварённая нормальная форма.*

Доказательство теоремы 3.69 использует индукцию по построению формулы. Для индуктивного шага нам потребуется набор пар эквивалентных формул, который обеспечивается следующей леммой.

Лемма 3.70. *Пусть A, B — формулы исчисления предикатов, причём B не содержит переменной x. Тогда следующие пары формул эквивалентны*

$$\neg\exists x A \text{ и } \forall x \neg A, \quad \neg\forall x A \text{ и } \exists x \neg A,$$

$$B \to \forall x A \text{ и } \forall x (B \to A), \quad \forall x A \to B \text{ и } \exists x (A \to B),$$

$$B \to \exists x A \text{ и } \exists x (B \to A), \quad \exists x A \to B \text{ и } \forall x (A \to B).$$

Доказательство. Эквивалентность первых двух пар формул сразу следует из эквивалентности формул A и $\neg\neg A$, так как $\neg\exists x A$ является сокращением для $\neg\neg\forall x \neg A$, а $\exists x \neg A$ — сокращением для $\neg\forall x \neg\neg A$.

Для доказательства остальных эквивалентностей рассмотрим некоторую интерпретацию с областью M. Зафиксируем значения всех параметров формул A и B кроме x. Тогда B приобретает определённое истинностное значение.

Используем тавтологии

$$0 \to x \sim 1, \quad 1 \to x \sim x, \quad x \to 0 \sim \neg x, \quad x \to 1 \sim 1.$$

Эквивалентность левых двух пар формул становится очевидной. Совпадение значений правых пар формул при $B = 1$ также очевидно. Остаётся проверить совпадение значений пары формул $\neg\forall x A$ и $\exists x \neg A$, $\neg\exists x A$ и $\forall x \neg A$. Это уже было сделано выше. □

Пример 3.71. Проиллюстрируем применение леммы 3.70. Построим для формулы

$$\forall\, x A(x) \to \exists\, x A(x)$$

эквивалентную ей предварённую нормальную форму. Вначале выполним замены на новые переменные. Получим эквивалентную формулу

$$\forall\, x A(x) \to \exists\, y A(y).$$

Теперь применим лемму 3.70 к посылке импликации и получим эквивалентную формулу

$$\exists\, x (A(x) \to \exists\, y A(y)).$$

К формуле под квантором $\exists\, x$ применим лемму 3.70 ещё раз. Получаем предварённую нормальную форму

$$\exists\, x\, \exists\, y (A(x) \to A(y)),$$

эквивалентную исходной формуле.

Доказательство теоремы 3.69. В общем случае действуем так же, как в примере 3.71. Вначале строим по формуле A эквивалентную формулу A_1 с разделёнными переменными (лемма 3.51). Затем последовательно выполняем следующее преобразование. В уже построенной формуле A_i ищем кванторную подформулу Qx_iB_i, которая входит в одну из пропозициональных связок, т. е. находим подформулу C_i вида $\neg Qx_iB_i$, или $D_i \to Qx_iB_i$, или $Qx_iB_i \to D_i$. В силу разделённости переменных к формуле C_i применима лемма 3.70. Согласно утверждению 3.42 заменяем C_i на эквивалентную ей формулу, в которой применение квантора и пропозициональной связки меняются местами.

Если формула C_i указанного вида не найдена, то A_i является предварённой нормальной формой (в дереве формулы выше кванторов нет пропозициональных связок).

Мы построили последовательность эквивалентных формул A_i. Осталось проверить, что эта последовательность конечна. В самом деле, применение леммы 3.70 уменьшает количество пар «связка – квантор», в которых квантор является потомком связки в дереве формулы. Поэтому рано или поздно такие пары исчезнут. □

3.8.2. Сколемизация

Теорема 3.69 показывает, что достаточно проверять выполнимость предварённых нормальных форм.

Следующий шаг состоит в том, чтобы упростить кванторную часть предварённой нормальной формы.

В общем случае кванторная часть содержит оба квантора $\forall$ и $\exists$. Мы хотим ограничиться случаем формул, в которых кванторная часть состоит из одинаковых кванторов. Этого невозможно добиться, если искать эквивалентные формулы в той же сигнатуре.

Пример 3.72. Докажем, что для формулы

$$\forall\, x\, \exists\, y A(x, y)$$

нет такой эквивалентной нормальной формы в сигнатуре $\{A\}$, $A \in \mathcal{P}_2$, все кванторы которой одинаковы.

Для интерпретации формулы в сигнатуре $\{A\}$ нужно указать интерпретацию предикатного символа A как бинарного отношения на множестве M. Бинарное отношение можно представить наглядно, как ориентированный граф без кратных рёбер с множеством вершин M. Из вершины x в вершину y ведёт ребро тогда и только тогда, когда $A(x, y)$.

Формула $\forall\, x\, \exists\, y A(x, y)$ выражает такое свойство графа: «из каждой вершины выходит по крайней мере одно ребро». Любая формула с кванторами одного

вида выражает некоторое свойство подграфов графа, количество вершин в которых фиксировано (оно равно количеству переменных в формуле).

Пусть формула Ф говорит о подграфах с числом вершин n. У ориентированного цикла на $(n+2)$-х вершинах и ориентированного пути длины n (граф на $n+1$ вершине) одинаковый набор подграфов с числом вершин n: путь длины $n-1$ и несвязные объединения путей длин k и $n-k-2$.

Формула Ф утверждает либо существование подграфа с каким-то свойством, либо выполнение некоторого свойства для всех подграфов. Поэтому она принимает одинаковое значение на двух указанных графах.

Так как формула $\forall x \exists y A(x,y)$ принимает разные значения на этих графах, она неэквивалентна Ф.

Определение 3.73. Предварённая нормальная форма, которая содержит только кванторы всеобщности, называется Π_1-*формулой*.

Чтобы перейти к Π_1-формулам, будем расширять сигнатуру формулы, добавляя новые функциональные символы. Рассмотрим основной пример.

Пример 3.74. Выполнимость формулы

$$S = \forall x \exists y A(x,y)$$

равносильна выполнимости формулы

$$F = \forall x A(x, f(x)),$$

где f — «новый» унарный функциональный символ, т. е. функциональный символ, который не входит в формулу A.

Действительно, пусть есть интерпретация $\mathcal{M}$ с областью интерпретации M, в которой истинна формула S. Построим интерпретацию $\mathcal{M}'$ с той же областью,

в которой истинна формула F. Предикатный символ A интерпретируем так же, как в $\mathcal{M}$. Интерпретацию символа f зададим следующим образом: для каждого элемента $m_1 \in M$ выберем такой элемент $m_2 \in M$, что $[\mathbf{A}](m_1, m_2)$, и определим $[\mathbf{f}](m_1)$ как m_2. Поскольку S истинна в интерпретации $\mathcal{M}$, такой выбор возможен для любого m_1.

В обратную сторону утверждение очевидно: если в интерпретации $\mathcal{M}$ истинна формула F, то в той же интерпретации истинна и формула S.

Обобщая пример, сформулируем процедуру *сколемизации* формулы[2)]. Пусть A — замкнутая формула в предварённой нормальной форме, причём в кванторной части содержится квантор существования. Рассмотрим самый левый квантор существования $\exists\, y$, обозначим количество кванторов всеобщности перед ним через k. Таким образом формула имеет вид

$$\forall\, x_1 \forall\, x_2 \ldots \forall\, x_k \exists\, y B \quad \text{или} \quad \exists\, y \ldots, \tag{3.29}$$

последний случай отвечает $k = 0$.

Введём новый функциональный символ f арности k. Удалим из формулы квантор $\exists\, x$ и заменим в бескванторной части формулы B все вхождения y на $f(x_1, \ldots, x_k)$ или на f при $k = 0$. Получим новую замкнутую формулу C.

Будем повторять описанную выше процедуру до тех пор, пока не получим (замкнутую) Π_1-формулу F. Она и называется сколемизацией исходной формулы.

Теорема 3.75 (Сколем). *Выполнимость замкнутой формулы равносильна выполнимости её сколемизации.*

[2)] Процедура названа в честь известного логика Т. А. Сколема.

Доказательство. Достаточно доказать утверждение теоремы для одного шага сколемизации, далее можно применить индукцию по числу шагов.

Повторим рассуждения из примера 3.74. Пусть есть интерпретация $\mathcal{M}$ с областью интерпретации M, в которой истинна формула A. Построим интерпретацию $\mathcal{M}'$ с той же областью, в которой истинна формула C, построение которой из формулы A описано выше. Все предикатные и функциональные символы, входящие в A, интерпретируем точно так же, как в $\mathcal{M}$. Интерпретацию символа f зададим следующим образом: для каждого набора $m_1, \ldots, m_k \in M$ выберем такой $m_0 \in M$, что $[\mathbf{B}]$ истинно на наборе свободных переменных, в котором $x_1 = m_1, \ldots, x_k = m_k$. Поскольку A истинна в интерпретации $\mathcal{M}$, такой выбор возможен для любого набора $m_1, \ldots, m_k$. Определим теперь $[\mathbf{f}](m_1, \ldots, m_k)$ как m_0.

В обратную сторону утверждение очевидно: если в интерпретации $\mathcal{M}$ истинна формула C, то в той же интерпретации истинна и формула A. □

Иногда сколемизация формулы вообще не содержит переменных.

Пример 3.76. Пусть $B = \exists x(A(x) \to A(x))$. Сколемизацией формулы B является формула $(A(c) \to A(c))$, где c — константа.

В общем случае сколемизация замкнутой формулы содержит столько переменных, сколько было кванторов всеобщности в исходной формуле. Это ясно из определения процедуры: каждая переменная, связанная квантором существования в исходной формуле, заменяется на новый функциональный символ.

3.8.3. Теорема Эрбрана

Теорема 3.75 утверждает, что для проверки выполнимости формул достаточно проверять выполнимость Π_1-формул. Оказывается, выполнимость Π_1-формул достаточно проверять в интерпретациях специального вида, похожего на те интерпретации, которые были использованы в доказательстве леммы 3.64. Мы будем называть такие интерпретации *нормальными*.

По определению, область нормальной интерпретации формул сигнатуры σ, содержащей константы, т. е. 0-арные функциональные символы, состоит из замкнутых термов сигнатуры σ. (Напомним, что замкнутые термы строятся аналогично произвольным термам, но только из констант.) Если сигнатура σ не содержит констант, то областью нормальной интерпретации будем полагать область интерпретации сигнатуры $\sigma \cup \{c\}$, где c — символ константы.

Кроме того, нормальная интерпретация функциональных символов задаётся формулой (3.26): применение функции $[\mathbf{f}]$ переводит термы-объекты $\langle t_1 \rangle$, $\langle t_2 \rangle$, $\ldots$, $\langle t_k \rangle$ в терм $\langle f(t_1, \ldots, t_k) \rangle$.

Обратите внимание, что в этом определении не накладывается никаких ограничений на интерпретацию предикатных символов.

Лемма 3.77. *Π_1-формула выполнима тогда и только тогда, когда она выполнима в нормальной интерпретации.*

Доказательство. В одну сторону утверждение леммы очевидно из определения. Докажем в другую сторону.

Без ограничения общности считаем, что сигнатура содержит константы (если нет, то добавим новую константу как это сделано выше в определении области

интерпретации). Пусть P — выполнимая в некоторой интерпретации $\mathcal{M}$ Π_1-формула. Значения замкнутых элементарных формул в интерпретации $\mathcal{M}$ определяют некоторую нормальную интерпретацию $\mathcal{N}$.

Докажем, что P истинна в этой интерпретации. Обозначим через F бескванторную часть P и через $x_1, \ldots, x_n$ — входящие в неё переменные. Тогда для любых замкнутых термов $t_1, \ldots, t_n$ выполняется выводимость в ИП

$$P \vdash F((x_1{:=}t_1), (x_2{:=}t_2), \ldots, (x_n{:=}t_n)). \qquad (3.30)$$

Действительно, замкнутый терм свободен для любой переменной, поскольку он не содержит никаких переменных вообще. Поэтому можно построить искомый вывод, чередуя аксиому (A4) и modus ponens.

Отсюда следует, что для любого набора замкнутых термов $t_1, \ldots, t_n$ бескванторная часть F формулы P истинна в интерпретации $\mathcal{N}$ (из истинной в некоторой интерпретации формулы можно вывести только истинные в той же интерпретации формулы). □

Пример 3.78. Рассмотрим конечную сигнатуру, в которую входят только 0-арные функциональные символы, т. е. константы. Тогда область нормальной интерпретации также конечна, поскольку все элементарные замкнутые формулы имеют вид $A(c_1, \ldots, c_n)$, где A — предикатный символ, а c_i — константы. Поэтому из леммы 3.77 следует, что выполнимость формулы в такой сигнатуре равносильна её выполнимости в конечных интерпретациях.

Выше мы определяли синтаксические и допустимые деревья для некоторых формул исчисления высказываний, а именно для КНФ. Сейчас мы обобщим это понятие для Π_1-формул. Синтаксическое дерево

является полным бинарным деревом, возможно бесконечной глубины. Уровни синтаксического дерева индексируются замкнутыми элементарными формулами в сигнатуре σ.

Более точно, пусть $A_1, A_2, \ldots$ — некоторая нумерация замкнутых элементарных формул в сигнатуре σ. Вершиной синтаксического дерева является конечная последовательность булевых значений $\alpha = (\alpha_1, \ldots, \alpha_h)$, потомками этой вершины являются последовательности $(\alpha_1, \ldots, \alpha_h, 0)$ и $(\alpha_1, \ldots, \alpha_h, 1)$. Корнем дерева является пустая последовательность, т. е. последовательность длины 0.

Максимальным путём в корневом бинарном дереве будем называть такой путь из корня, который нельзя продолжить. В конечном дереве максимальным путям отвечают пути из корня в листья, а в бесконечном — бесконечные ветви, т. е. последовательности вершин вида

$$\varepsilon, (\alpha_1), (\alpha_1, \alpha_2), \ldots, (\alpha_1, \alpha_2, \ldots, \alpha_n), \ldots$$

Таким образом, бесконечная ветвь задаётся произвольной последовательностью $(\alpha_1, \alpha_2, \ldots)$ булевых значений.

Максимальному пути в синтаксическом дереве соответствует некоторая нормальная интерпретация, а именно, ветви $(\alpha_1, \alpha_2, \ldots)$ соответствует такая нормальная интерпретация, для которой $[\mathbf{A_i}] = \alpha_i$ для i-й в выбранной нумерации элементарной формулы A_i. Это соответствие взаимно однозначно, поскольку интерпретация задаёт значения всех элементарных замкнутых формул.

Теперь рассмотрим Π_1-формулу P с бескванторной частью F. Будем называть вершину $(\alpha_1, \ldots, \alpha_h)$ синтаксического дерева *недопустимой* для формулы P,

если существует такая подстановка замкнутых термов в формулу F, что все элементарные замкнутые формулы, которые входят в эту подстановку, содержатся в множестве $\{A_1, A_2, \ldots, A_h\}$, а формула F ложна при выборе значений формул $[\mathbf{A_i}] = \alpha_i$.

Пример 3.79. Пусть $F = A(x) \to A(f(x))$, а нумерация замкнутых элементарных формул начинается с формул

$$A_1 = A(c),\ A_2 = A(f(c)),\ A_3 = A(f(f(c))).$$

Тогда вершина $(1, 0)$ недопустима: подставляя вместо переменной x константу c (замкнутый терм), мы получим, что F ложна, так как посылка $A(c)$ истинна, а заключение $A(f(c))$ ложно. Аналогично проверяется, что вершина $(1, 1)$ является допустимой в этом примере, а вершина $(1, 1, 0)$ — недопустимой (подставим вместо x замкнутый терм $f(c)$).

Допустимым деревом формулы P назовём поддерево синтаксического дерева, состоящее из допустимых вершин. Заметим, что корень всегда допустим, а также все вершины на пути из допустимой вершины в корень являются допустимыми: убирая часть формул, мы сокращаем возможности подстановок.

Лемма 3.80. Π_1*-формула выполнима тогда и только тогда, когда её допустимое дерево содержит максимальный путь синтаксического дерева.*

Доказательство. Пусть формула P выполнима. Тогда по лемме 3.77 она выполнима в некоторой нормальной интерпретации. Нормальной интерпретации соответствует максимальный путь синтаксического дерева. Этот путь входит в допустимое дерево, так как если на этом пути была бы недопустимая вершина,

то бескванторная часть F формулы P оказалась бы ложной в данной нормальной интерпретации.

В обратную сторону: пусть допустимое дерево формулы P содержит максимальный путь α. Докажем, что формула P истинна в нормальной интерпретации, соответствующей этому пути. Для любого набора замкнутых термов $t_1, \ldots, t_n$ найдётся такое k, что все элементарные формулы, получающиеся после подстановки термов t_i в F, содержатся среди формул $A_1, \ldots, A_k$. На пути α есть вершина $(\alpha_1, \ldots, \alpha_k)$ на глубине k. Эта вершина допустима, откуда следует, что F истинна в интерпретации, задаваемой путём α, после подстановки замкнутых термов $t_1, \ldots, t_n$. □

Лемма 3.81 (лемма о дереве). *Пусть степени всех вершин корневого дерева конечны, и дерево не содержит бесконечной ветви. Тогда дерево конечно.*

Другими словами, лемма о дереве (иначе называемая леммой Кёнига) утверждает, что для корневых деревьев с конечным ветвлением условия «дерево бесконечно» и «дерево содержит бесконечную ветвь» равносильны.

Доказательство. Построим в бесконечном дереве бесконечную ветвь. Построение происходит индуктивно. Предположим, что уже построена начальная часть пути от корня до такой вершины v, под которой находится бесконечно много вершин. Лишь конечное число вершин v_1, …, v_n являются непосредственными потомками вершины v. Выберем ту из них, скажем v_i, под которой находится бесконечно много вершин, и продолжим путь, добавив к нему вершину v_i.

Под корнем дерева находится бесконечно много вершин, так что описанное выше рассуждение к нему также применимо. □

Теорема 3.82 (Эрбран). *Если замкнутая* Π_1*-формула* P *с переменными* $x_1, \ldots, x_n$ *невыполнима, то существует такое конечное множество наборов замкнутых термов* $t_1^{(i)}, \ldots, t_n^{(i)}$, $1 \leqslant i \leqslant k$, *что формула*

$$\bigwedge_{i=1}^{k} F((x_1{:=}t_1^{(i)}), (x_2{:=}t_2^{(i)}), \ldots, (x_n{:=}t_n^{(i)})) \qquad (3.31)$$

невыполнима. Здесь F *— бескванторная часть формулы* P*.*

Доказательство. Из предыдущих лемм вытекает, что допустимое дерево для невыполнимой формулы конечно. Значит, на некотором уровне h все вершины синтаксического дерева недопустимы. По определению это означает, что для каждой вершины уровня h найдётся такая подстановка замкнутых термов, на которой формула F ложна (и, в частности, содержащие эти термы элементарные замкнутые подформулы F находятся среди первых h элементарных замкнутых формул в выбранной нумерации). Взяв все наборы термов, участвующие в этих подстановках, получим искомый набор термов. Действительно, любое присваивание булевых значений для элементарных замкнутых формул, которые входят в (3.31) является частью набора значений (α_i) для одной из вершин уровня h. По построению, на таком наборе значений один из членов конъюнкции (3.31) ложен. □

Замечание 3.83. Теорему Эрбрана можно понимать так, как написано в начале этого раздела — для невыполнимой Π_1-формулы существует такая конечная область и заданные на ней частично определённые интерпретации функциональных символов, что при любой интерпретации предикатных символов на этой области формула ложна.

3.8.4. Метод резолюций для исчисления предикатов

Если заменить бескванторную часть Π_1-формулы на эквивалентную ей КНФ, получим эквивалентную формулу. На полученную формулу C можно смотреть как на «КНФ», поскольку квантор всеобщности задаёт фактически конъюнкцию условий для каждого набора значений переменных.

Тогда теорему Эрбрана можно пересказать так: невыполнимость этой «КНФ» равносильна тому, что из некоторого конечного множества дизъюнктов выводится пустой дизъюнкт.

Опишем более точно эту идею. Напомним, что мы фиксировали нумерацию элементарных замкнутых формул. Выберем число h и построим КНФ $C^{(h)}$ как конъюнкциею дизъюнктов, которые получаются из дизъюнктов бескванторной части Π_1-формулы C такой подстановкой замкнутых термов вместо переменных, что все возникающие элементарные замкнутые подформулы содержатся среди первых h элементарных замкнутых формул в выбранной нумерации.

Из теоремы Эрбрана следует, что если формула C невыполнима, то невыполнима и некоторая КНФ $C^{(h)}$.

Как мы помним, проверку невыполнимости КНФ можно осуществить методом резолюций. Обобщение метода резолюций на случай Π_1-формул с бескванторной частью в виде КНФ состоит в том, чтобы строить последовательно КНФ $C^{(h)}$ для $h = 1, 2, \ldots$ и резолютивные выводы из них. Получив из $C^{(h)}$ непротиворечивый и полный набор дизъюнктов, мы теперь не можем утверждать выполнимость формулы, а должны перейти к следующему значению h.

Для невыполнимой формулы этот процесс завершится выводом нулевого дизъюнкта из некоторого набора $C^{(h)}$. В этот момент мы можем утверждать невыполнимость исходной формулы. Если же исходная формула выполнима, а сигнатура такова, что имеется бесконечное множество замкнутых элементарных формул, то описанный процесс никогда не завершается. Поэтому ни в какой момент мы не можем утверждать что-либо о выполнимости формулы C. Таким образом, возможны ситуации, когда метод резолюций при проверке общезначимости формулы не даёт ответа.

Пример 3.84. Рассмотрим формулу

$$\neg(\exists x \forall y A(x,y) \to \forall y \exists x A(x,y)).$$

Она невыполнима как отрицание общезначимой формулы.

Построим приведённую нормальную форму для этой формулы:

$$\begin{gathered}\big(\neg(\exists x \forall y A(x,y) \to \forall y \exists x A(x,y))\big) \sim \\ \big(\neg(\exists x \forall y A(x,y) \to \forall v \exists u A(u,v))\big) \sim \\ \big(\neg(\forall x \exists y \forall v \exists u (A(x,y) \to A(u,v)))\big) \sim \\ \exists x \forall y \exists v \forall u \neg(A(x,y) \to A(u,v))\end{gathered}$$

Сколемизация этой формулы имеет вид

$$\forall y \forall u \neg(A(c,y) \to A(u, f(y))),$$

где c — константа, а f — функциональный символ арности 1. Выражая бескванторную часть в виде КНФ, получаем

$$\forall y \forall u (A(c,y) \wedge \neg A(u, f(y))).$$

Считаем, что нумерация замкнутых элементарных

формул в этой сигнатуре начинается с

$$A(c,c),\ A(c,f(c)),\ A(c,f(f(c))).$$

Тогда $C^{(1)}$ содержит пустое множество дизъюнктов, $C^{(2)} = A(c,c) \wedge \neg A(c,f(c))$. Из дизъюнктов $C^{(2)}$ ничего вывести нельзя. Поэтому переходим к следующему шагу

$$C^{(3)} = C^{(2)} \wedge A(c,f(c)) \wedge \neg A(c,f(f(c))) =$$
$$= A(c,c) \wedge \neg A(c,f(c)) \wedge A(c,f(c)) \wedge \neg A(c,f(f(c))).$$

Из средних двух дизъюнктов выводится нулевой дизъюнкт. Значит, формула невыполнима.

Более подробное изложение различных вариантов метода резолюций можно найти в книге [19].

Задачи

3.22. Напишите предварённые нормальные формы, которые эквивалентны формулам

а) $\exists\, xA(x) \to \neg(\forall\, xA(x) \to \forall\, xB(x))$;

б) $\exists\, xA(x) \to \forall\, xA(x)$;

в) $\forall\, xA(x) \to \neg(\forall\, xA(x) \to \exists\, xB(x))$;

г) $\neg((\forall\, xB(x) \to \exists\, xA(x)) \to \forall\, xA(x))$;

д) $\neg(\forall\, xB(x) \to \forall\, xA(x)) \to \exists\, xA(x)$.

3.23. Проверьте методом резолюций общезначимость формулы $\forall\, xA(x) \to A(y)$.

Глава 4

Алгоритмы

Алгоритм — это явно описанный рецепт действий.

В этой главе мы будем изучать алгоритмы решения *массовых задач*. Неформально можно считать, что такой алгоритм — это компьютерная программа, которая получает на вход файл с описанием исходных данных и записывает результат в другой файл.

Такого неформального понимания алгоритма достаточно, если нужно построить алгоритм решения конкретной задачи. Однако если сравнивать алгоритмы между собой или доказывать алгоритмическую неразрешимость задачи, то возникает необходимость в формальном определении алгоритма.

В этой главе мы приведём начальные сведения из теории алгоритмов, имея в виду прежде всего вопрос об алгоритмической неразрешимости.

Есть много способов формального определения понятия «алгоритм». Большинство из них используют один из двух основных подходов: процедурный или функциональный. В первом разделе мы приведём одно из процедурных определений, так называемые машины Тьюринга.

В общем виде работа алгоритма (или вычисление) может быть описана так: алгоритм получает

входные данные, выполняет некоторую последовательность действий с ними и останавливается, выдавая *результат работы*, если выполняется *условие остановки*. В процессе вычисления алгоритм находится в одном из возможных состояний (конфигураций).

Алгоритм полностью определяется заданием входных данных; *программой работы*, т. е. частично определённым отображением множества конфигураций алгоритма в себя, которое задаёт одно элементарное действие алгоритма; правилом остановки и способом получения результата работы алгоритма в конце его работы.

В рассуждениях об алгоритмах мы встречаемся с общим затруднением: конкретные алгоритмы удобно описывать неформально, приводя рецепты действий на стандартном полуформализованном математическом языке; однако, для строгих доказательств эти неформальные рецепты нужно переписать формально, используя ту или иную модель вычислений. Такой перевод как правило несложен, но приводит к длинным доказательствам, которые не более пригодны для понимания, чем тексты длинных программ на каком-нибудь языке программирования. Поэтому в бóльшей части наших рассуждений мы игнорируем эту трудность, ограничиваясь только неформальными описаниями. Читателю предоставляется возможность самостоятельно заполнить возникающие в доказательствах пробелы.

4.1. Машины Тьюринга

Машина Тьюринга (сокращённо МТ) — одна из самых часто используемых моделей вычислений (алгоритмов).

Мы начнём с неформального описания этой модели. МТ состоит из бесконечной ленты, считывающей головки и программы работы (таблицы переходов). Работа МТ происходит по тактам (т. е. в дискретном времени). Лента МТ разбита на ячейки, каждая из которых имеет левого и правого соседа. В ячейке может быть записан один символ из алфавита МТ (некоторого фиксированного для данной машины конечного множества).

Считывающая головка может двигаться вдоль ленты на одну ячейку за такт работы. Головка может прочитать символ в текущей ячейке и записать туда новый. Сама головка может находиться в нескольких состояниях из фиксированного для данной машины множества состояний. (Состояние головки мы будем также называть состоянием МТ.)

Программа работы МТ определяет действия МТ следующим образом. Для каждой пары (состояние головки, читаемый символ в ячейке под головкой) программа определяет, какой символ головка напишет в ячейку, какое действие совершит головка (сдвиг влево, вправо или остаться над той же ячейкой), в какое состояние перейдёт головка к следующему такту. Программа определяет также, продолжает ли работу МТ на следующем такте.

В начале работы МТ во всех ячейках за исключением конечного числа записан особый символ алфавита Λ (пустой символ). Считывающая головка размещается над самым левым непустым символом, если на ленте есть непустые символы (в противном случае, как легко понять, положение головки не имеет никакого значения). Состояние головки в начале работы фиксировано для данной МТ (*начальное состояние*).

Далее на каждом такте работы МТ к текущей конфигурации (состояние ленты, положение головки на ленте и состояние головки) применяется программа и получается следующая конфигурация. Часть состояний головки объявляются *финальными*. Когда МТ достигает финального состояния, она прекращает работу.

Ясно, что в любой момент работы МТ на ленте записано лишь конечное число непустых символов. Поэтому, когда МТ останавливается, на ленте также записано конечное число непустых символов. Мы будем считать результатом работы МТ ту часть ленты в момент остановки МТ, которая получается отбрасыванием бесконечного «префикса» из пустых символов слева и бесконечного «суффикса» из пустых символов справа (точное определение см. ниже).

Хотя данное выше описание вполне достаточно для того, чтобы понять устройство и принципы работы МТ и даже писать программы для МТ, мы приведём и более формальное определение.[1)]

Определение 4.1. *Машина Тьюринга* — это набор

$$(A, Q, Q_f, \delta, q_0, \Lambda),$$

где A, Q — конечные множества, $A \cap Q = \varnothing$, $Q_f \subseteq Q$, $q_0 \in Q$, $\Lambda \in A$, $\delta\colon A \times Q \to A \times \{-1, 0, 1\} \times Q$ (функция, которая называется *таблицей переходов*).

В этом формальном определении A — алфавит МТ, Λ — пустой символ, Q — множество состояний головки, q_0 — начальное состояние, Q_f — множество

[1)] Следует иметь в виду, что в литературе встречается много определений машин Тьюринга, различающихся в несущественных деталях. Эти детали, тем не менее, могут играть важную роль в строгих доказательствах.

финальных состояний, δ — программа (или таблица переходов), неформальное описание которой приведено выше.

Замечание 4.2. В определении 4.1 мы для простоты считаем, что таблица переходов полностью заполнена (т. е. функция δ всюду определена). При описании конкретных МТ это неудобно. Ниже мы будем использовать и частично определённые таблицы переходов. При этом мы предполагаем, что МТ $(A, Q, Q_f, \delta, q_0, \Lambda)$ с частично определённой таблицей переходов соответствует МТ $(A, Q \cup \{q_t\}, Q_f \cup \{q_t\}, \tilde{\delta}, q_0, \Lambda)$ из определения 4.1, у которой ко множеству состояний добавлено ещё одно финальное состояние q_t, отличающееся от всех прочих состояний МТ, а функция $\tilde{\delta}$ доопределяет функцию δ по правилу:

$$\tilde{\delta}(a,q) = \begin{cases} \delta(a,q), & \text{если } (a,q) \text{ принадлежит области} \\ & \text{определения } \delta, \\ (a,0,q_t) & \text{иначе.} \end{cases}$$

Это соответствует простому неформальному правилу: если у машины нет инструкций для выполнения очередного шага работы, то она останавливается.

Замечание 4.3. В определении 4.1 речь идёт о произвольных конечных множествах. Это удобно при описании конкретных МТ (мы можем использовать все доступные нам символы, чтобы работа МТ была представлена максимально наглядно). Однако во многих отношениях оно неудобн. Скажем, невозможно перенумеровать все «машины», удовлетворяющие такому такому определению. При этом ясно, что возникающая проблема носит технический характер, так как без ограничения общности можно считать, что все символы и все состояния головки принадлежат

некоторому заранее заданному счётному множеству. Мы будем полагать, что это универсальное множество символов есть просто-напросто множество целых неотрицательных чисел.

Учитывая предыдущее замечание, сформулируем следующую теорему.

Теорема 4.4. *Множество всех машин Тьюринга счётно.*

Один из возможных способов доказать это утверждение — привести некоторый способ *описания* машин Тьюринга как слов в конечном алфавите. Отсюда следует счётность множества машин Тьюринга, так как множество слов в конечном алфавите счётно и любое его подмножество конечно или счётно.

Приведём один из возможных способов описания МТ, в котором используется алфавит $\{0, 1\}$.

Пусть имеется машина Тьюринга с состояниями q_1, $\ldots$, q_n и алфавитом $a_1, \ldots, a_m$. Символам движения сопоставим слова в алфавите $\{0, 1\}$:

на месте: 010, вправо: 0110, влево: 01110.

Состоянию q_i сопоставим слово $1\underbrace{00\ldots00}_{2i+2 \text{ нуля}}1$, символу алфавита a_i сопоставим слово $1\underbrace{00\ldots00}_{2i+3 \text{ нуля}}1$.

Таблицу переходов МТ закодируем списком всех пятёрок вида

$$(q_i, a_j, a_{ij}, d_{ij}, q_{ij}), \text{ где } \delta(a_i, q_j) = (a_{ij}, d_{ij}, q_{ij}).$$

Кодом (описанием) всей машины Тьюринга полагаем выписанные подряд коды символов алфавита, состояний головки, финальных состояний, начального состояния, пустого символа и таблицы переходов. (Для

определённости считаем, что перечисления символов и состояний идут в порядке возрастания номеров.) Разным МТ будут присвоены при этом разные коды.

Задача 4.1. Докажите последнее утверждение.

Важно заметить, что никакого выделенного способа описания МТ не существует и есть много эквивалентных способов описания.

Здесь под эквивалентностью мы понимаем существование двух алгоритмов, один из которых получает на вход описание МТ в первом формате и выдаёт как результат описание МТ во втором формате, а второй выполняет преобразование форматов в обратную сторону.

Далее мы будем использовать обозначение $\langle M \rangle$ для описания МТ M, предполагая некоторый способ описания, эквивалентный описанному выше. Мы будем также обозначать угловыми скобками совместные описания МТ и их входов и т. п.

Определение 4.5. *Конфигурацией МТ*

$$(A, Q, Q_f, \delta, q_0, \Lambda)$$

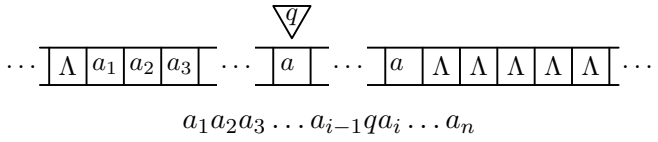
называется слово w в алфавите $A \cup Q$, которое содержит ровно один символ из множества Q, а первый и последний символы w не являются пустыми.

Начальная конфигурация МТ имеет вид $q_0 u$, где $u \in (A \setminus \{\Lambda\})^*$ — слово без пустых символов, которое называется *входом* МТ.

Замечание 4.6. Условие $A \cap Q = \varnothing$ из определения машины Тьюринга гарантирует, что конфигурация МТ однозначно задаёт положение головки на ленте: в любой конфигурации есть только один символ из множества состояний Q.

Произвольная конфигурация МТ имеет вид uqv, где $u, v \in A^*$, $q \in Q$, причём u не начинается, а v не заканчивается пустым символом.

