А. С. ГЕРАСИМОВ

КУРС МАТЕМАТИЧЕСКОЙ ЛОГИКИ И ТЕОРИИ ВЫЧИСЛИМОСТИ

УЧЕБНОЕ ПОСОБИЕ

Издание третье, исправленное и дополненное

Издательство «ЛЕМА»
Санкт-Петербург
2011

УДК 510.6+510.2+510.5+512.54.05+004.05
ББК 22.12
Г37

Герасимов А. С. Курс математической логики и теории вычислимости: Учебное пособие. 3-е изд., испр. и доп. — СПб.: Издательство «ЛЕМА», 2011. — 284 с.

ISBN 978-5-98709-292-7

Настоящее учебное пособие предназначено для изучения математической логики и теории алгоритмов. В нём описаны языки логики высказываний и логики предикатов первого порядка, семантика этих языков. На основе общего понятия исчисления изложены исчисления гильбертовского типа, секвенциальные исчисления и метод резолюций как способы формального математического доказательства. Рассмотрены основные формальные аксиоматические теории — элементарная арифметика и теория множеств Цермело-Френкеля. Теория алгоритмов представлена теорией вычислимости, в рамках которой дано несколько точных определений понятия алгоритма и доказана неразрешимость некоторых проблем. Дополнительная глава посвящена исчислению для формального доказательства правильности программ некоторого императивного языка программирования. В данной книге имеется более 190 упражнений.

Это учебное пособие адресовано в первую очередь студентам, специализирующимся по информатике, но будет полезно студентам разных математических специальностей (направлений подготовки), а также всем желающим начать систематическое изучение математической логики.

Александр Сергеевич Герасимов
Курс математической логики и теории вычислимости

Подписано в печать 12.01.2011.
Формат 60 × 84/16. Бумага офсетная.
Тираж 25 экз. Усл. п. л. 16,7.
Заказ N 1865.

Отпечатано в ООО «Издательство "ЛЕМА"»
199004, Россия, Санкт-Петербург,
В. О., Средний пр., д. 24, тел./факс: 323-67-74
email: izd_lema@mail.ru

ISBN 978-5-98709-292-7

Оглавление

Моим родителям:
Сергею Петровичу и
Анастасии Ивановне

Предисловие

Настоящее учебное пособие предназначено для изучения математической логики и смежной с ней области — теории вычислимости. Эта книга адресована в первую очередь студентам, специализирующимся по информатике, но будет полезна студентам разных математических специальностей (направлений подготовки), а также всем желающим начать систематическое изучение указанных областей математики. Традиционные названия дисциплин, для изучения которых можно использовать это учебное пособие, — «Математическая логика», «Математическая логика и теория алгоритмов». Теория вычислимости, также называемая общей теорией алгоритмов, является одним из основных разделов теории алгоритмов (наряду с более молодым разделом — теорией сложности вычислений).

Математическая логика — наука о принятых в математике способах рассуждений. Эта наука определяет, что считается доказательством в математике, тем самым формируя основания математики. Даваемые математической логикой определения доказательства настолько точны, что могут быть использованы для проверки и поиска доказательств с помощью компьютера. Это позволяет широко применять математическую логику в информатике, например, разрабатывать экспертные системы — программы, моделирующие рассуждения эксперта в какой-либо области знаний.

Информатика является наукой, близкой к математике, и занимается, в частности, точным описанием (формализацией) знаний, причём такое описание должно быть настолько точно, чтобы его понимал компьютер. Данная книга учит формальным языкам и представлению знаний с их помощью, формальным доказательствам и их алгоритмизации; таким образом, эта книга учит осуществлять формализацию знаний. Также читатель познакомится с теоретическими основами одной из парадигм декларативного программирования — логическим программированием; знание этих основ поможет эффективно использовать языки логического программирования, например, язык Prolog.

Теория вычислимости произошла из математической логики в результате попыток осуществить желание математиков иметь алгоритм доказательства математических утверждений, записанных на формальном логическом языке. В посвящённой теории вычислимости главе этой книги читатель познакомится с точными определениями алгоритма; каждое из них по сути задаёт язык программирования. Это знакомство крайне полезно для развития программистского опыта, поскольку некоторые из точных определений алгоритма легли в основу современных парадигм программирования, например, императивной и функциональной парадигм. Также в теории вычислимости раскрыты неустранимые ограничения алгоритмиче-

ского решения задач: доказано, что для решения некоторых важных задач не существует никакого алгоритма.

В последней главе книги рассматривается основополагающий метод формального доказательства правильности программ. Изучение этого метода не только развивает умение убеждаться в правильности программ (или находить в них ошибки) путём рассуждений, но и подготавливает читателя к использованию и разработке технологий программирования, предполагающих проверку правильности программ посредством доказательства.

Итак, данная книга создана с целью существенно способствовать качественной подготовке математиков, исследователей в области информатики, программистов, умеющих осуществлять формализацию знаний.

С более детальным содержанием этого учебного пособия можно ознакомиться в оглавлении и в аннотациях, с которых начинается каждая глава и приложение. Характерной особенностью изложения является определение общего понятия исчисления как, грубо говоря, некоторого механического способа получения заключений из ранее установленных или принятых в качестве аксиом утверждений. Это обеспечивает единый подход к различным способам формализации математических доказательств. В данной книге изучаются исчисления пяти принципиально различных видов: исчисления гильбертовского типа, секвенциальные и резолютивные исчисления, лямбда-исчисление и исчисление Хóара.

Излагаемый материал представляет собой лишь введение в соответствующие разделы математической логики и теории вычислимости. Однако автор уверен, что освоивший эту книгу читатель при желании будет готов к изучению более специальной литературы по математической логике, теории алгоритмов и их применениям в информатике.

Для изучения данной книги требуется знание начальных понятий наивной теории множеств (таких, как отношение, функция, отношение эквивалентности, классы эквивалентности по отношению эквивалентности, счётность множества), знание метода математической индукции, знание понятий полинома, полугруппы, группы, дерева, а также желательно знакомство с каким-нибудь языком программирования (например, с языком Pascal). Все эти знания студенты обычно получают из курсов алгебры, математического анализа и информатики в течение первого семестра обучения на математических факультетах университетов.

Автор благодарен С. П. Герасимову за поддержку и А. И. Герасимовой за поддержку и нахождение множества опечаток в первом издании этого учебного пособия.

Также автор выражает благодарность студентам математико-механического факультета СПбГУ, которым он преподавал и которые указали опечатки в первом издании данной книги.

А. С. Герасимов
Email: `Alexander.S.Gerasimov@ya.ru`

Начальные соглашения об обозначениях

Знаком $\triangleleft$ отмечается конец доказательства (а также определения, примера, замечания и упражнения), начинающегося соответствующим заголовком.

Принадлежность элемента x множеству X записывается как $x \in X$. Если неверно, что $x \in X$, то пишем $x \notin X$. Пустое множество обозначаем $\varnothing$. $\{x_1, \ldots, x_n\}$ есть множество, элементами которого являются в точности $x_1, \ldots, x_n$. Сходным образом можно записывать множество $\{x_1, \ldots, x_n, \ldots\}$, когда указан или общепринят способ получения элементов, подразумеваемых под последним многоточием. $\{x \in X \mid \Pi(x)\}$ есть множество всех элементов x множества X, для которых верно утверждение $\Pi(x)$.

Объединение, пересечение и разность множеств X и Y обозначаются $X \cup Y$, $X \cap Y$ и $X \setminus Y$ соответственно. $\cup_{i \in I} X_i$ есть объединение всех множеств X_i, где $i \in I$. $X \subseteq Y$ означает, что множество X является подмножеством множества Y.

Под $f : X \to Y$ понимается, что f является функцией (отображением) из множества X в множество Y. Область определения и область значений функции f обозначаются через $dom(f)$ и $rng(f)$ соответственно.

Множество $X_1 \times \ldots \times X_n$ (где $n \geqslant 1$) является декартовым произведением множеств $X_1, \ldots, X_n$, т. е. множеством всех упорядоченных n-ок $\langle x_1, \ldots, x_n \rangle$ элементов $x_1 \in X_1, \ \ldots, \ x_n \in X_n$. Множество X^n есть n-ая декартова степень множества X, иначе говоря, X^n есть $X_1 \times \ldots \times X_n$ при $X = X_1 = \ldots = X_n$, т. е. множество всех упорядоченных n-ок элементов множества X.

$\mathbb{N}$ обозначает множество всех натуральных чисел $\{0, 1, 2, \ldots\}$. Множество всех отличных от нуля натуральных чисел $\{1, 2, \ldots\}$ мы обозначаем через $\mathbb{N}_+$. Знак $\Rightarrow$ заменяет слово «влечёт», а $\Leftrightarrow$ — слова «тогда и только тогда, когда».

Остальные обозначения общеупотребительны (как, впрочем, и некоторые из вышеприведённых обозначений) или вводятся в тексте по мере необходимости. Подробный список обозначений является частью предметного указателя.

Греческий алфафит

Приведём греческий алфавит с названиями букв. (Далее в этом учебном пособии мы будем использовать только те буквы греческого алфавита, которые по начертанию отличны от букв латинского алфавита.)

$A,\ \alpha$ — альфа	$N,\ \nu$ — ню
$B,\ \beta$ — бета	$\Xi,\ \xi$ — кси
$\Gamma,\ \gamma$ — гамма	$O,\ o$ — омикрон
$\Delta,\ \delta$ — дельта	$\Pi,\ \pi$ — пи
$E,\ \varepsilon$ — епсилон	$P,\ \rho$ — ро
$Z,\ \zeta$ — дзета	$\Sigma,\ \sigma$ — сигма
$H,\ \eta$ — эта	$T,\ \tau$ — тау
$\Theta,\ \theta$ — тета	$\Upsilon,\ \upsilon$ — ипсилон
$I,\ \iota$ — йота	$\Phi,\ \varphi$ — фи
$K,\ \kappa$ — каппа	$X,\ \chi$ — хи
$\Lambda,\ \lambda$ — лямбда	$\Psi,\ \psi$ — пси
$M,\ \mu$ — мю	$\Omega,\ \omega$ — омега

Нестандартные варианты написания букв латинского алфавита

Укажем (более или менее) нестандартные варианты написания некоторых букв латинского алфавита, используемые в этой книге.

A — $\mathfrak{A}$	M — $\mathcal{M}$
B — $\mathbb{B}$, $\mathfrak{B}$	N — $\mathbb{N}$
C — $\mathcal{C}$	P — $\mathcal{P}$
D — $\mathcal{D}$	Q — $\mathcal{Q}$
F — $\mathcal{F}$	R — $\mathcal{R}$, $\mathbb{R}$
G — $\mathcal{G}$	S — $\mathcal{S}$, $\mathbb{S}$, $\mathfrak{S}$
H — $\mathcal{H}$	Z — $\mathbb{Z}$

Глава 1.