На рис. 4.1 дана иллюстрация этого определения. Конфигурация МТ кодирует в одном слове и состояние ленты, и положение головки, и её состояние. Символ состояния головки записывается слева от того символа, над которым находится головка.



$$a_1 a_2 a_3 \ldots a_{i-1} q a_i \ldots a_n$$

Рис. 4.1. Конфигурация машины Тьюринга

Определение 4.7. *Расширенной конфигурацией МТ* назовём двустороннюю бесконечную последовательность $\ldots \Lambda w \Lambda \ldots$, где w — конфигурация МТ, а на месте многоточий стоят пустые символы Λ.

Как видно из определений, между конфигурациями и расширенными конфигурациями МТ имеется взаимно однозначное соответствие (благодаря тому, что крайние символы в конфигурации отличны от пустого): чтобы по конфигурации построить расширенную конфигурацию, нужно приписать в ней слева и справа бесконечные последовательности пустых символов. Обратное преобразование состоит в том, что из расширенной конфигурации удаляются максимальные слева и справа бесконечные последовательности пустых символов. Конфигурации удобны тем, что это конечные слова, а расширенные конфигурации удобны тем, что на них проще задавать работу МТ.

Определение 4.8. *Отображение d_M на расширенных конфигурациях МТ $M = (A, Q, Q_f, \delta, q_0, \Lambda)$* определяется следующим образом:

$$d_M(uqv) = \begin{cases} ua'q'v', & \text{если} \quad v = av', \\ & \delta(a,q) = (a', +1, q'); \\ uq'a'v', & \text{если} \quad v = av', \\ & \delta(a,q) = (a', 0, q'); \\ u'q'ba'v', & \text{если} \quad u = u'b,\ v = av', \\ & \delta(a,q) = (a', -1, q'). \end{cases} \tag{4.1}$$

Отображение d_M на конфигурациях МТ определяется с помощью указанного выше взаимно однозначного соответствия между конфигурациями и расширенными конфигурациями: по конфигурации строится указанным выше способом расширенная конфигурация, к ней применяется отображение (4.1), после чего полученная расширенная конфигурация преобразуется в конфигурацию, в которой оказывается МТ после такта работы.

Отображение на конфигурациях формализует данное выше описание работы МТ за один такт. Работа МТ M полностью задаётся начальной конфигурацией и отображением на конфигурациях и может быть описана конечной или бесконечной последовательностью конфигураций

$$w_0,\ w_1,\ \ldots, w_n,\ \ldots, \tag{4.2}$$

где w_0 — начальная конфигурация, а для любого i выполнено $w_{i+1} = d_M(w_i)$. Если последовательность конфигураций конечна, для её последнего элемента w_n должно выполняться условие: входящий в w_n символ состояния головки q_f обязан принадлежать множеству финальных состояний Q_f.

Определение 4.9. Последовательность (4.2) называется *протоколом работы* машины Тьюринга.

Определение 4.10. *Результатом работы MT* на входе u является слово, которое получается из последней конфигурации, возникающей при работе МТ с начальной конфигурацией q_0u, вычёркиванием символа состояния головки.

Это общее определение можно конкретизировать для различных целей. Например, можно определить с его помощью понятие *вычислимой функции*. Для этого нужно задать способ записи чисел словами в некотором алфавите. В приводимом ниже представлении выбрана простейшая *унарная система счисления*: число n записывается словом из единиц длины $n+1$ (такое правило позволяет записать число 0 непустым словом длины 1).

Определение 4.11. (Частично определённая) функция $f\colon \mathbb{N}^n \to \mathbb{N}$ называется *вычислимой на машине Тьюринга*, если существует такая МТ M, что для любого набора $(m_1, \ldots, m_n)$, принадлежащего области определения f, работа M на входе

$$\underbrace{11\ldots1}_{m_1+1}\#\underbrace{11\ldots1}_{m_2+1}\#\ldots\#\underbrace{11\ldots1}_{m_n+1}$$

завершается с результатом

$$\underbrace{11\ldots1}_{f(m_1,\ldots,m_n)+1} \quad .$$

(Алфавит МТ может содержать и другие символы, кроме $\{1, \#, \Lambda\}$.)

Несмотря на весьма ограниченный набор действий, МТ является универсальной моделью вычислений.

Тезис Тьюринга. Любой алгоритм может быть реализован с помощью машины Тьюринга.

Важно понимать, что тезис Тьюринга не является математическим утверждением. К нему нужно относиться как к эмпирически установленному факту: любая из известных на данный момент моделей вычислений может моделироваться на машине Тьюринга.

Если принять тезис Тьюринга, то понятия *вычислимой функции* и функции, вычислимой на МТ, совпадают.

Приведём несколько простейших примеров вычисления функций на МТ.

Пример 4.12 (*добавление* 1). Функция $f(x) = x+1$ вычислима. Используя введённое выше соглашение о частично определённых таблицах переходов, МТ, которая вычисляет эту функцию, можно задать двумя строками:

$$\delta\colon (1, q_0) \mapsto (1, -1, q_0)$$
$$\delta\colon (\Lambda, q_0) \mapsto (1, 0, q_1).$$

Очевидно, что работа такой машины заканчивается не более, чем за 2 такта (см. рис. 4.2).

Пример 4.13 (*сложение*). Функция $f(x_1, x_2) = x_1 + x_2$, разумеется, также вычислима на МТ. Вот пример МТ, которая вычисляет эту функцию:

$$\delta\colon (1, q_0) \mapsto (1, 1, q_0), \qquad \delta\colon (\#, q_0) \mapsto (1, 1, q_1),$$
$$\delta\colon (1, q_1) \mapsto (1, 1, q_1), \qquad \delta\colon (\Lambda, q_1) \mapsto (\Lambda, -1, q_2),$$
$$\delta\colon (1, q_2) \mapsto (\Lambda, -1, q_3).$$

Предоставляем читателю самостоятельно нарисовать состояния этой машины в начале и в конце работы.

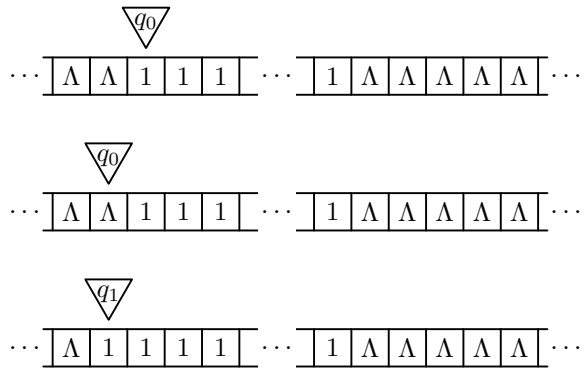


Рис. 4.2.

4.1.1. Описание машин Тьюринга

При реализации алгоритмов на машинах Тьюринга постоянно приходится использовать одни и те же стандартные действия. В этом подразделе мы опишем некоторые стандартные приёмы построения машин Тьюринга. В дальнейшем мы будем подразумевать эти приёмы и не выписывать явно результаты их применения.

Прежде всего обратим внимание на то, что обычно алгоритм разбивается на несколько шагов. Если каждый из таких шагов реализуется некоторой машиной Тьюринга, то весь алгоритм реализуется *последовательным соединением машин Тьюринга*. Приведём неформальное описание этой конструкции.

Переобозначим состояния соединяемых машин так, чтобы они стали попарно различны. После этого множеством состояний машины, которая является последовательным соединением, назовём объединение множеств состояний соединяемых машин. Таблица переходов соединения составлена из таблиц переходов

соединяемых машин за исключением того, что из финального состояния $(i-1)$-й машины осуществляется переход в начальное состояние i-й машины. Финальные состояния последней машины являются финальными состояниями всего соединения машин.

Один или несколько шагов алгоритма могут повторяться циклически до тех пор, пока выполняется некоторое условие. Такие алгоритмы мы представляем аналогично описанному выше соединению машин. Но теперь из части финальных состояний $(i-1)$-й машины осуществляется переход в начальное состояние $(i-1)$-й машины (условие повторения выполнено), а из остальных — в начальное состояние i-й машины. Аналогично можно описать более сложные циклы, в которых повторяется выполнение некоторого последовательного блока соединяемых машин.

Другой вариант последовательного соединения состоит в том, что вместо второй машины используется набор машин, каждая из которых отвечает одному из финальных состояний первой машины. В этом случае из финального состояния первой машины делается переход в начальное состояние той машины, которая отвечает этому финальному состоянию. Такое соединение будем называть *веерным*.

Теперь обратим внимание на двойственную роль множества состояний машины Тьюринга. Во-первых, состояние фиксирует инструкцию по дальнейшим действиям. Но состояния можно также использовать для хранения информации, привязанной к управляющей головке МТ. Такую информацию можно использовать в любой момент времени для выбора одного из возможных действий. Поскольку множество состояний МТ конечно, такая «оперативная память» имеет лишь конечную ёмкость. С формальной точки зрения

использование «оперативной памяти» МТ означает, что множество состояний МТ Q представлено в виде декартова произведения $C \times M$ двух конечных множеств. В состоянии $q = (c, m)$ вторая компонента m отвечает за содержимое «оперативной памяти».

Аналогично множеству состояний алфавит машины Тьюринга также можно расширять. Таким способом можно описывать расстановку «пометок» на рабочей ленте. «Пометки» хранят информацию в ячейках ленты.

Приведём пример использования этих приёмов в рассуждениях о машинах Тьюринга.

Пример 4.14 (*«чистое» соединение машин Тьюринга*). В дальнейшем мы часто будем использовать последовательное соединение машин Тьюринга для построения машины, вычисляющей композицию вычисляемых МТ функций[2)]. Обратим внимание на одну возникающую здесь трудность. В определении результата работы МТ ничего не говорится о положении головки. С другой стороны, в начальной конфигурации машины мы предполагаем, что головка находится над первым непустым символом. Поэтому описанная выше конструкция соединения машин напрямую непригодна для вычисления композиции функций.

Эта трудность преодолевается построением по МТ M такой МТ M', которая на любом входе даёт тот же самый результат, что и M, но в конце работы помещает головку над первым непустым символом.

Машина M' имеет алфавит $A \times \{0, 1\}$, который состоит из пар (символ алфавита A машины M, пометка). Неформально пометка 1 означает, что был

[2)] Под вычисляемой МТ функцией мы понимаем сейчас частично отображение входов на результаты работы.

выполнен такт работы, при котором прочитана ячейка, в которой находится символ. Пары вида $(a, 0)$ отождествляются с исходным алфавитом машины. В частности, пустым символом машины является символ $(\Lambda, 0)$, а запись входа машины M целиком состоит из символов машины M' с нулевой пометкой, так что описанное выше свойство пометок перед началом работы выполняется.

Опишем машину M' как последовательное соединение машин M_1, M_2 и M_3. Машина M_1 моделирует работу M, меняя первые компоненты символов алфавита точно так же, как это делает машина M. Во вторых компонентах машина M_1 поддерживает указанное выше свойство. Другими словами, её таблица переходов имеет вид

$$\delta_1((a,0),q) = ((b,1),r,q'),$$
$$\delta_1((a,1),q) = ((b,1),r,q'), \quad \text{где } \delta(a,q) = (b,r,q').$$

Финальные состояния у M_1 те же, что и у M.

Машина M_2 ищет самый левый символ, у которого пометка равна 1. Поскольку M_2 начинает работу после завершения работы M_1, можно без ограничения общности считать, что этот символ расположен не правее начального положения головки. В силу сформулированного выше свойства пометок первым слева символом с пометкой 0 будет пустой символ. Поэтому таблицу переходов машины M_2 можно задать следующим образом:

$$\delta_2((a,1),q_0) = ((a,1),-1,q_0),$$
$$\delta_2((\Lambda,0),q_0) = ((\Lambda,0),1,q_1).$$

Наконец, машина M_3 заменяет все пометки на нулевые (именно такие пары мы отождествили с символами алфавита A машины M) и останавливается

над первым непустым символом. (Если вся лента пуста, то машина может остановиться в любом месте.) Машина M_3 движется направо, меняя символы $(\Lambda, 1)$ на $(\Lambda, 0)$, пока не достигнет первого непустого символа в первой компоненте. Этот символ она не меняет и продолжает движение направо, заменяя пометки на 0, пока не достигнет символа с пометкой 0. В этот момент она начинает двигаться налево до тех пор, пока не обнаружит (уже единственный) символ с пометкой 1. Заменив пометку этого символа на 0, машина M_3 останавливается. Таблица переходов машины M_3 задаётся следующим образом (a пробегает множество непустых символов алфавита):

$$\begin{aligned}
&\delta_3((\Lambda,1),q_0) = ((\Lambda,0),1,q_0),\\
&\delta_3((\Lambda,0),q_0) = ((\Lambda,0),0,q_3),\\
&\delta_3((a,1),q_0) = ((a,1),1,q_1),\\
&\delta_3((\Lambda,1),q_1) = ((\Lambda,0),1,q_1),\\
&\delta_3((\Lambda,0),q_1) = ((\Lambda,0),-1,q_2),\\
&\delta_3((a,1),q_1) = ((a,0),1,q_1),\\
&\delta_3((a,0),q_1) = ((a,0),-1,q_2),\\
&\delta_3((\Lambda,0),q_2) = ((\Lambda,0),-1,q_2),\\
&\delta_3((a,1),q_2) = ((a,0),0,q_3),\\
&\delta_3((a,0),q_2) = ((a,0),-1,q_2).
\end{aligned}$$

Проверка корректности этой таблицы предоставляется читателю в качестве полезного упражнения.

Замечание 4.15. Иногда бывает желательно не только фиксировать положение головки в конце работы, но и потребовать, чтобы финальное состояние машины M' совпадало с финальным состоянием машины M. Для этого нужно использовать веерное соединение, в котором для каждого финального состояния

запускается своя копия машины M_2 из описанной выше конструкции.

Задача 4.2. Постройте машину Тьюринга, результат работы которой по входе $u\#v$, $u, v \in \{0,1\}^*$, равен 1, если u и v задают одно и то же натуральное число, и равен 0 в противном случае.

4.2. Алгоритмически неразрешимые проблемы

Начнём с того, что дадим точную формулировку вычислительной задачи (или алгоритмической проблемы — это синонимы).

Существуют два основных подхода к определению вычислительной задачи.

Во-первых, под вычислительной задачей можно понимать задачу вычисления значения некоторой функции при заданном значении аргументов.

Во-вторых, можно считать, что задача состоит в проверке некоторого свойства входного слова. Проверка свойства входного слова — это то же самое, что проверка принадлежности входного слова некоторому множеству слов (удовлетворяющих данному свойству). Подмножества слов в конечном алфавите традиционно называют *языками*.

При таком подходе под *массовой алгоритмической проблемой* понимают проверку принадлежности входного слова некоторому языку. В этом случае удобно использовать машины Тьюринга специального вида.

Определение 4.16. *Решающая машина Тьюринга* имеет ровно два финальных состояния: q_{yes} и q_{no}.

Определение 4.17. Язык L в конечном алфавите A называется *разрешимым*, если существует такая

решающая МТ с алфавитом $B \supset A$, которая на любом входе $w \in L$ останавливается в состоянии q_{yes}, а на любом входе $w \notin L$ останавливается в состоянии q_{no}. При этом пустой символ Λ не принадлежит алфавиту A.

Связь между этими двумя определениями очевидна. С каждым языком L можно связать *характеристическую функцию* χ_L:

$$\chi_L(w) = \begin{cases} 1, & \text{если } w \in L, \\ 0, & \text{если } w \notin L. \end{cases}$$

Разрешимость языка L равносильна вычислимости характеристической функции χ_L.

Задача 4.3. Пусть язык L содержит конечное множество слов. Докажите, что L разрешим.

4.2.1. Самоприменимость и остановка

Проблему самоприменимости можно сформулировать весьма общим способом. Поскольку множество алгоритмов счётно, их можно занумеровать. Фиксируем некоторую нумерацию алгоритмов $A_0, A_1, \ldots$ (повторения допустимы).

Кроме того, множество входов алгоритмов также счётно. Поэтому без ограничения общности считаем, что на вход алгоритму подаётся натуральное число.

Проблема самоприменимости. Определим функцию $F\colon \mathbb{N} \to \mathbb{N}$ следующими правилами: $F(i) = 1$, если алгоритм A_i останавливается на входе i, и $F(i) = 0$ в противном случае.

Проблема самоприменимости состоит в вычислении функции F.

Теорема 4.18. *Для любой нумерации алгоритмов проблема самоприменимости алгоритмически неразрешима, т. е. функция F не является вычислимой.*

Доказательство. Составим бесконечную таблицу, в которой элемент S_{ij}, стоящий в i-й строке и j-м столбце, равен 1, если алгоритм A_i останавливается на входе j, и равен 0 в противном случае. Заметим, что $F(i) = S_{ii}$.

Предположим, что существует алгоритм, который вычисляет F. Ясно, что тогда существует и такой алгоритм A, который на входе i вычисляет $F(i)$, после чего останавливается, если $F(i) = 0$, а если $F(i) = 1$, то алгоритм «зацикливается» (т. е. продолжает работу бесконечно долго).

У алгоритма A есть номер в нумерации алгоритмов. Пусть этот номер равен N. Чему равно S_{NN}? Предположим, что $S_{NN} = F(N) = 1$. Тогда с одной стороны это означает, что алгоритм A на входе с номером N останавливается ($S_{NN} = 1$), а с другой — что не останавливается (так как $F(N) = 1$). Если же $S_{NN} = F_N = 0$, то получаем, что алгоритм не останавливается на входе N, так как $S_{NN} = 0$, но поскольку $F_N = 0$, то из описания алгоритма A следует, что он останавливается. Итак, для любого из возможных значений S_{NN} мы пришли к противоречию. Отсюда следует, что ложно предположение о существовании алгоритма, вычисляющего F. □

Замечание 4.19. Применённый в этом доказательстве приём называется *диагональной конструкцией*. Впервые он был использован Г. Кантором для доказательства несчётности множества действительных чисел.

Замечание 4.20. Обратите внимание, что нумерация алгоритмов в проблеме самоприменимости произвольна и даже необязательно вычислима.

Замечание 4.21. В приведённом доказательстве есть фраза, начинающаяся словами «Ясно, что...». Большинство ошибок в математических рассуждениях скрываются именно за этим вводным оборотом и его синонимами.

Впрочем, в данном случае никакой ошибки нет. Однако уточнение рассуждения требует использования какого-нибудь формального определения алгоритма, например, машин Тьюринга.

Описанный в рассуждении алгоритм можно получить соединением машины, вычисляющей функцию F, и машины со следующим свойством: начиная работу на ленте, содержащей результат вычисления функции F, машина останавливается, если количество единиц на ленте равно 1, в противном случае она перемещает головку вправо бесконечно долго. Обратите внимание, что одна единица является кодом числа 0 в определении вычислимой функции на с. 178. Для корректности работы соединения машин нужно также потребовать, чтобы вторая машина выполняла указанные действия, начиная работу с любого места ленты.

Задача 4.4. Постройте машину Тьюринга, обладающую описанными в предыдущем замечании свойствами.

Проблему самоприменимости можно обобщить.

Проблема остановки. Она состоит в вычислении значения функции $f_{\mathrm{S}}\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, заданной правилами: $f_{\mathrm{S}}(i,j) = 1$, если алгоритм A_i останавливается на входе j, и $f_{\mathrm{S}}(i,j) = 0$ в противном случае.

Теорема 4.22. *Проблема остановки при любой нумерации алгоритмов является алгоритмически неразрешимой.*

Эта теорема легко следует из теоремы 4.18, поскольку

$$F(i) = \mathrm{stop}(i, i).$$

Поэтому из вычислимости функции f_{S} следует вычислимость функции самоприменимости F.

Мы сформулировали проблемы самоприменимости и остановки как задачи вычисления целочисленной функции от неотрицательных целочисленных аргументов. Есть другая формулировка этих задач, в которой нужно ответить на вопрос об остановке, если задано описание машины Тьюринга и входного слова. При такой формулировке удобно использовать введённое выше понятие разрешимого языка.

Определение 4.23. Обозначим через stop язык в алфавите $\{0,1\}^*$, состоящий из тех описаний $\langle M, w\rangle$ машины Тьюринга M и её входного слова w, для которых M останавливается на входе w.

Теорема 4.24. *Язык* stop *неразрешим.*

Доказательство этой теоремы следует из теоремы 4.22, поскольку с любым способом описания МТ и их входов можно связать нумерацию. Например, слова можно нумеровать в лексикографическом порядке.

Лексикографический порядок на словах в алфавите $\{0, 1, \ldots, n\}$ определяется следующим правилом: слово u предшествует слову v в лексикографическом порядке (обозначение $u \prec v$) тогда и только тогда, когда либо длина u меньше чем длина v, либо $u = pas_1$, $v = pbs_2$, где $p, s_1, s_2 \in \{0, 1, \ldots, n\}^*$, $a, b \in \{0, 1, \ldots, n\}$ и $a < b$.

Чтобы определить лексикографический порядок на словах в произвольном алфавите A, символы этого алфавита нужно занумеровать, после чего использовать предыдущее определение.

Задача 4.5. Постройте машины Тьюринга, которые

(а) по слову w строит следующее в лексикографическом порядке слово;

(б) по слову w определяет его номер при нумерации в лексикографическом порядке.

Используя машины из задачи 4.5, можно вывести теорему 4.24 из теоремы 4.22 с помощью стандартного приёма, называемого *сводимостью*.

Предположим, что существует решающий алгоритм R для языка stop. Тогда функция f_{S} вычисляется следующим алгоритмом: по входу (i, j) он строит описание $\langle M_i, 1^j \rangle$, где M_i — МТ, описание которой имеет номер i в лексикографическом порядке[3)], после чего применяет алгоритм R к этому описанию и выдаёт значение 1, если $\langle M_i, 1^j \rangle \in \mathrm{stop}$, или 0 в противном случае.

Поскольку не существует алгоритма вычисления f_{S}, то язык stop неразрешим.

4.2.2. Проблема равенства слов

Приведённые выше примеры алгоритмически неразрешимых проблем получались диагональной конструкцией. В этом разделе мы рассмотрим примеры

[3)] В этом месте имеется ещё одно формальное затруднение — не все слова являются описаниями МТ в смысле определения на с. 175. Преодолеть эту трудность можно сопоставлением таким словам некоторой фиксированной МТ M_0.

алгоритмически неразрешимых проблем, естественно возникающие при анализе преобразований слов.

Фиксируем некоторый конечный алфавит A.

Проблема достижимости в ассоциативном исчислении. Пусть задан набор правил преобразования слов

$$L_1 \to R_1,$$
$$L_2 \to R_2,$$
$$\dots$$
$$L_n \to R_n.$$

Здесь L_i, R_i — слова в алфавите A, быть может пустые (пустое слово будем обозначать λ). Правила преобразования могут применяться к слову $w \in A^*$ следующим образом. Если $w = uL_iv$, то слово w можно заменить на слово $w' = uR_iv$. Другими словами, любое вхождение левой части правила можно заменить на правую часть того же правила.

Задача достижимости состоит в том, чтобы по заданным правилам преобразования и по двум заданным словам решить, можно ли получить одно слово из другого с помощью нескольких преобразований указанного выше вида.

Замечание 4.25. Указанный набор правил преобразования слов называется *ассоциативным исчислением* или *полутуэвской системой*.

Рассмотрим несколько частных случаев проблемы достижимости.

Пример 4.26. Алфавит состоит из двух символов $\{a, b\}$, правила преобразования имеют вид:

$$aba \to bab,$$
$$bbb \to a,$$
$$aaab \to babbb.$$

С помощью этих преобразований слово $aaaaabbbbb$ можно перевести в слово $abbaab$:

$$aaaaabbbbb \to aababbbbbb \to ababbbbbbb \to abaabbbbb \to abaaabb \to abbabbbb \to abbaab$$

Заметим, что к слову $abbaab$ нельзя применить ни одного преобразования из данной системы, поэтому его нельзя преобразовать ни в какое другое слово.

Пример 4.27. Алфавит состоит из двух символов $\{a, b\}$, правила преобразования имеют вид:

$$aba \to bab,$$
$$bbaa \to aabb,$$
$$baba \to bbaa.$$

Легко видеть, что с помощью таких преобразований нельзя преобразовать слово aa в слово bbb. Действительно, указанные преобразования не меняют длину слов.

Замечание 4.28. *Инвариант* — свойство, которое сохраняется при преобразованиях. Очевидно, что если у двух слов значение инварианта различно, то нельзя перевести одно слово в другое.

Построение подходящего инварианта является одним из самых полезных способов доказательства невозможности преобразования. В предыдущем примере инвариантом является длина слова.

Пример 4.29. Алфавит состоит из двух символов $\{a, b\}$, правила преобразования имеют вид:

$$aa \to \lambda,$$
$$bbbb \to \lambda,$$
$$ba \to abbb.$$

Требуется выяснить, можно ли преобразовать с помощью этой системы слово $aababbbaaa$ в слово bab. Оказывается, что сделать это нельзя. Чтобы доказать невозможность такого преобразования, мы используем несколько более сложное рассуждение, чем применение инварианта.

Назовём индексом вхождения буквы a в слово w количество букв b правее этого вхождения буквы a. Индексом слова назовём сумму индексов по всем вхождениям буквы a. Легко видеть, что первые два правила не меняют чётности индекса, а третье правило всегда меняет. В самом деле, пусть на текущем шаге получено слово $PbaQ$. Тогда после применения третьего правила получается слово $PabbbQ$. Для букв a, входящих в слово P или в слово Q, индекс не изменился, а вот для буквы a между словами P и Q чётность количества букв b, стоящих правее, меняется, что и требовалось показать.

Так как чётность индексов слов $aababbbaaa$ и bab одинакова, то число применений третьего правила должно быть чётным. Следовательно, если третье правило хотя бы раз применялось, то разность между количеством букв b в двух словах должна делится на 4, но это не так в рассматриваемом случае. Значит, третье правило вообще ни разу не применялось. Откуда получаем, что слово $aababbbaaa$ нельзя преобразовать с помощью данных правил в слово bab.

В разных случаях проблемы достижимости мы применяли весьма различные рассуждения. Это не случайно — единого алгоритма решения этой проблемы не существует.

Теорема 4.30. *Проблема достижимости в ассоциативном исчислении алгоритмически неразрешима.*

Доказательство. Мы сведём решение проблемы остановки машины Тьюринга к решению проблемы достижимости.

Другими словами, мы построим *сводящий* алгоритм, который по описанию машины Тьюринга M и её входа w строит описание таких правил подстановок и двух слов s, f, что f достижимо из s тогда и только тогда, когда M останавливается на w.

Тогда из алгоритмической неразрешимости проблемы остановки следует алгоритмическая неразрешимость проблемы достижимости. Действительно, соединение сводящего алгоритма и алгоритма решения проблемы достижимости даёт в силу описанного выше свойства сводимости алгоритм решения проблемы остановки.

Оставшаяся часть доказательства состоит в описании сводящего алгоритма и доказательстве его корректности.

Итак, пусть имеется МТ $(A, Q, Q_f, \delta, q_0, \Lambda)$. Напомним, что по определению 4.1 функция δ всюду определена. Мы хотим описать такт работы МТ как результат применения правил подстановки. При этом удобно применять правила подстановки к словам более общего вида, чем конфигурации. А именно, расширим алфавит, добавив к алфавиту МТ A дополнительный символ Λ_0. Конфигурации МТ W будут соответствовать слова вида $\Lambda_0\Lambda^k W \Lambda^s \Lambda_0$. Правила преобразования будут иметь вид

$$aqb \to \Delta, \quad \text{где } q \in Q;\ a, b \in A \cup \{\Lambda_0\}.$$

Таких правил окажется достаточно, так как на каждом такте работы (расширенная) конфигурация МТ изменяется лишь в «окрестности» головки, т. е. на расстоянии не более 1 от (единственного) символа q из

множества состояний Q, который входит в конфигурацию.

Для определения слов Δ нужно рассмотреть все возможные случаи положения и движения головки.

Зададим систему правил преобразования слов следующим образом.

Вначале приведём правила *первого типа*, относящиеся к нефинальным состояниям МТ и описывающие такты работы МТ.

Пусть $q \notin Q_f$. Расширим функцию переходов на множество $(A \cup \{\Lambda_0\}) \times Q$ по правилу $\delta(\Lambda_0, q) = \delta(\Lambda, q)$.

Пусть $\delta(a, q) = (b, 0, q')$. Этому значению таблицы переходов мы сопоставим правила

$$xqa \to xq'b, \quad \text{для всех } x \in A \cup \{\Lambda_0\},\ a \neq \Lambda_0, \tag{4.3}$$

$$xq\Lambda_0 \to xq'b\Lambda_0, \quad \text{для всех } x \in A \cup \{\Lambda_0\}. \tag{4.4}$$

Если $\delta(a, q) = (b, 1, q')$, то добавим правила

$$xqa \to xbq', \quad \text{для всех } x \in A \cup \{\Lambda_0\},\ a \neq \Lambda_0, \tag{4.5}$$

$$xq\Lambda_0 \to xbq'\Lambda_0, \quad \text{для всех } x \in A \cup \{\Lambda_0\}. \tag{4.6}$$

Наконец, если $\delta(a, q) = (b, -1, q')$, то добавим правила

$$xqa \to q'xb, \qquad \text{для всех } x \in A,\ a \neq \Lambda_0, \tag{4.7}$$

$$\Lambda_0 qa \to \Lambda_0 q'\Lambda b, \qquad a \neq \Lambda_0, \tag{4.8}$$

$$xq\Lambda_0 \to q'xb\Lambda_0, \qquad \text{для всех } x \in A, \tag{4.9}$$

$$\Lambda_0 q\Lambda_0 \to \Lambda_0 q'\Lambda b\Lambda_0. \tag{4.10}$$

Теперь пусть $q \in Q_f$. Для таких состояний мы добавляем правила *второго типа*

$$xq \to q, \qquad \text{для всех } x \in A, \tag{4.11}$$

$$qx \to q, \qquad \text{для всех } x \in A, \tag{4.12}$$

$$\Lambda_0 q\Lambda_0 \to \Lambda_0\Lambda_0. \tag{4.13}$$

Мы определили по МТ некоторую систему правил преобразования в алфавите $A \cup \{\Lambda_0\}$. Построение системы правил преобразования было вполне конструктивным, так что по описанию МТ соответствующую систему правил можно построить алгоритмически[4)].

Теперь мы докажем, что МТ останавливается на входе W тогда и только тогда, когда слово $\Lambda_0 q_0 W \Lambda_0$ можно преобразовать в слово $\Lambda_0\Lambda_0$ с помощью введённой системы правил.

Прежде всего заметим, что применение любого из правил первого типа к слову вида $\Lambda_0\Lambda^k W \Lambda^s \Lambda_0$, где W — конфигурация МТ, приводит к слову такого же вида. Кроме того, для каждой пары (a, q), $a \in A$, $q \in Q \setminus Q_f$ и слова вида $\Lambda_0\Lambda^k W \Lambda^s \Lambda_0$ есть не более одного правила первого типа, которое можно применить к данному слову.

Значит, если МТ не останавливается, то преобразованиями из данной системы правил будут получаться слова, в которые обязательно входит символ из $Q \setminus Q_f$, т. е. слово $\Lambda_0\Lambda_0$ получить из начального слова нельзя.

Предположим, что МТ останавливается. Тогда, применяя правила соответственно тактам работы МТ, можно перевести начальное слово в слово, содержащее символ q из Q_f. Теперь можно, применяя правила второго типа, сократить из слова все символы из алфавита A и получить слово $\Lambda_0 q \Lambda_0$. Применяя к нему одно из последних правил второго типа, получим слово $\Lambda_0\Lambda_0$. □

[4)] Недоверчивый читатель должен в этом месте самостоятельно построить МТ, которая преобразует описание машины Тьюринга и её входа в описание указанной выше системы правил и пары слов $\Lambda_0 q_0 W \Lambda_0$, $\Lambda_0\Lambda_0$. Задача сильно облегчается, если использовать многоленточные машины Тьюринга, см. ниже раздел 4.3.2.

Проблема равенства слов в полугруппе. Это частный случай предыдущей задачи.

Пару правил преобразования слов $L \to R$, $R \to L$ будем называть *двусторонним правилом преобразования слов* и обозначать $L \leftrightarrow R$.

Пусть задан набор двусторонних правил преобразования слов

$$L_1 \leftrightarrow R_1,$$
$$L_2 \leftrightarrow R_2,$$
$$\dots$$
$$L_n \leftrightarrow R_n.$$

Здесь L_i, R_i — слова в алфавите A, быть может пустые. Правила преобразования могут применяться как слева направо, так и справа налево.

Задача состоит в том, чтобы по заданным правилам преобразования и по двум заданным словам решить, можно ли получить одно слово из другого с помощью нескольких преобразований указанного выше вида.

Замечание 4.31. Система двусторонних правил преобразования слов называется *туэвской системой*. Легко видеть, что отношение достижимости в туэвской системе является отношением эквивалентности. Можно проверить, что классы эквивалентности по отношению достижимости в туэвской системе образуют полугруппу относительно конкатенации слов. Построенная таким образом полугруппа называется полугруппой с порождающими из алфавита A и соотношениями, задаваемыми туэвской системой. Это объясняет название проблемы.

Теорема 4.32. *Проблема равенства слов в полугруппе алгоритмически неразрешима.*

Доказательство. Система правил подстановки такая же, как в предыдущей теореме, т. е. включает в себя правила (4.3–4.13) (теперь они понимаются как двусторонние правила преобразования). Рассуждение из предыдущей теоремы нужно модифицировать, чтобы доказать более сильное утверждение.

В одну сторону ничего не меняется: если МТ останавливается, то преобразованиями из данной системы правил можно получить слово $\Lambda_0\Lambda_0$.

Доказательство в обратную сторону требует теперь несколько больших усилий.

Назовём *активными* символы из множества Q. Все правила подстановки, за исключением последнего правила (4.13), содержат ровно один активный символ в левой части и ровно один активный символ в правой части. Поэтому при любом применении этих правил из начального слова, содержащего ровно один активный символ q_0, получается слово, в которое также входит ровно один активный символ, или получается слово $\Lambda_0\Lambda_0$. Обозначим через C множество тех слов, которые имеют вид $\Lambda_0 U \Lambda_0$, $U \in (A \cup Q)^*$, содержат ровно один активный символ q, и этот символ нефинален (т. е. $q \notin Q_f$), а через C_f — множество тех слов вида $\Lambda_0 U \Lambda_0$, $U \in (A \cup Q)^*$, которые содержат ровно один активный символ q, причём этот символ финальный ($g \in Q_f$).

Легко видеть, что все слова из C_f достижимы друг из друга с помощью правил второго типа (4.11–4.13), так как теперь эти правила можно применять в обе стороны.

Можно убедиться, просмотрев все правила, что для любого слова из C есть ровно одно правило подстановки, которое можно применить слева направо. И в этом нет ничего удивительного: мы составляли систему правил (4.3–4.10) таким образом, чтобы описать работу

МТ на одном такте, а действия МТ на каждом такте работы определены однозначно.

Заметим также, что никакое слово из C_f нельзя получить из слова C применением правила подстановки справа налево. Отсюда вытекает, что если слово $\Lambda_0 U \Lambda_0 \in C_f$ можно получить из начального слова $\Lambda_0 q_0 W \Lambda_0$ двусторонними подстановками, то его можно получить, используя лишь подстановки слева направо.

Действительно, возьмём такую цепочку преобразований от начального слова к слову $\Lambda_0 U \Lambda_0$, в которой используется минимальное количество преобразований справа налево. Рассмотрим последнее такое преобразование:

$$W_0 \leftrightarrow \cdots \leftrightarrow W_k \leftarrow W_{k+1} \rightarrow W_{k+2} \rightarrow \cdots \rightarrow \Lambda_0 U \Lambda_0.$$

Поскольку к слову W_{k+1} можно применить ровно одно преобразование слева направо, имеем равенство $W_k = W_{k+2}$. Но тогда в цепочке преобразований

$$W_0 \leftrightarrow W_1 \leftrightarrow \cdots \leftrightarrow W_k \rightarrow \cdots \rightarrow \Lambda_0 U \Lambda_0$$

на одно преобразование справа налево меньше. Полученное противоречие доказывает, что слово $\Lambda_0 U \Lambda_0$ можно получить только подстановками слева направо.

Теперь можно закончить доказательство так же, как и в случае ориентированных подстановок, считая цепочку преобразований до слова $\Lambda_0 U \Lambda_0$ ориентированной. □

4.2.3. Проблема соответствия Поста

В этом разделе мы рассмотрим ещё одну комбинаторную задачу — *проблему соответствия Поста*. Она также оказывается неразрешимой, а доказательство неразрешимости будет состоять в построении сводимости проблемы достижимости к проблеме соответствия.

Проблема соответствия. *Системой соответствия Поста* назовём набор пар слов в конечном алфавите:

$$\left[\frac{u_1}{b_1}\right], \left[\frac{u_2}{b_2}\right], \ldots, \left[\frac{u_k}{b_k}\right].$$

Здесь u_i, b_i — слова в алфавите A. Пару $\left[\frac{u}{v}\right]$ будем называть *связкой*, слово u — *верхним словом*, слово v — *нижним словом* в связке.

Решением системы соответствия называется такая последовательность чисел $i_1, i_2, \ldots, i_n$, что

$$u_{i_1} u_{i_2} \ldots u_{i_n} = b_{i_1} b_{i_2} \ldots b_{i_n}$$

(конкатенация верхних слов в последовательности связок, задаваемой числами $i_1, i_2, \ldots, i_n$, совпадает с конкатенацией нижних слов той же последовательности).

Алгоритмическая задача состоит в том, чтобы по заданной системе соответствия узнать, есть ли у неё решение.

Приведём алгебраическую формулировку проблемы соответствия. Имеется моноид $A^* \times A^*$, т. е. множество, состоящее из пар слов в алфавите A с операцией почленной конкатенации. Дано некоторое множество элементов $\{s_i\}$, $s_i \in A^* \times A^*$ (связки). Требуется узнать, можно ли выразить какой-нибудь диагональный элемент (u, u) как произведение элементов из заданного множества связок.

Пример 4.33. Рассмотрим систему соответствия

$$\left[\frac{a}{a}\right], \left[\frac{b}{b}\right], \left[\frac{bab}{bb}\right], \left[\frac{abba}{\lambda}\right], \left[\frac{\lambda}{ababa}\right], \tag{4.14}$$

здесь λ обозначает пустое слово.

Эта система имеет решение:

$$\left[\frac{abba}{\lambda}\right]\left[\frac{a}{a}\right]\left[\frac{bab}{bb}\right]\left[\frac{a}{a}\right]\left[\frac{\lambda}{ababa}\right] = \left[\frac{abba|a|bab|a|}{|a|bb|a|ababa}\right]. \quad (4.15)$$

В правой части мы для наглядности разграничили подслова, вошедшие из разных сомножителей в левой части.

Конечно, у системы (4.14) есть очевидное решение $\left[\frac{a}{a}\right]$.

Однако решение (4.15) показывает способ, которым мы будем имитировать в системах соответствия правила преобразования ассоциативного исчисления: обратите внимание, что слово $ababa$ получается из $abba$ применением правила преобразования $bb \to bab$.

Сводимость проблемы достижимости к проблеме соответствия будет опираться на идею, намеченную в предыдущем примере. Для корректного применения этой идеи нужно избавиться от паразитных решений (см. всё тот же пример 4.33). Ниже мы приводим доказательство, основанное на изложении М. Девиса в [6].

Теорема 4.34. *Проблема соответствия Поста алгоритмически неразрешима.*

Доказательство. Начнём с описания сводимости.

Пусть $\mathcal{A}$ — набор правил преобразования

$$\begin{aligned} &L_1 \to R_1,\\ &L_2 \to R_2,\\ &\dots\\ &L_n \to R_n. \end{aligned}$$

слов в алфавите $A = \{a_1, \dots, a_n\}$, а u, v — два слова в алфавите A.

По этим данным мы сейчас построим систему соответствия $\mathcal{C}$. Алфавит этой системы

$$\{a_1, \ldots, a_n, a'_1, \ldots, a'_n, \triangleleft, \triangleright, *, *'\}$$

состоит из двух копий алфавита A и вспомогательных символов $\triangleleft, \triangleright, *, *'$.

Связки в системе соответствия $\mathcal{C}$ разделим для удобства на несколько типов.

Снятие и добавление штриха:

$$\left[\frac{a'_i}{a_i}\right], \left[\frac{a_i}{a'_i}\right], \left[\frac{*'}{*}\right], \left[\frac{*}{*'}\right]. \tag{4.16}$$

Начальная и конечная связки:

$$\left[\frac{\triangleleft u *}{\triangleleft}\right], \left[\frac{\triangleright}{*' v \triangleright}\right]. \tag{4.17}$$

Связки для правил преобразования $L_i \to R_i$:

$$\left[\frac{R_i}{L'_i}\right], \left[\frac{R'_i}{L_i}\right]. \tag{4.18}$$

В этом наборе связок мы используем преобразование слова u в алфавите A в слово u' в алфавите $A' = \{a'_1, \ldots, a'_n\}$, которое заключается в добавлении штриха к каждому символу слова u.

Нетрудно видеть, что существует алгоритм, который по описанию ассоциативного исчисления $\mathcal{A}$ и пары слов u, v строит описание системы соответствия $\mathcal{C}$. (Как и в случае проблемы достижимости, рекомендуем упорному читателю доказать это утверждение.)

Теперь докажем, что достижимость слова v из слова u по правилам ассоциативного исчисления $\mathcal{A}$ равносильна существованию решения для системы соответствия $\mathcal{C}$.

Прежде всего заметим, что снятие и добавление штриха к символу алфавита A можно рассматривать

как частный случай связки, отвечающей правилу преобразования, если в правила включены тавтологические преобразования $a_i \to a_i$, которые фактически ничего не меняют. Поэтому далее мы считаем, что ассоциативное исчисление $\mathcal{A}$ содержит правила $a_i \to a_i$.

Пусть имеется последовательность преобразований

$$u = u_1 \to u_2 \to \cdots \to u_m = v, \tag{4.19}$$

переводящая слово u в слово v. Поскольку мы предположили, что $\mathcal{A}$ содержит тавтологические правила, то без ограничения общности полагаем, что m нечётно. Докажем, что

$$\left[\frac{\triangleleft u_1 * u'_2 *' u_3 * \ldots u'_{m-1} *' u_m \triangleright}{\triangleleft u_1 * u'_2 *' u_3 * \ldots u'_{m-1} *' u_m \triangleright}\right] \tag{4.20}$$

можно представить как произведение связок системы $\mathcal{C}$, что и означает существование решения.

Переход к следующему слову в последовательности (4.19) состоит в замене одного из вхождений слова L_i на слово R_i. Поэтому соседние в последовательности слова связаны между собой соотношениями

$$u_i = \ell_i L_{j(i)} r_i, \quad u_{i+1} = \ell_i R_{j(i)} r_i. \tag{4.21}$$

Разобьём верхнее и нижнее слово в (4.20) на части, используя соотношения (4.21):

$$\left[\frac{\triangleleft u_1| * |\ell'_1|R'_{j(1)}|r'_1| *' | \ldots | *' |\ell_{m-1}|R_{j(m-1)}|r_{m-1}|\triangleright}{\triangleleft|\ell_1|L_{j(1)}|r_1| * | \ldots | *' |\ell'_{m-1}|L'_{j(m-1)}|r'_{m-1}| *' |u_m\triangleright}\right] \tag{4.22}$$

Это разбиение порождает решение, так как пары вида

$$\left[\frac{\ell'_i}{\ell_i}\right], \left[\frac{\ell_i}{\ell'_i}\right], \left[\frac{r'_i}{r_i}\right], \left[\frac{r_i}{r'_i}\right]$$

выражаются как произведения связок снятия и добавления штриха, а остальные части являются связками системы $\mathcal{C}$.

Теперь проверим обратное утверждение. Пусть имеется решение системы соответствия $\mathcal{C}$. Оно обязано начинаться с начальной связки $\left[\frac{\triangleleft u *}{\triangleleft}\right]$, поскольку во всех остальных связках начальные символы верхнего и нижнего слов различны. Решение также обязано содержать конечную связку: во всех связках кроме начальной и конечной одинаковое количество символов $*$ (со штрихом или без штриха).

Рассмотрим часть решения от начала до первого вхождения конечной связки. Она разбивается вхождениями символов $*$ так, как указано в (4.20). Но такая пара может получиться только способом, указанным в (4.22), — это можно доказать индукцией по индексу i слов u_i. Значит, слово $u_m = v$ получается из слова $u_1 = u$ последовательностью преобразований по правилам ассоциативного исчисления $\mathcal{A}$.

Если существует алгоритм проверки разрешимости систем соответствия, то можно построить алгоритм проверки достижимости в ассоциативном исчислении, который по входу задачи достижимости $\mathcal{A}, u, v$ строит систему соответствия $\mathcal{C}$, а затем проверяет её разрешимость. Корректность алгоритма следует из доказанного выше свойства: разрешимость $\mathcal{C}$ равносильная достижимости v из u по правилам $\mathcal{A}$.

Из неразрешимости проблемы достижимости следует неразрешимость проблемы соответствия. □

4.2.4. Функция трудолюбия Радо

Приведём пример такой невычислимой функции, доказательство невычислимости которой не использует проблему остановки.

Определение 4.35. Функция Радо $R(n)$ определяется как максимальное количество единиц в результате

работы на пустом входе машины Тьюринга с n состояниями и алфавитом $\{1, \Lambda\}$ при условии, что МТ останавливается на пустом входе.

Другими словами, между машинами Тьюринга с n состояниями происходит соревнование: какая из них напишет больше всего единиц на изначально пустую ленту и остановится. Функция Радо равна результату победителя в этом соревновании.

Теорема 4.36. *Функция Радо невычислима.*

Доказательство. Очевидно, что $R(n)$ неубывающая функция. При этом $R(n)$ растёт достаточно быстро.

Рассмотрим МТ

$$\delta(\Lambda, q_{3i+0}) = (\Lambda, +1, q_{3i+1}), \quad \delta(1, q_{3i+0}) = (1, +1, q_{3i+0}),$$
$$\delta(\Lambda, q_{3i+1}) = (1, -1, q_{3i+2}), \quad \delta(1, q_{3i+1}) = (1, +1, q_{3i+2}),$$
$$\delta(\Lambda, q_{3i+2}) = (1, +1, q_{3i+1}), \quad \delta(1, q_{3i+2}) = (1, +1, q_{3i+3})$$

$(0 \leqslant i < n)$. Выполняя такты работы этой машины, мы увидим, что из начальной конфигурации q_0 она переходит в конфигурацию $111q_31$, затем в $1111q_3$, затем в $1111111q_61$ и т. д.. Останавливается такая машина в конфигурации

$$\underbrace{11\ldots11}_{4n-1} q_{3n}1.$$

Таким образом, $R(3n+1) \geqslant 4n$.

Предположим, что $R(n)$ вычислима. Тогда при достаточно больших n выполняется неравенство

$$R(C + 3n + 1) \geqslant R(4n) \tag{4.23}$$

при некоторой константе C. Действительно, нужно запустить после описанной выше машины такую машину, которая переводит головку на самую левую единицу и запускает МТ, вычисляющую функцию R.

Полученное соединение машин будет иметь $C + 3n + 1$ состояний и оставлять на ленте $R(4n)$ единиц.

Осталось заметить, что при достаточно больших n выполнено неравенство $C + 3n + 1 \leqslant 4n$. Поэтому неравенство (4.23) противоречит монотонности функции R. □

4.3. Моделирование на машинах Тьюринга других моделей вычислений

Выше был сформулирован тезис Тьюринга. Согласно этому тезису любой алгоритм может быть реализован в виде машины Тьюринга. Один из способов подтверждения этого тезиса состоит в моделировании на машинах Тьюринга других моделей вычислений. В этом разделе мы приведём ряд результатов на эту тему. Мы ограничимся лишь моделями, определяемыми процедурным способом. С функциональными определениями алгоритма, на которые опираются такие языки программирования, как Lisp и Haskell, читатель может познакомиться по книгам [12, 11, 16] (рекурсивные функции), [2, 18] (λ-исчисление Чёрча), [7] (λ-исчисление и язык Haskell).

Как понимать моделирование одной модели вычислений на другой? Можно считать, что вычислительная модель задаёт семейство частично определённых отображений (алгоритмов)

$$\langle\text{алгоритм}\rangle\colon \langle\text{вход}\rangle \mapsto \langle\text{результат}\rangle.$$

Если у двух вычислительных моделей множества входов и результатов работы совпадают, то естественно считать, что та модель сильнее (может моделировать другую), у которой семейство алгоритмов шире (а если эти семейства алгоритмов совпадают, то будем говорить, что модели *эквивалентны*).

Но в общем случае множества входов и результатов могут различаться в разных моделях. Чтобы задать связь между такими моделями, мы используем отображения кодировки (переводит входы первой модели во входы второй модели) и декодировки (переводит результаты работы второй модели в результаты работы первой модели).

Мы будем говорить, что модель вычислений 1 моделируется на модели вычислений 2 (при заданных отображениях кодировки/декодировки), если всякий алгоритм первой модели можно задать в виде композиции следующих отображений:

$$\langle \text{вход}_1 \rangle \xmapsto{\text{кодировка}} \langle \text{вход}_2 \rangle \xmapsto{\text{алгоритм}_2} \langle \text{результат}_2 \rangle \xmapsto{\text{декодировка}} \langle \text{результат}_1 \rangle.$$

Модели будем считать эквивалентными, если они взаимно моделируют друг друга.

Замечание 4.37. В данных выше пояснениях есть существенный пробел. Мы ничего не сказали о свойствах отображений кодировки/декодировки. Неформально говоря, мы хотим, чтобы эти отображения были «конструктивными». Наше определение алгоритма недостаточно общее, чтобы формализовать это понятие. Однако в каждом из рассматриваемых далее случаев «конструктивность» отображений кодировки/декодировки не будет вызывать сомнений.

4.3.1. Машины Тьюринга в алфавите $\{0, 1\}$

В качестве первого примера моделирования рассмотрим машины Тьюринга, алфавит которых состоит из двух символов. Обозначим эти элементы 0 и 1 и будем считать символ 0 пустым.

Входы и результаты работы машины Тьюринга с алфавитом A являются словами в алфавите A. Для моделирования такой машины на машине с алфавитом $\{0,1\}$, нужно ввести *отображение кодировки*

$$\varphi\colon A^* \to \{0,1\}^*,$$

которое сопоставляет слову в алфавите A некоторое слово в алфавите $\{0,1\}$. Это отображение должно удовлетворять некоторым условиям. Прежде всего, оно должно быть инъективным: разным словам из A^* должны отвечать разные слова из $\{0,1\}^*$. Если это условие нарушается, моделирующая машина будет вести себя одинаково при моделировании двух разных входов. Следующая простая задача показывает, что корректное моделирование в таком случае невозможно.

Задача 4.6. Докажите, что для любых двух слов $u, v \in A^*$ существует машина Тьюринга с алфавитом A, результаты работы которой на словах u, v различны.

Однако инъективности отображения φ недостаточно. Рассмотрим такое отображение слов в алфавите $\{0,1,2,3\}$ в двоичные слова:

$$\varphi(0) = 00,\ \varphi(1) = 01,\ \varphi(2) = 10,\ \varphi(3) = 11,$$
$$\varphi(s_1 s_2 \dots s_k) = \varphi(s_1)\varphi(s_2)\dots\varphi(s_k), \quad s_i \in \{0,1,2,3\}.$$

Это отображение инъективно.

Но возникает другая трудность: в определении входа указано, что первый и последний символы входного слова должны быть непустыми. Кодировки слов 1 и 2 не удовлетворяют этому свойству. Поэтому потребуем, чтобы коды всех слов должны начинались и заканчивались на 1.

Приведём пример подходящей кодировки слов в алфавите $A = \{a_0, \ldots, a_{n-1}\}$ (без ограничения общности, считаем, что $a_0 = \Lambda$ является пустым символом). Кодом символа a_i будем считать слово

$$c(a_i) = 11\underbrace{000\ldots0}_{i+1 \text{ раз}}1\ \underbrace{000\ldots0}_{n-i+1 \text{ раз}}\ 11, \tag{4.24}$$

а кодом слова $a_{i_1}a_{i_2}\ldots a_{i_k}$ — конкатенацию кодов символов $c(a_{i_1})c(a_{i_2})\ldots c(a_{i_k})$. Код любого символа имеет длину $n+7$, поэтому это кодирование равномерно — слову $w \in A^*$ длины k оно сопоставляет слово $c(w) \in \{0,1\}^*$ длины $(n+7)k$.

Задача 4.7. Проверьте, что описанное кодирование обладает следующими свойствами: (а) оно инъективно; (б) код любого слова начинается и заканчивается на 1.

В качестве отображения декодировки, которое нужно для определения эквивалентности вычислительных моделей, возьмём (частично определённое) обратное к отображению кодировки.

Теперь можно уточнить утверждение о моделировании произвольной машины Тьюринга машиной Тьюринга в алфавите $\{0,1\}$.

Машина Тьюринга N в алфавите $\{0,1\}$ *моделирует* машину Тьюринга $M = (A, Q, Q_f, \delta, q_0, \Lambda)$, где $|A| = n$, если она удовлетворяет следующему свойству: на входе вида $c(w)$, где w — вход машины M, результатом работы МТ N является $c(u)$, где u — результат работы машины M на входе w.

Построить такую машину довольно просто. Её множество состояний имеет вид $Q \times \{0,1\}^{n+7} \times Q'$. Другими словами, в «оперативной памяти» машины N поддерживается информация о состоянии моделируемой машины M и о текущем символе моделируемой

машины (см. пояснения на с. 182). Управление работой N осуществляется состояниями из множества Q'.

Начальным состоянием машины N является состояние $(q_0, 0^{n+7}, q'_0)$, финальными — состояния вида (q, w, q'), где $q \in Q_f$, а $q' \in Q'$.

Мы опишем работу машины N как выполнение последовательности итераций. Перед началом каждой итерации будут выполнены условия корректности: машина находится в состоянии вида (q, w, q'_0); на ленте записано слово $c(u)$, которое является кодом слова $u \in A^*$; головка машины находится над самым левым символом кода некоторого символа a слова u. Каждая итерация занимает не более $3(n+7)+2$ тактов работы, каждому из которых соответствует своё собственное состояние из множества Q'. Как будет видно, вариантов каждого такта не более трёх, так что общий размер множества Q' не превосходит $9(n + 7) + 6$.

Итерация состоит из следующих шагов (предполагаются выполненными условия корректности перед началом итерации):

- Прочитать в «оперативную память» текущий символ на ленте. Для этого машина делает $n + 7$ тактов работы, каждый раз запоминая в памяти прочитанный с ленты символ и сдвигаясь вправо (кроме последнего такта). В конце этого шага головка находится над последним символом кода текущего символа. Состояние машины в этот момент имеет вид $(q, c(a), q'_{n+7})$.

- Изменить текущий символ в соответствии с таблицей переходов машины M. Этот шаг также требует $n + 7$ тактов работы: машина движется справа налево, записывая на очередное место очередной символ кода символа a', который

машина M пишет, когда находится в состоянии q и видит символ a.

- Изменить положение головки. После предыдущего шага головка будет находиться над крайним левым символом кода символа. Если в таблице переходов M в строке (a, q) записана инструкция «остаться на месте», то ничего делать не нужно. Если записана инструкция «сдвинуться влево», то нужно выяснить, записан ли слева код какого-нибудь символа (для этого достаточно сделать шаг влево и проверить, записана ли в этой ячейке 1). После этого нужно сделать $n + 7$ тактов работы, сдвигая головку влево (и записывая при необходимости биты кода пустого символа, в этом случае необходим ещё один дополнительный такт работы для записи последнего бита, после которого машина остаётся на месте). Аналогичные действия происходят при обработке инструкции «сдвинуться вправо».

- Изменить первую компоненту состояния в соответствии с таблицей переходов машины M, а третью сделать равной q'_0. После этого все условия корректности становятся выполненными.

Фактически, мы доказали следующую теорему.

Теорема 4.38. *Для любой машины Тьюринга M существует машина в двухбуквенном алфавите M', моделирующая работу M.*

Задача 4.8 (*«МТ с засорившейся головкой»*). МТЗГ — это такая машина Тьюринга, которая может писать на ленту только символ $*$. Формально это условие можно выразить так: если $\delta(a, q) = (a', r, q')$, то

$a' \in \{a, *\}$. Докажите, что для любой МТ M, алфавит которой не содержит $*$, существует такая МТЗГ M', что на любом входном слове w машина M останавливается тогда и только тогда, когда на этом входе останавливается машина M'.

4.3.2. Многоленточные машины Тьюринга

Одним из существенных ограничений машин Тьюринга является ограниченность движений управляющей головки на одном такте. Эта трудность частично преодолевается при использовании *многоленточных машин Тьюринга*.

У многоленточной МТ несколько лент, по каждой из которых движется своя головка. Программа такой машины описывает элементарные действия (движение головок, чтение/запись на ленты) в зависимости от состояния машины и содержимого ячеек, над которыми находятся головки.

4.3.2.1. Определения

Из сделанного выше описания ясно, какие изменения нужно сделать в формальных определениях.

Определение 4.39. *k-ленточная машина Тьюринга* — это набор $(A, Q, Q_f, \delta, q_0, \Lambda)$, где A, Q — конечные множества, $Q_f \subseteq Q$, $q_0 \in Q$, $\Lambda \in A$, $\delta\colon A^k \times Q \to A^k \times \{-1, 0, 1\}^k \times Q$ (функция, которая называется *таблицей переходов*).

Как видим, изменения касаются только формата таблицы переходов. Можно, конечно, дать более общее определение, в котором для разных лент используются разные алфавиты, но такое обобщение не представляет интереса, так как не меняет сколь-нибудь существенным образом возможности вычислительной модели.

Определение 4.40. Назовём *конфигурацией k-ленточной машины Тьюринга* $(A, Q, Q_f, \delta, q_0, \Lambda)$ набор из k слов $(w_1, \ldots, w_k)$ в алфавите $A \cup Q$, каждое из которых содержит ровно один символ из множества Q, причём этот символ одинаков для всех слов w_i, а первые и последние символы w_i не являются пустыми.

Начальная конфигурация k-ленточной MT имеет вид $(q_0u, q_0, \ldots, q_0)$, $u \in (A \setminus \{\Lambda\})^*$.

Таким образом, многоленточная машина Тьюринга начинает работу, имея на одной ленте входные данные, причём головка стоит над первым символом входа, а остальные ленты пусты.

Определения расширенной конфигурации, отображений на расширенных конфигурациях и на конфигурациях аналогичны 1-ленточному случаю. Например, для определения отображения на расширенных конфигурациях нужно повторить определение 4.8 для каждого слова w_i конфигурации $(w_1, \ldots, w_k)$, используя вместо $\delta(a, q)$ тройку (a_i, r_i, q'), где a_i, r_i — i-е компоненты в первых двух компонентах $\delta(a, q)$.

Результатом работы многоленточной МТ, которая остановилась в конфигурации $(w_1, \ldots, w_k)$, является слово w_1, из которого вычеркнут символ состояния машины, т. е. содержимое первой ленты в конце работы.

Определения функции, вычислимой на многоленточной машине и языка, распознаваемого многоленточной машиной, сохраняются буквально.

4.3.2.2. Моделирование k-ленточной машины на одноленточной

Теорема 4.41. *k-ленточные машины Тьюринга эквивалентны одноленточным.*

В одну сторону утверждение очевидно: ясно, что добавление лент и переопределение таблицы переходов так, чтобы она не зависела от содержимого добавленных лент, не изменит отображения, реализуемого машиной.

Набросок доказательства обратного утверждения приведём для случая двух лент, общий случай доказывается аналогично.

План доказательства таков. Пусть имеется 2-ленточная машина $M = (A, Q, Q_f, \delta, q_0, \Lambda)$. Одноленточная машина M', которая моделирует M, работает следующим образом. Состояние 2-ленточной машины кодируется на одной ленте так, чтобы в каждой ячейке были записаны данные о соответственных ячейках 2-ленточной машины (можно считать для определённости, что ячейки на лентах перенумерованы). Каждый такт работы двухленточной машины моделируется в два этапа. На первом M' просматривает все используемые ячейки и определяет, в каком состоянии находится моделируемая машина, и какие символы прочитаны её головками. На втором M' меняет содержимое ячеек в соответствии с таблицей перехода машины M. Перед началом этого двухэтапного процесса машина M' должна привести входные данные к тому виду, который используется при моделировании. После окончания работы M' должна преобразовать полученные данные к слову в алфавите A, отвечающему результату работы.

Опишем устройство M' подробнее. Алфавит машины M' — это множество

$$A' = A \cup (A \times (Q \cup \Lambda)) \times (A \times (Q \cup \Lambda)),$$

пустой символ тот же, что и у моделируемой машины, — Λ. Неформально нужно понимать это так, что

M' использует помимо алфавита A символы, которые являются записями, фиксирующими содержимое двух ячеек машины M, расположенных на двух лентах одинаковым образом. Помимо собственно содержимого (символа из алфавита A) машине M' потребуется информация о положении головок на лентах. Поэтому для записи содержимого ячейки используется не множество A, как можно было ожидать, а множество пар $A \times (Q \cup \Lambda)$, где вторая компонента указывает, что над данной ячейкой нет головки (в этом случае вторая компонента равна Λ), либо описывает состояние машины M (это происходит в той ячейке, над которой расположена головка).

Такое правило задаёт кодировку конфигураций машины M словами в алфавите

$$(A \times (Q \cup \Lambda)) \times (A \times (Q \cup \Lambda)).$$

Моделирующая машина M' является последовательным соединением трёх машин.

Первая машина M_1 подготавливает содержимое ленты к двухэтапному моделированию. Она просматривает ячейки по очереди, заменяя первый символ a_1 на $((a_1, q_0), (\Lambda, q_0))$, а каждый следующий символ входа a на $((a, \Lambda), (\Lambda, \Lambda))$ (напомним, что в начальном состоянии вторая лента пуста). Обнаружив пустой символ Λ, эта машина записывает на его место символ $((\Lambda, \Lambda), (\Lambda, \Lambda))$ и начинает двигаться налево до тех пор, пока не встретит символ вида $((a, q_0), (\Lambda, q_0))$, после чего переходит в финальное состояние.

Задача 4.9. Задайте таблицу переходов для описанной выше машины M_1. Сколько состояний вам потребуется?

Вторая машина M_2 используется в цикле. Её работа разбита на итерации. В начале каждой итерации мы

потребуем выполнения следующих условий: на ленте записана кодировка конфигурации (w_1, w_2) машины M, а головка M_2 находится над самой левой ячейкой этой кодировки. В конце итерации на ленте записана кодировка конфигурации M, которая получается из (w_1, w_2) за один такт работы M, а головка M_2 опять находится над самой левой ячейкой.

Множество состояний второй машины M_2 имеет вид

$$Q' = \{q_0\} \cup Q_f \cup (A \times A \times Q \times (\{L, R\} \cup S)).$$

Неформально первая компонента состояния, отличного от q_0, $q \in Q_f$, хранит данные о содержимом ячейки первой ленты, над которой находится головка, вторая компонента хранит аналогичные данные о второй ленте, а третья — о состоянии 2-ленточной МТ на данном такте её работы. Последняя компонента хранит информацию о направлении движения машины M_2.

Финальными для машины M_2 являются состояния вида (a, b, q, L), где $q \in Q_f$ — финальное состояние машины M и состояние q_f. Поскольку машина M_2 используется при соединении в цикле, укажем заранее, что условием повторения является завершение работы в состоянии q_f.

В состоянии q_0 или состоянии вида (a, b, q, R) машина M_2 движется направо, пока не находит символ Λ. В процессе движения она меняет состояние по следующему правилу: пусть текущий символ на ленте имеет вид $((a, q), (b, \Lambda))$, $q \neq \Lambda$, тогда машина меняет состояние q_0 на (a, Λ, q, R), в остальных случаях она изменяет состояние (a', c, q', R) на (a, c, q, R). Аналогично (с перестановкой компонент) изменяются состояния в случае, когда символ на ленте имеет вид $((b, \Lambda), (a, q))$, $q \neq \Lambda$. Если машина M_2 встречает

символ вида $((a, q), (b, q))$, $q \neq \Lambda$, она изменяет своё состояние на (a, b, q, R).

Обнаружив символ Λ, машина M_2 меняет своё состояние (a, b, q, R) на (a, b, q, L).

Описанные выше правила гарантируют, что если машина M_2 начинает работу на кодировке некоторой конфигурации моделируемой машины M, находясь в самой левой ячейке, и переходит в состояние (a, b, q, L), то в данной конфигурации состояние моделируемой машины M равно q, а под головками на лентах расположены символы a и b соответственно.

В состояниях вида (a, b, q, L), $q \notin Q_f$, машина M_2 движется налево до тех пор, пока не встретит пустой символ Λ, после чего она переходит в состояние q_f и сдвигается вправо.

Попутно машина M_2 меняет содержимое ленты так, чтобы оно кодировало конфигурацию машины M после выполнения такта работы. Для этого она выполняет следующие действия. Если прочитан символ вида $(p, (b, q))$, машина находится над той ячейкой, над которой расположена головка второй ленты моделируемой машины. Поэтому M_2 должна изменить вторые компоненты символов в соответствии с таблицей переходов моделируемой машины M. Аналогичные действия нужно выполнить с первыми компонентами соседних символов, если прочитан символ вида $((a, q), p)$.

При этом нужно учесть такие ситуации, когда обе головки находятся над одной и той же ячейкой (напомним, что мы пронумеровали ячейки обеих лент 2-ленточной машины), а также когда изменения требуют увеличения длины кодировки конфигурации.

Все эти действия M_2 выполняет в состояниях вида (a, b, q, s), где $s \in S$. Каждое из них осуществляется за

конечное число шагов. Выбирая для каждого из этих шагов и каждого из этих действий своё s, мы можем записать таблицу переходов для M_2, хотя и довольно громоздкую.

Если машина M_2 завершает работу в состоянии q_f, то в соединении машин её работа повторяется циклически.

Если машина M_2 завершает работу в финальном состоянии вида (a, b, q, L), то моделируемая машина M находится в финальном состоянии, и машина M_2 передаёт управление последней машине M_3 в соединении. Эта машина переписывает содержимое ленты обратным по отношению к машине M_1 образом. А именно, машина M_3 заменяет символ $((a, q), (b, q))$ на a, стирая информацию о состоянии и о содержимом второй ленты.

Замечание 4.42. Аналогично одноленточным распознающим МТ можно определить многоленточные распознающие машины. В качестве упражнения читателю предлагается доказать, что классы языков, распознаваемых одноленточными и многоленточными МТ, совпадают.

4.3.2.3. Использование многоленточных машин Тьюринга

Во многих случаях описание алгоритма с помощью многоленточных машин Тьюринга проще, чем с помощью одноленточных.

Задача 4.10. Постройте многоленточную машину Тьюринга, которая

а) копирует вход (т. е. результатом работы на входе w является слово ww);

б) переписывает вход в обратном порядке, т. е. результатом работы на входе $w_1 w_2 \dots w_n$ является $w_n w_{n-1} \dots \dots w_2 w_1$;

в) сравнивает числа, заданные в двоичной записи;

г) выполняет сложение чисел в двоичной системе счисления;

д) выполняет умножение чисел в двоичной системе счисления;

е) выполняет целочисленное деление чисел в двоичной системе счисления (т. е. находит частное и остаток при делении с остатком).

Чтобы сравнить возможности одноленточных и многоленточных машин, полезно решить предыдущую задачу и для одноленточных машин.

4.3.3. Модель RAM

Использование машин Тьюринга для записи алгоритмов не слишком удобно. В этом разделе мы опишем другую модель вычислений — машины с произвольным доступом к памяти (RAM). Эта аббревиатура означает Random Access Machine и указывает на главную особенность модели: возможность обращаться к ячейкам памяти по их адресу.

Основанный на модели RAM псевдокод часто используется для записи и анализа алгоритмов. (См. книги [1, 9].)

В отличие от машины Тьюринга, одна ячейка RAM хранит целое число. Ячейки RAM разделяются на три группы: входная лента, память, выходная лента. В каждой группе ячейки занумерованы неотрицательными целыми числами. Помимо этих ячеек

RAM включает в себя программу — список команд. Команды RAM имеют вид

⟨наименование команды⟩⟨тип операнда⟩⟨операнд⟩

Операнд — это целое число. Команды RAM включают в себя арифметические операции (второй операнд для арифметических операций всегда берётся из фиксированной ячейки памяти, которая имеет номер 0), операции чтения с входной ленты и записи на выходную ленту. Типов операндов три: константы, переменные и указатели, мы их будем обозначать $=$, λ (пустая строка), $*$ соответственно. Значение операнда зависит от типа. Значение константы совпадает с ней самой. Значение переменной i равно числу, записанному в ячейке памяти с номером i. Значение указателя i равно числу, записанному в ячейке с номером, который содержится в ячейке с номером i (если в i записано отрицательное число, значение указателя не определено).