Исчисления высказываний

Под *высказыванием* мы понимаем повествовательное предложение, которое является либо истинным, либо ложным, но не тем и другим сразу. *Логика высказываний*, или *пропозициональная логика*, изучает способы математических рассуждений о высказываниях. Высказывания могут быть простыми (неделимыми) или составными. Примеры простых высказываний: «4 — чётное число», «5 — нечётное число», «идёт дождь», «я прогуливаюсь». Примеры составных высказываний: «4 — чётное число *и* 5 — нечётное число», «*если* идёт дождь, *то* я *не* прогуливаюсь». В этих примерах курсивом выделены слова, связывающие высказывания для образования составных высказываний. Из последнего составного высказывания и высказывания «идёт дождь» можно заключить, что «я *не* прогуливаюсь»; причём это рассуждение считается правильным. Из того же составного высказывания и высказывания «я прогуливаюсь», можно вывести «*не* идёт дождь»; и это рассуждение мы считаем правильным. А какие ещё рассуждения (о высказываниях) считать правильными, можно ли их как-нибудь точно описать? Основной целью данной главы является ответ на этот вопрос.

1.1. Пропозициональные формулы и булевы функции

Для изучения способов рассуждений о произвольных высказываниях мы введём обозначения для высказываний и для слов, связывающих высказывания, задав тем самым некоторый формальный язык. Кроме таких обозначений, существенной особенностью этого формального языка, отличающей его от естественного (например, русского) языка, будет однозначность его понимания. Мы укажем, как можно определить, является ли записанное на этом формальном языке высказывание истинным или ложным, и рассмотрим некоторые свойства таких высказываний.

1.1.1. Формальные языки

Мы запланировали ввести некоторый формальный язык для записи высказываний. Однако сначала будет полезно определить общее понятие

формального языка и сопутствующие понятия.

Под *символом*, или *буквой*, понимается знак, который рассматривается как единое целое. Понятие символа является исходным и не определяется, а лишь поясняется. Предполагается, что про любые два символа можно с уверенностью сказать ровно одно из двух: одинаковы они или различны. Тот факт, что два символа, обозначаемые через a_1 и a_2, одинаковы, мы будем записывать как $a_1 = a_2$.

Алфавитом называется произвольное непустое конечное множество символов.

Словом в алфавите Σ называется конечная цепочка (в том числе цепочка, не содержащая ни одного символа) записанных подряд символов этого алфавита, а число символов (не обязательно различных) в этой цепочке называется *длиной этого слова*. Цепочка, не содержащая ни одного символа, называется *пустым словом*.

Пусть $a_1, \ldots, a_m, b_1, \ldots, b_n$ — (не обязательно различные) символы некоторого алфавита, перечисленные через запятую. Будем говорить, что два слова $a_1 \ldots a_m$ и $b_1 \ldots b_n$ *равны* (*совпадают* или *одинаковы*) и писать $a_1 \ldots a_m = b_1 \ldots b_n$, если $m = n$ и $a_i = b_i$ для каждого $i = 1, \ldots, m$. В противном случае будем говорить, что эти слова *неравны* (*не совпадают* или *различны*).

Конкатенацией слов α и β называется слово $\alpha\beta$, получающееся приписыванием слова β к слову α. Говорят, что слово $\alpha\beta$ *начинается* со слова α, и слово α называют *началом* слова $\alpha\beta$.

Говорят, что слово α *входит* в слово β (α является *подсловом* слова β), если найдутся такие слова γ и δ, что $\beta = \gamma\alpha\delta$. При этом, если длина слова γ равна k, то будем говорить, что имеется *k-вхождение* слова α в слово β. Для любого $n \in \mathbb{N}$ n-вхождение слова α в слово β будем называть *вхождением* слова α в слово β. Вхождения (если они есть) слова α в слово β естественным образом упорядочены (k-вхождение предшествует m-вхождению, если $k < m$), поэтому можно говорить о первом, втором и т. д. вхождении α в β.

Пусть имеется k-вхождение слова α в слово β, т. е. существуют такие слова γ и δ, что $\beta = \gamma\alpha\delta$ и длина слова γ равна k. Тогда *результатом замены этого вхождения слова α в слово β на слово χ* (*результатом подстановки слова χ вместо этого вхождения слова α в слово β*) назовём слово $\gamma\chi\delta$.

Пример 1.1.1. Пусть рассматривается алфавит $\{A, a, B, b, C, c, \ldots, Z, z\}$.[1]

Слово *mathematics* имеет длину 11. Имеется ровно 2 вхождения слова *ma* в слово *mathematics*: 0-вхождение и 5-вхождение. Результатом замены 5-вхождения слова *ma* в слово *mathematics* на *FGH* является слово *matheFGHtics*.

Пустое слово имеет длину 0 и входит ровно 1 раз в пустое слово, причём это единственное вхождение является 0-вхождением.

Имеется ровно 2 вхождения слова *aba* в слово *ababa*: 0-вхождение и 2-вхождение. ◁

Множество всех слов в алфавите Σ обозначается через Σ^*. (Отметим, что пустое слово принадлежит Σ^*.)

[1] Для экономии места мы не перечисляем все элементы этого множества, полагая, что это множество состоит в точности из всех заглавных и всех строчных английских букв.

Формальным языком (или просто *языком*) *в алфавите* Σ называется какое угодно подмножество множества Σ^*.

Пример 1.1.2. Пусть алфавитом LettersAndDigits является множество символов $\{A, a, B, b, C, c, \ldots, Z, z, 0, 1, 2, \ldots, 9\}$. Определим язык Identifiers в этом алфавите как множество всех слов в алфавите LettersAndDigits, которые начинаются с любого символа, принадлежащего множеству $\{A, a, B, b, C, c, \ldots, Z, z\}$. Слова $Clause33, dvgjf683Xitr$ принадлежат языку Identifiers, а пустое слово и слово $73abc$ — нет.

Предпочтительнее задать язык Identifiers с помощью следующего индуктивного определения:

1) любой символ из $\{A, a, B, b, C, c, \ldots, Z, z\}$ является словом языка Identifiers;

2) если β — слово языка Identifiers, и γ — символ алфавита LettersAndDigits, то $\beta\gamma$ (т. е. конкатенация β и γ) является словом языка Identifiers.

Мы отдаём предпочтение этому индуктивному определению, поскольку оно описывает процесс построения (или конструирования) слов из символов по заданным правилам. ◁

Итак, мы ввели общее понятие (формального) языка. Теперь определим язык для записи высказываний.

1.1.2. Язык логики высказываний

Определение 1.1.3. Обозначим через $Vars$ язык $\{V|V,\ V||V,\ V|||V,\ \ldots\}$ в алфавите $\{V, |\}$. Каждое слово языка $Vars$ будем называть *пропозициональной переменной* (или, короче, *переменной*). *Пропозициональная формула* (*формула логики высказываний* или просто *формула*) задаётся следующим индуктивным определением:

1) любая пропозициональная переменная является пропозициональной формулой;

2) символы $\mathbf{F}$ и $\mathbf{T}$ являются пропозициональными формулами и называются *истинностными константами* (*ложь* и *истина* соответственно);

3) если A и B — пропозициональные формулы, то $\neg A$, $(A \wedge B)$, $(A \vee B)$ и $(A \supset B)$ — пропозициональные формулы.

Множество всех пропозициональных формул обозначим через PL и назовём *языком логики высказываний* (или *пропозициональным языком*).[2] ◁

Таким образом, алфавит языка PL есть множество символов $\{V, |, \mathbf{F}, \mathbf{T}, (,), \neg, \wedge, \vee, \supset\}$.

Пример 1.1.4. Разделённые запятыми слова

$$V||V,\ \neg V|V,\ (\neg\mathbf{T} \vee V|V),\ (\neg V|V \vee V|V),\ (\neg(V|V \vee V||||V) \wedge V||V)$$

[2] PL является сокращением словосочетания «Propositional Language».

являются пропозициональными формулами. Ни одно из слов

$$(\neg V|V \vee V||V,\ (\neg V|V - V||V),\ (\neg V|V \vee Z),\ \neg V|V \vee V||V$$

не является пропозициональной формулой. ◁

Замечание 1.1.5. Поясним, почему мы выбрали в качестве пропозициональных переменных несколько неестественные слова, например, $V|V$ и $V||V$, вместо немного более естественных слов $V|$ и $V||$. Но тогда, если бы мы заменили все вхождения $V|$ в $(V| \wedge V||)$ на A, то получили бы $(A \wedge A|)$. Нам же, как правило, будет удобно без дополнительных оговорок заменять все вхождения $V|V$ в $(V|V \wedge V||V)$ на A и получать в результате $(A \wedge V||V)$. Поэтому пропозициональные переменные мы выбрали так, что ни одна из них не входит ни в какую отличную от неё переменную. ◁

В дальнейшем изложении для удобства записи мы будем обозначать произвольную пропозициональную переменную словом из букв латинского алфавита, которое начинается с одной из букв p, q, r, причём такое слово может иметь индекс, являющийся натуральным числом. Например, обозначениями пропозициональных переменных являются следующие слова: $p, p_1, pf, pf_2, q_0, r, r_{2007}$. С такими обозначениями переменных мы будем считать формулами, например, следующие слова: p_3, $\neg p$, $((p \vee q) \supset p)$. Условимся, что в данной главе в каждой записи формулы различны переменные, имеющие различные обозначения, если явно не оговорено другое. Также условимся, что в данной главе буквы A, B, C, D, E и G (возможно, с индексами) будут обозначать произвольные пропозициональные формулы, если иное не оговорено явно.

Символы $\neg$ (*отрицание*), $\wedge$ (*конъюнкция*), $\vee$ (*дизъюнкция*) и $\supset$ (*импликация*) называются *логическими связками* (иногда просто *связками*). Будем говорить, что отрицание является *унарной* связкой, а остальные из этих связок являются *бинарными* связками. (Связка названа унарной или бинарной по числу формул, которые она связывает в одну формулу.)

Укажем, как обычно логические связки представляются словами русского языка в речи математиков:

1) $\neg A$ — «отрицание A», «не A», «неверно, что A»;

2) $(A \wedge B)$ — «конъюнкция A и B», «A и B»;

3) $(A \vee B)$ — «дизъюнкция A и B» (здесь союз «и» означает перечисление, а не конъюнкцию), «A или B»;

4) $(A \supset B)$ — «импликация с *посылкой* A и *заключением* B», «если A, то B», «B, если A», «A, только (лишь) если B», «A влечёт B», «из A следует B», «B является следствием A», «B имеет место при A», «B имеет место в случае A», «A только (лишь) при B», «A только (лишь) в случае B», «A достаточно для B», «B необходимо для A», «A сильнее B», «B слабее A».

Индуктивный характер определения пропозициональной формулы позволяет использовать следующий метод доказательства того, что любая формула C обладает свойством $\mathcal{P}$. Для этого достаточно доказать

1) базу индукции: любая переменная и любая истинностная константа обладает свойством $\mathcal{P}$; и

2) индукционный переход: из предположения (называемого индукционным предположением) того, что формулы A и B обладают свойством $\mathcal{P}$, следует, что $\neg A$, $(A \wedge B)$, $(A \vee B)$ и $(A \supset B)$ обладают свойством $\mathcal{P}$.

Этот метод доказательства называют *индукцией по построению формулы* C, а также *индукцией по построению множества формул* PL.

Докажем, например, что в любой формуле C число вхождений левых скобок «(» равно числу вхождений правых скобок «)».

Доказательство. Обозначим утверждение «в формуле C число вхождений левых скобок равно числу вхождений правых скобок» через $\mathcal{P}[C]$. Доказательство того, что для любой формулы C верно $\mathcal{P}[C]$, проведём индукцией по построению формулы C.

База индукции. Для любой формулы C, являющегося пропозициональной переменной или истинностной константой, выполняется $\mathcal{P}[C]$, поскольку в C имеется 0 вхождений левых и 0 вхождений правых скобок.

Индукционный переход. Предположим, что для формулы A верно $\mathcal{P}[A]$, и для формулы B верно $\mathcal{P}[B]$. Тогда $\mathcal{P}[\neg A]$ выполняется, поскольку при построении $\neg A$ из A ни одно вхождение скобки не добавляется, и $\mathcal{P}[A]$ верно по индукционному предположению. $\mathcal{P}[(A \wedge B)]$ выполняется потому, что при построении $(A \wedge B)$ из A и B добавляется ровно одно вхождение левой скобки и ровно одно вхождение правой скобки, а $\mathcal{P}[A]$ и $\mathcal{P}[B]$ верны по индукционному предположению. Аналогично верны $\mathcal{P}[(A \vee B)]$ и $\mathcal{P}[(A \supset B)]$. ◁

Упражнение 1.1.6. Докажите, что в любой пропозициональной формуле число вхождений бинарных логических связок не превосходит числа вхождений переменных. ◁

Индуктивный характер определения пропозициональной формулы также позволяет задавать функции, определённые на множестве формул PL, *индукцией* (или *рекурсией*) *по построению множества формул* PL, а именно, следующим образом:

1) пусть каждой переменной p сопоставлен объект $\mathcal{F}[p]$, и каждой истинностной константе Q сопоставлен объект $\mathcal{F}[Q]$;

2) пусть задано правило такое, что если формулам A и B сопоставлены объекты $\mathcal{F}[A]$ и $\mathcal{F}[B]$, то однозначно находится каждый из объектов $\mathcal{F}[\neg A]$, $\mathcal{F}[(A \wedge B)]$, $\mathcal{F}[(A \vee B)]$ и $\mathcal{F}[(A \supset B)]$.

Тогда для любой формулы A определён объект $\mathcal{F}[A]$.

Упражнение 1.1.7. Для каждой формулы A определим её *логическую сложность* $l[A]$ следующим образом (а именно, индукцией по построению множества формул PL): 1) $l[A] = 0$, если A — переменная или истинностная константа; 2) $l[\neg A] = l[A] + 1$, $l[(A \odot B)] = l[A] + l[B] + 1$, где A, B — формулы, $\odot$ — одна из связок $\wedge$, $\vee$, $\supset$. Докажите, что $l[A]$ равно числу вхождений логических связок в формулу A. ◁

Определим бинарную логическую связку $\equiv$ (*эквивалентность*), являющуюся вспомогательной. Пусть A и B — формулы. Тогда $(A \equiv B)$ служит сокращённой записью формулы $((A \supset B) \wedge (B \supset A))$. Формуле $(A \equiv B)$ обычно соответствуют следующие выражения русского языка: «A, если и

только если B», «A тогда и только тогда, когда B», «A равносильно B», «A эквивалентно B», «для A необходимо и достаточно B».

Введём нередко используемое сокращение: $\Psi \leftrightharpoons \Phi$ заменяет слова «Ψ по определению есть Φ» (или слова «Φ, обозначаемое через Ψ,»), где Φ — ранее определённое понятие, а Ψ определяется данным сокращением. Ниже мы будем нередко пользоваться таким сокращением, а также понимать под записью $\Psi \leftrightharpoons \Phi \leftrightharpoons \Upsilon$, что сначала вводится обозначение $\Phi \leftrightharpoons \Upsilon$, а затем — $\Psi \leftrightharpoons \Phi$.

С использованием только что введённого сокращения определение логической связки $\equiv$ можно переписать так: $(A \equiv B) \leftrightharpoons ((A \supset B) \wedge (B \supset A))$.

Наконец, дадим ещё одно определение, используемое в дальнейшем, и договоримся о сокращении числа скобок в записи формул.

Формула, входящая в формулу A, называется *подформулой* формулы A.

Для более экономной и иногда более наглядной записи формул примем соглашения об опускании некоторых скобок.

Во-первых, можно опускать пару внешних скобок в записи отдельно стоящей формулы. Например, формулу $(p \vee \neg q)$ можно записать в виде $p \vee \neg q$.

Во-вторых, будем считать, что каждая бинарная логическая связка левоассоциативна. Это означает, что в произвольной формуле любую подформулу вида $((A \odot B) \odot C)$ можно записать в виде $(A \odot B \odot C)$, где все выделенные вхождения $\odot$ являются вхождениями какой угодно фиксированной бинарной связки. Например, формулу $p \wedge ((q \vee (r \supset \neg p)) \vee r)$ можно записать как $p \wedge (q \vee (r \supset \neg p) \vee r)$.

В-третьих, будем считать, что приоритет связок в последовательности $\wedge$, $\vee$, $\supset$, $\equiv$ строго убывает. Подробнее, пусть $\odot$ и $\oplus$ обозначают такие связки, что приоритет $\odot$ выше, чем приоритет $\oplus$ (т. е. $\odot$ стоит в указанной последовательности левее $\oplus$). Тогда любую подформулу вида $((A \odot B) \oplus C)$ можно записать как $(A \odot B \oplus C)$, а также любую подформулу вида $(A \oplus (B \odot C))$ можно записать как $(A \oplus B \odot C)$. Например, формулу $((p \wedge q) \supset r) \wedge p$ можно записать как $(p \wedge q \supset r) \wedge p$, а формулу $(p \vee (q \wedge r)) \wedge p$ — как $(p \vee q \wedge r) \wedge p$.

Пример 1.1.8. Формула $((((p \vee \neg q) \supset r) \supset \neg(p \wedge q)))$ может быть записана в виде $p \vee \neg q \supset r \supset \neg(p \wedge q)$, причём в этой записи уже нельзя опустить пару скобок.

Отметим, что $\neg p \supset p$ является, а $p \supset q$ не является подформулой формулы $\neg p \supset p \supset q$, поскольку последняя формула подробнее записывается как $(\neg p \supset p) \supset q$.

$\neg p$ является, а $\neg(p \wedge q)$ не является подформулой формулы $\neg p \wedge q \vee r$, поскольку последняя может быть подробнее записана как $(\neg p \wedge q) \vee r$. Вхождение логической связки $\neg$ относится к минимальной по длине подформуле, стоящей сразу за рассматриваемым вхождением $\neg$. $\triangleleft$

Упражнение 1.1.9. Разработайте алгоритм восстановления опущенных скобок в любой заданной сокращённой записи формулы. $\triangleleft$

1.1.3. Семантика языка логики высказываний. Логические законы

До сих пор пропозициональные формулы были для нас лишь специальным образом построенными цепочками символов, лишёнными какого-либо содержательного значения (смысла). В данном разделе мы придадим пропозициональным формулам некоторое содержательное значение, иначе говоря, зададим семантику языка PL логики высказываний, а также изучим некоторые семантические свойства этого языка.

Рассмотрим следующее высказывание математика: «У меня с собой зонт, если я не знаю прогноз погоды, или я знаю прогноз погоды, и прогнозируются дожди». Нам понятен его смысл: если мы знаем, истинно или нет каждое простое высказывание в составе этого составного высказывания, то мы сможем определить, говорит ли этот математик правду (истину). Для удобства введём обозначения: $pu \leftrightharpoons$ «у меня (этого математика) с собой зонт», $pf \leftrightharpoons$ «я знаю прогноз погоды», $pr \leftrightharpoons$ «прогнозируются дожди», всё исходное высказывание обозначим через A.

Если мы знаем, что pf, pr и pu все истинны, то мы считаем, что математик говорит правду, т. е. высказывание A истинно. Если же мы знаем, что pf и pr истинны, а pu ложно, то мы считаем, что математик говорит ложь, т. е. высказывание A ложно.

Из предыдущего раздела мы знаем, какие логические связки соответствуют словам «если», «не», «или», «и», поэтому мы можем записать A в виде формулы $(\neg pf \vee (pf \wedge pr)) \supset pu$, где pf, pr и pu играют роль пропозициональных переменных. Расстановка скобок в этой формуле соответствует интуитивному смыслу высказывания A, в частности, pf и pr связаны союзом «и» сильнее, чем $\neg pf$ и pf союзом «или». Рассуждения, приведённые выше, интуитивно показывают, что можно определить, истинна ли формула A, если известно, истинна ли каждая входящая в A переменная. Такой подход мы и приме́ним по отношению к любой пропозициональной формуле.

Если высказывание истинно, то также говорят, что это высказывание имеет *истинностное значение* 1 (*истина*). Если высказывание ложно, то также говорят, что это высказывание имеет *истинностное значение* 0 (*ложь*). Множество $\{0, 1\}$ всех истинностных (или булевских) значений обозначим через $\mathbb{B}$.

Для того, чтобы любая пропозициональная формула получила истинностное значение, прежде всего каждой пропозициональной переменной сопоставим истинностное значение с помощью так называемой интерпретации.