Работа RAM начинается с исполнения первой команды. Все команды делятся на две группы: операции и переходы. После исполнения операции происходит выполнение следующей в списке команды. Исполнение перехода определяет номер команды, которая должна исполняться следующей.

Работа заканчивается в одном из следующих случаев: исполнена последняя команда программы и это операция; исполнена команда перехода, но полученное в результате её исполнения число не является номером команды; очередная команда не может быть исполнена (например, машина получает инструкцию разделить на 0).

В начале работы ячейки RAM пусты, кроме нескольких первых ячеек входной ленты. Поэтому работа RAM задаёт (частичное) отображение входов

(последовательностей целых чисел) на выходы (также последовательности целых чисел).

Точные определения работы RAM, результата работы и связанные с ними понятия функции, вычислимой на RAM, и языка, распознаваемого RAM, приводятся в следующем подразделе.

4.3.3.1. Определения

Состояние RAM — это четвёрка (n, x, r, y), где n — положительное целое число (счётчик команд), а x, r, y — бесконечные последовательности целых чисел (они задают состояния входной ленты, памяти и выходной ленты соответственно). Нам будет удобно считать, что последовательность — это функция из множества натуральных (т. е. целых неотрицательных) чисел в целые числа.

Уточним определение операнда и значения операнда. Операнд p — это пара (t, i), где i — целое число, а $t \in \{=, \lambda, *\}$. *Значение* $v(u)$ операнда определяется так: $v(=, i) = i$, $v(\lambda, i) = r(i)$, $v(*, i) = r(r(i))$. В последнем случае значение не определено при $r(i) < 0$.

Команда — это отображение на множестве состояний. При описании команд мы будем использовать следующие соглашения: состояние перед началом выполнения команды обозначается (n, x, r, y), после выполнения — n', x, r', y', если не указано, как изменяется элемент состояния, то он сохраняет прежнее значение. Входная лента не изменяется никогда.

Опишем вначале команды-операции. Для всех этих команд $n' = n + 1$.

Команда `store` u записывает $r(0)$ в ячейку с номером $v(u)$, т. е. $r'(v(u)) = r(0)$.

Команда `load` u записывает в нулевую ячейку содержимое ячейки с номером $v(u)$, т. е. $r'(0) = r(v(u))$.

Команды `add` u (сложение), `sub` u (вычитание), `mul` u (умножение), `div` u (целочисленное деление) выполняют арифметические действия с первой ячейкой памяти. Точные правила изменения состояния для этих команд такие: для `add` u: $r'(0) = r(0) + v(u)$, для `sub` u: $r'(0) = r(0) - v(u)$, для `mul` u: $r'(0) = r(0) - v(u)$, для `div` u: $r'(0) = \lfloor r(0)/v(u) \rfloor$.

Команда `read` u читает в нулевую ячейку число из входной ленты: $r'(0) = x(v(u))$.

Команда `write` u пишет содержимое нулевой ячейки на выходную ленту: $y'(v(u)) = r(0)$.

Теперь опишем команды-переходы. Они меняют только счётчик команд n. Для команды `goto` u: $n' = v(u)$; для команды `ifzero` u: если $r(0) = 0$, то $n' = v(u)$, иначе $n' = n + 1$; для команды `ifpos` u: если $r(0) > 0$, то $n' = v(u)$, иначе $n' = n + 1$.

Определение 4.43. RAM — это конечная последовательность команд $c_1, \ldots, c_m$ описанного выше вида.

Из определения ясно, что множество RAM счётно.

Работа RAM M — это последовательность состояний c_0, c_1, c_2, ... Начальное состояние $c_0 = (n_0, x, r, y)$ удовлетворяет следующим условиям: $n_0 = 1$, $r(k) = y(k) = 0$ для всех k. Состояние c_{k+1} получается из состояния $c_k = (n_k, x, r, y)$ применением команды n_k к состоянию c_k. Как уже говорилось выше, в некоторых случаях исполнение команды приводит к остановке. Таких случаев три: (1) после применения команды $n' < 1$ или $n' > m$; (2) значение операнда не определено; (3) невозможно выполнить операцию деления (для команды `div` u значение $v(u) = 0$). В таком случае работа RAM конечна.

Как и для всякой вычислительной модели, нужно определить отображение из множества входов в

множество результатов, вычисляемое RAM. Есть несколько способов определить такое отображение.

Например, определим класс функций вида $\mathbb{Z}^n \to \mathbb{Z}$, вычислимых на RAM, полагая входом конечную последовательность чисел $x_0, \ldots, x_{n-1}$, а результатом (в случае конечной работы RAM) значение $y(0)$ первой ячейки выходной ленты. При этом о начальном состоянии делается дополнительное предположение, что $x(k) = 0$ при $k \geqslant n$.

Более общее определение состоит в следующем. Входом для RAM объявляется конечная последовательность положительных чисел $a_1, \ldots, a_n$, о начальном состоянии делается предположение, что $x(k) = a_k$ при $k \leqslant n$ и $x(k) = 0$ при $k > n$. Результатом работы считается начальная последовательность значений на выходной ленте до первого неположительного числа.

4.3.3.2. Моделирование RAM на машинах Тьюринга

Отображение входов в результаты работы, задаваемое данной RAM, можно реализовать на машине Тьюринга. Опишем кратко, как это делается.

Для моделирования RAM на машине Тьюринга нужно кодировать состояние RAM и менять его в соответствии с работой RAM. Для этого удобно использовать многоленточные машины Тьюринга.

Кодировать целые числа мы будем их двоичной записью. Команд RAM фиксированное число, поэтому их можно хранить в «оперативной памяти» машины Тьюринга, как это описано выше в разделе 4.1.1.

Состояние группы ячеек RAM мы будем представлять в виде слова

$$\#a_1\#v_1\#\#a_2\#v_2\#\ldots\#a_n\#v_n\#,$$

где a_i — номер ячейки памяти, v_i — её содержимое. При этом предполагается, что в остальных ячейках записаны нули. Каждое из таких слов (коды входной и выходной ленты, а также памяти) будем записывать на отдельную ленту моделирующей МТ.

Так как RAM начинает работу, имея ненулевые значения лишь в конечном числе ячеек, то после любого числа шагов ненулевые значения будут лишь в конечном числе ячеек. Поэтому описанный выше способ позволяет корректно кодировать состояние RAM.

Для выполнения команд необходимо уметь определять значения операндов, выполнять с ними арифметические действия и записывать полученные значения в ту или иную ячейку RAM. Кроме того, нужно уметь выполнять команды-переходы. Нужно уметь сравнивать число с нулём и менять подходящим образом номер исполняемой команды.

Задача 4.11. Постройте машины Тьюринга, реализующие описанные операции.

4.3.4. Машины Минского

Необычайно простую универсальную модель вычислений предложил Марвин Минский [20, 21]. *Машина Минского* снабжена конечным числом регистров $R_1, \ldots, R_n$, каждый из которых содержит неотрицательное целое число. Договоримся обозначать через v_i содержимое регистра с номером i. *Программа для машины Минского* состоит из конечного нумерованного списка команд. Команды бывают трёх следующих типов:

- `inc` i, j увеличить значение регистра i на 1 и передать управление команде с номером j;

- `dec` i, j, k если значение регистра i равно 0, то передать управление команде с номером j, в противном случае уменьшить на 1 значение регистра i, а управление передать команде с номером k;
- `halt` остановиться.

В начальной конфигурации значения всех регистров, кроме первого, равны 0. Результатом работы является значение первого регистра после остановки.

Как это ни удивительно, но такая вычислительная модель универсальна.

Выше была доказана теорема 4.38, которая фактически утверждает универсальность машин Тьюринга в двухбуквенном алфавите. Так что достаточно показать, что на машинах Минского можно моделировать работу машины Тьюринга в алфавите $\{0, 1\}$.

Приведём несколько примеров программ для машин Минского, они потребуются в дальнейшем для моделирования работы машин Тьюринга. Рассматривайте эти примеры и как задачи — докажите сделанные в них утверждения.

Пример 4.44 (*Обнуление регистра*). Рассмотрим такую программу:

1: `dec` $i, n, 1$

Она уменьшает значение регистра i до нуля, после чего передаёт управление команде с номером n.

Пример 4.45 (*Копирование регистра*). Следующая программа

1: `dec` $i, n, 2$

2: `inc` $j, 3$

3: `inc` $k, 1$

увеличивает содержимое регистров j, k на содержимое регистра i, а после этого передаёт управление команде с номером n. Содержимое регистра i в конце работы программы равно 0. Если перед началом выполнения этой программы регистры j, k были пусты (содержали нули), а регистр i содержал число v_i, то после выполнения этой программы уже два регистра будут содержать значение v_i. Поэтому мы называем эту программу копированием регистра.

Упрощённый вариант этой программы

1: `dec` $i, n, 2$

2: `inc` $j, 1$

переносит значение регистра i в регистр j (если последний пуст перед началом выполнения программы).

Комбинируя эти две программы, можно написать программу, которая копирует значение регистра i в регистр j, сохраняя в конце работы значение в регистре i. Для этого требуется один вспомогательный регистр.

Пример 4.46 (*Проверка чётности*). Программа

1: `dec` $i, n, 2$

2: `dec` $i, m, 1$

уменьшает значение регистра i до 0 и передаёт управление команде с номером n, если в начале работы регистр i содержал чётное число, в противном случае управление передаётся команде с номером m.

Задача 4.12. Напишите программы для машины Минского, которые решают следующие задачи:

(a) удвоение регистра i (в начале работы значение регистра равно v, в конце $2v$);

(б) нахождение остатка от деления на 2 (в начале работы значение регистра равно v, в конце $v \bmod 2$);

(в) нахождение частного от деления на 2 (в начале работы значение регистра равно v, в конце $\lfloor v/2 \rfloor$).

Используйте в пунктах (а) и (в) не более двух вспомогательных регистров. Для пункта (б) можно составить программу без вспомогательных регистров.

Описанных в этих примерах возможностей машин Минского уже достаточно для моделирования работы произвольной машины Тьюринга в алфавите $\{0,1\}$. Напомним, что символ 0 мы считаем пустым. Поэтому любой конфигурации такой машины Тьюринга uqv, $u, v \in \{0,1\}^*$, можно сопоставить состояние q и два числа ℓ и r, двоичные записи которых имеют вид u, v^R соответственно (v^R обозначает слово, записанное в обратном порядке).

На каждом такте машина Тьюринга меняет символ, над которым находится головка (младший бит числа r), и сдвигает головку на одну позицию влево или вправо. Изменение символа меняет число r на $r-1$ (если 1 заменяется на 0) или на $r+1$ (если 0 меняется на 1). Возможные изменения чисел ℓ и r зависят от направления движения и чётности чисел ℓ и r. Например, если оба числа чётные, а сдвиг происходит вправо, то ℓ увеличится в два раза (к двоичной записи добавляется 0 в конце), а r уменьшится в два раза. Если ℓ чётное, а r нечётное и головка сдвигается вправо, то ℓ заменяется на $2\ell + 1$, а r — на $(r-1)/2$.

Задача 4.13. Составьте таблицу изменений чисел ℓ и r, кодирующих состояние ленты машины Тьюринга,

для всех возможных вариантов чётности чисел и направления движения (всего будет, как нетрудно понять, 8 различных вариантов).

Задача 4.14. Напишите программы для машин Минского, которые меняют значения ℓ и r по правилам, найденным в предыдущей задаче. Используйте при этом не более двух вспомогательных регистров.

После решения этих задач легко построить машину Минского, которая моделирует работу некоторой машины Тьюринга в алфавите $\{0, 1\}$. У этой машины два основных регистра (которые хранят числа ℓ и r) и два вспомогательных регистра.

Команды машины Минского разбиваются на группы, которые соответствуют состояниям машины Тьюринга. Каждая группа начинается с команд, копирующих во вспомогательный регистр число r и определяющих его чётность, т. е. символ под головкой моделируемой машины. Далее меняется значение r (или не меняется — в зависимости от текущего состояния и символа); после чего выполняется одна из найденных в решении задачи 4.14 программ. Далее управление передаётся начальной команде той группы, которая отвечает новому состоянию моделируемой машины Тьюринга.

Замечание 4.47. Машины Минского являются фактически ограниченным вариантом RAM. Фрагменты RAM-программ, приведённые в таблице 4.1 реализуют команды машины Минского.

Здесь мы предполагаем, что регистры машины Минского — это ячейки памяти RAM. Нетрудно дополнить эти фрагменты командами чтения входа и записи выхода и получить моделирующую машину Минского RAM-программу.

Таблица 4.1. Фрагменты RAM-программы

для `inc` i, j		для `dec` i, j, k	
s:	`load` i	t:	`load` i
$s+1$:	`add` $= 1$	$t+1$:	`ifzero` $= j$
$s+2$:	`store` i	$t+2$:	`add` $= -1$
$s+3$:	`goto` $= j$	$t+3$:	`store` i
		$t+4$:	`goto` $= k$

Отсюда следуют два важных вывода. Во-первых, машины Минского моделируются на машинах Тьюринга с помощью описанного в разделе 4.3.3.2 моделирования RAM на машине Тьюринга. Во-вторых, из доказанной выше универсальности машин Минского следует универсальность RAM.

4.3.5. Нормальные алгорифмы Маркова

Нормальные алгорифмы, как и машины Тьюринга, работают со словами в некотором алфавите[5)]. Но действия нормального алгорифма напоминают преобразования слов в ассоциативном исчислении: они заключаются в заменах одного подслова другим. В отличие от ассоциативного исчисления порядок выполнения замен в нормальном алгорифме однозначно определён.

Дадим формальные определения.

Определение 4.48. Пусть A — конечный алфавит.

Нормальным алгорифмом в алфавите A (для краткости, НА) называется конечная последовательность

[5)] Понятие нормального алгорифма введено А. А. Марковым, который называл алгоритмы алгорифмами.

правил подстановки. Правило подстановки имеет вид $u \to v$ или $u \to \cdot v$, где u, v — слова в алфавите A. Правила подстановки вида $u \to \cdot v$ называются *заключительными*.

Нормальным алгорифмом над алфавитом A называется нормальный алгорифм в некотором алфавите B, содержащем алфавит A.

Последовательность правил подстановки называется также *схемой алгорифма*.

Определение 4.49. *Результат* применения правила подстановки $u \to v$ (или $u \to \cdot v$) к слову w определяется так: пусть $w = w_1 u w_2$, причём $w_1 u$ содержит единственное вхождение слова u; тогда результатом является слово $w_1 v w_2$. Если слово w не содержит вхождений слова u, то в таком случае слово w *не поддаётся* правилу $u \to v$.

Другими словами, применение правила подстановки состоит в замене самого левого вхождения слова u на слово v. Выбор самого левого вхождения делает результат применения однозначным, если слово вообще поддаётся данному правилу.

Пример 4.50. Результатом применения правила $ab \to ba$ к слову $aabab$ является слово $abaab$.

Результатом применения правила $\lambda \to a$ к слову b является слово ab. Напомним, что через λ мы обозначаем пустое слово. Самое левое вхождение пустого слова находится, конечно же, перед первой позицией в слове. Поэтому и получается слово ab.

Определение 4.51. *Результат* применения схемы алгорифма к слову w определяется как результат применения к слову w первого в списке правила подстановки, которому поддаётся слово w.

Если такого правила подстановки нет, то результат не определён, а мы говорим, что слово *не поддаётся* схеме алгорифма.

Выбор первой по порядку применимого правила делает результат применения схемы алгорифма однозначным, если слово поддаётся схеме алгоритма.

Пример 4.52. Применим схему

$$\begin{cases} a \to aa \\ aa \to \lambda \end{cases}$$

к слову bab. Оно поддаётся первому же правилу, и результатом будет слово $baab$.

Применим схему

$$\begin{cases} aa \to \lambda \\ a \to aa \end{cases}$$

к слову bab. Первому правилу это слово не поддаётся, так как в слове bab нет вхождений слова aa. Но оно поддаётся второму правилу, и результатом опять будет слово $baab$.

Результаты применения двух схем, различающихся порядком правил подстановки, в данном случае совпали. В общем случае результаты будут, конечно же, различными. Например, применение первой схемы к слову $baab$ даёт результат $baaab$, а применение второй — результат bb.

Применение схемы алгорифма $\mathcal{A}$ задаёт частично определённое отображение $\mathcal{A}\colon A^* \to A^*$ на множестве слов в алфавите A. Работа НА состоит в итерациях этого отображения с уточнением, касающимся заключительных правил.

Определение 4.53. *Протокол работы НА* со схемой $\mathcal{A}$ — это такая последовательность слов

$$w_0, w_1, \ldots, w_n, \ldots, \tag{4.25}$$

что $\mathcal{A}(w_i) = w_{i+1}$, если $i+1$ не превосходит длину последовательности. Если последовательность конечна, то либо последнее слово w_n не поддаётся схеме $\mathcal{A}$, либо к слову w_{n-1} применяется заключительное правило.

Неформально говоря, если правило помечено как заключительное, то это является указанием для алгорифма прекратить работу после применения этого правила.

Определение 4.54. *Результатом работы НА* со схемой $\mathcal{A}$ является последнее слово в протоколе работы, если он конечен. В противном случае результат работы не определён.

Будем обозначать результат работы алгорифма со схемой $\mathcal{A}$ на слове w через $\mathcal{A}^*(w)$.

Итак, НА в алфавите A определяет некоторое (частично определённое) отображение на множестве слов в алфавите A. Это отображение не совсем то, что нужно называть вычислимым по Маркову отображением. Как мы уже видели в случае МТ, часто бывает полезно расширить алфавит.

Определение 4.55. *Алгорифм* $\mathcal{A}$ над алфавитом A *вычисляет* частично определённое отображение $f\colon A^* \to A^*$, если для любого слова w в алфавите A выполнено равенство $f(w) = \mathcal{A}^*(w)$ в расширенном смысле (если хотя бы одна часть равенства определена, то определена и другая часть равенства, причём обе части совпадают).

Отображения, вычислимые НА, называются *вычислимыми по Маркову*.

Тонкости этого определения проще понять на следующем примере.

Таблица 4.2. Схема алгорифма для отображения (4.26)

$$\begin{cases} q11 \to 11q \\ q1 \to \cdot 11 \\ q \to \cdot 11 \\ \lambda \to q \end{cases}$$

Задача 4.15. Докажите, что не существует НА в алфавите $\{1\}$, результат работы которого определяется как

$$\mathcal{A}^*(w) = \begin{cases} w11, & \text{если длина } w \text{ чётна,} \\ w1, & \text{если длина } w \text{ нечётна.} \end{cases} \tag{4.26}$$

Указание: проверьте, что всякое правило в схеме такого алгорифма должно быть заключительным.

Тем не менее, отображение (4.26) вычислимо по Маркову. В таблице 4.2 приведена схема $\mathcal{E}$ алгорифма над алфавитом $\{1\}$, который вычисляет это отображение. Как видно из описания, это алгорифм в алфавите $\{1, q\}$.

Рассмотрим применение $\mathcal{E}$ к слову $w \in \{1\}^*$.

На первом шаге применяется четвёртое правило подстановки, так как символ q не входит в слово w.

Количество символов q в слове не изменяется при применении первых трёх правил подстановки. Поэтому на последующих шагах алгоритма до тех пор, пока применяются только эти подстановки, слова в протоколе вычисления будут иметь вид $1^s q 1^r$.

Заметим, что пока применяется первое правило, количество единиц в слове также не изменяется, а номер позиции, на которой стоит символ q,

увеличивается. Первое правило неприменимо к словам вида $1^k q 1$ или $1^k q$. В первом случае будет применено второе (заключительное) правило подстановки и алгоритм остановится, увеличив количество единиц по сравнению с входным словом на одну. Аналогично, во втором случае алгоритм останавливается и увеличивает количество единиц на 2 (применяется третье правило подстановки).

Осталось заметить, что два варианта применения заключительных правил из схемы $\mathcal{E}$ соответствуют нечётной или чётной длине входного слова.

Разобранный пример показывает путь к доказательству следующей леммы.

Лемма 4.56. *Для любой машины Тьюринга*

$$M = (A, Q, Q_f, \delta, q_0, \Lambda)$$

существует такой нормальный алгорифм Маркова $\mathcal{A}$ над алфавитом $A \cup Q$, результат работы которого на любом слове $w \in A^*$ совпадает с результатом работы M на w.

Доказательство. При анализе проблемы достижимости в ассоциативном исчислении (раздел 4.2.2) мы уже разобрали подробно изменения конфигураций МТ за один шаг работы.

Искомый алгорифм Маркова содержит все правила подстановки, отвечающие правилам (4.3–4.10). Как было показано при анализе этих правил, если состояние МТ M не является финальным, то к конфигурации МТ применимо ровно одно из них. Поэтому порядок записи правил в данном случае неважен.

Обозначим соответствующую часть схемы алгорифма через $\mathcal{A}_0$.

Перед $\mathcal{A}_0$ в схеме алгорифма записаны правила подстановки

$$\Lambda_1 a \to a\Lambda_1, \qquad a \in A, \tag{4.27}$$

$$\Lambda_1 \to \Lambda_0, \tag{4.28}$$

а после $\mathcal{A}_0$ — правила подстановки

$$\Lambda_0\Lambda \to \Lambda_0, \tag{4.29}$$

$$\Lambda_0 a \to a, \qquad a \in (Q \cup A) \setminus \{\Lambda\}, \tag{4.30}$$

$$\Lambda\Lambda_0 \to \Lambda_0, \tag{4.31}$$

$$a\Lambda_0 \to a, \qquad a \in (Q \cup A) \setminus \{\Lambda\}, \tag{4.32}$$

$$q \to \cdot\lambda, \qquad q \in Q_f, \tag{4.33}$$

$$\lambda \to \Lambda_0 q_0 \Lambda_1. \tag{4.34}$$

В приведённом описании предполагается, что вспомогательные символы Λ_0, Λ_1 не входят ни в алфавит МТ, ни в её множество состояний.

Рассмотрим применение построенной схемы алгорифма к некоторому слову w в алфавите A. На первом шаге слово поддаётся лишь последнему правилу, и результатом применения этого правила будет слово $\Lambda_0 q_0 \Lambda_1 w$. Такое слово поддаётся первому правилу подстановки (4.27). Легко видеть, что первое правило будет применяться до тех пор, пока не получится слово $\Lambda_0 q_0 w \Lambda_1$. Это слово поддаётся второму правилу (4.28).

Результатом применения второго правила подстановки будет слово $\Lambda_0 q_0 w \Lambda_0$. Этому слово не поддаётся первым двум правилам подстановки, зато к нему применимо одно из правил $\mathcal{A}_0$. Правила из $\mathcal{A}_0$ будут применяться до тех пор, пока символ состояния q в текущем слове не является финальным.

Рассмотрим заключительную часть работы алгорифма, когда текущее слово имеет вид

$$\Lambda_0 \Lambda^k u_1 q_f u_2 \Lambda^s \Lambda_0.$$

Заметим, что по определению u_1u_2 — результат работы МТ M на входном слове w.

Слово указанного вида не поддаётся правилам из $\mathcal{A}_0$. Поэтому будет применяться правило (4.29) до тех пор, пока не будут стёрты все k пустых символов в левой части слова. Затем применение правила (4.30) сотрёт первое вхождение символа Λ_0.

Далее аналогичным образом после применения правил (4.31) и (4.32) будут стёрты последние s пустых символов и символ Λ_0.

Первое правило, которое применимо к получившемуся слову $u_1q_fu_2$, — это одно из заключительных правил (4.33). Оно стирает символ финального состояния.

Итак, результат работы алгоритма — слово u_1u_2 совпадает с результатом работы МТ M. □

Мы проверили, что вычислимые по Тьюрингу функции являются вычислимыми по Маркову. Верно и обратное.

Теорема 4.57. *Вычислимость по Маркову равносильна вычислимости по Тьюрингу.*

Доказательство. В силу леммы 4.56 осталось доказать, что вычислимая по Маркову функция может быть вычислена на некоторой машине Тьюринга.

Искомая МТ получается циклическим соединением машин, проверяющих вхождение некоторого слова u в слово w и последующей замене на слово v. Здесь u, v — два фиксированных слова.

Дадим описание машин, которые осуществляют поиск и замену. Описание соединения таких машин, которое обеспечивает моделирование работы НА,

мы оставляем читателю в качестве самостоятельного упражнения.

Машина M_u начинает работу, имея головку над самым левым символом слова. Множество её состояний Q_u индексировано префиксами слова u:

$$Q_u = \{q_x : x \text{ — префикс слова } u\}.$$

Начальное состояние — q_λ, оно индексировано пустым префиксом λ. Поэтому в начальном состоянии выполняется условие «машина находится в состоянии q_x, а слева от символа под головкой записаны символы префикса x». Таблица переходов устроена так, что это свойство выполняется при работе МТ до тех пор, пока не найдено вхождение слова u или не достигнут конец слова. Этого можно добиться, если команды перехода имеют вид

$$\begin{aligned} &\delta\colon (q_x, a) \mapsto (q_{xa}, 1, a) && \text{если } xa \text{ — префикс } u,\\ &\delta\colon (q_x, a) \mapsto (q_y, 1, a) && \text{иначе.} \end{aligned}$$

Здесь y — это наибольший суффикс слова xa, который является префиксом слова u. Выбор наибольшего суффикса гарантирует, что вхождение слова u не будет пропущено.

Если машина M_u читает пустой символ (слово просмотрено до конца), она переходит в финальное состояние q_{no}. Ещё одним финальным состоянием является состояние q_u (найдено самое левое вхождение слова u).

Через состояние q_u к машине M_u присоединена машина N_v, которая заменяет найденное вхождение u на v.

Такую машину проще описать, используя две ленты. На первом этапе работы машина N_v переносит на вторую ленту содержимое первой ленты от текущего

символа вправо, заменяя на первой ленте перемещаемые символы пустыми.

На втором этапе машина N_v двигается по первой ленте влево до непустого символа, после чего стирает $|u|$ символов, продолжая движение влево.

На третьем этапе машина движется вправо и записывает слово v на первую ленту, начиная с текущего положения головки. На последнем этапе машина N_v двигается вправо по первой ленте и возвращает на неё содержимое второй ленты. □

4.4. Универсальный алгоритм

Под универсальным алгоритмом мы понимаем такую машину Тьюринга, которая моделирует работу любой другой машины Тьюринга.

Входом такой *универсальной машины Тьюринга* является описание машины Тьюринга и её входа. Универсальная машина преобразует описание входа в описание состояния ленты после первого такта работы моделируемой машины, затем процесс повторяется. Результатом работы универсальной машины является описание результата работы моделируемой машины.

Моделирование работы одноленточной машины проще реализовать, используя трёхленточную машину и равномерное кодирование символов алфавита и состояний моделируемой машины. Последнее означает, что длина кода каждого символа алфавита одинакова.

Приведём пример такой кодировки. Пусть в алфавите $k < 2^n$ символов. Тогда кодом символа мы будем считать слово $\#w\#$, где w — двоичная запись числа k, дополненная слева нулями так, чтобы длина w равнялась k.

Для удобства моделирования мы также изменим формат описания МТ, предложенный на с. 175. А именно, начальному состоянию и пустому символу присвоим максимальные номера в перечислении соответственно состояний и символов алфавита, а символы движения закодируем равномерно тем же способом, что описан выше. Будем считать описанием МТ следующее слово:

$$\langle\text{код начального состояния}\rangle\langle\text{код пустого символа}\rangle f\langle\text{список кодов финальных состояний}\rangle t\langle\text{код таблицы переходов}\rangle e$$

В этом слове используются дополнительные символы e, f, t. Код таблицы переходов состоит из списка пятёрок вида

$$\langle q_i\rangle\langle a_j\rangle\langle a_{ij}\rangle\langle d_{ij}\rangle\langle q_{ij}\rangle$$

которые кодируют пары (образ – значение).

Описанием МТ и её входа будем считать конкатенацию описания МТ в указанном выше формате и посимвольного кода входного слова.

Теорема 4.58. *Существует такая трёхленточная машина Тьюринга U, которая по входу $\langle M, w\rangle$, являющемуся описанием машины Тьюринга M и её входа w выдаёт результат $\langle u\rangle$, который является описанием результата работы M на входе w. Если M на w не останавливается, то U на $\langle M, w\rangle$ также не останавливается.*

Следствие 4.59. *Существует одноленточная универсальная машина Тьюринга.*

Это комбинация теорем 4.58 и 4.41 (моделирование многоленточных машин Тьюринга на одноленточных).

Следствие 4.60. *Существует универсальная машина Тьюринга в двухбуквенном алфавите.*

Действительно, по теореме 4.38 всякую МТ можно моделировать на МТ в двухбуквенном алфавите.

Теперь обсудим доказательство теоремы 4.58. Мы опишем один из возможных способов построения универсальной машины.

Наша машина поддерживает на одной ленте (входной) описание состояния ленты моделируемой машины в процессе работы, на второй ленте (лента описания) хранит описание моделируемой машины Тьюринга, на третьей (лента состояния) — описание текущего состояния моделируемой машины.

Она начинает работу, приводя начальную конфигурацию к указанному виду. Для этого нужно скопировать на вторую ленту описание моделируемой машины (т. е. ту часть входа, которая заканчивается символом e), одновременно стирая это описание с первой ленты (заменяя все символы на пустые). Дополнительно потребуем, чтобы головка входной ленты после такого копирования оказалась над первым символом кода первого символа входа моделируемой машины. Если вход пуст, то запишем на входную ленту код пустого символа, скопировав его из описания МТ (второй символ описания).

После этого нужно скопировать код начального состояния, с которого начинается описание МТ, на третью ленту и перевести головку второй ленты на символ f (начало списка финальных состояний), а головку третьей ленты — влево до символа # (начало описания текущего состояния).

Итак, перед началом моделирования такта работы на входной ленте головка стоит над первым символом

кода текущего символа на ленте моделируемой машины, на ленте описания головка стоит над символом f, на ленте состояния — над первым символом описания текущего состояния моделируемой машины.

Моделирование такта работы состоит в выполнении следующих шагов:

(а) Проверить, является ли текущее состояние моделируемой машины финальным. Для этого оно сравнивается с элементами списка финальных состояний, который является частью описания машины Тьюринга. Если текущее состояние финальное, то закончить работу, в противном случае выполнить следующие шаги моделирования.

(б) Найти пятёрку в таблице значений, которая соответствует текущему состоянию и символу моделируемой машины. Для каждой пятёрки проверка сводится к проверке равенства двух первых слов пятёрки со словом на входной ленте и ленте состояния соответственно. Если искомой пятёрки не найдено, машина останавливается. В противном случае она изменяет:

(в) слово на ленте состояния на то слово, которое записано на пятом месте в пятёрке;

(г) код текущего символа на входной ленте на то слово, которое записано на третьем месте в описании таблицы переходов;

(д) положение головок в следующей последовательности

- головка на входной ленте перемещается в соответствии с движением, указанным четвёртым словом в пятёрке (здесь нужно также учесть, что мы кодируем только часть

ленты моделируемой машины и при необходимости пополнить входную ленту описанием пустого символа);

- головка на ленте состояния переводится на первый символ кода текущего состояния;
- головка на ленте описания возвращается в положение перед началом моделирования такта работы (т. е. вновь оказывается над символом f).

Задача 4.16. Составьте машины Тьюринга, выполняющие описанные выше действия, и укажите способ их соединения в универсальную МТ.

Приведём простой пример использования универсальной машины Тьюринга в доказательствах алгоритмической неразрешимости. Частным случаем проблемы остановки является проблема остановки МТ на пустом входе.

Определение 4.61. Обозначим через stop_Λ язык в алфавите $\{0,1\}^*$, состоящий из тех описаний $\langle M\rangle$ машины Тьюринга M, для которых M останавливается на пустом входе.

Теорема 4.62. *Язык* stop_Λ *неразрешим.*

Доказательство. Мы опять применим тот же приём, что и в доказательстве алгоритмической неразрешимости проблемы достижимости. А именно, в предположении существования алгоритма, разрешающего язык stop_Λ мы построим алгоритм, разрешающий язык stop.

Нам потребуется сводящий алгоритм, преобразующий описание машины Тьюринга M и её входа w в

описание такой машины Тьюринга M_w, которая останавливается на пустом входе тогда и только тогда, когда M останавливается на w.

Используя сводящий алгоритм и алгоритм, разрешающий язык stop_Λ, построим алгоритм A', разрешающий язык stop. Этот алгоритм применяет ко входу $\langle M, w\rangle$ сводящий алгоритм и получает описание МТ M_w. К описанию M_w алгоритм A' применяет алгоритм, разрешающий язык stop_Λ, и его результат работы является результатом работы алгоритма A'.

Корректность алгоритма A' сразу следует из определения сводящего алгоритма.

Осталось описать сводимость.

Вначале построим МТ $M'_{M,w}$, которая дописывает к своему входу описание $\langle M, w\rangle$. Искомая машина M_w получается последовательным соединением машины $M'_{M,w}$ и универсальной машины Тьюринга. □

Задача 4.17. Дайте явное описание машины $M'_{M,w}$.

Замечание 4.63. Разрешимость проблемы остановки на пустом входе зависит от способа описания МТ. Например, занумеруем машины, останавливающиеся на пустом входе, чётными числами, а машины, не останавливающиеся на пустом входе — нечётными. Проблема остановки на пустом входе в этом случае состоит в определении чётности числа и потому разрешима.

4.5. Неразрешимость нетривиальных свойств вычислимых функций

Вычислимые функции можно описывать, приводя описания вычисляющих их МТ. Это описание конструктивно. Однако в некотором смысле, который

ниже уточняется, оно «менее конструктивно», чем описание натуральных чисел в позиционной системе счисления. В частности, оно «бесполезно» — из него нельзя алгоритмически извлечь никаких полезных сведений о функции.

Для уточнения сделанных утверждений обратим внимание на то, что по двум двоичным записям натуральных чисел легко понять, указывают ли они на одно и то же число. «Легко» здесь означает, что существует алгоритм проверки равенства.

Можно привести много примеров унарных предикатов на множестве натуральных чисел (другими словами, свойств натуральных чисел), которые являются вычислимыми, если числа заданы двоичной записью: чётность, делимость на заданное натуральное число, простота и т. д.

С вычислимыми функциями дело обстоит совершенно иначе. Список свойств вычислимых функций, которые можно алгоритмически проверить, весьма краток. Он содержит лишь *тривиальные* свойства.