Определение 1.1.10. *Интерпретацией* (или *моделью*) *языка* PL называется какое угодно отображение $M : Vars \to \mathbb{B}$, которое каждой переменной p сопоставляет истинностное значение $M(p)$. ◁

Далее, предположим, что формулы A и B получили истинностные значения α и β соответственно. Зададим, как по истинностным значениям α и β вычисляются истинностные значения формул $\neg A$, $(A \wedge B)$, $(A \vee B)$ и $(A \supset B)$. Для этого определим одноместную функцию $F_\neg$ из множества $\mathbb{B}$ в $\mathbb{B}$ и двухместные функции $F_\wedge$, $F_\vee$, $F_\supset$ из множества $\mathbb{B}^2$ в $\mathbb{B}$ с помощью таблиц, изображённых на рис. 1.1. Эти функции (таблицы) назовём *функцией*

(*таблицей*) *истинности отрицания, конъюнкции, дизъюнкции* и *импликации* соответственно. Теперь положим, что истинностные значения формул

α	$F_\neg(\alpha)$
0	1
1	0

α	β	$F_\wedge(\alpha,\beta)$	$F_\vee(\alpha,\beta)$	$F_\supset(\alpha,\beta)$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	1

Рис. 1.1. Таблицы, определяющие функции $F_\neg$, $F_\wedge$, $F_\vee$ и $F_\supset$.

$\neg A$, $(A \wedge B)$, $(A \vee B)$ и $(A \supset B)$ есть $F_\neg(\alpha)$, $F_\wedge(\alpha,\beta)$, $F_\vee(\alpha,\beta)$ и $F_\supset(\alpha,\beta)$ соответственно.

Замечание 1.1.11. В соответствии с определением функции $F_\supset$, импликация $A \supset B$ ложна, если и только если одновременно формула A истинна и формула B ложна; в остальных случаях эта импликация истинна: если A истинна и B истинна, а также, если A ложна (B может быть как истинна, так и ложна). Возможно, вызывает сомнение то, что при ложной посылке A мы посчитали импликацию истинной вне зависимости от истинностного значения заключения B.

Чтобы развеять это сомнение, для любого натурального числа k рассмотрим такое высказывание: «Если k больше 5, то k больше 2». Для каждого k эту импликацию мы, конечно, хотим считать истинной. При $k = 1$ посылка этой импликации ложна и заключение ложно; при $k = 3$ посылка ложна и заключение истинно; поэтому мы признаём, что при ложной посылке импликацию целесообразно считать истинной. Таким образом, определение функции $F_\supset$ согласуется с нашим интуитивным пониманием импликации. ◁

Теперь мы готовы дать определение истинностного значения любой формулы в произвольной интерпретации.

Определение 1.1.12. Пусть задана какая угодно интерпретация M языка PL. Для любой формулы A определим *истинностное значение формулы A в интерпретации M*. Это значение мы будем обозначать через $|A|_M$ (или иногда через $|A|$, если из контекста понятно, какая интерпретация рассматривается). Зададим $|A|_M$ с помощью следующего индуктивного определения[3]:

1) если A является переменной, то $|A|_M = M(A)$;
2) если A является истинностной константой $\mathbf{F}$, то $|A|_M = 0$;
3) если A является истинностной константой $\mathbf{T}$, то $|A|_M = 1$;
4) если A имеет вид $\neg B$, то $|A|_M = F_\neg(|B|_M)$;
5) если A имеет вид $(B \wedge C)$, то $|A|_M = F_\wedge(|B|_M, |C|_M)$;
6) если A имеет вид $(B \vee C)$, то $|A|_M = F_\vee(|B|_M, |C|_M)$;

[3]Это определение является определением индукцией по построению множества формул PL (см. раздел 1.1.2). В дальнейшем мы не будем явно отмечать такие факты.

7) если A имеет вид $(B \supset C)$, то $|A|_M = F_{\supset}(|B|_M, |C|_M)$.

Если $|A|_M = 1$, то будем говорить, что формула A *истинна в (при) интерпретации* M, и сокращённо записывать этот факт как $M \vDash A$.

Если $|A|_M = 0$, то будем говорить, что формула A *ложна в (при) интерпретации* M, и сокращённо записывать этот факт как $M \nvDash A$. ◁

Очевидно, что истинностное значение формулы определяется лишь значениями переменных, входящих в неё, и не зависит от значений переменных, не входящих в эту формулу.

Пример 1.1.13. Пусть $A \leftrightharpoons ((p \vee q) \supset p)$.

Зададим интерпретацию M_1. Положим $M_1(p) = 0,\ M_1(q) = 1$. Отличным от p и q переменным можно сопоставить любые истинностные значения, положим для определённости $M_1(r) = 1$ для любой переменной $r \in Vars \setminus \{p, q\}$. Тогда имеем

$$|A| = F_{\supset}(|p \vee q|,\ |p|) = F_{\supset}(F_{\vee}(|p|, |q|),\ |p|) = \\ F_{\supset}(F_{\vee}(M_1(p), M_1(q)),\ M_1(p)) = F_{\supset}(F_{\vee}(0, 1),\ 0) = F_{\supset}(1,\ 0) = 0,$$

таким образом, $M_1 \nvDash A$.

Если же задать интерпретацию M_2 так, что $M_2(r) = 1$ для любой переменной r, то $M_2 \vDash (p \vee q)$ и $M_2 \vDash p$, поэтому $M_2 \vDash A$. ◁

Среди пропозициональных формул выделяют формулы, которые являются истинными при любых значениях входящих в них переменных. Такие формулы представляют в формальном виде принятые в математике способы рассуждений, иначе говоря, логические законы. Например, формула $p \supset (p \vee q)$ является логическим законом, поскольку, имея какие угодно высказывания, обозначенные через p и q, мы наверняка считаем, что если истинно p, то истинно хотя бы одно из двух: p или q. Эти соображения приводят нас к следующему определению.

Определение 1.1.14. Формула, истинная в любой интерпретации, называется *общезначимой* формулой (*тождественно истинной* формулой, *тавтологией* или *логическим законом*). То, что формула A общезначима, обозначается через $\vDash A$.

Формула, не являющаяся общезначимой, называется *необщезначимой*. То, что формула A необщезначима, обозначается через $\nvDash A$.

Формула, ложная в любой интерпретации, называется *невыполнимой* (*противоречивой* или *тождественно ложной*).

Формула, не являющаяся противоречивой, называется *выполнимой* (или *непротиворечивой*). ◁

Пример 1.1.15. Формула $p \supset (p \vee q)$ является общезначимой, выполнимой. Формула $\neg(p \supset (p \vee q))$ является невыполнимой, необщезначимой. Формула $p \vee q$ является выполнимой, необщезначимой. ◁

Выясним, как проверять, является ли произвольная формула A общезначимой. Если в A не входит никакая переменная, то A имеет одно и то же истинностное значение в любой интерпретации, которое легко вычислить согласно определению 1.1.12. Иначе в A входит лишь конечное число n попарно различных переменных, каждая из которых получает одно из двух

истинностных значений при задании интерпретации. Следовательно, имеется лишь конечное число (именно, 2^n) различных наборов значений этих переменных. Поэтому проверить, является ли данная формула общезначимой или нет, можно путём нахождения истинностного значения этой формулы для каждого такого набора значений переменных. Такую проверку удобно представлять в виде таблицы, общий вид которой мы сейчас зададим.

Пусть $p_1, \ldots, p_n$ — попарно различные переменные, $n > 0$, A — какая угодно пропозициональная формула, в которую могут входить переменные только из списка $p_1, \ldots, p_n$ (в частности, в A может не входить никакая переменная). *Таблица истинности* формулы A имеет вид, соответствующий таблице 1.1. Кроме верхней (заголовочной) строки, эта таблица имеет 2^n строк, в которых записаны все различные последовательности $\alpha_1, \ldots, \alpha_n$ истинностных значений и истинностные значения α формулы A в какой угодно интерпретации M такой, что $M(p_1) = \alpha_1$, $\ldots$, $M(p_n) = \alpha_n$. (Очевидно, $\alpha = |A|_M$ не зависит от выбора такой интерпретации M, поскольку истинностное значение формулы не зависит от значений переменных, не входящих в эту формулу.)

p_1	$\ldots$	p_n	A
$\ldots$	$\ldots$	$\ldots$	$\ldots$
α_1	$\ldots$	α_n	α
$\ldots$	$\ldots$	$\ldots$	$\ldots$

Таблица 1.1. Общий вид таблицы истинности формулы.

Пример 1.1.16. Таблица истинности формулы $p \equiv q$ есть таблица 1.2. $\triangleleft$

p	q	$p \equiv q$
0	0	1
0	1	0
1	0	0
1	1	1

Таблица 1.2. Таблица истинности формулы $p \equiv q$.

Пример 1.1.17. Составим таблицу истинности (см. таблицу 1.3) формулы $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$ и увидим, что в последнем столбце стоят только 1, таким образом, эта формула общезначима. Обязательными столбцами в этой таблице являются те, в которых указаны значения переменных (первый и второй столбцы) и значение исходной формулы (последний столбец), а остальные столбцы — вспомогательные и могут быть опущены. $\triangleleft$

Приведём список некоторых логических законов с их традиционными названиями.

Законы коммутативности и ассоциативности конъюнкции:

1) $(p \wedge q) \equiv (q \wedge p)$;

2) $((p \wedge q) \wedge r) \equiv (p \wedge (q \wedge r))$.

p	q	$\neg(p \wedge q)$	$\neg p \vee \neg q$	$\neg(p \wedge q) \equiv (\neg p \vee \neg q)$
0	0	1	1	1
0	1	1	1	1
1	0	1	1	1
1	1	0	0	1

Таблица 1.3. Таблица истинности формулы $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$.

Законы коммутативности и ассоциативности дизъюнкции:

3) $(p \vee q) \equiv (q \vee p)$;

4) $((p \vee q) \vee r) \equiv (p \vee (q \vee r))$.

Законы дистрибутивности:

5) $(p \wedge (q \vee r)) \equiv ((p \wedge q) \vee (p \wedge r))$;

6) $(p \vee (q \wedge r)) \equiv ((p \vee q) \wedge (p \vee r))$.

Законы поглощения:

7) $(p \vee (p \wedge q)) \equiv p$;

8) $(p \wedge (p \vee q)) \equiv p$.

Законы Де Моргана:

9) $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$;

10) $\neg(p \vee q) \equiv (\neg p \wedge \neg q)$.

Закон исключённого третьего:

11) $p \vee \neg p$.

Закон двойного отрицания:

12) $p \equiv \neg\neg p$.

Закон контрапозиции:

13) $(p \supset q) \equiv (\neg q \supset \neg p)$.

Отрицание посылки:

14) $\neg p \supset (p \supset q)$.

Выражение импликации через отрицание и дизъюнкцию:

15) $(p \supset q) \equiv (\neg p \vee q)$.