Теорема 4.64. *Пусть $\mathcal{A}$ — некоторое нетривиальное семейство вычислимых функций, т. е. существуют вычислимые функции f_1, f_2 такие, что $f_1 \in \mathcal{A}$, $f_2 \notin \mathcal{A}$. Тогда алгоритмически неразрешима следующая проблема: по описанию МТ M решить, принадлежит ли вычисляемая машиной M функция семейству $\mathcal{A}$.*

Доказательство. Предположим, что для нетривиального семейства $\mathcal{A}$ вычислимых функций существует алгоритм, который проверяет принадлежность этому семейству функции, заданной описанием вычисляющей её МТ. Пусть f_0 — нигде не определённая вычислимая функция. Предположим, что $f_0 \notin \mathcal{A}$.

Поскольку семейство функций $\mathcal{A}$ нетривиально, найдётся принадлежащая ему функция f_1. Обозначим МТ, которая вычисляет функцию f_1, через M_1.

Опираясь на сделанные предположения, построим алгоритм, который проверяет, что машина M останавливается на входе w. Алгоритм проверяет принадлежность семейству $\mathcal{A}$ функции f, задаваемой машиной Тьюринга M_f. Машина M_f является последовательным соединением машин M_0 и M_1. Машина M_0 «запоминает» вход, после чего стирает его и имитирует работу машины M на пустом входе. После завершения моделирования работы M машина M_0 «восстанавливает» входное слово и останавливается. Таким образом, если машина M останавливается на слове w, то M_f вычисляет функцию f_1. В противном случае M_f не останавливается ни на одном входе.

Описание машины M_f строится по описанию M алгоритмически. Операции «запоминания» и «восстановления» входа легко реализовать на двухленточной машине, которая по теореме 4.41 эквивалентна некоторой одноленточной. Моделирование работы машины M по её описанию осуществляется универсальной машиной Тьюринга.

Корректность полученного алгоритма ясна из построения: если M останавливается на входе w, то f совпадает с f_1, т. е. $f \in \mathcal{A}$. Если же M не останавливается на входе w, то функция f нигде не определена, поэтому она совпадает с f_0, т. е. $f \notin \mathcal{A}$.

Мы рассмотрели случай, когда $f_0 \notin \mathcal{A}$. Если $f_0 \in \mathcal{A}$, то построения аналогичны. Нужно выбрать функцию $f_1 \notin \mathcal{A}$ и построить ту же самую машину M_f. Ответ алгоритма проверки остановки противоположен ответу алгоритма проверки принадлежности f семейству $\mathcal{A}$. □

Использование описаний машин Тьюринга для указания на вычислимую функцию — не единственный способ описаний вычислимых функций. Приведённую выше теорему можно обобщить на многие такие способы (включающие в себя, в частности, описания функций с помощью языков программирования). Такое обобщение называется *теоремой Успенского – Райса*. Мы не будем формулировать эту теорему во всей общности, заметим лишь, что указанный выше эффект сохраняется — при «разумном» задании вычислимых функций проверка любого их нетривиального свойства алгоритмически неразрешима.

4.6. Алгоритмы и логика

В этом заключительном разделе мы обсудим связи между логикой и теорией алгоритмов. Изложение будет более кратким, чем в остальных разделах. Многие важные факты оставлены в качестве задач для самостоятельного решения.

На основе теории алгоритмов можно уточнить требования к формальным системам, которые описывают математическое рассуждение. Интуитивно ясное требование состоит в том, что множество аксиом и все правила вывода должны быть разрешимы. В самом деле, мы хотим разбить математическое рассуждение на элементарные шаги. Но тогда каждый такой шаг должен быть достаточно прост, чтобы допускать механическую проверку.

Из разрешимости множества формул, аксиом и правил вывода следует разрешимость множества всех выводов в данной формальной системе.

Задача 4.18. Дайте строгую формулировку предыдущего утверждения и докажите его.

Однако множество формальных теорем формальной системы с разрешимыми аксиомами и правилами вывода может оказаться неразрешимым.

4.6.1. Неразрешимость проверки общезначимости формул

Теорема 4.65. *Множество общезначимых формул неразрешимо.*

Конечно, такая краткая формулировка требует уточнения — формулы это слова в бесконечном алфавите, а мы определяли разрешимость множеств лишь для слов в конечном алфавите. Поэтому нужно зафиксировать способ описания формул в конечном алфавите.

Мы сделаем это в два шага. Сначала сопоставим формулам конечные последовательности натуральных чисел. Для этого занумеруем символы в алфавите исчисления предикатов следующим образом:

$$\begin{array}{rccccc} \text{символ} & \forall, & \neg, & \to, & (, &) \\ \text{номер} & 0, & 1, & 2, & 3, & 4, \end{array} \tag{4.35}$$

символу переменной x_i сопоставим номер $5(i+1)$ $(i \geqslant 0)$, предикатному символу P_i^k (i-й символ арности k) сопоставим номер $5(\binom{k+i+1}{2}+i+1)+1$, функциональному символу f_i^k (i-й символ арности k) сопоставим номер $5(\binom{k+i+1}{2}+i+1)+2$. Странные формулы в этой нумерации имеют простое объяснение, которое мы сформулируем в виде задачи.

Задача 4.19. Докажите, что отображение

$$(x,y) \mapsto \binom{x+y+1}{2} + y$$

является биекцией $\mathbb{N}\times\mathbb{N}$ на $\mathbb{N}$. Постройте алгоритмы, вычисляющее это отображение и обратное к нему.

Итак, мы сопоставляем формуле в исчислении предикатов (конечная последовательность символов в алфавите исчисления предикатов) конечную последовательность номеров символов.

Второй шаг состоит в том, чтобы закодировать конечные последовательности натуральных чисел словами в некотором конечном алфавите. Для этого будем использовать уже рассмотренную выше кодировку. Последовательность v_1, v_2, …, v_n будем кодировать словом

$$\#\bar{v}_1\#\#\bar{v}_2\#\ldots\#\bar{v}_n\#$$

в алфавите $\{\#, 0, 1\}$. Здесь $\bar{v}$ обозначает двоичную запись числа v.

Теперь можно дать более точную формулировку теоремы 4.65: множество кодов общезначимых формул неразрешимо.

Доказательство теоремы 4.65. Сведём проблему равенства слов в полугруппах к проверке общезначимости формул в языке первого порядка.

Рассмотрим набор правил преобразования слов

$$L_1 \leftrightarrow R_1,$$
$$L_2 \leftrightarrow R_2,$$
$$\ldots$$
$$L_n \leftrightarrow R_n$$

в алфавите A и два слова U, V.

Напомним, что проблема равенства слов заключается в том, чтобы решить, эквивалентны ли слова U и V относительно заданных правил преобразования.

Мы построим формулу исчисления предикатов, которая будет общезначима тогда и только тогда, когда эти слова эквивалентны.

Построенное соответствие будет вычислимым, т. е. существует алгоритм, который по описанию слов U, V и правил подстановки строит код соответствующей формулы.

Далее рассуждаем стандартно: если бы существовал алгоритм проверки общезначимости формул, его можно было бы использовать для проверки общезначимости построенной формулы и получить тем самым алгоритм для решения проблемы равенства слов в полугруппах.

Искомая формула имеет вид $B \to C$, где B является конъюнкцией[6)] $n + 5$ формул. Первые пять из них перечислим явно

$$\forall x E(x, x)$$
$$\forall x \forall y (E(x, y) \sim E(y, x))$$
$$\forall x \forall y \forall z ((E(x, y) \wedge E(y, z)) \to E(x, z))$$
$$\forall x \forall y \forall u \forall v ((E(x, y) \wedge E(u, v)) \to E(m(x, u), m(y, v)))$$
$$\forall x \forall y \forall z E(m(x, m(y, z)), m(m(x, y), z))$$

В этих формулах использован бинарный предикатный символ E и бинарный функциональный символ m. Для кодирования считаем, что $E = P_0^2$, $m = f_0^2$.

При интерпретации символу E соответствует бинарное отношение. Из истинности первых трёх формул следует, что это отношение является отношением эквивалентности (рефлексивность, симметричность и транзитивность в точности описываются этими формулами).

Интерпретацией символа m является некоторая бинарная операция, будем называть её умножением.

[6)] Напомним, что конъюнкция (и остальные использованные в рассуждении пропозициональные связки) выражается через связки $\neg$ и $\to$.

Четвёртая формула означает, что умножение корректно: класс произведения не зависит от выбора представителей в классах сомножителей. Пятая формула утверждает ассоциативность умножения.

Таким образом, любая интерпретация с областью интерпретации M, в которой указанные выше формулы истинны, корректно определяет полугруппу на множестве M/E классов эквивалентности отношения E.

Остальные n членов конъюнкции B соответствуют правилам подстановки и имеют вид

$$E(m(\ell_1, m(\ell_2, m(\dots, \ell_j)\dots)), \\ m(r_1, m(r_2, m(\dots, r_k)\dots))) \quad (4.36)$$

В этих формула использованы константы (0-арные функциональные символы) f_i^0, которые соответствуют символам алфавита a_i. Константа ℓ_i соответствует i-му символу левой части правила подстановки, а константа r_i — i-му символу правой части.

Формула C имеет вид, аналогичный (4.36). В ней ℓ_i соответствует i-му символу слова U, а r_i — i-му слова V.

Описанное соответствие вычислимо (проверку этого утверждения оставляем читателю в качестве самостоятельного упражнения).

Если построенная формула $B{\to}C$ общезначима, то слова U и V эквивалентны относительно данных правил подстановки. В самом деле, пусть множество слов в алфавите A служит областью интерпретации формулы $B{\to}C$, предикатный символ E интерпретируется как отношение эквивалентности, а функциональный символ m как умножение (конкатенация) слов. Тогда истинность $B{\to}C$ в этой интерпретации означает равенство $U = V$ в полугруппе A^*/E, что равносильно

достижимости слова V из слова U преобразованиями из заданного набора правил.

В обратную сторону. Пусть слова U и V эквивалентны относительно данных правил подстановки. Рассмотрим некоторую интерпретацию формулы $B{\to}C$ на области M, в которой B истинна (в противном случае $B{\to}C$ истинна). Это означает, что в полугруппе M/E выполняются равенства, задаваемые всеми правилами подстановки. Но тогда в полугруппе выполняются равенства $U_1L_iU_2 = U_1R_iU_2$ для любых слов U_1, U_2 и любого правила подстановки $L_i \leftrightarrow R_i$. Поэтому последовательность подстановок определяет цепочку равенств, которая начинается со слова U и заканчивается на слове V. Из транзитивности отношения E заключаем, что истинна формула C. Это доказывает общезначимость формулы $B{\to}C$. □

Из теоремы 4.65 и теоремы 3.65 о полноте исчисления предикатов следует, что исчисление предикатов является примером формальной системы, в которой аксиомы и правила вывода разрешимы (см. задачи 4.21–4.23), а множество выводимых формул неразрешимо (по теореме 4.65).

Задачи

4.20. Предложите алгоритм проверки общезначимости тех формул, которые содержат только унарные предикатные и функциональные символы.

4.21. Докажите, что множество кодов формул исчисления предикатов разрешимо.

4.22. Докажите разрешимость множества троек вида $\langle A, A{\to}B, B\rangle$, где A, B — формулы исчисления предикатов.

4.23. Докажите, что множество кодов аксиом исчисления предикатов разрешимо.

4.6.2. Перечислимость и выводимость

Множество формальных теорем, т. е. выводимых формул, в формальной системе с разрешимыми множествами аксиом и правил вывода обладает более слабым алгоритмическим свойством — перечислимостью.

Определение 4.66. 2-ленточная МТ *перечисляет* язык $L \subset A^*$, если она, работая на пустом входе, печатает на одной из лент (*выходная лента*) список всех слов языка L (возможно, с повторениями), разделённых специальным знаком $\# \notin A$. При этом машина никогда не изменяет непустые символы на выходной ленте.

Язык называется *перечислимым*, если какая-то машина его перечисляет.

Можно охарактеризовать перечислимые множества условиями, которые ослабляют условия для разрешимых множеств.

Определение 4.67. Пусть для языка L в конечном алфавите A существует такая решающая МТ с алфавитом $B \supset A$, которая останавливается в состоянии q_{yes} в точности на входах из языка L. (При этом пустой символ Λ не принадлежит алфавиту A.) Тогда будем говорить, что такая машина *допускает* язык L.

Из этого определения сразу следует, что для любого разрешимого множества существует допускающая машина. Обратное, вообще говоря, неверно: машина, которая допускает L, может не останавливаться на одном из входов, не принадлежащих L.

Теорема 4.68. *Язык перечислим тогда и только тогда, когда некоторая машина Тьюринга допускает его.*

Доказательство. Пусть для языка L есть перечисляющий алгоритм. Опишем допускающую L машину (многоленточную). Она запускает перечисляющий алгоритм и после печати очередного слова сравнивает напечатанное слово и то слово, которое ей подано на вход. В случае совпадения допускающая машина останавливается в состоянии q_{yes}, иначе продолжает выполнение перечисляющего алгоритма. Поскольку перечисляющий алгоритм печатает в точности слова из языка L, допускающая машина останавливается в состоянии q_{yes} в точности на входах из L.

И в обратную сторону: пусть машина M допускает язык L. Перечисляющий алгоритм работает итерациями. На итерации с номером i он проверяет для каждого из первых i слов $w_1, \ldots, w_i$ в алфавите A (порядок лексикографический) остановится ли машина M на входе w_j за i тактов работы в состоянии q_{yes}. Если обнаруживается слово, для которого это условие выполнено, оно печатается на выходную ленту.

Такой алгоритм печатает только слова из языка L. С другой стороны, на каждом слове из языка L машина M останавливается в состоянии q_{yes}. Поэтому каждое слово из L будет рано или поздно напечатано. □

Следствие 4.69. *Пересечение перечислимого и разрешимого множества перечислимо.*

Доказательство. Пусть множество A разрешается машиной M_1, а множество B допускается машиной M_2. Машина M, которая допускает язык $A \cap B$, использует машины M_1 и M_2 следующим образом. Вначале она сохраняет копию входного слова и запускает M_1. Если работа M_1 заканчивается в состоянии q_{no}, то машина M также останавливается в отвергающем состоянии. Если работа M_1 заканчивается в состоянии

q_{yes}, то машина M запускает M_2 на сохранённой копии входного слова. □

Для перечислимых множеств есть несколько других характеризаций.

Задача 4.24. Докажите, что язык $L \subset A^*$ перечислим тогда и только тогда, когда

(а) L является проекцией разрешимого множества, т. е. существует такое разрешимое множество $R \subset (A \cup \{\#\})^*$, $\# \notin A$, что $w \in L$ равносильно существованию такого $u \in R$, что $u = w\#w'$;

(б) L является областью определения некоторой вычислимой функции, т. е. множеством тех слов, на которых останавливается некоторая машина Тьюринга;

(в) L является областью значений некоторой всюду определённой вычислимой функции, т. е. существует такая машина Тьюринга, которая останавливается на любом входе, а результаты её работы в точности совпадают с языком L.

Задача 4.25. Докажите, что если слова языка L можно перечислить в лексикографическом порядке, то язык L разрешим.

Теорема 4.70. *Если множество аксиом и правила вывода формальной системы разрешимы, то множество выводимых в этой формальной системе формул перечислимо.*

Доказательство. Перечисляющий алгоритм работает следующим образом. Он перебирает все слова в лексикографическом порядке и для каждого слова

проверяет, является ли оно описанием вывода. Если найдено слово, которое является описанием вывода, то алгоритм печатает описания всех формул, входящих в этот вывод.

Ясно, что такой алгоритм напечатает только выводимые формулы. С другой стороны, для каждой выводимой формулы F есть вывод, описание которого является конечным словом. Поэтому перечисляющий алгоритм рано или поздно обработает это слово и напечатает F. □

Задача 4.26. Докажите, что если формулы, аксиомы и правила вывода формальной системы перечислимы, то множество выводимых в этой формальной системе формул также перечислимо.

Указание: используйте результат задачи 4.24(а).

Замечание 4.71. Из этих фактов можно сделать вывод, что из неперечислимости некоторого множества слов сразу следует, что нельзя построить такую формальную систему, в которой выводятся в точности слова из этого множества, а множества формул, аксиом и правил вывода перечислимы.

В дальнейшем окажется очень полезной связь между перечислимостью и разрешимостью, устанавливаемая следующей теоремой.

Теорема 4.72 (Пост). *Если язык L и его дополнение $A^* \setminus L$ перечислимы, то язык L разрешим.*

Доказательство. В одну сторону теорема тривиальна: дополнение к разрешимому множеству разрешимо. Чтобы доказать её в обратную сторону, рассмотрим алгоритм D, который выполняет поочерёдно алгоритмы, перечисляющие множество и его дополнение (на

разных лентах) и сравнивает каждое из напечатанных слов со своим входом. Если вход совпал со словом, которое напечатал алгоритм, перечисляющий L, то алгоритм D останавливается в состоянии q_{yes}, если вход совпал со словом, которое напечатал алгоритм, перечисляющий дополнение к L, то алгоритм D останавливается в состоянии q_{no}.

Поскольку любое слово будет напечатано на одной из лент, алгоритм D всегда останавливается. □

Из теоремы Поста получаем примеры неперечислимых множеств: это дополнения к неразрешимым перечислимым множествам.

Задача 4.27. Докажите перечислимость множества описаний машин Тьюринга, останавливающихся на пустом входе.

Указание: используйте теорему 4.68.

Из этой задачи и теоремы Поста следует, что множество описаний тех машин Тьюринга, которые не останавливаются на пустом входе, неперечислимо. То же самое справедливо и для других универсальных моделей вычислений. Важный для дальнейшего случай сформулируем явно.

Следствие 4.73. *Неперечислимо множество описаний машин Минского, которые не останавливаются, начиная работу с нулевыми регистрами.*

4.6.3. Теорема Гёделя

Теорема 4.70 имеет такое следствие: из неперечислимости множества формул следует невозможность построения формальной системы с разрешимыми множествами аксиомам и правил вывода, которая выводила бы в точности формулы из этого множества. В этом

разделе мы рассмотрим наиболее известный пример такого рода.

В примере 3.31 мы ввели язык формальной арифметики. В него входят формулы в сигнатуре $0 \in \mathcal{F}_0$, $E \in \mathcal{P}_2$, $\nu \in \mathcal{F}_1$, $\times \in \mathcal{F}_2$, $+ \in \mathcal{F}_2$. В примере 3.33 были выписаны аксиомы формальной арифметики. Выводимые из этих аксиом утверждения называются *теоремами* формальной арифметики.

Стандартная модель формальной арифметики имеет в качестве области интерпретации множество натуральных чисел $\mathbb{N} = \{0, 1, \dots\}$, константа 0 интерпретируется как 0 из множества $\mathbb{N}$ натуральных чисел, E как предикат равенства, ν как функция следования, $\times$ как функция умножения, $+$ как функция сложения. Это действительно модель, поскольку легко проверить истинность аксиом. Поэтому все теоремы формальной арифметики истинны в стандартной модели (рассуждение аналогично доказательству корректности исчисления предикатов). Другими словами, формальная арифметика корректна относительно стандартной модели и, значит, непротиворечива (например, формула $E(0, \nu(0))$ невыводима, поскольку она интерпретируется как ложное равенство $0 = 1$).

Замечание 4.74. Важно отметить, что приведённые выше доводы в пользу корректности и непротиворечивости формальной арифметики опираются на нашу уверенность в том, что содержательно арифметика непротиворечива. Анализ этой правдоподобной гипотезы представляет куда более сложную задачу.

4.6.3.1. Арифметические предикаты

Предикаты, выразимые в формальной арифметике, называются *арифметическими*.

В примере 3.31 уже были указаны некоторые арифметические предикаты: $L(x,y) =$ «x не больше y»; предикат делимости $D(x,y) =$ «x делит y»; предикат $Q(r,x,y) =$ «r есть остаток от деления x на y»; унарные предикаты $\mathrm{Is}_n =$«$x = n$».

Из этих формул легко построить формулы, выражающие предикаты сравнения значений многочленов с целыми коэффициентами (равенства, неравенства и сравнения по модулю). Такие формулы получаются громоздкими. Для краткости мы будем использовать обычные математические формулы. Чтобы различать формальные части формул и неформальные сокращения, мы будем ограничивать последние кавычками (что уже делали выше). Например, утверждение «существует такое x, что $x^3 = x + 6$», мы будем записывать в виде

$$\exists\, x\text{«}x^3 = x + 6\text{»},$$

что намного короче полной записи этого утверждения

$$\exists\, xE(\times(\times(x,x),x),+(x,\nu(\nu(\nu(\nu(\nu(\nu(0)))))))). \quad (4.37)$$

Ещё один простой пример арифметических предикатов предоставляют финитные предикаты.

Определение 4.75. Предикат $P(x_1,\ldots,x_n)$ от натуральных чисел называется *финитным*, если существует такое число M, что $P(x_1,\ldots,x_n)$ ложен для любого набора чисел, в котором найдётся число, превышающее M.

Лемма 4.76. *Любой финитный предикат арифметичен.*

Доказательство. Количество наборов, на которых финитный предикат истинен, конечно. Поэтому финитный предикат можно выразить формулой, которая

является дизъюнкцией конъюнкций (ДНФ), причём каждая из конъюнкций соответствует одному из наборов $(t_1, \ldots, t_n)$, на которых предикат истинен, и имеет вид

$$«x_1 = t_1» \wedge «x_2 = t_2» \wedge \cdots \wedge «x_n = t_n».$$

(Обратите внимание, что здесь t_i — не переменные, а константы, каждую из которых можно записать термом аналогично тому, как в формуле (4.37) записано число 6.) □

Чтобы выражать более сложные предикаты, нужна предварительная подготовка. Основой её служит следующий факт из теории чисел.

Лемма 4.77 (бета-функция Гёделя). *Для любой конечной последовательности $x_1, \ldots, x_n$ натуральных чисел найдутся такие два натуральных числа a, b, что для всех $1 \leqslant i \leqslant n$ выполнены сравнения $x_i \equiv a \pmod{ib+1}$ и неравенства $x_i < b$.*

Доказательство. Подберём такое b, чтобы $b > x_i$ для всех $1 \leqslant i \leqslant n$, и все числа вида $ib+1$ были бы взаимно простыми (т. е. не имели общих простых делителей). Можно указать такое b явно. Например,

$$b = n! \prod_{i=1}^{n} (x_i + 1). \tag{4.38}$$

Тогда любой простой делитель числа

$$(ib+1) - (kb+1) = (i-k)b$$

является также делителем b (так как $|i-k| < n$), а значит, не может быть делителем $ib+1$.

Из китайской теоремы об остатках (см., например, [8]) следует существование числа a, удовлетворяющего условию леммы. □

Лемма 4.77 предоставляет способ выражать формулами формальной арифметики более общие условия, включающие в себя утверждения о конечных последовательностях натуральных чисел. Конечной последовательности $x_1, \ldots, x_n$ соответствует в таком переводе тройка чисел a, b, n, удовлетворяющих лемме 4.77; чтобы сослаться на i-й член последовательности, нужно ввести вспомогательную переменную t и записать формально условия $t < b$, $t \equiv a \pmod{ib+1}$. Итерируя этот приём, можно превращать в формулы формальной арифметики утверждения о конечных последовательностях конечных последовательностей натуральных чисел и т. д.

Вместо того, что формулировать общую технику применения леммы 4.77, рассмотрим два типичных примера.

Пример 4.78. Числа Фибоначчи задаются рекуррентно: $f_0 = f_1 = 1$ и $f_i = f_{i-1} + f_{i-2}$ при $i \geqslant 2$. Докажем арифметичность предиката

$$\mathrm{Fib}(x, n) = «x = f_n».$$

Для этого используем те числа a, b, которые «кодируют» последовательность первых n чисел Фибоначчи. В лемме 4.77 идёт речь о последовательностях, первый член которых имеет индекс 1. Поэтому мы должны сдвинуть нумерацию чисел Фибоначчи на единицу. Предикат $\mathrm{Fib}(x, n)$ выразим формулой

$$\exists\, a\, \exists\, b «x \equiv a \pmod{(n+1)b+1}» \wedge \ldots,$$

где на место многоточий нужно поставить условия, которым должны удовлетворять числа Фибоначчи. Во-первых, нужно записать начальные условия

$$«1 \equiv a \pmod{b+1}» \wedge «1 \equiv a \pmod{2b+1}».$$

Во-вторых, нужно записать рекуррентное соотношение

$$\begin{aligned}&\forall\, i(\text{«}i \leqslant n\text{»} \to \exists\, x\, \exists\, y\, \exists\, z \text{«}x \equiv a \pmod{(i+1)b+1}\text{»} \wedge \\ &\quad \text{«}x < b\text{»} \wedge \text{«}y \equiv a \pmod{(i+2)b+1}\text{»} \wedge \text{«}y < b\text{»} \wedge \\ &\text{«}z \equiv a \pmod{(i+3)b+1}\text{»} \wedge \text{«}z < b\text{»} \wedge \text{«}z = x + y\text{»})\end{aligned}$$

Подставив конъюнкцию всех этих условий вместо многоточий в написанной выше формуле, получим формулу, выражающую предикат $\mathrm{Fib}(x, n)$.

Пример 4.79. Вопрос об арифметичности предиката «x равен y в степени z» был оставлен в примере 3.31 без ответа. Теперь можно доказать, что этот предикат арифметичен.

Для этого нужно закодировать последовательность степеней $y, y^2, \ldots, y^z$ числами a, b и записать естественное рекуррентное условие для членов этой последовательности. Формула, выражающая предикат, будет иметь вид

$$\begin{aligned}&\text{«}x = 1\text{»} \vee \\ &\exists\, a\, \exists\, b \text{«}x \equiv a \pmod{zb+1}\text{»} \wedge \text{«}y \equiv a \pmod{b+1}\text{»} \wedge \\ &\forall\, i(\text{«}i < z\text{»} \to \exists\, u\, \exists\, v \text{«}u \equiv a \pmod{ib+1}\text{»} \wedge \text{«}u < b\text{»} \wedge \\ &\quad \text{«}v \equiv a \pmod{(i+1)b+1}\text{»} \wedge \text{«}v < b\text{»} \wedge \text{«}v = yu\text{»})\end{aligned}$$

Задачи

4.28. Докажите, что следующие предикаты выражаются в формальной арифметике

а) «x имеет ровно 7 делителей»;

б) «первая цифра десятичной записи x — единица»;

в) «в двоичной записи x нет двух единиц подряд»;

г) «$x = \binom{n}{k}$».

4.29. Пусть множество A натуральных чисел таково, что предикат $x \in A$ выражается формулой формальной арифметики. Докажите, что предикат $C(n) =$«множество A содержит ровно n чисел» также можно выразить формулой формальной арифметики.

4.30. Можно ли выразить в формальной арифметике предикат «ℓ — наименьшая длина формулы формальной арифметики, которой выражается предикат Is_n»? (Напомним, что $\mathrm{Is}_n(x)$ означает «$x = n$».)

4.6.3.2. Неперечислимость арифметических истин

Используя бета-функцию Гёделя, для машины Минского можно написать формулу в формальной арифметике, которая выражает тот факт, что машина Минского не останавливается на пустом входе (начальные значения всех регистров нулевые).

Достаточно рассмотреть машины Минского с четырьмя регистрами (напомним, что такого количества регистров достаточно для моделирования машин Тьюринга).

Работа такой машины Минского M полностью описывается пятью последовательностями натуральных чисел: номерами исполняемых команд и состояниями регистров. Таким образом, нужная нам формула имеет вид

$$\forall\, t(\exists\, a_1\, \exists\, b_1 \ldots \exists\, a_5\, \exists\, b_5 F_M(t, a_1, \ldots, b_5)), \qquad (4.39)$$

где формула F_M выражает тот факт, что последовательности $x_j^{(i)}$, $i = 1, \ldots, 5$, $j = 1, \ldots, t$, кодируемые тройками (a_i, b_i, n), корректно описывают работу машины Минского M на t шагах вычисления, начиная с пустого входа, и при этом машина не останавливается за t шагов. Как видно из этого описания, формула F_M

имеет вид

$$F_M = F_1(a_1, \ldots, b_5) \wedge \forall\, j\big(L(j,t) \wedge \neg E(j,t) \rightarrow \\ F_2(j, a_1, \ldots, b_5) \wedge F_3(t, a_1, \ldots, b_5)\big).$$

Формула F_1 выражает корректность начальных условий: исполняется первая команда и значения всех регистров нулевые. Формула F_2 описывает изменение состояния машины Минского на шаге i, т. е. связывает $x_{j+1}^{(i)}$ и $x_j^{(i)}$. Наконец, последняя формула выражает тот факт, что команда с номером t не является командой `halt` .

Идея построения этих формул понятна, если использовать следующее описание машины Минского. Пусть команды машины Минского и регистры нумеруются положительными числами. Тогда машина Минского описывается набором пятёрок вида (n, c, i, j, k), где n — номер команды, $c \in \{1, 2, 3\}$ — её тип, а i, j, k являются параметрами этой команды, дополненными при необходимости нулями. Итак, арифметическое описание машины Минского является набором из $5q$ чисел, где q — число команд. Мы считаем, что эти числа являются параметрами в формуле F_M.

Задача 4.31. Докажите, что множество формул вида (4.39), в которых параметры описания машины Минского заменены на замкнутые термы вида $\nu(\nu(\ldots(0)\ldots))$, разрешимо.

Предположим, что перечислимо множество формул формальной арифметики, истинных в стандартной интерпретации.

По следствию 4.69 и задаче 4.31 перечислимо и пересечение этого множества с множеством формул вида (4.39), в которые вместо параметров описания

машины Минского подставлены замкнутые термы вида $\nu(\nu(\dots(0)\dots))$.

Но такое пересечение выделяет в точности формулы вида (4.39) для машин Минского, не останавливающихся на пустом входе. Поскольку по формуле можно восстановить машину Минского, то перечислимым оказывается и множество машин Минского, которые не останавливаются на пустом входе, что противоречит следствию 4.73.

Из полученного противоречия следует

Теорема 4.80 (Тарский). *Множество формул формальной арифметики, истинных в стандартной интерпретации, неперечислимо.*

Если учесть, что аксиомы и правила вывода формальной арифметики разрешимы, то как следствие из теорем 4.70 и 4.80 получаем одну из форм знаменитой теоремы Гёделя о неполноте.

Следствие 4.81 (Гёдель). *Формальная арифметика неполна.*

Замечание 4.82. Напомним, что мы предполагаем непротиворечивость стандартной интерпретации арифметики. Таким образом, мы доказали довольно слабое утверждение. Оказывается, справедлив более сильный, чисто синтаксический, факт — формальная арифметика либо противоречива, либо неполна. Для его доказательства необходимо построить выводы в формальной арифметике для всех содержательных утверждений, которыми мы пользовались в нашем рассуждении.

Замечание 4.83. Описанный подход к доказательству теоремы Гёделя был предложен А. Н. Колмогоровым (см. [17]).

Задачи

4.32. Разрешима ли проблема остановки машины Тьюринга M на входе w, если алфавит машины состоит из одного символа?

4.33. Покажите, что любая модель вычислений, в которой результат вычисления определён для любого входа, не является универсальной (т. е. существует алгоритм, который нельзя реализовать в этой модели).

4.34. Функция $f\colon \mathbb{N} \to \mathbb{N}$ неубывающая. Верно ли, что f вычислима на машине Тьюринга?

4.35. Функция $f\colon \mathbb{N} \to \mathbb{N}$ невозрастающая. Верно ли, что f вычислима на машине Тьюринга?

4.36. Верно ли, что для всякой всюду определённой функции $f\colon \mathbb{N} \to \{0,1\}$ существует машина Тьюринга, результат работы которой на входе n совпадает с $f(n)$ для бесконечно многих n?

4.37. (*«Машины Тьюринга с односторонней лентой»*). Определение такой МТ отличается от нашего основного определения лишь тем, что лента машины ограничена слева и работа всегда начинается в самой левой ячейке. Сам процесс работы не отличается от стандартного за исключением дополнительного правила остановки: если машина пытается сдвинуться влево в самой левой ячейке, то она останавливается. Докажите, что проблема остановки МТ с односторонней лентой алгоритмически неразрешима.

4.38. Существует ли алгоритм, который по описанию МТ M и её входа w проверяет, что на входе w головка M хотя бы один раз возвращается в первую ячейку?

4.39. Существует ли алгоритм, который по описанию МТ M и её входа w проверяет, что на входе w машина M не меняет состояние ленты?

4.40. Существует ли алгоритм, который по описанию МТ M и её входа w проверяет, что на входе w машина M изменяет хотя бы раз символы в ячейках, не содержащих символов входного слова w?

4.41. Существует ли алгоритм, который по описанию МТ M и её входа w проверяет, что на входе w головка M побывает над не более чем $2|w|$ ячейками памяти? ($|w|$ — длина слова w.)

4.42. Функция $u(M)$ равна 1, если машина Тьюринга M на любом входном слове работает не более 1000 тактов, в противном случае $u(M) = 0$. Вычислима ли функция $u(M)$ на машине Тьюринга?

4.43. Функция $u(M)$ равна количеству тех слов длины 10, на которых машина Тьюринга M не останавливается. Вычислима ли функция $u(M)$ на машине Тьюринга?

4.44. Функция $u(M)$ равна 1, если головка машины Тьюринга M на любом входном слове движется только вправо, в противном случае $u(M) = 0$. Существует ли алгоритм, который по описанию M находит значение функции $u(M)$?

4.45. Пусть $T(M, w)$ — номер того такта работы машины Тьюринга M на входе w, на котором головка в последний раз оказывается над пустым символом. (Если головка никогда не оказывается над пустым символом или оказывается над ним бесконечно много раз, $T(M, w)$ не определена.) Вычислима ли функция $T(M, w)$ на машине Тьюринга?

4.46. Пусть $T(M, w)$ — номер того такта работы машины Тьюринга M на входе w, на котором головка в первый раз оказывается над непустым символом алфавита a. (Если головка никогда не оказывается над символом a, $T(M, w)$ не определена.) Вычислима ли функция $T(M, w)$ на машине Тьюринга?

4.47. Пусть $T(M, q)$ — номер того такта работы машины Тьюринга M на пустом входе, после которого состояние машины в первый раз оказывается равно q. (Если машина никогда не переходит в состояние q, то $T(M, q) = 0$.) Вычислима ли функция $T(M, q)$ на машине Тьюринга?

4.48. Приведите пример такой функции от двух переменных $f(x, y)$, что для любого u функция $f(x, u)$ (как функция от x) вычислима на машине Тьюринга, а функция $f(u, y)$ (как функция от y) невычислима. (Значения функции и значения переменных — слова в алфавите $\{0, 1\}$.)

4.49. Число k назовём *рекордом-минимумом* функции $f\colon \mathbb{N} \to \mathbb{N}$, если $f(s) = k$, а при всех $t < s$ выполняется $f(t) > f(s) = k$. Обозначим $\mathrm{Recmin}_f(x)$ функцию, которая равна 1, если x — рекорд-минимум f, а иначе равна 0. Докажите, что Recmin_f вычислима.

4.50. Число k назовём *рекордом-максимумом* функции $f\colon \mathbb{N} \to \mathbb{N}$, если $f(s) = k$, а при всех $t < s$ выполняется $f(t) < f(s) = k$. Обозначим $\mathrm{Recmax}_f(x)$ функцию, которая равна 1, если x — рекорд-максимум f, и 0 в противном случае. Существует ли такая вычислимая функция f, что Recmax_f невычислима?

4.51. Существует ли алгоритм, который по описаниям двух машин Тьюринга M_1, M_2 проверяет истинность утверждения «машины M_1 и M_2 на любом входном слове w порождают одинаковую последовательность конфигураций»?

4.52. (*«Машины Тьюринга с хрупкой лентой»*) Это МТ со следующим ограничением: если машина меняет состояние одной из ячеек ленты второй раз, то она останавливается («лента рвётся»). Докажите, что проблема остановки машины Тьюринга с хрупкой лентой алгоритмически неразрешима.

4.53. Верно ли, что для всякой машины Тьюринга M существует такая машина Тьюринга M', что для любого входного слова x справедливы утверждения:

а) если M останавливается на x, то M' не останавливается на x;

б) если M не останавливается на x, то M' останавливается на x?

Ответы, указания, решения

1.1. Ответ: неверно.

1.3. Ответ: нет. Любая выводимая формула входит в бесконечное множество выводов.

1.4. Ответ: верно. Примените лемму 1.15.

1.5. Ответ: верно. Примените лемму 1.16.

1.6. Ответ: существует. Годится любая формула, которая не является тавтологией.

1.7. Ответ: неверно. Проверьте, что значение формулы $F_1 \to F_2$ на любом наборе переменных ложно.

1.8. Ответ: неверно. Например, x и $y{\to}x$ невыводимы, так как не являются тавтологиями, а $x{\to}(y{\to}x)$ выводима (это аксиома (A1)).

1.9. Ответ: да. При $F_1 = F_2$ обе формулы становятся частными случаями аксиомы (A1).

1.10. Указание. Докажите индукцией по построению формулы более сильное утверждение: если все переменные в формуле различны, то эта формула принимает как значение «истина», так и значение «ложь».

1.12. По теореме дедукции из $B \vdash A$ следует $\vdash B{\to}A$. Это означает, что формулы $B{\to}A$ являются тавтологиями для всех B, т. е. $\neg A{\to}A$ — тавтология. Поскольку $\neg 0{\to}0 = 0$, формула A также является тавтологией.

1.13. Аналогично предыдущей задаче заключаем, что все формулы вида $A{\to}B$ являются тавтологиями. Отсюда следует, что формула A невыполнима. Доказательство от противного. Допустим, что формула A истинна на наборе значений переменных $x_1 = \alpha_1, \ldots, x_n = \alpha_n$. Возьмём в качестве B формулу

$$x_1^{\neg\alpha_1} \vee x_2^{\neg\alpha_2} \vee \cdots \vee x_n^{\neg\alpha_n},$$

которая ложна на наборе α. Тогда и $A{\to}B$ ложна на наборе α. Пришли к противоречию.

1.14. Применяя теорему дедукции, заключаем, что $\vdash A{\to}B$ и $\vdash B{\to}\neg A$. Из леммы 1.23 следует, что $\vdash A{\to}\neg A$. Поэтому $A{\to}\neg A$ — тавтология. Но тогда и $\neg A$ — тавтология.

1.16. Формула $x{\to}y$ ложна при $x = 1$, $y = 0$, формула $y{\to}x$ ложна при $x = 0$, $y = 1$, а формула $(x{\to}x){\to}x$ — ложна при $x = 0$. Если $x = y = 1$, то любая формула ИВ без отрицаний истинна (это легко доказать индукцией по построению формулы).

Применим теперь критерий условной выполнимости. Так как

$$(\neg 1{\to}1){\to}\neg(1{\to}\neg 1) = 1{\to}\neg 0 = 1,$$

то формула $A = (\neg x \to y) \to \neg(x \to \neg y)$ истинна при $x = y = 1$, т. е. она истинна на любом наборе значений переменных, на котором истинны все формулы из Γ. Значит, $\Gamma \vdash A$.

1.17. Ответ: верно. Формула $\neg(F_1 \to \neg F_2)$ реализует ту же функцию, что и формула $F_1 \wedge F_2$, которая ложна каждый раз, когда одна из формул F_1, F_2 ложна. По теореме о полноте невыводимая формула F_1 не является тавтологией.

1.18. Из выводимости A, B следует, что это тавтологии. В силу теоремы о полноте достаточно доказать,

что приведённая в условии формула — тавтология. Вычислим значение формулы на наборе α:

$$(((C \to A)\to\neg B) \to C)(\alpha) = (((C\to 1)\to\neg 1)\to C)(\alpha) = \\ = (1\to 0)\to C(\alpha) = 0\to C(\alpha) = 1.$$

1.19. а) Формула $x_2\to\neg x_1$ выводима из множества формул Γ (и даже из одной формулы $x_1\to\neg x_2$).

б) Формула $\neg x_2\to\neg x_1$ невыводима из множества формул Γ. Действительно, на наборе значений $x_1 = \\ = 1$, $x_2 = x_3 = \cdots = x_n = \cdots = 0$ все формулы из множества Γ истинны, а формула $\neg x_2\to\neg x_1$ ложна. Невыводимость следует теперь из теоремы 1.37 об условной выводимости.

1.20. Ответ: противоречиво. Достаточно выбрать в качестве формул A и B любое отрицание тавтологии, например, $A = B = x \wedge \neg x$. Тогда формула $C = \neg A\to B$ ложна на любом наборе переменных. По теореме об условной выводимости отсюда следует выводимость из C любой формулы.

1.21. Ответ: для любого n.

Для $n = 1$ можно взять любое отрицание тавтологии.

В общем случае нужно предъявить такое несовместное множество формул, каждое подмножество которого совместно.

При $n = 2$ возьмём множество $S_2 = \{x\to\neg x, \neg x\to x\}$.

При $n > 2$ условию задачи будет удовлетворять следующее множество формул:

$$S_n = \{x\to y_1, y_1\to y_2, \ldots, y_{n-3}\to y_{n-2}, y_{n-2}\to\neg x, \neg x\to x\}.$$

Множество S_n несовместно, так как из него выводятся обе формулы множества S_2. С другой стороны, при удалении любой формулы из множества S_n получаем совместное множество:

- формулы из множества $S_n \setminus \{x{\to}y_1\}$ истинны на наборе $x = 1$, $y_1 = \dots = y_{n-2} = 0$;
- формулы из множеств $S_n \setminus \{y_i{\to}y_{i+1}\}$ истинны на наборах $x = 1$, $y_1 = \dots = y_i = 1$, $y_{i+1} = \dots = y_{n-2} = 0$;
- формулы из множества $S_n \setminus \{y_{n-2}{\to}\neg x\}$ истинны на наборе $x = 1$, $y_1 = \dots = y_{n-2} = 1$;
- формулы из множества $S_n \setminus \{\neg x{\to}x\}$ истинны на наборе $x = 0$, $y_1 = \dots = y_{n-2} = 0$.

1.22. а) Так как $\Gamma \vdash x_i{\to}\neg x_i$ (вывод по транзитивности из первых трёх формул в условии) и $\Gamma \vdash \neg x_i{\to}x_i$ (вывод по транзитивности из последних трёх формул), множество Γ противоречиво и, следовательно, полно.

Пункты б) и в) — аналогично.

г) Заметим, что при $x = 0$ все формулы из множества Γ истинны. Но формула y ложна на наборе $x = 0$, $y = 0$, а формула $\neg y$ ложна на наборе $x = 0$, $y = 1$. Применяя критерий условной выводимости (теорема 1.37), получаем, что $\Gamma \nvdash y$ и $\Gamma \nvdash \neg y$. Значит, множество Γ неполно.

1.23. Ответ: да.

Докажем более общее утверждение. Пусть T — тавтология, а Γ — некоторое множество формул. Обозначим через Γ_T множество всех формул вида $T{\to}A$, $A \in \Gamma$.

Утверждение: выводимости $\Gamma \vdash X$ и $\Gamma_T \vdash X$ равносильны. Отсюда следует ответ задачи: достаточно взять любую тавтологию, включающую более двух переменных, и применить утверждение к полному и непротиворечивому множеству, которое существует по лемме 1.36.

Таблица Р.1. Модель для задачи 1.24 а)

x	$f_\neg(x)$
0	0
1	1

x	y	$f_\to(x,y)$
0	0	1
0	1	0
1	0	0
1	1	1

Для доказательства утверждения достаточно доказать выводимости $\Gamma \vdash T{\to}A$, $A \in \Gamma$, и $\Gamma_T \vdash A$, $A \in \Gamma$. Они легко следуют из критерия условной выводимости.

1.24. а) Все схемы аксиом задаются тавтологиями, поэтому данная формальная система корректна.

Неполнота данной системы следует из невыводимости аксиомы (А1). Для доказательства невыводимости используем модель, заданную таблицей Р.1. Отрицание в данной модели задано тождественной функцией, а импликация соответствует стандартной связке эквивалентности $\sim$. Поэтому все аксиомы данной формальной системы реализуют в данной модели функции, тождественно равные 1.

Осталось заметить, что формула $x{\to}(y{\to}x)$ представляет функцию, обращающуюся в 0: согласно таблице Р.1, $[0{\to}(0{\to}0)] = [0{\to}1] = 0$.

б) Корректность следует из того, что аксиомы (А1), (А2) и формула $A \to (\neg A \to B)$ являются тавтологиями.

Данная формальная система неполна. Это следует из того, что все её схемы аксиом являются интуиционистскими тавтологиями. (А значит, в ней выводимы лишь интуиционистские тавтологии.)

1.25. Заметим, что из $A = 1$ и $A{\to}(\neg B{\to}\neg A) = 1$ следует, что $B = 1$. Это значит, что с помощью данного правила вывода из тавтологий выводятся только тавтологии.

Условию задачи будет удовлетворять такой набор $\mathcal{A}$ схем аксиом, который состоит из аксиом ИВ и схемы аксиом

$$(A{\to}B){\to}(\neg(A{\to}(\neg B{\to}\neg A)){\to}\neg(A{\to}B)). \qquad \text{(D1)}$$

Формула (D1) является тавтологией. Поэтому построенная формальная система корректна.

Для доказательства полноты достаточно проверить, что из формул A, $A{\to}B$ в этой формальной системе можно вывести формулу B (после чего любой вывод в ИВ преобразуется в вывод в данной формальной системе заменой применения modus ponens на указанный вывод).

Приведём нужный вывод явно:

1. A (гипотеза)
2. $A{\to}B$ (гипотеза)
3. $(A{\to}B){\to}(\neg(A{\to}(\neg B{\to}\neg A)){\to}\neg(A{\to}B))$ (D1)
4. $A{\to}(\neg B{\to}\neg A))$ (вывод из 2,3)
5. B (вывод из 1,4)

1.26. Ответ: изменится.

Заметим, что если $A = 1$, то $(B{\to}A){\to}A = 1$. Значит, $A \vdash (B{\to}A){\to}A$ в ИВ. Поэтому новое правило вывода позволяет из формулы A вывести любую формулу вида $A{\to}B$. Среди формул такого вида есть не только тавтологии.

1.27. Каждый из дизъюнктов ложен на $1/2^8 = 1/256$ доле всех наборов значений переменных, входящих в КНФ. Всего дизъюнктов 200, так что на по

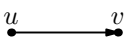
крайней мере $(256-200)/256 = 7/32$ доле всех наборов КНФ истинна.

1.28. Пусть КНФ C состоит из всех дизъюнктов вида

$$\bigvee_{j\in S} x_j, \quad \bigvee_{j\in S} \neg x_j, \quad |S| = n.$$

Любая нетавтологичная резольвента из этих дизъюнктов содержит $(n-1)+(n-1) = 2n-2 > n$ литералов. С другой стороны, C невыполнима, поскольку любой набор значений её переменных содержит либо n нулей (и тогда ложен один из дизъюнктов первого вида), либо n единиц (и тогда ложен один из дизъюнктов второго вида).

2.2. Рассмотрим шкалу Крипке



и на ней зададим модель Крипке таблицей

	u	v
$[a]$	0	0
$[b]$	0	1

В этой модели аксиома (А3) ложна в мире u, что можно проверить прямым вычислением:

	u	v
$[a]\to\bot$	1	1
$[b]\to\bot$	0	0
$[(b\to\bot)\to a]$	1	1
$[((b\to\bot)\to a)\to b]$	0	1
$[(b\to\bot)\to(a\to\bot)]$	1	1
$[((b\to\bot)\to(a\to\bot))\to(((b\to\bot)\to a)\to b)]$	0	1

2.3. Ответ: первая формула является И-тавтологией, а вторая нет.

И-тавтологичность первой формулы доказывается стандартными рассуждениями.

Контрмодель ко второй формуле можно построить на шкале Крипке из предыдущей задачи 2.2. Переменную x нужно положить истинной в обоих мирах, а переменную y — истинной только в мире v.

2.4. Для шкалы Крипке из решения задачи 2.2 рассмотрим модель

	u	v
$[a]$	0	1
$[b]$	0	0

В этой модели формула $a{\to}b{\to}a{\to}a$ ложна в мире u:

	u	v
$[a]{\to}b$	0	0
$[a{\to}b{\to}a]$	1	1
$[a{\to}b{\to}a{\to}a]$	0	1

Значит, формула $a{\to}b{\to}a{\to}a$ не является И-тавтологией. Поэтому она невыводима в ИИВ$_{\to}$ и тем более невыводима из аксиом (A1), (A2).

2.5. Проверка ложности формулы в мире u рутинна. Доказательство истинности этой формулы в любом мире любой линейной модели Крипке аналогично рассуждениям из примера 2.23 на с. 89.

2.6. Рассмотрим шкалу Крипке из решения задачи 2.2 и модель

	u	v
$[a]$	0	1
$[b]$	0	0
$[c]$	0	1

В этой модели формула

$$(a{\to}c){\to}((b{\to}c){\to}(a{\to}{\perp}{\to}b{\to}c))$$

ложна в мире u:

	u	v
$[a{\to}c]$	1	1
$[b{\to}c]$	1	1
$[a{\to}{\perp}]$	0	0
$[a{\to}{\perp}{\to}b]$	1	1
$[a{\to}{\perp}{\to}b{\to}c]$	0	1
$[(b{\to}c){\to}(a{\to}{\perp}{\to}b{\to}c)]$	0	1
$[(a{\to}c){\to}((b{\to}c){\to}(a{\to}{\perp}{\to}b{\to}c))]$	0	1

2.7. По теореме о полноте достаточно доказать, что если формулы A, B не являются И-тавтологиями, то и формула $A \vee B$ также не является И-тавтологией.

Пусть формула A ложна в мире u_A некоторой модели Крипке на шкале $(S_A, \leqslant_A)$, а формула B ложна в мире u_B некоторой модели Крипке на шкале $(S_B, \leqslant_B)$. Без ограничения общности считаем, что $S_A \cap S_B = \varnothing$.

Построим шкалу $(S, \leqslant)$ следующим образом: $S = S_A \cup S_B \cup \{d\}$, отношения достижимости на множествах S_A и S_B совпадают с $\leqslant_A$ и $\leqslant_B$ соответственно, а элемент d непосредственно накрывается элементами u_A и u_B.

Рассмотрим модель на шкале $(S, \leqslant)$, в которой значения всех переменных в мире d ложны, в мирах S_A переменные принимают такие значения, при которых формула A ложна в мире u_A, в мирах S_B переменные принимают такие значения, при которых формула B ложна в мире u_B.

Если формула A истинна в мире d построенной модели, то A истинна в мире u_A (монотонность значений формулы). Значит, A ложна в мире d. Аналогично

доказывается, что B также ложна в мире d. Поэтому формула $A \vee B$ также ложна в мире d в соответствии с определением истинности дизъюнкции в модели Крипке.

3.2. Указание: докажите аналогичное утверждение для термов и примените индукцию по построению формулы.

3.3. Ответ: x, y (свободны первое вхождение x и последнее вхождение y).

3.4. 1) Утверждение ложно. Формула

$$\forall\, x A(x) \to A(x)$$

даёт контрпример.

2) Утверждение истинно.

В кванторный блок должна входить хотя бы одна элементарная формула. По определению 3.8 на с. 105 она имеет вид $A(\dots)$ (в частности, для предикатного символа X арности 0 элементарная формула имеет вид $X()$). Поэтому второе из указанных вхождений переменной x входит в кванторный блок.

3) Утверждение ложно. Формула

$$\forall\, x A(x, y) \to A(y, x)$$

даёт контрпример.

4) Утверждение истинно.

3.5. Ответ: $A(x, y)$.

3.6. Ответ: нет.

Докажем, что любая формула $A(x, y)$ ИП с двумя параметрами в сигнатуре из унарных предикатных и функциональных символов в любой интерпретации на бесконечной области M истинна на некоторой паре значений параметров $\xi \neq \eta$, $\xi, \eta \in M$.

Заметим, что любая элементарная подформула формулы A зависит лишь от одной переменной, так

как и предикатные, и функциональные символы унарны. Обозначим через X_1, X_2, ..., X_k те элементарные подформулы A, которые зависят от переменной x, а через Y_1, Y_2, ..., Y_s — те, которые зависят от переменной y. *Цветом* элемента ξ области интерпретации назовём набор значений формул X_i при оценке $\pi(x) = \xi$ и набор значений формул Y_i при оценке $\pi(y) = \xi$. Таким образом, различных цветов не более 2^{k+s}.

Цвета $c(\xi), c(\eta)$ элементов ξ, η полностью определяют значение формулы A при оценке $\pi(x) = \xi$, $\pi(y) = \eta$. Чтобы формула A была истинна при оценках вида $\pi(x) = \pi(y) = \xi$, значение формулы при одинаковых цветах элементов ξ и η обязано быть истинным. Но тогда формула A истинна и при любой оценке вида $\pi(x) = \xi$, $\pi(y) = \eta$, для которой $c(\xi) = c(\eta)$.

Так как область интерпретации бесконечна, какой-то из цветов должен содержать бесконечно много элементов. Но это означает, что предикат, выражаемый формулой A, отличается от предиката равенства.

3.7. Пусть формула F истинна в интерпретации $\mathcal{M}$ на области M. Построим интерпретацию $\mathcal{M}'$ на бесконечной области $M \times \mathbb{Z}$, в которой формула F также истинна.

(Унарный) предикатный символ A интерпретируем в $\mathcal{M}'$ как $[\mathbf{A}] \times \mathbb{Z}$, где $[\mathbf{A}] \subseteq M$ — интерпретация A в $\mathcal{M}$ (значение предиката A на паре (m, n) зависит лишь от первой компоненты пары). Интерпретацию функциональных символов зададим аналогично: $[\mathbf{f}]'(m, n) = ([\mathbf{f}](m), n)$.

Аналогично предыдущей задаче заметим, что каждая элементарная подформула E формулы F зависит только от одной переменной. Более того, по построению значение формулы E при оценке $\pi'(x) = (m, n)$ в интерпретации $\mathcal{M}'$ совпадает со значением той же

элементарной подформулы при оценке $\pi(x) = m$ в интерпретации $\mathcal{M}$.

Осталось доказать индукцией по построению формулы, что для любой подформулы G формулы F с параметрами $x_1, \ldots, x_k$ значение G в интерпретации $\mathcal{M}'$ при оценке $\pi'_1(x_1) = (m_1, n_1)$, $\pi'(x_2) = (m_2, n_2)$, $\ldots$, $\pi'(x_k) = (m_k, n_k)$ совпадает со значением G в интерпретации $\mathcal{M}$ при оценке $\pi(x_1) = m_1$, $\pi'(x_2) = m_2$, $\ldots$, $\pi'(x_k) = m_k$. (Базой индукции является случай элементарных подформул.)

3.8. Пусть $b_0, \ldots, b_k$ — последовательность двоичных цифр числа n. Ясно, что $k = O(\log n)$. Запишем арифметическую формулу, выражающую n как результат вычислений по схеме Горнера, и сравним её с x:

$$x = (b_0 + 2(b_1 + 2(\cdots + 2b_k)) \ldots)$$

В эту формулу входят k сложений и k умножений на числа 0, 1, 2. Их можно представить переменными x_0, x_1, x_2 и добавить условия $\mathrm{Is}_i(x_i)$, длина записи которых $O(1)$. Поэтому длина всей формулы $O(k) = O(\log n)$.

3.9. Ответ: да.

Для любого натурального числа n есть формула $\mathrm{Is}_n(x)$, которая выражает унарный предикат «$x = n$». Формула $\mathrm{Is}_0(x)$ имеет вид

$$\forall\, y E(a(x, y), y),$$

формула $\mathrm{Is}_1(x)$ —

$$\forall\, y(\neg \mathrm{Is}_0(y) \to E(m(x, y), y)),$$

а остальные формулы $\mathrm{Is}_n(x)$ определяются рекуррентным соотношением

$$\mathrm{Is}_{n+1}(x) = \exists\, y\, \exists\, z(\mathrm{Is}_n(y) \wedge \mathrm{Is}_1(z) \wedge E(x, a(y, z))).$$

Искомая формула является дизъюнкцией формул $\mathrm{Is}_p(x)$, где p пробегает множество простых десятизначных чисел (таких чисел конечное число).

Заметим, что труднее доказать невыразимость предиката «x — простое число» в данной интерпретации.

3.10. *Линейным порядком* называется такое частично упорядоченное множество $(M, \leqslant)$, что для любых $x \neq y$ либо $x \leqslant y$, либо $y \leqslant x$.

Теория линейного порядка задаётся формулами в сигнатуре из одного бинарного предикатного символа Less, который соответствует строгому сравнению ($x < y$ означает, что $x \neq y$ и $x \leqslant y$). Свойства линейного порядка выражаются такими формулами:

$$A = \forall x \forall y(\mathrm{Less}(x, y) \to \neg\, \mathrm{Less}(y, x)), \qquad \text{(антисимметричность)}$$

$$T = \forall x \forall y \forall z(\mathrm{Less}(x, y) \wedge \mathrm{Less}(y, z) \to \mathrm{Less}(x, z)), \qquad \text{(транзитивность)}$$

$$L = \forall x \forall y(\mathrm{Less}(x, y) \vee \mathrm{Less}(y, x)). \qquad \text{(линейность)}$$

Теория с аксиомами A, T, L или, что равносильно, с одной аксиомой $A \wedge T \wedge L$ называется теорией линейного порядка. Моделями этой теории служат как раз всевозможные линейные порядки. (Заметим, что из истинности формулы A в некоторой интерпретации следует истинность формулы $R = \forall x \neg\, \mathrm{Less}(x, x)$, выражающей свойство антирефлексивности строгого сравнения.)

Формула $M = \exists m \forall x \neg\, \mathrm{Less}(m, x)$ выражает свойство линейного порядка «существует максимальный элемент».

Рассмотрим теперь формулу $F = A \wedge T \wedge L \wedge \neg M$. Она истинна для тех линейных порядков, в которых нет максимального элемента. Легко проверить,

что в каждом конечном линейном порядке максимальный элемент есть. Примером бесконечного линейного порядка без максимального элемента являются натуральные числа с обычным отношением сравнения. Поэтому формула F удовлетворяет условиям задачи.

3.11. Ответ: да.

Рассмотрим теорию линейного порядка с константами 0, 1. Формулы этой теории записываются в сигнатуре $(\mathrm{Less}, 0, 1)$, где Less — бинарный предикатный символ, а 0, 1 — константы (0-арные функциональные символы). К аксиомам A, T, L (см. решение предыдущей задачи) добавляется ещё одна аксиома $\mathrm{Less}(0, 1)$.

Заметим, что в любой модели теории линейного порядка формула $E(x, y) = \neg\mathrm{Less}(x, y) \wedge \neg\mathrm{Less}(y, x)$ выражает предикат равенства $x = y$.

Рассмотрим формулу $B = \forall x(E(x, 0) \vee E(x, 1))$. Эта формула истинна только для линейного порядка с константами 0, 1, который не содержит других элементов. Поэтому искомая формула имеет вид $F = A \wedge T \wedge L \wedge \mathrm{Less}(0, 1) \wedge B$.

3.12. Ответ: да.

Указание. Постройте формулу, которая утверждает, что в группе нет элемента порядка 11 или больше. Бинарный функциональный символ $m(x, y)$ в этой формуле интерпретируется как умножение в группе, а унарный предикатный символ $E(x)$ как равенство $x = e$, где e — единица группы.

3.13. Ответы для (а)–(б):

(а) $S_0 = \exists z((\forall i \neg P(i, z)) \wedge \neg B(z))$,

(б) $F_1 = \exists z((\forall i \neg P(z, i)) \wedge B(z))$.

Формулу для множества из пункта (в) удобно записать, введя предикат «позиция j непосредственно следует за позицией i».

Этот предикат выражается формулой

$$N(i,j) = P(i,j) \wedge \neg\exists\, k(P(i,k) \wedge P(k,j)).$$

Формула для пункта (в) имеет вид

$$\exists\, i\, \exists\, j\, \exists \big(N(i,j) \wedge N(j,m) \wedge B(i) \wedge B(j) \wedge B(m)\big).$$

Множество слов из пункта (г) можно описать следующим образом: это слова, которые начинаются на 0, заканчиваются на 1 и символы в этих словах чередуются. Такое описание выражается следующей формулой

$$S_0 \wedge F_1 \wedge \big(\forall\, i \forall\, j(N(i,j) \rightarrow \\ ((B(i) \rightarrow \neg B(j))) \wedge (\neg B(j) \rightarrow B(i)))\big).$$

Множество слов из пункта (д) можно описать следующими условиями: слово начинается с 00, слово заканчивается на 1, после 00 следует 1, а после 1 — 00. Эти свойства выражаются аналогично приведённым выше свойствам.

Ответ для (е) и (ж): нет. В этой книге мы не обсуждали способы доказательства невыразимости свойств формулами первого порядка, поэтому решения для этих пунктов не приводятся.

Идея доказательства невыразимости состоит в том, чтобы указать два таких слова v и w, что одно принадлежит рассматриваемому множеству (скажем, для пункта (е) это значит, что оно содержит чётное число единиц), а второе не принадлежит, причём все замкнутые формулы нашей сигнатуры принимают в интерпретациях на этих словах одинаковые значения. Такие интерпретации называются *элементарно эквивалентными*.

Удобный метод проверки элементарной эквивалентности интерпретаций основан на так называемых

играх Эренфойхта (см., например, [4]). Этот метод сравнительно несложно применить в данной задаче.

Заметим также, что известна полная характеризация выразимых множеств в терминах регулярных выражений — выразимы в точности те множества, которые можно представить регулярным выражением, не использующим итерации (звёздочки Клини).

3.14. Ответы: а), б), в) формулы не общезначимы, г), д) формулы общезначимы.

3.15. Формула ложна в интерпретации A как тождественно ложного предиката.

3.16. Формулы а) и б) выполнимы, поскольку истинны в интерпретации A как тождественно ложного предиката.

Формула в) невыполнима. Проще всего заметить, что её отрицание эквивалентно формуле д) из задачи 3.14.

3.17. Ответ: да.

3.18. Другими словами вопрос задачи формулируется так: существует ли такая формула $A(x, y)$ с двумя параметрами, что формулы $\forall x \exists y A(x, y)$ и $\forall x \exists y \neg A(x, y)$ общезначимы? Ответ отрицательный. Действительно, будем рассматривать интерпретации, в которых каждый предикатный символ интерпретируется как тождественная ложь или тождественная истина. Тогда истинность $A(x, y)$ не зависит от значений x, y. Поэтому в такой интерпретации одна из указанных выше формул будет ложной.

3.19. Формулы неэквивалентны. Интерпретацию, в которой они принимают разные значения построим так, чтобы предикаты A и B не зависели от первого аргумента.

В этом случае истинность первой формулы совпадает с истинностью формулы $(\exists y A'(y)) \to (\exists y B'(y))$,

а истинность второй — с истинностью формулы $\exists\, y(A'(y)\to B'(y))$. Нетрудно подобрать такие A' и B', что первая формула ложна, а вторая — истинна (B' можно выбрать тождественно ложным).

3.20. Ответы: 1) ложно; 2) истинно; 3), 4) ложно.

3.21. Ответы: а) да; б) не обязательно.

Количество различных наборов истинностных значений четырёх унарных предикатов равно 16. Истинность формулы с этими предикатами зависит лишь от того, какие из наборов истинностных значений встречаются при данной интерпретации. Поэтому истинность F в любой интерпретации на области из 16 элементов означает, что F общезначима.

Если F истинна в интерпретациях с меньшим количеством элементов, то она может и не быть общезначимой. Достаточно рассмотреть дизъюнкцию формул, каждая из которых отрицает, что существует элемент, на котором реализуется некоторый набор истинностных значений предикатов. В интерпретации на области из не более чем 15 элементов такая формула будет истинной, так как истинен хотя бы один из членов дизъюнкции. Однако существует интерпретация на 16 элементах, реализующая все возможные наборы истинностных значений четырёх унарных предикатов, и в этой интерпретации такая дизъюнкция будет ложной.

3.22. Ответы:

а) $\forall\, x\,\forall\, y\,\exists\, z(A(x)\to\neg(A(y)\to B(z)))$;

б) $\forall\, x\,\forall\, y(A(x)\to A(y))$;

в) $\exists\, x\,\forall\, y\,\forall\, z(A(x)\to\neg(A(y)\to B(z)))$;

г) $\exists\, x\,\exists\, y\,\exists\, z\neg((B(x)\to A(y))\to A(z))$;

д) $\exists\, x\,\forall\, y\,\exists\, z(\neg(B(x)\to A(y))\to A(z))$.

3.23. Отрицание замыкания формулы из условия имеет вид

$$\neg(\forall\, y(\forall\, x A(x) \to A(y))),$$

приведённая нормальная форма

$$\exists\, y \forall\, x \neg(A(x) \to A(y)),$$

её сколемизация

$$\forall\, x \neg(A(x) \to A(c)).$$

Заменяя бескванторную часть на КНФ, получаем эквивалентную формулу

$$\forall\, x(A(x) \wedge \neg A(c)).$$

Пустой дизъюнкт можно вывести, как только в перечислении замкнутых элементарных формул встретится формула $A(c)$ (из $A(c) \wedge \neg A(c)$).

4.2. Задачу проще решить, используя многоленточные машины Тьюринга (см. задачу 4.10). Тем не менее мы приведём решение для основной модели (одноленточной МТ), которое иллюстрирует использование приёмов построения машин, описанных в разделе 4.1.1.

Для сравнения соответственных символов слов u и v используем циклическое соединение машин. Сравнения нужно производить от младших битов к старшим, чтобы результат работы был корректным и при наличии незначащих нулей в начале записи числа.

Опишем простые машины, из которых построена искомая машина сравнения.

Машина M_1 переводит головку на младший бит первого числа. Её таблица переходов приведена в таблице Р.2, q_1 — финальное состояние.

Машина M_2 запоминает очередной бит первого числа и заменяет его на символ #, после чего находит

Таблица Р.2. Машина M_1 для решения задачи 4.2

	#	0	1
q_0	$(\#, -1, q_1)$	$(0, +1, q_0)$	$(1, +1, q_0)$

Таблица Р.3. Машина M_2 для решения задачи 4.2

	#	0	1	Λ
q_0		$(0, +1, q_{10})$	$(1, +1, q_{11})$	$(\Lambda, +1, q_{10})$
q_{10}	$(\#, +1, q_{10})$	$(0, +1, q_{10})$	$(1, +1, q_{10})$	$(\Lambda, -1, q_{20})$
q_{11}	$(\#, +1, q_{11})$	$(0, +1, q_{11})$	$(1, +1, q_{11})$	$(\Lambda, -1, q_{21})$
q_{20}	$(\Lambda, 0, q_{f0})$	$(\Lambda, -1, q_3)$	$(\Lambda, -1, q_{f0})$	$(1, 0, q_{f2})$
q_{21}	$(\Lambda, -1, q_{f0})$	$(\Lambda, -1, q_{f0})$	$(\Lambda, -1, q_3)$	
q_3	$(\#, -1, q_3)$	$(0, 0, q_{f1})$	$(1, 0, q_{f1})$	$(\Lambda, 0, q_{f1})$

соответственный ему бит второго числа и сравнивает их. При неравенстве она переходит в финальное состояние q_{f0}, а в случае равенства возвращается к (новому) младшему биту первого числа и переходит в финальное состояние q_{f1} или q_{f2} (последний случай означает успешное завершение сравнения чисел). Таблица переходов машины M_2 представлена в таблице Р.3.

Машина M_3 заменяет стирает все непустые символы на ленте слева от начального положения головки и после этого пишет символ 0. Таблица переходов машины M_3 представлена в таблице Р.4.

Теперь уже нетрудно проверить, что искомая машина сравнения получается соединением машин M_1, M_2 и M_3, при котором M_2 из состояния q_{f1} переходит в своё начальное состояние, из состояния q_{f0} переходит в начальное состояние машины M_3, а состояние q_{f2} является финальным для всего соединения машин.

Таблица Р.4. Машина M_3 для решения задачи 4.2

	#	0	1	Λ
q_0	$(\Lambda, -1, q_0)$	$(\Lambda, -1, q_0)$	$(\Lambda, -1, q_0)$	$(0, 0, q_1)$

4.3. Указание: в «оперативной памяти» МТ можно запомнить любую конечную информацию, в частности, любой конечный список слов.

4.4. Искомая машина получается последовательным соединением двух машин M_1 и M_2.

Машина M_1 ищет на ленте символ 1, после чего смещается до первого пустого символа перед 1 и останавливается.

Машина M_2 определяет, есть ли справа от начального положения головки две единицы, после чего останавливается в случае отрицательного ответа и двигается вправо бесконечно долго в случае положительного ответа. Соединение этих машин удовлетворяет условиям, описанным в замечании 4.21.

Машина M_1 ищет единицу и слева, и справа от своего начального положения, увеличивая область поиска. Для этого она вынуждена использовать помимо символов $\Lambda, 1$ ещё один вспомогательный символ 0 (пометка, указывающая на ячейки просмотренной области).

В табл. Р.5 представлена таблица переходов машины M_1 (строки — первый аргумент δ_{M_1}, столбцы — второй).