Упражнение 1.1.18. Проверьте общезначимость формул 1–15, приведённых выше. ◁

Продолжаем изучение семантических свойств языка логики высказываний. Рассмотрим, например, формулы $G_1 \leftrightharpoons p$, $G_2 \leftrightharpoons q$ и $A \leftrightharpoons (p \vee q)$. Легко видеть, что в любой интерпретации, если истинны G_1 и G_2, то истинна и A. Про такие формулы говорят, что A является логическим следствием множества формул $\{G_1, G_2\}$. Перейдём к определению, которое уточняет и обобщает это наблюдение.

Определение 1.1.19. Формула A называется *логическим следствием* (или *семантическим следствием*) множества формул Γ (это обозначается как $\Gamma \vDash A$), если для всякой интерпретации M выполняется: если для любой формулы $G \in \Gamma$ верно $M \vDash G$, то $M \vDash A$.

Если неверно, что $\Gamma \vDash A$, то пишем $\Gamma \nvDash A$. ◁

Если множество формул Γ пусто, то $\Gamma \vDash A$ в точности означает, что $\vDash A$.

Если Γ состоит из конечного числа формул $G_1, \ldots, G_n$ и $\Gamma \vDash A$, то этот факт будем также записывать как $G_1, \ldots, G_n \vDash A$ и говорить, что формула A является логическим (семантическим) следствием формул $G_1, \ldots, G_n$. Для такого Γ факт $\Gamma \nvDash A$ будем также записывать как $G_1, \ldots, G_n \nvDash A$

Пример 1.1.20. Легко проверить, что $p \vDash p \vee q$, $p \wedge q, r \vDash p \wedge r$ и $p, p \supset q \vDash q$. Но $p \nvDash p \wedge q$ (рассмотрите интерпретацию, в которой переменная p истинна, а q ложна). ◁

Теорема 1.1.21 (о логическом следствии)**.** *Пусть $G_1, \ldots, G_n$ $(n \in \mathbb{N}_+)$, A — пропозициональные формулы. Тогда $G_1, \ldots, G_n \vDash A$, если и только если $\vDash G_1 \wedge \ldots \wedge G_n \supset A$.*

Доказательство. $\vDash G_1 \wedge \ldots \wedge G_n \supset A$ по определению общезначимой формулы означает, что (a) для любой интерпретации M выполняется $M \vDash (G_1 \wedge \ldots \wedge G_n) \supset A$. В соответствии с таблицей истинности импликации утверждение (a) равносильно следующему утверждению (b): для всякой интерпретации M, если $M \vDash G_1 \wedge \ldots \wedge G_n$, то $M \vDash A$. Согласно таблице истинности конъюнкции, утверждение (b) равносильно следующему: для любой интерпретации M, если для любой формулы $G \in \{G_1, \ldots, G_n\}$ $M \vDash G$, то $M \vDash A$, т. е. (по определению логического следствия) $G_1, \ldots, G_n \vDash A$. ◁

Рассмотрим теперь формулы $A \leftrightharpoons \neg(p \wedge q)$ и $B \leftrightharpoons (\neg p \vee \neg q)$. Таблицы истинности этих формул (являющиеся подтаблицами таблицы 1.3 на с. 21) показывают, что A и B истинны или ложны при одних и тех же значениях переменных. Про такие формулы говорят, что они равносильны. Уточним это наблюдение, дав следующее определение.

Определение 1.1.22. Формулы A и B называются *равносильными* (*логически эквивалентными* или *семантически эквивалентными*), если для всякой интерпретации M выполняется: $M \vDash A$ тогда и только тогда, когда $M \vDash B$ (утверждение, стоящее после двоеточия, можно записать иначе — как $|A|_M = |B|_M$). То, что формулы A и B равносильны, будем обозначать через $A \sim B$. ◁

Очевидно, что формулы A и B равносильны, если и только если формула $(A \equiv B)$ является общезначимой. Ещё один критерий равносильности

формул — A и B равносильны, если и только если $A \vDash B$ и $B \vDash A$ — по сути является переформулировкой только что данного определения. Также очевидно, что $A \sim A$ (т. е. отношение $\sim$ рефлексивно); из $A \sim B$ следует $B \sim A$ (т. е. $\sim$ симметрично); и из $A \sim B$ и $B \sim C$ следует $A \sim C$ (т. е. $\sim$ транзитивно); таким образом, отношение $\sim$ является отношением эквивалентности на множестве всех пропозициональных формул.

Примеры равносильных формул мы получим, если заменим $\equiv$ на $\sim$ в каждом из вышеприведённых логических законов 1–15, в который входит $\equiv$.

Имея некоторые общезначимые формулы, мы можем получать новые общезначимые формулы следующим образом. Рассмотрим, например, общезначимую формулу $p \supset (p \vee q)$. Подставим вместо всех вхождений p какую угодно формулу, скажем, $(\neg q \wedge r)$. В результате получим формулу $(\neg q \wedge r) \supset ((\neg q \wedge r) \vee q)$. Легко проверить, что она общезначима. Можно предположить, что общезначимость будет сохраняться для любой исходной формулы, лишь бы все вхождения некоторой переменной заменялись на одну и ту же (хотя и выбранную произвольно) формулу. Сформулируем это предположение в обобщённом виде и докажем его. Но прежде уточним понятие подстановки формул вместо вхождений пропозициональных переменных.

Пусть $p_1, \ldots, p_n$ — попарно различные пропозициональные переменные; $A, B_1, \ldots, B_n$ — пропозициональные формулы; θ обозначает множество слов $\{B_1/p_1, \ldots, B_n/p_n\}$ (таким образом, каждый элемент этого множества есть слово, состоящее из формулы, символа-разделителя «/» и переменной). Положим по определению, что через $A\theta$ обозначается результат одновременной замены всех вхождений переменных $p_1, \ldots, p_n$ в A на $B_1, \ldots, B_n$ соответственно.

Дадим также индуктивное определение $A\theta$:

1) $p_i\theta \leftrightharpoons B_i$ для любого $i = 1, \ldots, n$;

2) если A есть истинностная константа или переменная, отличная от каждой из $p_1, \ldots, p_n$, то $A\theta \leftrightharpoons A$;

3) если A есть $\neg C$, то $A\theta \leftrightharpoons \neg C'$, где $C' \leftrightharpoons C\theta$;

4) если A есть $(C \odot D)$, где $\odot$ — одна из связок $\wedge$, $\vee$, $\supset$, то $A\theta \leftrightharpoons (C\theta \odot D\theta)$.

Замечание 1.1.23. С помощью индукции по построению формулы A можно доказать, что два данных определения $A\theta$ равносильны. (Отметим и другой возможный подход: можно было оставить лишь индуктивное определение, а слова «$A\theta$ — результат одновременной замены всех вхождений переменных $p_1, \ldots, p_n$ в A на $B_1, \ldots, B_n$ соответственно» считать пояснением.)

Следующее объяснение адресовано прежде всего читателям, изучающим информатику и сомневающимся в целесообразности индуктивных определений, нередко громоздких. Данное индуктивное определение $A\theta$ не следует считать бесполезным из-за того, что неиндуктивное определение того же понятия было дано раньше. Как мы увидим, индуктивное определение позволит легко доказывать некоторые свойства с помощью индукции. Кроме того, предположим, что требуется разработать компьютерную программу, реализующую подстановку $A\theta$, причём изначально такая подстановка определена неиндуктивно. Также предположим, что формула пред-

ставляется в программе не просто в виде строки, а в виде так называемого синтаксического дерева формулы.

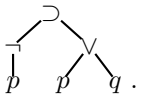
Пусть A — формула. *Синтаксическим деревом формулы* A называют дерево, корень которого помечен (или является)

1) самой формулой A, если A есть переменная или истинностная константа;

2) логической связкой $\neg$, если A имеет вид $\neg C$, причём в этом случае поддеревом этого узла является синтаксическое дерево формулы C;

3) логической связкой $\odot$, если A имеет вид $C \odot D$ ($\odot$ — одна из связок $\wedge$, $\vee$, $\supset$), причём в этом случае левым и правым поддеревьями этого узла являются синтаксические деревья формул C и D соответственно.

Например, синтаксическое дерево формулы $\neg p \supset (p \vee q)$ выглядит так:



.

При представлении формулы в программе в виде синтаксического дерева одним из естественных подходов для разработчика программы будет уточнение того, как вычисляется $A\theta$, в духе данного нами индуктивного определения. Это индуктивное определение по сути является рекурсивным алгоритмом вычисления $A\theta$.

В дальнейшем мы обычно будем опускать очевидные индуктивные определения и давать более короткие неиндуктивные, будучи уверенными, что читатель при необходимости сможет записать соответствующие индуктивные определения. ◁

Пример 1.1.24. Пусть $A \leftrightharpoons (p \supset (p \vee q \vee r))$. Тогда $A\{(\neg q \wedge r)/p,\ \neg r_1/q, \neg q/r_2\}$ есть $(\neg q \wedge r) \supset ((\neg q \wedge r) \vee \neg r_1 \vee r)$. ◁

Теорема 1.1.25 (о подстановке пропозициональных формул в пропозициональную тавтологию)**.** *Пусть* $p_1, \ldots, p_n$ *— попарно различные пропозициональные переменные,* A, $B_1, \ldots, B_n$ *— пропозициональные формулы,* $\theta \leftrightharpoons \{B_1/p_1, \ldots, B_n/p_n\}$. *Тогда если* $\vDash A$, *то* $\vDash A\theta$.

Доказательство. Пусть M — произвольная интерпретация языка PL. Определим интерпретацию M' так, что $M'(p_i) = |B_i|_M$ для каждого $i = 1, \ldots, n$ и $M'(q) = M(q)$ для любой переменной $q \in Vars \setminus \{p_1, \ldots, p_n\}$. Если мы докажем, что для любой формулы E верно $|E\theta|_M = |E|_{M'}$, то, взяв в качестве E общезначимую формулу A, получим $|A\theta|_M = |A|_{M'} = 1$. Поскольку M — произвольная интерпретация, то $\vDash A\theta$, что и утверждает данная теорема.

Таким образом, для завершения доказательства данной теоремы осталось установить, что для любой формулы E верно $|E\theta|_M = |E|_{M'}$. Применим индукцию по построению формулы E.

База индукции. E является пропозициональной переменной или истинностной константой. Если E есть p_i для некоторого $i = 1, \ldots, n$, то

$$|E\theta|_M = |p_i\theta|_M = |B_i|_M = M'(p_i) = |E|_{M'}.$$

Если же E есть переменная из $Vars \setminus \{p_1, \ldots, p_n\}$ или истинностная константа, то $|E\theta|_M = |E|_M = |E|_{M'}$.