В состоянии q_0 машина M_1 перемещается вправо по ячейкам, содержащим пометку 0, в состоянии q_1 — в противоположном направлении.

Таблица Р.5. Машина M_1 для решения задачи 4.4

	Λ	0	1
q_0	$(0, -1, q_1)$	$(0, +1, q_0)$	$(1, -1, q_3)$
q_1	$(0, +1, q_0)$	$(0, -1, q_1)$	$(1, -1, q_2)$
q_2	$(\Lambda, 0, q_3)$		$(1, -1, q_2)$
q_3		$(\Lambda, 0, q_3)$	

Если M_1 видит пустой символ в состоянии q_0, то это означает, что она находится справа от просмотренной области (заполненной 0). Записав в текущую ячейку 0, машина увеличивает просмотренную область на одну ячейку вправо и переходит в состояние q_1.

Аналогично, обнаружив пустую ячейку в состоянии q_1, машина M_1 увеличивает просмотренную область на одну ячейку влево и переходит в состояние q_0.

Если машина видит 1 в состоянии q_0, то это означает, что слева расположена просмотренная область. Поэтому машина сдвигается влево и останавливается (быть может, заменив 0 на пустой символ). Тут полезно вспомнить, что по договорённости из замечания 4.2 неопределённые элементы таблицы соответствуют переходу в финальное состояние.

Если же машина M_1 видит 1 в состоянии q_1, то это означает, что просмотренная область расположена справа. Поэтому машина находит левый край последовательности из 1 и лишь там останавливается. Это условие выполняется после перехода в состояние q_2.

Машина M_2 проще (см. табл. Р.6), и читателю предлагается самостоятельно проверить корректность её работы.

Таблица Р.6. Машина M_2 для решения задачи 4.4

	Λ	0	1
q_0	$(0, +1, q_0)$		$(1, +1, q_1)$
q_1			$(1, +1, q_2)$
q_2	$(\Lambda, +1, q_2)$		$(1, +1, q_2)$

4.5. Считаем, что алфавит $\{0, 1, \ldots, n\}$.

(а) Искомая машина является соединением машины M_1, которая переводит головку на последний символ входного слова; машины M_2, которая двигается справа налево, каждый раз заменяя символ n на 0, и останавливается, достигнув отличного от n символа; и машины M_3, которая заменяет символ i под головкой на $i+1$, а если под головкой находится пустой символ, M_3 заменяет его на 0.

(б) Такую машину легче построить, используя две вспомогательные ленты. На одной из них машина перечисляет слова в лексикографическом порядке, на другой — считает количество сделанных шагов. Машина каждый раз сравнивает слово на первой вспомогательной ленте со входом и, когда эти слова совпадают, выдаёт в качестве результата работы содержимое второй вспомогательной ленты.

4.6. Достаточно построить такую МТ M_u, которая останавливается в точности на входах, начинающихся со слова $u = u_1 u_2 \ldots u_n$.

Приведём явную конструкцию такой машины. Машина M_u имеет $n + 2$ состояния. Таблица переходов:

$$\delta(u_i, q_i) = (u_i, +1, q_{i+1}), \qquad 1 \leqslant i \leqslant |u|,$$
$$\delta(a, q_i) = (a, +1, q_{n+2}), \qquad a \neq u_i,\ 1 \leqslant i \leqslant |u|,$$
$$\delta(a, q_{n+2}) = (a, +1, q_{n+2}).$$

Начальным является состояние q_1. Легко проверить, что в состоянии q_{i+1}, $0 \leqslant i \leqslant n$, машина оказывается тогда и только тогда, когда она прочла первые i символов слова u, а в состоянии q_{n+2} — тогда и только тогда, когда один из прочитанных символов отличается от соответствующего символа слова u. Из последней строки в описании таблицы переходов видно, что после перехода в состояние q_{n+2} машина M_u не останавливается.

4.8. Приведём набросок решения. Нужно использовать две идеи. Одна изложена при доказательстве теоремы 4.38: работу любой машины можно моделировать на МТ в двухсимвольном алфавите.

Здесь такими символами будут пустой символ Λ и символ $*$, который МТЗГ может писать на ленту.

Вторая идея состоит в том, чтобы вместо моделирования самого процесса вычисления записать на ленту *протокол вычисления* (т. е. последовательность конфигураций) в подходящей кодировке.

Будем представлять протокол вычисления в виде слова $c_0\#c_1\#\ldots\#c_n\#\ldots$. Предполагаем, что вспомогательный символ $\#$ не входит ни в алфавит A, ни во множество состояний Q.

Протокол вычисления мы будем кодировать как в доказательстве теоремы 4.38 (0 нужно заменить на Λ, 1 — на $*$, кодируются все символы из множества $A \cup Q \cup \{\#\}$). Однако теперь мы увеличим кодировку, добавляя после кода каждого символа $n + 3$ пустых символа, где n — длина кода символа в кодировке теоремы 4.38.

На бесконечной вправо части ленты, заполненной пустыми символами, можно записать код любого символа, а значит, и любой конфигурации, используя только запись $*$.

В процессе моделирования нам будет нужно копировать конфигурации. Вставка новых символов всегда происходит на правом конце записи. Признаком правого конца являются $2n + 4$ пустых символа подряд. Чтобы отметить место, до которого уже выполнено копирование, используются пустые вставки в кодировке: записываем вместо $n + 3$ пустых символов слово $\Lambda *^{n+1} \Lambda$.

Заметим, что перед началом моделирования нужно привести входное слово к указанной кодировке. Для этого его также нужно скопировать, меняя символы на их коды и вставляя пробелы в указанном выше количестве. Правый край (место, куда нужно скопировать код символа) определяется точно так же. А место, до которого уже скопированы символы нужно отмечать $*$. (Поэтому $*$ не должна входить в алфавит моделируемой машины).

Завершение конструкции рутинно. Искомая машина комбинируется из машин, выполняющих такие действия:

- записать справа от входного слова код начальной конфигурации, копируя по очереди символы входного слова, после этого записать код символа-разделителя #;
- повторять циклически следующие действия:
  - найти в записи кода текущей конфигурации код символа из множества Q;
  - определить соседей этого символа и записать полученные данные в «оперативную память» (используя вспомогательные состояния);

– копировать коды символов текущей конфигурации на свободную справа часть ленты, перед началом каждого копирования проверять, не является ли текущий код или следующий за ним кодом состояния МТ (если обнаружен код состояния, то в запись новой конфигурации символы не копируются, а вычисляются на основе записанной на предыдущем шаге информации);

– завершить цикл, скопировав код разделителя #, после чего остановиться, если текущее состояние моделируемой машины финальное.

4.9. Достаточно использовать четыре состояния, а если допускать частично определённые таблицы переходов (см. замечание 4.2 на с. 173), то всего три. Мы приведём решение для последнего случая.

Обозначим состояния M_1 через q_0, $\rightarrow$ и $\leftarrow$. Состояние q_0 является начальным. Таблица переходов M_1 задаётся следующим образом:

$$\begin{aligned}
\delta(a, q_0) &= (((a, q_0), (\Lambda, q_0)), +1, \rightarrow), \\
&\qquad a \in A, \\
\delta(a, \rightarrow) &= (((a, \Lambda), (\Lambda, \Lambda)), +1, \rightarrow), \\
&\qquad a \in A \setminus \{\Lambda\}, \\
\delta(\Lambda, \rightarrow) &= (((\Lambda, \Lambda), (\Lambda, \Lambda)), -1, \leftarrow), \\
\delta(((a, \Lambda), (\Lambda, \Lambda)), \leftarrow) &= (((a, \Lambda), (\Lambda, \Lambda)), -1, \leftarrow), \\
&\qquad a \in A \setminus \{\Lambda\}.
\end{aligned}$$

(Напомним, что алфавит машины M содержит помимо символов из A символы, которые индексируются парами из пар ⟨символ, состояние моделируемой машины⟩.)

Читателю предоставляется проверить, что такая машина останавливается в точности над той же ячейкой, с которой она начинает работу.

4.10. При записи решения этой задачи мы ограничимся неформальным описанием искомых машин, достаточно детальным, чтобы было ясно, как строить таблицы переходов. Наши примеры не оптимальны ни по количеству лент (всегда можно ограничиться одной), ни по числу тактов работы.

а) Используем 2-ленточную машину.

Первый этап работы состоит в том, чтобы скопировать вход на вторую ленту. Для этого достаточно одного состояния, в котором машина пишет на вторую ленту символ, который она читает с первой ленты, и сдвигает вправо обе головки. Этот этап заканчивается, когда с первой ленты прочитан пустой символ.

На втором этапе машина переводит головку второй ленты влево, пока не достигнет пустого символа, после чего сдвинет головку на один символ вправо. Головка первой ленты не меняет своего положения.

Третий этап аналогичен первому с заменой ролей лент: теперь символ со второй ленты копируется на первую ленту и обе головки сдвигаются вправо. Машина останавливается, прочитав пустой символ.

б) Конструкция машины аналогична. Но здесь на первом этапе машина копирует вход на вторую ленту, заменяя символы на первой ленте пустыми. На втором этапе машина читает символы со второй ленты, двигаясь справа налево, и записывает их на первую ленту, двигаясь по ней слева направо. Машина останавливается, когда прочитывает со второй ленты пустой символ.

в) Сравнение чисел удобно выполнять от младших битов к старшим.

Вначале машина копирует одно из чисел на вторую ленту, как это описано в предыдущих пунктах.

Далее происходит сравнение. Головки машины движутся по записям чисел справа налево. Если прочитаны одинаковые символы, то машина переходит к следующему символу. Если же прочитаны разные символы, то по ним можно определить, какое из чисел больше.

На последнем этапе машина стирает входную ленту и оставляет на ней результат сравнения, который можно хранить в «оперативной памяти» машины, увеличив число состояний.

г) Сложение чисел также удобно выполнять от младших битов к старшим и использовать при этом три ленты.

На первом этапе работы машина копирует записи слагаемых со входной ленты на две вспомогательные, стирая их с первой ленты.

На втором этапе выполняется собственно сложение. Машина движется справа налево, каждый раз выполняя сложение соответственных двоичных цифр (с учётом бита переноса) и меняя бит переноса. Бит переноса хранится в «оперативной памяти» машины. Если запись одного слагаемого короче другого, машина обрабатывает пустой символ как 0. Когда с обоих вспомогательных лент прочитаны пустые символы, машина делает последний такт работы, записывая при необходимости бит переноса (если он равен 1).

Алгоритм вычитания реализуется аналогично.

д) Проще всего реализовать алгоритм умножения «в столбик», используя пять лент. На две вспомогательные ленты копируются (со стиранием на входной ленте) сомножители a, b. На ленту 3 по очереди записываются числа вида $b \cdot 2^k$, где k-й бит числа a равен 1.

На ленте 4 хранится текущая сумма всех таких чисел. Сложение содержимого лент 3 и 4 даёт новое значение этой суммы, которое копируется (со стиранием) на ленту 4. Работа продолжается, пока не достигнут самый старший бит числа a.

Заметим также, что запись числа $b \cdot 2^k$ отличается от записи числа b только k нулями в конце. Поэтому эта запись легко поддерживается при чтении битов первого сомножителя (каждый прочитанный символ добавляет 0 на ленту 3 справа).

е) Из описанных выше алгоритмов нетрудно собрать алгоритм деления положительных целых чисел, который последовательно проверяет возможные значения частного, начиная от нуля. На вспомогательной ленте хранится запись «остатка» от такого «неправильного деления» (т. е. если мы проверяем возможное частное k, то записываем на эту ленту число $a - bk$). Каждый раз мы сравниваем $a - bk$ с делителем b. Если $a - bk < b$, то деление закончено. В противном случае процесс повторяется.

Такой алгоритм работает долго. Даже школьный алгоритм деления «уголком» работает быстрее. Предоставляем читателю самостоятельно продумать детали реализации алгоритма деления «уголком», а также придумать алгоритм деления, который работает ещё быстрее.

4.11. Указания. Выполнение арифметических операций на МТ описано в решении задачи 4.10.

Определить значение операнда-переменной i можно, просматривая код памяти RAM. Он имеет вид

$$\#a_1\#v_1\#\#a_2\#v_2\#\ldots\#a_n\#v_n\#.$$

Машина Тьюринга проверяет по очереди пары $\#a_j\#v_j$ и сравнивает a_j с i. Если найдена такая пара, что

$a_j = i$, то значением операнда будет v_j. В противном случае значение равно 0.

Для операнда-указателя описанную выше процедуру нужно повторить дважды. Здесь уже $v(i)$ может быть сколь угодно большим числом. Поэтому сравнивать на втором проходе $v(i)$ с a_j нужно, используя алгоритм, описанный в решении задачи 4.10в).

Выше мы предполагали, что значения операндов извлекаются на вспомогательные ленты. Чтобы сохранить новые значения на основных лентах, содержащих коды групп ячеек RAM, нужно выполнять операцию вставки содержимого одной ленты L_1 вместо заданной части другой ленты L_2. Эту операцию легко реализовать. Считаем, что заменяемая часть ленты отмечена пометками на ленте (о пометках на ленте см. раздел 4.1.1). Тогда нужная последовательность действий состоит в том, что а) копируем на дополнительную ленту содержимое L_2 до отмеченной позиции, б) копируем содержимое ленты L_1, в) после чего копируем ту часть ленты L_2, которая идёт вслед за заменяемым куском. Дополнительаня лента теперь содержит ленту L_2 со вставленной частью ленты L_1.

Выполнение остальных действий RAM реализуется ещё проще. Например, изменение номера команды заключается фактически в изменении состояния моделирующей машины (мы договорились хранить описание RAM в «оперативной памяти», т. е. во множестве состояний MT).

4.12.

(а) Скопируем регистр i в регистр j с сохранением значения в регистре i (см. пример 4.45). После этого применим следующие команды:

1: `dec` $j, n, 2$

2: `inc` $i, 1$

После перехода на команду n регистр j будет содержать 0, а регистр i — своё удвоенное первоначальное значение.

(б) Расширим пример 4.46:

1: `dec` $i, n, 2$

2: `dec` $i, 3, 1$

3: `inc` $i, m, 1$

Теперь не только управление передаётся в зависимости от чётности числа, но и регистр i содержит остаток от деления своего первоначального значения на 2.

(в) Как в (а), скопируем регистр i в регистр j с сохранением значения в регистре i. Выполнение команд

1: `dec` $j, n, 2$

2: `dec` $j, 3, 3$

3: `dec` $i, 1, 1$

приведёт к тому, что после перехода к команде n значение регистра i будет равно $\lfloor v/2 \rfloor$, где v — его первоначальное значение.

4.13. Приведём ответ (в таблице через ℓ', r' обозначены новые значения ℓ, r).

ℓ	r	d	ℓ'	r'
чётно	чётно	вправо	2ℓ	$r/2$
чётно	нечётно	вправо	$2\ell+1$	$\lfloor r/2 \rfloor$
нечётно	чётно	вправо	2ℓ	$r/2$
нечётно	нечётно	вправо	$2\ell+1$	$\lfloor r/2 \rfloor$
чётно	чётно	влево	$\ell/2$	$2r$
чётно	нечётно	влево	$\ell/2$	$2r$
нечётно	чётно	влево	$\lfloor \ell/2 \rfloor$	$2r+1$
нечётно	нечётно	влево	$\lfloor \ell/2 \rfloor$	$2r+1$

4.14. Нужные программы описаны фактически в решении задачи 4.12 (а), (в) (для преобразования $x \mapsto 2x + 1$ к программе пункта (а) нужно ещё добавить прибавление 1). Эти программы основаны на примерах 4.44–4.46. Можно проверить, что каждая из программ использует не более двух вспомогательных регистров.

4.15. Пусть алгорифм в алфавите $\{1\}$ со схемой $\mathcal{A}$ вычисляет отображение

$$\mathcal{A}^*(w) = \begin{cases} w11, & \text{если длина } w \text{ чётна,} \\ w1, & \text{если длина } w \text{ нечётна.} \end{cases}$$

Поскольку результат начинается со входного слова, каждое правило подстановки должно быть заключительным.

Применим алгорифм к достаточно длинному слову w (длина w превышает длины всех левых частей формул подстановки из схемы алгорифма). Слово w поддаётся первому правилу в схеме. Оно изменяет длину слова на s. Но то же самое справедливо и для слова $w1$. А отображение, приведённое в условии задачи, изменяет длину слов w и $w1$ по-разному.

4.17. Машина, которая дописывает к своему входу заданное слово u, может быть построена соединением машин, первая из которых находит конец входного слова (т. е. двигается вправо, пока не достигнет пустого символа), а вторая машина M_2 имеет на единицу больше состояний, чем длина дописываемого слова. В каждом из состояний M_2 записывает в текущую ячейку соответствующий этому состоянию символ слова u и сдвигается вправо. Последнее состояние — финальное.

4.18. Строго говоря, нужно перейти от слов в бесконечном алфавите к их кодам в конечном алфавите.

При этом предполагается, что разным формулам отвечают разные коды. Пример такого перехода см. в разделе 4.6.1.

После такого перехода разрешимость множества формул означает разрешимость предиката «слово w является кодом некоторой формулы». Аналогично формулируется разрешимость множества аксиом. Разрешимость множества правил вывода означает разрешимость предиката «слово $w_1\#\ldots\#w_k$ таково, что слова w_i являются кодами таких формул F_i, что $(F_1,\ldots,F_k)\in\mathcal{R}$ для некоторого правила вывода $\mathcal{R}$» (другими словами, является ли слово кодом вывода). Здесь $\#$ — специальный символ, который не входит в алфавит кодирования.

Вывод является последовательностью формул. Последовательности формул будем кодировать словами вида $w_1\#w_2\#\ldots\#w_n$.

Аналогично предыдущему, разрешимость множества выводов формулируется как разрешимость предиката «слово w является кодом некоторого вывода».

Доказательство утверждения достаточно простое. Искомый алгоритм перебирает по очереди коды формул в выводе (т. е. слова, которые стоят между разделителями $\#$).

Для каждого такого слова w_k алгоритм проверяет, является ли слово w_k кодом формулы. Если нет, то алгоритм останавливается в отвергающем состоянии. Если да, то алгоритм проверяет, является ли слово w_k кодом аксиомы. Если да, то алгоритм переходит к анализу следующего подслова w_{k+1} входа, заключённого между разделителями $\#$ (если следующего слова нет, то алгоритм останавливается в принимающем состоянии). Если нет, то алгоритм перебирает все подмножества множества $\{1,\ldots,k-1\}$. Для каж-

дого такого подмножества $S = \{i_1, \ldots, i_s\}$ алгоритм проверяет, является ли слово $w_{i_1}\# \ldots \# w_{i_s}\# w_k$ кодом некоторого вывода. Если да, то алгоритм переходит к анализу следующего подслова w_{k+1} (аналогично предыдущему переходу). Если для всех подмножеств результат проверки отрицательный, алгоритм останавливается в отвергающем состоянии.

Существенно в этом построении, что множество всех подмножеств $\{1, \ldots, k-1\}$ конечно. Поэтому алгоритм обязательно остановится (при сделанных выше предположениях о разрешимости формул, аксиом и правил вывода).

Перебирать подмножества можно разными способами. Один из них заключается в том, чтобы закодировать подмножество словом из 0 и 1, в котором 1 стоят в точности на местах $i_1, \ldots, i_s$. Перебирать такие слова можно в лексикографическом порядке, переход к следующему элементу в котором соответствует прибавлению 1 к числу, записываемому в двоичной системе данным словом.

4.19. Указание: данное отображение соответствует следующему способу нумерации точек положительного квадранта $\{(x, y) : x \geqslant 0, y \geqslant 0, x, y \in \mathbb{Z}\}$.

Перебираем по очереди все неотрицательные целые k. Для каждого такого k занумеруем последовательно пары $(k, 0)$, $(k-1, 1)$, $\ldots$, $(0, k)$. Для каждого k занумерована $k+1$ пара. Поэтому нумерация пар, соответствующих некоторому k, начинается с числа $\binom{k+1}{2}$.

Для вычисления отображения нужно уметь выполнять арифметические операции, см. задачу 4.10.

Для вычисления обратного отображения можно воспользоваться указанное выше процедурой последовательного перебора пар целых неотрицательных чисел.

Замечание. Для обратного отображения существует более эффективный алгоритм, который выполняет значительно меньшее количество элементарных операций при вычислении обратного отображения.

4.20. Построим вначале алгоритм проверки общезначимости формул, содержащих лишь унарные предикатные символы.

Рассуждаем как в задаче 3.21. Количество наборов истинностных значений n унарных предикатов равно 2^n. Истинность формулы с этими предикатами зависит лишь от того, какие из наборов истинностных значений встречаются при данной интерпретации. Поэтому истинность формулы F с n унарными предикатными символами в любой интерпретации на области из 2^n элементов означает, что F общезначима.

Алгоритм проверки общезначимости проверяет по очереди все интерпретации на области из 2^n элементов. Кодировать такие интерпретации можно наборами из n двоичных слов длины 2^n, т. е. одним двоичным словом длины $n2^n$.

Алгоритм оценки формулы в конечной интерпретации нетрудно построить: он следует определению истинностного значения формулы, в котором все шаги (для конечной интерпретации) занимают лишь конечное время.

Сведём случай формул, содержащих также и унарные функциональные символы к уже рассмотренному. Так как и предикатные, и функциональные символы унарны, то любая элементарная подформула имеет вид $A(f(f(\dots(x)\dots)))$.

Введём вспомогательные предикатные символы $A^{f,k}$, которые отвечают таким формулам с k функциональными символами. Для каждой формулы нам

потребуется лишь конечное число n таких вспомогательных предикатов.

Построим по исходной формуле F формулу F', в которой каждая элементарная подформула заменена на соответствующую формулу вида $A^{f,k}(x)$.

Общезначимость формулы F равносильна истинности формулы F' во всех специальных интерпретациях.

Специальные интерпретации удовлетворяют свойству согласованности: для каждого функционального символа f исходной формулы существует такое отображение $[\mathbf{f}]\colon M \to M$, что предикаты

$$[\mathbf{A}^{\mathbf{f},\mathbf{k}}](x) \text{ и } [\mathbf{A}^{\mathbf{f},\mathbf{0}}]([\mathbf{f}]^k(x))$$

равносильны, а также предикаты

$$[\mathbf{A}^{\mathbf{f},\mathbf{0}}](x) \text{ и } [\mathbf{A}^{\mathbf{g},\mathbf{0}}](x)$$

равносильны для любых двух функциональных символов f, g исходной формулы F.

Рассуждение об интерпретациях на 2^n элементах сохраняет силу и для специальных интерпретаций. Общий алгоритм при проверке интерпретации проверяет также условия согласованности для неё. Таких условий конечное число и каждое из них сводится в проверке равенства двух предикатов на конечной области.

4.21. Приведём краткое описание одного из возможных алгоритмов для решения этой задачи.

Алгоритм читает коды символов, описанные на с. 247. Для каждого кода алгоритм должен определять тип символа, представленного данным кодом. Процедуры идентификации символов используют арифметические операции с числовыми кодами символов (включая деление с остатком), сравнение чисел (см.

решение задачи 4.10) и вычисление обратного отображения, устанавливающего биекцию $\mathbb{N} \times \mathbb{N}$ и $\mathbb{N}$ (см. задачу 4.19).

В дальнейшем описании алгоритма мы опускаем переходы от кодов символов к символам.

Алгоритм следует рекурсивному определению формул и термов. Для проверки того, что данное слово является формулой (или термом), он выделяет из этого слова несколько подслов и выполняет проверку для каждого из них. Такая рекурсивная процедура может быть реализована, скажем, на многоленточных МТ с помощью вспомогательной ленты, на которую записываются проверяемые слова с указанием типа проверки (формула или терм). В начале работы алгоритма на эту ленту помещается входное слово.

Если при одной из проверок получен отрицательный ответ, то алгоритм останавливается и даёт отрицательный ответ. Если список слов для проверки исчерпан (на ленте проверки не осталось слов), то алгоритм останавливается и даёт положительный ответ.

Основной шаг алгоритма (разбор формулы или терма) состоит в том, что он копирует последнее из записанных на ленту проверки слов и выделяет части слова, которые нужно проверять. Каждое из таких подслов помещается на ленту проверки.

Опишем разбор терма. По определению (см. с. 104) терм либо состоит из одного символа, который является 0-арным функциональным символом или переменной (так что разбор сводится к проверке типа символа), либо имеет вид $f(t_1, \ldots, t_k)$, где f — функциональный символ арности k, а все слова t_i являются термами. Слова t_i выделяются следующими условиями: (а) первый символ слова t_i не является скобкой или запятой и имеет скобочный итог $+1$; (б) следующий за

t_i символ является первой (при движении направо от начала выделяемого подслова) закрывающей скобкой или запятой со скобочным итогом $+1$.

Теперь опишем разбор формулы. По определению формулы исчисления предикатов (см. с. 105) первый символ любой формулы является либо предикатным символом (элементарная формула), либо открывающей скобкой.

В первом случае слово обязано иметь вид

$$A(t_1, \ldots, t_k),$$

где A — предикатный символ арности k, а все слова t_i являются термами. Слова t_i выделяются следующими условиями: (а) первый символ не является скобкой или запятой и имеет скобочный итог $+1$; (б) символ, следующий за последним символом t_i, является первой (при движении направо от начала выделяемого подслова) закрывающей скобкой или запятой со скобочным итогом $+1$.

Если первый символ является открывающей скобкой, то возможны два варианта: либо второй символ s разбираемого слова является символом отрицания или квантора всеобщности, либо он является открывающей скобкой. В первом случае слово имеет вид

$$(\neg w) \text{ или } (\forall\, x w),$$

где x — переменная. В этом случае на ленту проверки нужно поместить слово w.

Во втором в силу леммы 1.8 слово имеет вид

$$w = (w_1 \to w_2),$$

где слова w_1 и w_2 разделяет левый символ импликации, для которого скобочный итог равен $+1$. На ленту проверки нужно поместить слова w_1 и w_2.

4.22. Указание. Алгоритм проверяет, что вход имеет вид $\langle A, C, B\rangle$, причём слово C имеет вид $(A{\to}B)$.

4.23. Алгоритм основывается на модификации алгоритма проверки формулы (задача 4.21).

Сравнительно легко модифицировать алгоритм из задачи 4.21 так, чтобы он определял также тип подслова, помещаемого на ленту проверки: посылка импликации, заключении импликации, формула под отрицанием, формула под квантором по переменной x.

Проверка аксиом (A1–A3) не вызывает никаких затруднений. При проверке аксиом (A4–A5) необходимо проверять также условия корректности.

Проверка условия корректности для аксиомы (A5) сводится к тому, чтобы проверить, есть ли в некоторой формуле свободное вхождение указанной переменной x (далее, как и в решении задачи 4.21, мы говорим о символах, а не о кодах символов). Алгоритм также основан на алгоритме разбора формулы. Он дополнительно сохраняет с каждым словом информацию о вхождении в кванторный блок по x (один бит информации — «да – нет»). В начале разбора этот бит равен «нет». Если разбирается слово вида

$$(\forall\, xw),$$

то этот бит устанавливается в значение «да». В остальных случаях этот бит наследуется при разборе слова.

При разборе односимвольного терма алгоритм сверяет его с переменной x. Если терм равен x и бит вхождения равен «нет», то обнаружено свободное вхождение переменной. Алгоритм заканчивает работу с ответом «свободные вхождения есть». Если лента проверки пуста, то разбор формулы закончен и свободные вхождения x не обнаружены. Алгоритм выдаёт ответ «свободных вхождений нет».

Аналогично можно проверить, что терм t свободен для переменной x в данной формуле (условие корректности для аксиомы (A4)). Прежде всего нужно составить список переменных терма t. Далее работает аналогичный предыдущему алгоритм, но теперь он сохраняет информацию о вхождении не только в кванторный блок по x, но и по каждой переменной терма t.

Кроме того, проверка аксиомы (A4) требует проверки следующего условия: формула B получается из формулы A подстановкой вместо всех свободных вхождений переменной x терма t.

Это условие можно проверить, модифицировав алгоритм разбора формул. Разбираем формулы A и B одновременно, используя две ленты проверок и делая шаги разбора формул поочередно. Как и выше, сохраняем информацию о свободном вхождении переменной x. Все шаги разбора формулы должны быть одинаковы. При разборе термов также должны получаться одинаковые шаги за исключением случая, когда в формуле A встречается односимвольный терм x в свободном вхождении. В этом случае нужно проверить, что подслово формулы B, которое должно быть разобрано а этом шаге, совпадает с термом t.

4.24. Конечный язык разрешим (см. задачу 4.3), откуда легко следуют свойства (а)–(в) из условия задачи.

Теперь рассмотрим случай бесконечных языков. Равносильность перечислимости и свойств (а)–(в) вытекает из следующих импликаций.

Если язык L перечислим, то выполнено свойство (б). Пусть машина Тьюринга M запускает алгоритм перечисления для языка L и сравнивает каждое напечатанное слово со входом. При совпадении машина останавливается.

Такая машина M вычисляет функцию, область определения которой совпадает с L.

Из свойства (б) следует свойство (в).

Пусть машина M вычисляет функцию с бесконечной областью определения L. Выберем некоторое слово w_0, принадлежащее языку L. Построим машину M', которая работает следующим образом: на входе $\langle w, t\rangle$ (описание слова w в алфавите машины M и целого положительного числа t) машина M' моделирует t тактов работы машины M на входе w. Если за это время M останавливается, то M' выдаёт результат w. В противном случае она выдаёт результат w_0.

Результат w_0 выдаётся также на всех входах, которые не являются описаниями пар указанного вида.

Из свойства (в) следует свойство (а). Пусть L — область значений всюду определённой функции f. Тогда множество R слов вида $u\#v$, где $u = f(v)$, разрешимо. Язык L является проекцией R по определению области значений.

Из свойства (а) следует перечислимость языка L. Алгоритм перечисления перебирает слова вида $w\#w'$ в лексикографическом порядке и проверяет принадлежность очередного такого слова множеству R. Если ответ положительный, то алгоритм печатает w на выходную ленту.

4.25. Если язык конечен, то он разрешим.

В случае бесконечного языка L разрешаем его следующим образом: запускаем перечисление в лексикографическом порядке и сравниваем каждое напечатанное на выходной ленте слово со входом w. Если оно совпало со входом, то принимаем w. Если напечатанное слово больше w, то отвергаем w. Рано или поздно встретится один из этих случаев, поэтому такой алгоритм всегда останавливается.

4.26. Перечислимое множество является проекцией разрешимого (задача 4.24(а)). Обозначим через A разрешимое множество пар $f\#v$, проекцией которого являются аксиомы формальной системы, а через R_k — разрешимое множество пар вида $w_1\#\ldots\#w_s\#v$, проекцией которого являются наборы из s слов, образующие отношение k-го вывода.

Построим разрешимое множество R пар $w\#w'$, проекцией которого являются все формальные теоремы, включив в него все такие пары $w\#w'$, что

$$w' = \#w_1\#\#w_2\#\ldots\#w_k\#,$$

где слова w_i описывают формулы в некотором выводе w. Более точно это означает следующее. Если i-я формула f_i в выводе является аксиомой, то $w_i = f_i\#v_i$ и $w_i \in A$. Если i-я формула f_i в выводе получается применением k-го правила вывода к формулам с номерами $j_1,\ldots, j_s$, то

$$w_i = \#k\#\langle j_1\rangle,\ldots\#\langle j_s\rangle\#g_1\#\ldots\#g_{s-1}\#f_i\#v_i,$$

где $g_1\#\ldots\#g_{s-1}\#f_i\#v_i \in R_k$. Кроме того, в последнем слове w_k так его часть, которая обозначена f_k в предыдущих условиях, должна совпадать с w.

Ясно, что множество R разрешимо: двигаемся вдоль описания вывода, каждый раз проверяя справедливость указанных выше условий.

4.27. Машина M, которая допускает описания МТ, останавливающихся на пустом входе, работает следующим образом.

Вначале она проверяет, что вход является описанием некоторой машины M'. Если это не так, то M останавливается в отвергающем состоянии q_{no}).

После этого машина M моделирует работу универсальной машины M' на пустом входе. Если машина

M' останавливается, то и M останавливается в принимающем состоянии q_{yes}.

4.28. а) Указанный предикат равносилен такому: «существует семь таких чисел $d_1, \ldots, d_7$, что все они различны и если y является делителем числа x, то он совпадает с одним из чисел $d_1, \ldots, d_7$». Свойство, описанное в кавычках, переписывается рутинным образом в виде формулы формальной арифметики.

б) Если первая цифра десятичной записи числа x равна 1, то это означает, что для некоторых y, n выполнено $x = 10^n + y$, $y < 10^n$. Предикат $z = 10^n$ выражается в формальной арифметике, поэтому указанные условия также записываются в виде формулы формальной арифметики.

в) Используя β-функцию Гёделя, закодируем последовательности $b_0, \ldots, b_n$ двоичных цифр числа x и $x_0, \ldots, x_n$ результатов вычисления x по его двоичной записи схемой Горнера:

$$b_n = x_n; \quad b_{k-1} + 2x_k = x_{k-1}\ (1 \leqslant k \leqslant n); \quad x_0 = x.$$

Эти последовательности однозначно определяются, если добавить к предыдущим соотношениям следующие:

$$b_k = x_k \bmod 2\ (0 \leqslant k \leqslant n-1).$$

Выразим в терминах последовательности b_i свойство «нет двух единиц подряд» в виде сокращённой записи, которую легко превратить в формулу формальной арифметики:

$$\forall i((i < n) \wedge (0 \leqslant i) \rightarrow ((b_i = 0) \vee (b_{i+1} = 0))).$$

г) Используя β-функцию Гёделя, закодируем последовательность факториалов $f_k = k!$, которая однозначно определяется следующими соотношениями:

$$f_0 = 1;\ f_{k+1} = (k+1)f_k \quad (0 \leqslant k \leqslant n-1).$$

Дальнейшее очевидно, так как

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}\,.$$

4.29. Используя β-функцию Гёделя, закодируем последовательность a_k элементов множества A, упорядоченных по возрастанию. Она однозначно задаётся условиями

$$a_k \in A\ (1 \leqslant k \leqslant n); \qquad a_{k+1} > a_k\ (1 \leqslant k \leqslant n-1),$$

которые по условию задачи выражаются формулами формальной арифметики.

4.30. Ответ: нет. Неформально этот ответ можно объяснить парадоксом Берри (см. пример В.2).

Этот парадокс можно преобразовать в противоречие в арифметике, если предположить, что наименьшая длина формулы, которой выражается предикат Is_n, выразима в арифметике.