Индукционный переход. Предположим, что для формул C и D устанавливаемое утверждение верно.

Пусть E имеет вид $\neg C$. Тогда $|E\theta|_M = |\neg C'|_M$, где $C' \leftrightharpoons C\theta$. По индукционному предположению $|C\theta|_M = |C|_{M'}$. Следовательно,

$$|E\theta|_M = F_\neg(|C\theta|_M) = F_\neg(|C|_{M'}) = |E|_{M'}.$$

Наконец, пусть E имеет вид $(C \odot D)$, где $\odot$ — одна из связок $\wedge$, $\vee$, $\supset$. Тогда $|E\theta|_M = |C\theta \odot D\theta|_M$. По индукционному предположению $|C\theta|_M = |C|_{M'}$ и $|D\theta|_M = |D|_{M'}$. Следовательно,

$$|E\theta|_M = F_\odot(|C\theta|_M,\ |D\theta|_M) = F_\odot(|C|_{M'},\ |D|_{M'}) = |E|_{M'}.$$

◁

Следствие 1.1.26. *Пусть $p_1, \ldots, p_n$ — попарно различные пропозициональные переменные, A, $B_1, \ldots, B_n$, C — пропозициональные формулы, $\theta \leftrightharpoons \{B_1/p_1, \ldots, B_n/p_n\}$. Тогда если $A \sim C$, то $A\theta \sim C\theta$.*

Доказательство. Пусть верно $A \sim C$. Тогда имеет место $\vDash (A \equiv C)$, отсюда по предыдущей теореме 1.1.25 получаем $\vDash (A \equiv C)\theta$. $(A \equiv C)\theta$ есть $(A\theta \equiv C\theta)$, значит, $\vDash (A \equiv C)\theta$ влечёт $A\theta \sim C\theta$, что и требовалось доказать. ◁

Мы бы хотели производить равносильные преобразования формул примерно так же, как и известные преобразования алгебраических выражений. Например, поскольку $\neg(p \wedge q) \sim (\neg p \vee \neg q)$, то кажется естественным, что должно быть верно

$$(\neg p \vee \underline{\neg(p \wedge q)}\ ) \wedge r \sim (\neg p \vee \underline{(\neg p \vee \neg q)}\ ) \wedge r.$$

Здесь подчёркнутое вхождение подформулы слева от знака $\sim$ заменено на подчёркнутую формулу справа от этого знака. Следующая теорема обосновывает семантическую эквивалентность исходной формулы и формулы, полученной в результате такой замены.

Теорема 1.1.27 (о семантически эквивалентной замене)**.** *Пусть A, B и B' — пропозициональные формулы, A' есть результат замены некоторого вхождения B в A на B'. Тогда если $B \sim B'$, то $A \sim A'$.*

Пример 1.1.28. Проиллюстрируем эту теорему:

$$\underbrace{\neg(p \wedge q)}_{B} \sim \underbrace{(\neg p \vee \neg q)}_{B'} \quad \Rightarrow \quad \overbrace{(\neg p \vee \underbrace{\neg(p \wedge q)}_{B}\ ) \wedge r}^{A} \sim \overbrace{(\neg p \vee \underbrace{(\neg p \vee \neg q)}_{B'}\ ) \wedge r}^{A'}.$$

◁

Упражнение 1.1.29. Докажите предыдущую теорему, воспользовавшись индукцией по построению формулы A. ◁

1.1.4. Выражение булевой функции формулой. Дизъюнктивная и конъюнктивная нормальные формы. Полиномы Жегалкина

Для какого угодно $n \in \mathbb{N}_+$ любая функция из множества $\mathbb{B}^n$ в множество $\mathbb{B}$ называется n-местной *булевой функцией*.

Любая пропозициональная формула A, в которую входят переменные только из списка попарно различных переменных $p_1, \ldots, p_n$, естественным образом определяет булеву функцию $f(p_1, \ldots, p_n)$, значения которой вычисляются по формуле A. Подробнее, для любых $\alpha_1, \ldots, \alpha_n \in \mathbb{B}$ $f(\alpha_1, \ldots, \alpha_n) = |A|_M$, где M — любая интерпретация такая, что $M(p_1) = \alpha_1$, $\ldots$, $M(p_n) = \alpha_n$ (очевидно, что $f(\alpha_1, \ldots, \alpha_n)$ не зависит от выбора такой интерпретации M). С другой стороны, если даны формула A и булева функция $f(p_1, \ldots, p_n)$, удовлетворяющие указанным условиям, то мы будем говорить, что *формула A выражает функцию $f(p_1, \ldots, p_n)$*.

Пример 1.1.30. Формула $\neg p_1$ выражает как функцию $F_\neg(p_1)$, так и функцию $f(p_1, p_2) = F_\neg(p_1)$. Формула $p_1 \wedge \neg p_2$ выражает функцию $g(p_1, p_2) = F_\wedge(p_1,\ F_\neg(p_2))$. ◁

Теорема 1.1.31. *Для любой булевой функции $f(p_1, \ldots, p_n)$ найдётся пропозициональная формула A, которая выражает f.*

Доказательство. Для простоты изложения возьмём функцию f, заданную таблицей 1.4. Аналогичное доказательство будет справедливо для любой булевой функции.

p_1	p_2	$f(p_1, p_2)$
0	0	1
0	1	0
1	0	0
1	1	1

Таблица 1.4. Таблица, задающая булеву функцию f.

Запишем пропозициональную формулу A, истинную при тех и только тех значениях переменных, при которых истинна функция f. Для этого сначала для каждой строки таблицы со значением 1 функции f запишем формулу, истинную при тех и только тех значениях переменных, которые стоят в рассматриваемой строке. Для первой строки[4] таблицы 1.4 записываем формулу $(\neg p_1 \wedge \neg p_2)$, а для последней строки — формулу $(p_1 \wedge p_2)$. Искомая формула A получается соединением полученных формул с помощью дизъюнкции: $(\neg p_1 \wedge \neg p_2) \vee (p_1 \wedge p_2)$.

Этот способ выражения f формулой годится, если f не является тождественно ложной; иначе f можно выразить, например, формулой $(p_1 \wedge \neg p_1)$.

Излагаемый ниже второй способ выражения f формулой применим, если f не является тождественно истинной; иначе f можно выразить, например, формулой $(p_1 \vee \neg p_1)$.

Запишем пропозициональную формулу A, ложную при тех и только тех значениях переменных, при которых ложна функция f. Для этого сначала

[4]Заголовочную строку мы считаем нулевой.

для каждой строки таблицы со значением 0 функции f запишем формулу, ложную при тех и только тех значениях переменных, которые стоят в рассматриваемой строке. Для второй строки таблицы 1.4 записываем формулу $(p_1 \vee \neg p_2)$, а для третьей строки — формулу $(\neg p_1 \vee p_2)$. Искомая формула A получается соединением полученных формул с помощью конъюнкции: $(p_1 \vee \neg p_2) \wedge (\neg p_1 \vee p_2)$. $\triangleleft$

Формулы, построенные в только что проведённом доказательстве, имеют некий специальный вид, к определению которого мы сейчас перейдём.

Определение 1.1.32. Пусть $A_1, \ldots, A_n$ $(n \geqslant 1)$ — формулы. Формула $A_1 \vee \ldots \vee A_n$ называется *дизъюнкцией формул* $A_1, \ldots, A_n$. Формула $A_1 \wedge \ldots \wedge A_n$ называется *конъюнкцией формул* $A_1, \ldots, A_n$. Каждая формула A_i $(i = 1, \ldots, n)$ называется *членом этой дизъюнкции* (соответственно, *конъюнкции*). $\triangleleft$

Литерой называется любая пропозициональная переменная и отрицание любой пропозициональной переменной.

Определение 1.1.33. *Конъюнктом* называется конъюнкция литер. Говорят, что формула *находится в дизъюнктивной нормальной форме*, если она является дизъюнкцией конъюнктов. Также такая формула называется *дизъюнктивной нормальной формой* (сокращённо — *ДНФ*). $\triangleleft$

Пример 1.1.34. Следующие формулы находятся в ДНФ: p, $\neg p$, $\neg p \vee q$, $p \wedge \neg q$, $(p \wedge \neg q) \vee (\neg p \wedge r \wedge p) \vee \neg q$, причём все скобки в последней формуле можно опустить. $\triangleleft$

Определение 1.1.35. *Дизъюнктом* называется дизъюнкция литер. Говорят, что формула *находится в конъюнктивной нормальной форме*, если она является конъюнкцией дизъюнктов. Также такая формула называется *конъюнктивной нормальной формой* (сокращённо — *КНФ*). $\triangleleft$

Пример 1.1.36. Следующие формулы находятся в КНФ: p, $\neg p$, $\neg p \vee q$, $p \wedge \neg q$, $(p \vee \neg q) \wedge (\neg p \vee r \vee p) \wedge \neg q$. $\triangleleft$

Будем говорить, что *формула B выражает формулу A*, если $A \sim B$.

Определение 1.1.37. *Дизъюнктивной* (соответственно, *конъюнктивной*) *нормальной формой формулы A* называется любая формула, находящаяся в ДНФ (соответственно, КНФ) и выражающая формулу A. $\triangleleft$

Пример 1.1.38. Легко проверить, что таблица 1.4 из доказательства теоремы 1.1.31 является таблицей истинности формулы $A \leftrightharpoons (p_1 \equiv p_2)$. Построенные в доказательстве этой теоремы формулы $(\neg p_1 \wedge \neg p_2) \vee (p_1 \wedge p_2)$ и $(p_1 \vee \neg p_2) \wedge (\neg p_1 \vee p_2)$ являются, соответственно, ДНФ и КНФ формулы A. $\triangleleft$

Вышеприведённые определения и доказательство теоремы 1.1.31 дают нам следующую теорему.

Теорема 1.1.39. *Любую булеву функцию (и любую пропозициональную формулу) можно выразить как формулой, находящейся в ДНФ, так и формулой, находящейся в КНФ.*

Упражнение 1.1.40. Докажите, что КНФ общезначима, если и только если в каждом её дизъюнкте имеется переменная и отрицание этой переменной. Сформулируйте и докажите похожее утверждение о невыполнимости ДНФ. ◁

Упражнение 1.1.41. Уточните следующий способ выражения булевой функции формулой, находящейся в КНФ, и обоснуйте его: сначала построить ДНФ «отрицания» этой функции, а затем построить отрицание полученной ДНФ. ◁

Упражнение 1.1.42. Опишите два алгоритма, один из которых любую формулу выражает формулой, находящейся в КНФ, а другой — в ДНФ. Эти алгоритмы должны опираться на равносильные преобразования формул. ◁

Введём бинарные логические связки *штрих Шеффера* $|$ и *стрелку Пирса* $\downarrow$: положим $(A \mid B) \leftrightharpoons \neg(A \wedge B)$ и $(A \downarrow B) \leftrightharpoons \neg(A \vee B)$.