В самом деле, пусть формула $L(\ell, n)$ задаёт предикат из условия задачи. Для каждого ℓ рассмотрим формулу $M_\ell(x)$, которая имеет вид:

$$\exists\, u\big(\,\mathrm{Is}_\ell(u) \wedge L(u,x) \wedge \forall\, j(\text{«}j < x\text{»} \to \neg L(u,j))\big).$$

Предикат Is_ℓ выражен здесь формулой длины $O(\log \ell)$ (см. задачу 3.8).

Формула $M_\ell(x)$ выражает предикат $\mathrm{Is}_{s(\ell)}(x)$, где $s(\ell)$ — наименьшее число, для которого кратчайшая формула, выражающая предикат $\mathrm{Is}_{s(\ell)}$, имеет длину ℓ.

Длина формулы $M_\ell(x)$ ограничена $O(\log \ell)$, поэтому при достаточно большом ℓ она меньше, чем ℓ. Приходим к противоречию, так как выразили предикат $\mathrm{Is}_{s(\ell)}$ формулой, которая короче, чем ℓ.

4.31. Указание: Разрешающий алгоритм должен проверить структуру формулы F_M и согласованность

параметров (вместо одного и того же параметра подставлен один и тот же терм).

4.32. Ответ: да.

Если алфавит машины состоит из одного символа, то таблица переходов задаёт отображение δ на множестве состояний Q. МТ останавливается тогда и только тогда, когда итерации δ, применённые к начальному состоянию q_0, приводят в финальное состояние.

Если после $|Q| + 1$ итерации финальное состояние не достигнуто, то МТ «зацикливается»: какое-из состояний повторилось. Таким образом, достаточно моделировать работу МТ за $|Q| + 1$ такт.

4.33. Заметим, что задача сформулирована не вполне корректно. В решении будем полагать, что алгоритмы, входы и результаты работы в данной модели описываются словами в алфавите, который содержит символы 0 и 1.

Применим диагональное рассуждение. Зафиксируем вычислимую нумерацию алгоритмов и входов в данной модели и построим следующий алгоритм: на входе i применяется алгоритм i, результат его работы изменяется на 0, если он отличен от 0, и на 1, если он равен 0. Этот алгоритм по построению отличен от всех алгоритмов данной модели.

4.34. Ответ: нет. Пример: функция трудолюбия Радо.

4.35. Ответ: да.

Невозрастающая функция $f\colon \mathbb{N} \to \mathbb{N}$ постоянна на некотором интервале $n_0 \leqslant n < \infty$. МТ, которая её вычисляет, должна хранить в своей «оперативной памяти» таблицу значений f вплоть до n_0. Машина сравнивает вход k с n_0; если $k < n_0$, то выдаётся результат $f(k)$ из таблицы значений; в противном случае выдаётся $f(n_0)$.

4.36. Ответ: да.

Пусть результат работы останавливающейся на каждом входе машины M отличается от $f(n)$ на бесконечном множестве S. Рассмотрим машину M', которая выдаёт результат, противоположный результату M. На множестве S этот результат совпадает с $f(n)$.

4.37. Работу обычной МТ можно моделировать на МТ с односторонней лентой. Состояние двусторонней бесконечной ленты в алфавите A

$$\dots a_{-k} \dots a_{-2} a_{-1} a_0 a_1 a_2 \dots a_k \dots$$

будем кодировать словом вида

$$\$(a_0, a_{-1})(a_1, a_{-2}) \dots (a_k, a_{-k-1}) \dots$$

Здесь \$ — дополнительный символ, указывающий на начало ленты. Кроме того, алфавит односторонней ленты содержит множество $A \times A$ всех пар символов алфавита A. Пустой символ Λ мы отождествляем с парой (Λ, Λ).

Пусть M — МТ, работающая на двусторонней ленте. Опишем машину N, которая моделирует работу M на односторонней ленте. Машина N является соединением двух машин N_1 и N_2.

Машина N_1 преобразует входное слово $w_1 \dots w_n$ в его представление в выбранной кодировке двусторонней ленты

$$\$(w_1, \Lambda)(w_2, \Lambda) \dots (w_n, \Lambda) \dots$$

(на месте правого многоточия стоят пустые символы). Машина N_1 запоминает в «оперативной памяти» очередной символ входа и записывает на его место \$ на первом такте или (w_{i-1}, Λ) на последующих (w_{i-1} — символ, который машина извлекает из своей «оперативной памяти»).

Достигнув символа Λ, машина N_1 начинает двигаться влево, пока не достигнет символа \$. После этого она сдвигается вправо и заканчивает работу.

Машина N_2 реализует моделирование работы M. Помимо состояний M она хранит в «оперативной памяти» бит положения головки машины M_1 (0, если головка находится в правой части ленты, и 1 в противном случае). Таблица переходов N_2 для состояний в правой части ленты копирует таблицу переходов для M за тем исключением, что изменяется первая компонента символа (у N_2 символы являются парами символов M).

Таблица переходов N_2 для состояний в левой части ленты повторяет таблицу переходов для M с переменой направления движения (так как левая часть ленты закодирована в обратном порядке).

Когда машина читает символ \$, она сдвигается вправо и изменяет бит положения головки моделируемой машины.

Описание машины N можно построить по описанию машины M алгоритмически. Поэтому получаем сводимость задачи остановки МТ на двусторонней ленте к задаче остановки МТ на односторонней ленте.

4.38. Ответ: нет.

Указание: Проблему остановки МТ на односторонней ленте (задача 4.37) можно свести к проблеме возвращения в первую ячейку.

4.39. Ответ: да.

Из решения задачи 4.32 можно извлечь следующую лемму: если МТ, двигаясь по бесконечной вправо (влево) части ленты, заполненной пустыми символами, не изменяла символы на ленте и сдвинулась по крайней мере на $|Q| + 1$ ячеек вправо (влево), то эта машина

никогда не останавливается и не изменяет ни одного символа на ленте.

Алгоритм, отвечающий на вопрос задачи, моделирует работу M на входе w в пределах зоны $|w|+2|Q|+2$ (ко входному слову слева и справа добавляются по $|Q|+1$ пустому символу), составляя список последовательных конфигураций.

Как только какой-либо символ на ленте изменяется, алгоритм останавливается и выдаёт ответ «нет».

Алгоритм останавливается и выдаёт ответ «да», если работа M завершилась, или конфигурация повторилась (для проверки этого условия нужно сравнивать очередную конфигурацию со всеми предыдущими), или машина M собирается выйти за границы отведённой зоны.

Корректность алгоритма следует из приведённой выше леммы.

4.40. Ответ: да.

Сводим эту задачу к предыдущей. Для этого заметим, что общее количество слов длины w в алфавите A равно $|A|^{|w|}$. Это значение вычислимо по описанию МТ M и её входа. Поэтому можно построить описание такой машины M', которая моделирует работу M, не меняя состояния ячеек, в которых записано входное слово. Вместо этого M' хранит в «оперативной памяти» состояние ячеек при работе M и изменяет его в соответствии с правилами работы M.

Из построения ясно, что M изменяет хотя бы раз символы в ячейках, не содержащих символов входного слова w, тогда и только тогда, когда M' изменяет хотя бы один символ на ленте.

4.41. Ответ: да.

Из условия следует, что количество возможных конфигураций МТ конечно и ограничено вычислимой

функцией от описания МТ и входа. Если у МТ Q состояний, а алфавит ленты содержит A символов, то общее количество состояний не превышает $2|w|QA^{2|w|}$ (первый множитель оценивает количество возможных положений головки, второй — количество состояний МТ, третий — количество возможных записей в $2|w|$ ячейках).

Далее рассуждаем аналогично задаче 4.39.

4.42. Ответ: да.

Указание: множество конфигураций МТ, от которых зависит ответ на данный вопрос, конечно и ограничено вычислимой от описания МТ функцией. Дальнейшие рассуждения аналогичны предыдущей задаче.

4.43. Ответ: нет.

К вычислению этой функции можно свести проблему остановки МТ на пустом входе. Действительно, построим по описанию МТ M описание МТ M', которая сравнивает длину входа с 10 и останавливается, если длина входа не равна 10. В противном случае M' стирает вход и моделирует работу M на пустом входе.

Из построения ясно, что $u(M')$ равно 0, если M останавливается на пустом входе и равно $(|A|-1)^{10}$, если M не останавливается на пустом входе.

4.44. Ответ: да.

Обозначим через Q_L множество тех состояний q, для которых найдётся такое a, что таблица переходов машины M предписывает на паре (a,q) движение налево или остановку на месте. Через Q_Λ обозначим множество тех q, для которых таблица переходов машины M предписывает на паре (Λ,q) движение налево или остановку на месте.

Функция $u(M)$ равна 0 в двух случаях:

– на некотором входе машина M достигает состояния из Q_L, двигаясь только вправо и не выходя за пределы входа вплоть до последнего такта работы;

– на некотором входе машина M прочитывает всё входное слово, двигаясь только вправо, после чего продолжает движение вправо по части ленты, заполненной пустыми символами, и достигает в некоторый момент состояния из Q_Λ.

Построим на множестве состояний Q ориентированный граф G_1, рёбрами которого являются пары (q_1, q_2), для которых при некотором $a \neq \Lambda$ выполнено $\delta_M(a, q_1) = (a', +1, q_2)$.

Построим также аналогичным образом граф G_2, рёбрами которого являются такие пары (q_1, q_2), что $\delta_M(\Lambda, q_1) = (a', +1, q_2)$.

Равенство $u(M) = 0$ равносильно дизъюнкции следующих условий:

(i) из состояния q_0 достижимо какое-то состояние из Q_L в графе G_1;
или

(ii) из состояния q, достижимого в графе G_1 из состояния q_0, достижимо в графе G_2 какое-то состояние из Q_Λ.

Проверка обоих условий сводится к решению задачи о достижимости в ориентированном графе. Для этой задачи есть стандартные алгоритмы (поиск в глубину, поиск в ширину). Ознакомиться с ними можно, например, в книге [9].

4.45. Ответ: да.

Вычисление функции $T(M, w)$ организовано следующим образом. Моделируется работа машины M

на входе w, при этом выполняются вспомогательные вычисления на двух дополнительных лентах (ленте-счётчике и ленте результата).

В начале работы содержимое дополнительных лент устанавливается в 0.

За каждый шаг моделирования одного такта работы M выполняются следующие действия:

– если головка машины M читает пустой символ, то на ленту результата копируется содержимое ленты счётчика;

– счётчик увеличивается на 1;

– вычисляется разность между показанием счётчика t и содержимым результата τ. Если $t - \tau > s|Q| \cdot |A|^s$, где $s = \max(|w|, t)$, то моделирование заканчивается.

После завершения моделирования выполняются следующие действия. Если лента результата содержит 0, то алгоритм не останавливается никогда. Если же на ленте результата записано ненулевое число, то оно выдаётся как значение функции $T(M, w)$.

Корректность этого алгоритма следует из такого наблюдения. Если головка машины M после такта t не оказывается над пустым символом, то дальнейшая работа машины происходит в зоне из $s = \max(|w|, t)$ ячеек. Если при этом машина не останавливается после $s|Q| \cdot |A|^s$ тактов работы, то она зацикливается и уже никогда не увидит пустого символа.

Указанный алгоритм не останавливается в двух случаях: (i) после достижения машиной M финального состояния лента результата содержит 0; (ii) бесконечно много раз головка оказывается над пустым

символом. Функция $T(M, w)$ в каждом из этих случаев не определена.

4.46. Ответ: да.

Вычисление функции $T(M, w)$ организовано так же, как в предыдущей задаче. Моделируется работа машины M на входе w, при этом на дополнительной ленте поддерживается счётчик.

Если машина оказывается в некоторый момент над символом a, то моделирование заканчивается и значение счётчика является результатом вычисления.

Если машина M достигла финального состояния, то моделирование заканчивается тем, что алгоритм вычисления $T(M, w)$ не останавливается.

4.47. Ответ: нет.

Функция $T(M, q)$ всюду определена и может быть использована для решения проблемы остановки.

Сведём проблему остановки машины M на пустом входе к вычислению функции $T(M', \$)$, где \$ — некоторое выделенное состояние.

Машина M' работает так же, как и машина M (мы предполагаем, что таблица переходов M всюду определена, иначе нужно преобразовать M как указано в замечании 4.2 на с. 173). Но финальные состояния M перестают быть финальными, и в них делается переход в состояние \$.

Значение $T(M', \$)$ равно 0 тогда и только тогда, когда M не останавливается на пустом входе.

4.48. Пусть $g(x)$ — некоторая невычислимая функция одного аргумента. Тогда функция $f(x, y) = g(y)$ удовлетворяет условию задачи.

При фиксированном втором аргументе функция $f(x, y)$ постоянна и потому вычислима. При фиксированном значении первого аргумента она совпадает с невычислимой функцией g.

4.49. Функция Rec_f отлична от нуля только при $n \leqslant f(0)$. Поэтому она финитна, следовательно, вычислима.

4.50. Ответ: нет.

Если f ограничена, то Rec_f отлична от нуля только при $n \leqslant \max_m f(m)$. Как и в предыдущей задаче, Rec_f финитна и, следовательно, вычислима.

Если f неограничена, то для любого k существует такое n, что $k < f(n)$. Чтобы проверить, является ли число k рекордом-максимумом, достаточно вычислить f для чисел от 0 до n.

Интересно отметить, что различить эти два случая алгоритмически невозможно. В частности, не существует алгоритма, который преобразует описание МТ, вычисляющей функцию f в МТ, вычисляющую функцию Rec_f.

4.51. Ответ: нет.

Проблему остановки на пустом входе можно свести к вопросу задачи.

По машине M построим две машины M_1 и M_2. Каждая из машин M_i имеет одно и то же начальное состояние q_0. В этом состоянии обе машины переходят в финальное состояние q_f, если читают любой непустой символ.

В противном случае обе машины моделируют работу M на пустом входе. Разница между ними состоит в том, что M_1 останавливается, как только останавливается M, а M_2 в этом случае переходит в состояние, в котором на ленту записывается вспомогательный символ \$, не входящий в алфавит M.

Из описания ясно, что M_1 и M_2 на любом входном слове w порождают одинаковую последовательность конфигураций тогда и только тогда, когда M не останавливается на пустом входе.

4.52. *Указание*: используйте решение задачи 4.8.

4.53. Ответ: нет.

Из задачи 4.24(б) следует, что множества слов, на которых останавливаются машины Тьюринга, это в точности перечислимые языки. Поэтому утверждение задачи можно переформулировать так: дополнение ко всякому перечислимому языку перечислимо. Это утверждение неверно, так как перечислимый язык с перечислимым дополнением разрешим в силу теоремы Поста (с. 255).

Справочное приложение

Мы всюду используем в этой книге теоретико-множественный подход, определяя все математические объекты как множества с некоторыми свойствами. За подробным изложением теории множеств мы отсылаем читателя к вводным курсам (например, [3]) и монографиям (скажем, [10]).

Для удобства читателя приведём определения тех понятий, которые используются в основном тексте, но не определяются в нём, в терминах теории множеств.

Множества, их подмножества и элементы

Само понятие множества остаётся неопределяемым. Интуитивно множество — это совокупность *элементов* произвольной природы.

Множество полностью и однозначно определяется своими элементами.

Например, имеется ровно одно *пустое* множество, которое не содержит ни одного элемента. Пустое множество обозначается $\varnothing$.

Если a является элементом множества m, то говорят также, что элемент a принадлежит множеству m или что m содержит элемент a. Принадлежность множеству обозначается $a \in m$.

На множествах определены следующие операции.

Объединение $a \cup b$ множеств a и b содержит в точности те элементы, которые входят хотя бы в одно из них.

Пересечение $a \cap b$ множеств a и b содержит в точности те элементы, которые входят в оба эти множества.

Разность $a \setminus b$ множеств a и b содержит в точности те элементы, которые входят в a и не входят в b.

Множество a содержится в множестве b (обозначение $a \subseteq b$), если $a \setminus b = \varnothing$. Говорят также, что a является *подмножеством* b.

Подмножество $a \subseteq b$ называется *собственным*, если $a \neq \varnothing$ и $b \setminus a \neq \varnothing$.

Подмножества множества m образуют множество, которое обозначается 2^m.

Бинарные отношения и отображения

Упорядоченная пара — это множество, которое содержит два элемента a и b, причём b является множеством, которое содержит a и, быть может, ещё один элемент. Элемент a называется *первым элементом пары*. *Второй элемент пары* равен a, если в b нет других элементов, и равен c, если b содержит элемент $c \neq a$.

Упорядоченная пара обозначается (a', a''), где a' — первый, а a'' — второй элемент пары.

Декартово произведение $a \times b$ множеств a и b состоит из всех упорядоченных пар вида (x, y), где $x \in a$, $y \in b$, и только из них.

Бинарное отношение r на множестве a — это подмножество $a \times a$.

Отношение $r \subseteq a \times a$ называется *отношением эквивалентности*, если выполняются свойства *рефлексивности*: $(x, x) \in r$ для любого $x \in a$; *симметричности*:

если $(x, y) \in r$, то $(y, x) \in r$; *транзитивности*: если $(x, y) \in r$ и $(y, z) \in r$, то $(x, z) \in r$.

Классом эквивалентности $r \subseteq a \times a$ называется такое подмножество $e \subseteq a$, что $(x, y) \in r$ для любых $x, y \in e$ и $(x, y) \notin r$ для любых $x \in e$, $y \in a \setminus e$. Классы эквивалентности либо совпадают, либо не пересекаются. Множество классов эквивалентности обозначается a/r.

Частично определённое отображение f из множества a в множество b — это такое подмножество декартова произведения $a \times b$, что если $(x', y') \in f$, $(x'', y'') \in f$ и $x' = x''$, то $y' = y''$.

Областью определения отображения $D(f) \subseteq a$ называется множество тех $x \in a$, для которых существует такая пара $(x', y') \in f$, что $x = x'$.

Областью значений отображения $R(f) \subseteq b$ называется множество тех $y \in b$, для которых существует такая пара $(x', y') \in f$, что $y = y'$.

(Всюду определённое) *отображение* f из множества a в множество b удовлетворяет свойству $D(f) = a$.

Отображение обозначается как $f\colon a \to b$. Для пары $(x, y) \in f$ используются обозначения $f\colon x \mapsto y$ или $y = f(x)$. При этом y называется *образом* x, а x — *прообразом* y.

Заметим, что отображения чаще понимаются как соответствия между элементами множеств. Данные выше определение отображения задаёт при таком понимании *график отображения*.

Отображение $f\colon a \to b$ называется *сюръективным*, если $R(f) = b$.

Отображение $f\colon a \to b$ называется *инъективным*, если $(x', y') \in f$, $(x'', y'') \in f$ и $y' = y''$, то $x' = x''$.

Биективное (или взаимно однозначное) отображение (или биекция) одновременно сюръективно и

инъективно. Таким образом, если $f\colon a \to b$ — биективное отображение, то у любого элемента a есть ровно один образ (общее свойство всюду определённых отображений), а у любого элемента b — ровно один прообраз (свойства сюръективности и инъективности).

Упорядоченные множества

Отношение $r \subseteq a \times a$ называется *отношением частичного порядка*, если выполняются свойства *рефлексивности*: $(x, x) \in r$ для любого $x \in a$; *антисимметричности*: если $(x, y) \in r$ и $(y, x) \in r$, то $x = y$; *транзитивности*: если $(x, y) \in r$ и $(y, z) \in r$, то $(x, z) \in r$.

Частично упорядоченное множество (a, r) — это множество a и отношение частичного порядка r на a.

Элемент m подмножества q частично упорядоченного множества r называется *минимальным* в q, если нет такого элемента $x \in q$, что $x \neq m$ и $(x, m) \in r$.

Порядок r называется *линейным*, если для любых $x \neq y$ либо $(x, y) \in r$, либо $(y, x) \in r$.

Любое подмножество линейно упорядоченного множества имеет не более одного минимального элемента. Если любое непустое подмножество имеет ровно один минимальный элемент, то такое упорядоченное множество называется *вполне упорядоченным*.

В стандартной теории множеств справедлива *лемма Цорна*: любое множество можно вполне упорядочить.

Конечные и счётные множества, натуральные числа

Множества a и b называются *равномощными*, если существует биективное отображение $f\colon a \to b$.

Множество называется *конечным*, если оно не равномощно никакому своему собственному подмножеству, и *бесконечным* в противном случае.

Конечное множество можно задать списком его элементов, что обозначается как $\{a_1, a_2, \ldots, a_n\}$.

Примером бесконечного множества является множество натуральных чисел $\mathbb{N} = \{0, 1, 2, \ldots\}$. Основной операцией на множестве натуральных чисел является прибавление 1. С помощью этой операции на множестве натуральных чисел можно определить порядок: число x больше числа y, если y можно получить из x последовательными прибавлениями единицы. Данный порядок является вполне упорядочением, т. е. каждое подмножество множества натуральных чисел имеет наименьший элемент. Кроме того, любое натуральное число кроме 0 можно получить из 0 последовательным добавлением 1.

Бесконечное множество называется *счётным*, если оно равномощно множеству натуральных чисел. Биекция множества a и множества натуральных чисел называется *нумерацией*.

Можно дать эквивалентные определения конечного и счётного множества, используя упорядочения.

Множество конечно тогда и только тогда, когда любой линейный порядок на нём вполне упорядочен. Можно также доказать, что относительно любого линейного порядка на конечном множестве любое его непустое подмножество имеет как минимальный, так и максимальный элемент (аналогично минимальному, *максимальный* элемент m подмножества q частично упорядоченного множества (s, r) обладает тем свойством, что из $(m, x) \in r$ следует $m = x$).

Множество s счётно тогда и только тогда, когда существует такое его вполне упорядочение r, что для

любого элемента $x \in s$ конечно множество

$$\{y : (y, x) \in r\}$$

тех элементов y, которые не больше x.

Натуральные числа можно определить как счётное множество s с вполне упорядочением r указанного вида. При этом 0 — это минимальный элемент во всем множестве s, а $x + 1$ определяется как минимальный элемент в подмножестве тех элементов, которые больше x. Из указанного выше свойства порядка r следует, что для любого $x \in s$, $x \neq 0$, существует такой y, что $x = y+1$. Действительно, таким элементом y является максимальный элемент среди тех, которые меньше x (их конечное число для данного вполне упорядочения по определению и для любого ненулевого элемента это множество непусто, так как содержит 0).

Последовательности и слова

Начальный отрезок натурального ряда — это множество $[n]$, состоящее из тех элементов, которые меньше n. В этом множестве n элементов.

Конечная последовательность s элементов множества a — это отображение начального отрезка натурального ряда $[n]$ в множество a. Число n называется длиной последовательности. Мы обозначаем длину последовательности $|s|$.

Бесконечная последовательность s элементов множества a — это отображение натурального ряда в множество a.

Хотя последовательности определены как отображения, для них используется специальная терминология. Если $s\colon i \mapsto a_i$, то говорят, что *на i-м месте последовательности* (конечной или бесконечной) стоит символ a_i.

Конкатенация $u \circ v$ последовательностей u и v определяется следующим правилом:

$$u \circ v(i) = \begin{cases} u(i), & \text{если } i < |u|, \\ v(i - |u|), & \text{иначе.} \end{cases}$$

Конкатенация ассоциативна

$$u \circ (v \circ w) = (u \circ v) \circ w.$$

Знак конкатенации принято опускать, например, предыдущее соотношение обычно записывается как $u(vw) = (uv)w$.

Множество конечных последовательностей элементов a с операцией конкатенации образует *моноид*, единичным элементом которого является *пустое слово*, т. е. последовательность длины 0. Этот моноид обозначается a^* и его элементы называют также *словами* в алфавите a.

Если $w = uv$, то u называется *префиксом*, а v — *суффиксом* слова w.

Слово u называется *подсловом* слова w, если $w = v'uv''$ для некоторых слов v', v''.

Вхождением слова u в слово w называется такое число i, что $w = v'uv''$ и $|v'| = i$.

Отношения и функции нескольких переменных

Декартова степень a^n множества a состоит из всех последовательностей длины n из элементов a.

n-арное отношение r на множестве a — это подмножество a^n.

Функцией от n переменных на множестве a со значениями в множестве b называется отображение $f\colon a^n \to b$.

Графы и деревья

Ориентированный граф Γ — это два конечных множества V и E и отображение $i\colon E \to V \times V$. Элементы V называются *вершинами* графа, а элементы E — *рёбрами*.

Если $i(e) = (v_1, v_2)$, то v_1 называется *начальной вершиной* ребра, а v_2 — *конечной*. Говорят также, что ребро e *выходит* из вершины v_1 и *входит* в вершину v_2.

Неориентированный граф Γ — это два конечных множества V и E и отображение $i\colon E \to V^{(2)}$ из E в множество неупорядоченных пар элементов множества V.

Неупорядоченные пары можно определить как классы отношения эквивалентности $\sim$ на $V \times V$, для которого из $(x, y) \sim (u, v)$ следует $x = u$, $y = v$ или $x = v$, $y = u$. Неупорядоченные пары также обозначаются (x, y), при этом подразумевается, что $(x, y) = (y, x)$.

Если $i(e) = \{v_1, v_2\}$, то v_1, v_2 называются *концами* ребра. При это говорят, что ребро e *инцидентно* вершине v_i, $i = 1, 2$.

Рёбра, у которых начальная и конечная вершины (в случае неориентированного графа — концы ребра) совпадают, называются *петлями*.

Граф *не имеет параллельных рёбер*, если отображение i инъективно, т. е. ребро однозначно определяется своими начальной и конечной вершинами (концами в случае неориентированного графа).

Маршрутом τ в графе называется последовательность нечётной длины с элементами из $V \cup E$, такая что $\tau(2i) \in V$, $\tau(2i + 1) \in E$ и для любого $0 \leqslant i \leqslant (|\tau| - 3)/2$ выполняется условие: вершины $\tau(2i)$

и $\tau(2i+2)$ являются концами ребра $\tau(2i+1)$ (в случае ориентированного графа — начальной и конечной вершинами ребра).

Если первая и последняя вершины в маршруте совпадают, что такой маршрут называется *замкнутым*. Если все рёбра и вершины маршрута различны, то такой маршрут называют *путём*. Если все рёбра и вершины замкнутого маршрута различны, за исключением первой и последней вершин, то такой маршрут называют *циклом*.

Если для любых двух вершин v_1, v_2 в графе существует маршрут, первой вершиной которого является v_1, а последней — v_2, то такой неориентированный граф называют *связным* (*сильно связным* в случае ориентированного графа).

Связный неориентированный граф без циклов называется *деревом*. В дереве между любыми двумя вершинами есть ровно один путь.

Степенью вершины в неориентированном графе называют количество инцидентных ей рёбер. *Полустепенью исхода* вершины в ориентированном графе называют количество исходящих из неё рёбер. Аналогично *полустепенью захода* называют количество входящих в вершину рёбер.

Концевой вершиной называют вершину степени 1 в неориентированном графе.

Плоское корневое дерево имеет выделенную вершину (*корень*) и его можно нарисовать на координатной плоскости так, чтобы изображения рёбер не пересекались во внутренних точках, а путь от корня к любой вершине был направлен строго вниз (т. е. y-координата вдоль пути убывает). Комбинаторное определение плоского корневого дерева: это дерево, в котором выделен корень, а для каждой некорневой вершины

задано упорядочение рёбер, выходящих из этой вершины и не лежащих на пути из неё в корень (*нижние рёбра*). (Мы соотносим это определение с предыдущим так, чтобы упорядочение нижних рёбер указывало порядок слева направо, в котором идут изображения нижних рёбер.)

Верхним ребром называется ребро, которое выходит из данной вершины и лежит на пути в корень. *Листьями* плоского корневого дерева являются те его концевые некорневые вершины, из которых не выходит нижних рёбер.

Рекомендуемая литература

[1] А. Ахо, Дж. Хопкрофт, Дж. Ульман. Построение и анализ вычислительных алгоритмов. — М.: Мир, 1979.

[2] Х. Барнедрегт. Бестиповое λ-исчисление // Справочная книга по математической логике. Часть IV. Теория доказательств и конструктивная математика. — М.: Наука, 1983. С. 278–318.

[3] Н. В. Верещагин, А. Шень. Лекции по математической логике и теории алгоритмов. Часть 1. Начала теории множеств. — М.: МЦНМО, 2008.

[4] Н. В. Верещагин, А. Шень. Лекции по математической логике и теории алгоритмов. Часть 2. Языки и исчисления. — М.: МЦНМО, 2008.

[5] Н. В. Верещагин, А. Шень. Лекции по математической логике и теории алгоритмов. Часть 3. Вычислимые функции. — М.: МЦНМО, 2008.

[6] М. Девис. Неразрешимые проблемы // Справочная книга по математической логике. Часть III. Теория рекурсии. — М.: Наука, 1982. С. 51–76.

[7] Р. В. Душкин. Функциональное программирование на языке Haskell. — М.: ДМК Пресс, 2007.

[8] Ю. И. Журавлёв, Ю. А. Флёров, М. Н. Вялый. Дискретный анализ. Основы высшей алгебры. — М.: МЗ Пресс, 2007.

[9] Т. Кормен, Ч. Лейзерсон, Р. Ривест. Алгоритмы: построение и анализ. — М.: МЦНМО, 2000.

[10] К. Куратовский, А. Мостовский. Теория множеств. М.: Мир, 1970.

[11] А. И. Мальцев. Алгоритмы и рекурсивные функции. — М.: Наука, 1965.

[12] Ю. И. Манин. Вычислимое и невычислимое. М.: Советское радио, 1980.

[13] Ю. В. Матиясевич. Десятая проблема Гильберта. — М.: Наука. Физматлит, 1993.

[14] С. С. Марченков. Рекурсивные функции. — М.: Физматлит, 2007.

[15] Э. Мендельсон. Введение в математическоую логику. — М.: Наука, 1976.

[16] Х. Роджерс. Теория рекурсивных функций и эффективная вычислимость. — М.: Мир, 1972.

[17] В. А. Успенский. Теорема Гёделя о неполноте и четыре дороги, ведущие к ней. Материалы VII летней школы «Современная математика», 2007.
`http://www.mccme.ru/dubna/2007/notes/vau-dorogi.pdf`

[18] А. Филд, П. Харрисон. Функциональное программирование. — М.: Мир, 1993.

[19] Ч. Чень, Р. Ли. Математическая логика и автоматическое доказательство теорем. — М.: Наука, 1983.

[20] M. Minsky. Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines // Annals of Mathematics, 74, p. 437–455, 1961.

[21] M. Minsky. Computation: Finite and Infinite Machines. — Prentice-Hall International, 1967.

Журавлёв Юрий Иванович,
Флёров Юрий Арсеньевич,
Вялый Михаил Николаевич,

ДИСКРЕТНЫЙ АНАЛИЗ.
ФОРМАЛЬНЫЕ СИСТЕМЫ И АЛГОРИТМЫ

Москва ООО Контакт Плюс 2010

Учебное издание

Компьютерная верстка: *М.Н. Вялый*

Подписано в печать 05.07.2010 г. Формат 60x84/16
Бумага офсетная. Печать офсетная. Усл. печ. л. 21,0
Заказ № ____

В издательстве **«МЗ Пресс»**
при участии Московского физико-технического института
в серии «Естественные науки. Математика. Информатика.»
выпущены следующие книги:

1. ЛАБОРАТОРНЫЙ ПРАКТИКУМ ПО КУРСУ «ОСНОВЫ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ». *Иванов В.Д., Косарев В.И., Лобанов А.И., Петров И.Б. и др. 1-е изд.(2001), 2-е изд.(2003), 194 стр.*
2. ТЕОРИЯ И РЕАЛИЗАЦИЯ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ. Учебное пособие. *Серебряков В.А., Галочкин М.П., Гончар Д.Р., Фуругян М.Г., 1-е изд., 2003, 296 стр.*
3. ОСНОВЫ ТЕОРИИ СЛУЧАЙНЫХ ПРОЦЕССОВ. Учебное пособие. *Натан А.А., Горбачев О.Г., Гуз С.А., 2003, 168 стр.*
4. ОПТИМИЗАЦИЯ. ЭЛЕМЕНТЫ ТЕОРИИ. ЧИСЛЕННЫЕ МЕТОДЫ. Учебное пособие. *Бирюков С.И., 2003, 248 стр.*
5. ВВЕДЕНИЕ В НЕЛИНЕЙНУЮ ФИЗИКУ ПЛАЗМЫ. Учебное пособие. *Кингсеп А.С., 2004, 264 стр.*
6. РЯДЫ ФУРЬЕ.ИНТЕГРАЛЫ, ЗАВИСЯЩИЕ ОТ ПАРАМЕТРА, И ОБОБЩЕННЫЕ ФУНКЦИИ. Курс лекций. *Черняев А.П., 2004, 149 стр.*
7. МАТЕМАТИЧЕСКАЯ СТАТИСТИКА. *Натан А.А., Горбачев О.Г., Гуз С.А., 2005, 160 стр.*
8. СОВРЕМЕННЫЕ ПРОБЛЕМЫ ПРИКЛАДНОЙ МАТЕМАТИКИ. Сборник научно-популярных статей. *П/р Петрова А.А., 2005, 232 стр.*
9. ОПЫТ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ПРИ АНАЛИЗЕ СОЦИАЛЬНО-ЭКОНОМИЧЕСКИХ ЯВЛЕНИЙ. *Павловский Ю.Н., Белотелов Н.В., Бродский Ю.И., Оленев Н.Н., 2005, 136 стр.*

10. СТРУКТУРНАЯ СОГЛАСОВАННОСТЬ ДАННЫХ И ЗНАНИЙ. Учебное пособие. *Дулин С.К., 2005, 144 стр.*

11. ТЕОРИЯ И РЕАЛИЗАЦИЯ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ. Учебное пособие. *Серебряков В.А., Галочкин М.П., Гончар Д.Р., Фуругян М.Г., 2-е изд., допол. и испр., 2006, 352 стр.*

12. ДИСКРЕТНЫЙ АНАЛИЗ. ОСНОВЫ ВЫСШЕЙ АЛГЕБРЫ. Учебное пособие. *Журавлев Ю.И., Флёров Ю.А., Вялый М.Н., 2006, 208 стр.*

13. СИММЕТРИИ УРАВНЕНИЙ ГАМИЛЬТОНА И ЛАГРАНЖА. *Яковенко Г.Н., 2006, 120 стр.*

14. МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ ГАЗОДИНАМИЧЕСКИХ ПРОЦЕССОВ С ИСТОЧНИКАМИ. *Волосевич П.П., Ермолин Е.В., Леванов Е.И., 2006, 216 стр.*