Теорема 1.1.43. *Любую булеву функцию можно выразить пропозициональной формулой, в которую из логических связок входят только:* (a) $\neg$ *и* $\wedge$, (b) $\neg$ *и* $\vee$, (c) $|$, (d) $\downarrow$.

Доказательство. По теореме 1.1.39 любую булеву функцию можно выразить формулой C, в которую из связок входят только $\neg$, $\wedge$ и $\vee$.

(a) Докажем, что C можно выразить можно выразить формулой, в которую из связок входят только $\neg$ и $\wedge$. Воспользуемся индукцией (точнее, методом математической индукции[5]) по числу вхождений связки $\vee$ в C.

База индукции. Имеется ровно 0 вхождений связки $\vee$ в C. Доказываемое утверждение верно.

Индукционный переход. Предположим, что для любой формулы C с числом вхождений связки $\vee$, равным n, доказываемое утверждение верно. Рассмотрим формулу C с ровно $n+1$ вхождением связки $\vee$. Легко проверить, что $(p \vee q) \sim \neg(\neg p \wedge \neg q)$; отсюда по следствию 1.1.26 теоремы о подстановке имеем $(A \vee B) \sim \neg(\neg A \wedge \neg B)$ для любых формул A и B. Заменим в формуле C одно из вхождений формулы вида $(A \vee B)$ на $\neg(\neg A \wedge \neg B)$. Обозначим получившуюся в результате этой замены формулу через C'. По теореме 1.1.27 о семантически эквивалентной замене справедливо $C \sim C'$. В формуле C' имеется ровно n вхождений связки $\vee$. Применив к C' индукционное предположение, получим формулу C'', которая из связок имеет лишь $\neg$ и $\wedge$, причём $C' \sim C''$. Воспользовавшись транзитивностью отношения $\sim$, получим $C \sim C''$. Таким образом, C'' — искомая формула, и пункт (a) доказан.

Очевидно, для любых формул A и B имеем следующее:

(b) $(A \wedge B) \sim \neg(\neg A \vee \neg B)$;

(c) $\neg A \sim (A \mid A)$, $(A \wedge B) \sim \neg(A \mid B)$;

(d) $\neg A \sim (A \downarrow A)$, $(A \vee B) \sim \neg(A \downarrow B)$.

[5] Пусть $k \in \mathbb{N}$. Напомним, что для доказательства методом математической индукции того, что утверждение $\mathcal{P}(n)$ верно при любом натуральном $n \geqslant k$, устанавливают

1) базу индукции: $\mathcal{P}(k)$; и

2) индукционный переход: для любого натурального $n \geqslant k$ из предположения (называемого индукционным предположением) верности $\mathcal{P}(n)$ следует верность $\mathcal{P}(n+1)$.

Отсюда доказательства пунктов (b), (c) и (d) получаются таким же образом, как и доказательство пункта (a). ◁

Замечание 1.1.44. Доказательство пункта (a) предыдущей теоремы было весьма подробным. В дальнейшем мы иногда будем приводить менее подробные доказательства, будучи уверенными, что читатель при необходимости сможет восстановить детали. Менее подробное доказательство пункта (a) могло бы быть таким: очевидно, $(A \vee B) \sim \neg(\neg A \wedge \neg B)$, поэтому, заменив в формуле C каждую подформулу вида $(A \vee B)$ на $\neg(\neg A \wedge \neg B)$, в результате получим формулу, равносильную формуле C и из связок имеющую только $\neg$ и $\wedge$. ◁

Упражнение 1.1.45. Докажите, что любую булеву функцию можно выразить пропозициональной формулой, в которую из связок входят только $\neg$ и $\supset$. ◁

Замечание 1.1.46. Теорема 1.1.39 установила, что любую булеву функцию можно выразить формулой, в которую из связок входят лишь $\neg$, $\wedge$ и $\vee$. Этот факт можно переформулировать так: любая булева функция есть некоторая композиция функций $F_\neg$, $F_\wedge$ и $F_\vee$. Например, если некоторая функция $f(p_1, p_2)$ выражена формулой $(\neg p_1 \wedge \neg p_2) \vee (p_1 \wedge p_2)$, то $f(p_1, p_2)$ равна $F_\vee(F_\wedge(F_\neg(p_1), F_\neg(p_2)),\ F_\wedge(p_1, p_2))$. ◁

Упражнение 1.1.47. Докажите, что, кроме функций $F_|(p_1, p_2)$ и $F_\downarrow(p_1, p_2)$, которые выражаются формулами $(p_1 \mid p_2)$ и $(p_1 \downarrow p_2)$ соответственно, не существует двухместной булевой функции F такой, что любая булева функция есть некоторая композиция функций, среди которых имеется только функция F. *Указание.* Сколько всего имеется двухместных булевых функций? Какие из этих функций F сохраняют 0 или 1 (т. е. $F(\alpha, \alpha) = \alpha$ при $\alpha = 0$ или при $\alpha = 1$ соответственно)? Можно ли выразить любую булеву функцию только с помощью функции, сохраняющей 0 или 1? ◁

Полиномы Жегалкина

Рассмотрим (известное из алгебры) поле вычетов по модулю 2, обозначаемое здесь через $\mathbb{Z}_2$. Напомним, что в $\mathbb{Z}_2$ лишь два элемента 0 и 1; операции $+$ и $\cdot$ над ними определены таким образом: $0 + 0 = 0$, $0 + 1 = 1$, $1 + 0 = 1$, $1 + 1 = 0$, $0 \cdot 0 = 0$, $0 \cdot 1 = 0$, $1 \cdot 0 = 0$, $1 \cdot 1 = 1$.

Полиномом Жегалкина от переменных $x_1, \ldots, x_n$ называется любой полином над $\mathbb{Z}_2$ вида

$$\sum_{\{i_1,\ldots,i_m\} \subseteq \{1,2,\ldots,n\}} a_{\{i_1,\ldots,i_m\}} x_{i_1} \ldots x_{i_m},$$

где суммирование идёт по всем подмножествам $\{i_1, \ldots, i_m\}$ множества $\{1, 2, \ldots, n\}$, каждое $a_{\{i_1,\ldots,i_m\}} \in \mathbb{Z}_2$, причём член, соответствующий пустому подмножеству, является константой.

Для любого $x \in \mathbb{Z}_2$ и любого $n \in \mathbb{N}_+$ имеем $x^n = x$ (грубо говоря, степени выше первой не нужны). Поэтому любой полином $P(x_1, \ldots, x_n)$ над $\mathbb{Z}_2$ функционально равен некоторому полиному Жегалкина $P'(x_1, \ldots, x_n)$, т. е. для любых $x_1, \ldots, x_n \in \mathbb{Z}_2$ верно $P(x_1, \ldots, x_n) = P'(x_1, \ldots, x_n)$.

Отождествим элементы 0 и 1 множества $\mathbb{Z}_2$ с истинностными значениями 0 и 1 соответственно. Докажем, что тогда любая булева функция

выражается некоторым полиномом Жегалкина, т. е. значения этой функции вычисляются как значения этого полинома.

Согласно замечанию 1.1.46 любая булева функция есть некоторая композиция функций $F_\neg$, $F_\wedge$ и $F_\vee$. Поэтому достаточно выразить полиномами Жегалкина эти три функции. Легко проверить, что

$$F_\neg(q) = q + 1,$$
$$F_\wedge(q_1, q_2) = q_1 \cdot q_2.$$

Учитывая, что $q_1 \vee q_2 \sim \neg(\neg q_1 \wedge \neg q_2)$, имеем

$$F_\vee(q_1, q_2) = F_\neg(F_\wedge(F_\neg(q_1),\ F_\neg(q_2))) = (q_1 + 1) \cdot (q_2 + 1) + 1 = q_1 \cdot q_2 + q_1 + q_2.$$

Итак, всякая булева функция может быть выражена некоторым полиномом Жегалкина.

Пример 1.1.48. Выразим полиномом Жегалкина булеву функцию, заданную таблицей 1.4 на с. 26. Следующая ДНФ выражает эту функцию:

$$D \leftrightharpoons (\neg p_1 \wedge \neg p_2) \vee (p_1 \wedge p_2).$$

Конъюнкт $(p_1 \wedge p_2)$ выражается полиномом $p_1 \cdot p_2$ (от переменных p_1 и p_2); конъюнкт $(\neg p_1 \wedge \neg p_2)$ — полиномом $(p_1 + 1) \cdot (p_2 + 1) = p_1 \cdot p_2 + p_1 + p_2 + 1$. Наконец, D выражается полиномом

$$(p_1 \cdot p_2 + p_1 + p_2 + 1) \cdot p_1 \cdot p_2 + (p_1 \cdot p_2 + p_1 + p_2 + 1) + p_1 \cdot p_2 = p_1 + p_2 + 1.$$

◁

Упражнение 1.1.49. Докажите, что для любой булевой функции существует ровно один полином Жегалкина, выражающий эту функцию. *Указание.* Одним из возможных подходов к доказательству является сравнение числа n-местных булевых функций и числа полиномов Жегалкина от n переменных. ◁

В заключение данного раздела скажем несколько слов о применении теории булевых функций к построению цифровых схем, в том числе интегральных микросхем. Такие схемы состоят из *функциональных элементов*. Функциональный элемент f имеет несколько входов и один выход. На каждый из входов элемента f подаётся цифровой сигнал (т. е., по сути, 0 или 1). То, какой сигнал получается на выходе элемента f, определяется тем, какую булеву функцию *реализует* элемент f. Например, если f реализует функцию $F_\vee$, то на выходе будет 0, когда на оба входа f подаётся 0, в остальных случаях на выходе будет 1. Схему образуют из функциональных элементов путём соединения входов одних элементов и выходов других. Так как любая булева функция есть некоторая композиция функций $F_\neg$, $F_\wedge$ и $F_\vee$ (см. замечание 1.1.46), то любую булеву функцию можно реализовать схемой из функциональных элементов трёх типов: реализующих функции $F_\neg$, $F_\wedge$ и $F_\vee$ соответственно. Естественно возникает задача: как минимизировать количество элементов в схеме. Однако эта задача выходит за рамки данной книги, а наша цель — лишь показать, что теория булевых функций, основ которой мы лишь коснулись, находит важные применения.

1.2. Общее понятие исчисления

Не всегда осознавая это, мы имеем дело с исчислением, когда заданы некоторые исходные объекты (аксиомы) и правила, по которым можно получать новые объекты из исходных и ранее полученных (правила вывода).

Пример 1.2.1. Примером исчисления, хорошо известного из базового курса математического анализа, является исчисление для нахождения производных функций (в символьном виде). Аксиомы этого исчисления суть равенства, задающие производные некоторых элементарных функций, например, константы, степенной, показательной и логарифмической функций. Правила вывода этого исчисления суть правила нахождения производной функции f, которая является суммой, произведением, частным, композицией функций f_1 и f_2, по уже найденным производным функций f_1 и f_2. Более конкретно, одна из аксиом может быть записана в общем виде как «$(x^\alpha)' = \alpha x^{\alpha-1}$». Одно из правил вывода можно записать в виде «из $(f_1)' = g_1$ и $(f_2)' = g_2$ получается $(f_1 + f_2)' = g_1 + g_2$» или в виде

$$\frac{(f_1)' = g_1; \quad (f_2)' = g_2}{(f_1 + f_2)' = g_1 + g_2}.$$

Тогда из аксиом $(x^5)' = 5x^4$ и $(x^6)' = 6x^5$ по этому правилу вывода получаем $(x^5+x^6)' = 5x^4+6x^5$. Далее, из последнего равенства и аксиомы $(x^7)' = 7x^6$ по данному правилу вывода получаем $(x^5 + x^6 + x^7)' = 5x^4 + 6x^5 + 7x^6$. Это равенство, конечно, не оказывается неожиданным, но замечательно то, что при получении этого равенства мы пользовались лишь аксиомами и правилами вывода исчисления, причём делали это чисто механически и не вникали в содержательное значение (семантику) объектов, которыми оперировали. ◁

Конкретизируем понятие исчисления применительно к формальным языкам (в этом случае получаемые объекты будут словами) и уточним, как получаются (иначе говоря, выводятся) новые объекты с помощью исчисления.

1.2.1. Исчисление и вывод в исчислении

Пусть Σ — произвольный алфавит, L — произвольный язык в алфавите Σ. Говорят, что задано *исчисление* (или *дедуктивная система*) $\mathcal{C}$, если задано множество Ax слов языка L и конечное множество R, каждый элемент которого является не менее чем двухместным отношением на L. Каждый элемент множества Ax называют *аксиомой* исчисления $\mathcal{C}$. Каждый элемент множества R называют *правилом вывода* исчисления $\mathcal{C}$. Будем называть Σ *алфавитом исчисления* $\mathcal{C}$, L — *языком исчисления* $\mathcal{C}$, а также будем говорить, что $\mathcal{C}$ является *исчислением в алфавите* Σ (*исчислением в языке* L). Если потребуется подчеркнуть, что рассматривается исчисление $\mathcal{C}$ в языке L, то вместо $\mathcal{C}$ будем использовать $\mathcal{C}(L)$.

Пусть $\mathcal{R}$ — правило вывода, являющееся $(n+1)$-местным отношением, где $n \in \mathbb{N}_+$. Если $\langle \alpha_1, \ldots, \alpha_n, \alpha \rangle \in \mathcal{R}$, то говорят, что α получается (или получено) из $\alpha_1, \ldots, \alpha_n$ по правилу вывода $\mathcal{R}$, или что α является результатом *применения* правила вывода $\mathcal{R}$ к $\alpha_1, \ldots, \alpha_n$. Каждое слово α_i $(i = 1, \ldots, n)$

называют *посылкой*, а слово α — *заключением* данного (применения) правила. Правило вывода, являющееся $(n+1)$-местным отношением, называют *n-посылочным* правилом вывода.

Часто n-посылочное правило вывода записывают в следующем виде:

$$\frac{A_1;\ \ldots;\ A_n}{A},$$

где $A_1, \ldots, A_n$ задают общий вид посылок, а A — общий вид заключения. При этом предполагается, что символ «;» не входит в алфавит Σ, а служит разделителем. Такая запись и сопутствующие комментарии определяют отношение, являющееся правилом вывода. (Обычно правило вывода таково, что можно построить алгоритм, проверяющий по произвольным словам $\alpha_1, \ldots, \alpha_n, \alpha \in \Sigma$, получается ли α из $\alpha_1, \ldots, \alpha_n$ по этому правилу вывода или нет.)

Выводом (или *доказательством*) *в исчислении* $\mathcal{C}$ называется конечная последовательность слов $\alpha_1, \alpha_2, \ldots, \alpha_k$, в которой каждое слово α_i $(i = 1, 2, \ldots, k)$ является аксиомой исчисления $\mathcal{C}$ или получается по некоторому правилу вывода исчисления $\mathcal{C}$ из некоторых слов $\alpha_{i_1}, \ldots, \alpha_{i_n}$ этой последовательности таких, что каждое $i_1, \ldots, i_n$ меньше i. (Конец предыдущего предложения, начинающийся со слова «из», можно менее точно переформулировать так: «из предшествующих слов этой последовательности»). Число k называется *длиной этого вывода*. Предполагается, что символ «,», служащий разделителем слов языка L в выводе, не входит в алфавит Σ (иначе в качестве такого разделителя возьмём символ, не входящий в Σ). Таким образом, вывод является словом в алфавите $\Sigma \cup \{,\}$.

Выводом слова α *в исчислении* $\mathcal{C}$ называется вывод $\alpha_1, \ldots, \alpha_k$ в этом исчислении, где α_k совпадает с α.

Если существует вывод слова α в исчислении $\mathcal{C}$, то слово α называют *выводимым в исчислении* $\mathcal{C}$ (или *теоремой исчисления* $\mathcal{C}$) и обозначают это как $\vdash_{\mathcal{C}} \alpha$.

Если слово α не является выводимым в исчислении $\mathcal{C}$, то α называют *невыводимым в* $\mathcal{C}$ и обозначают это как $\nvdash_{\mathcal{C}} \alpha$.

Обозначения вида $\vdash_{\mathcal{C}} \alpha$ и $\nvdash_{\mathcal{C}} \alpha$ часто сокращают до $\vdash \alpha$ и $\nvdash \alpha$ соответственно, если из контекста понятно, какое исчисление рассматривается.

Зачастую полезно дополнять вывод его *анализом*: каждое слово в выводе снабжать номером и указанием на то, что оно является аксиомой, или на то, из каких слов и по какому правилу вывода получено данное слово. (Можно условиться дополнять вывод анализом другого вида, если это потребуется.)

Пример 1.2.2. Пусть дан алфавит $\Sigma \leftrightharpoons \{a, b\}$ и язык $L \leftrightharpoons \Sigma^*$. Напомним, что палиндромом называется слово, которое одинаково читается слева направо и справа налево, и дадим определение, уточняющее понятие палиндрома применительно к алфавиту Σ. Слово $x_1 \ldots x_n$ (где $x_i \in \Sigma$ для каждого $i = 1, \ldots, n$) назовём *палиндромом*, если $n > 0$ и для каждого $j = 1, \ldots, n$ выполняется $x_j = x_{n+1-j}$.

Зададим исчисление Pal, в котором, как мы докажем, будут выводимы все палиндромы языка L и только они.

Аксиомами этого исчисления являются слова: 1) a, 2) b, 3) aa и 4) bb.

Правила вывода исчисления Pal таковы:

$$\frac{\gamma}{a\gamma a}\ (a), \qquad \frac{\gamma}{b\gamma b}\ (b),$$

где γ — произвольное слово языка L. Справа от каждого из этих правил приведено его обозначение. По правилу (a) (соответственно, по правилу (b)) из слова γ получается слово $a\gamma a$ (соответственно, $b\gamma b$). Определение исчисления Pal закончено.

Выводом слова $abababa$ в исчислении Pal является приведённая ниже последовательность слов, разделённых запятыми (точка служит для завершения текущего предложения и не входит в этот вывод):

$$b,\ aba,\ babab,\ abababa\ . \qquad (*)$$

Этот же вывод с анализом выглядит так:

1) b — аксиома 2;

2) aba получается из слова 1 по правилу (a);

3) $babab$ получается из слова 2 по правилу (b);

4) $abababa$ получается из слова 3 по правилу (a).

Докажем, что для любого слова δ языка L имеет место: $\vdash \delta$ тогда и только тогда, когда δ является палиндромом.

Доказательство. *Необходимость.* Пусть $\vdash \delta$, т. е. существует вывод слова δ в исчислении Pal. Индукцией (точнее, методом возвратной индукции[6]) по длине вывода слова δ докажем, что δ является палиндромом.

База индукции. Длина вывода слова δ равна 1. Тогда δ является аксиомой. Любая аксиома, очевидно, является палиндромом.

Индукционный переход. Предположим, что любое слово, являющееся последним в выводе длины, меньшей n $(n > 1)$, есть палиндром. Рассмотрим вывод длины n слова δ. Если δ является аксиомой, то, как мы уже установили, оно является палиндромом. Иначе δ получено из некоторого предшествующего слова γ этого вывода по правилу (a) или по правилу (b). Тогда $\delta = a\gamma a$ или $\delta = b\gamma b$. По индукционному предположению γ является палиндромом. Следовательно, δ является палиндромом.

Достаточность. Пусть δ — палиндром. Индукцией по длине слова δ докажем, что $\vdash \delta$.

База индукции. Длина δ равна 1 или 2. Тогда, поскольку δ — палиндром, δ является одним из слов a, b, aa или bb. Каждое из этих слов является аксиомой, поэтому $\vdash \delta$.

[6]Пусть $k \in \mathbb{N}$. Напомним, что для доказательства методом возвратной индукции того, что утверждение $\mathcal{P}(n)$ верно при любом натуральном $n \geqslant k$, устанавливают, что для любого натурального $n \geqslant k$ из предположения (называемого индукционным предположением) верности всех $\mathcal{P}(k)$, $\mathcal{P}(k+1)$, $\ldots$, $\mathcal{P}(n-1)$ следует верность $\mathcal{P}(n)$ (в частности, при $n = k$ непосредственно устанавливают верность $\mathcal{P}(k)$). Базы индукции как таковой в этом методе не требуется, однако нам будет удобно выделять доказательство $\mathcal{P}(n)$ при начальных значениях n в виде базы индукции. В дальнейшем и о методе математической индукции, и о методе возвратной индукции мы для краткости будем говорить как об индукции (по натуральному параметру).

Индукционный переход. Предположим, что любой палиндром, длина которого меньше n ($n > 2$), выводим. Рассмотрим палиндром δ длины n. Палиндром δ, очевидно, должен иметь вид $x\gamma x$, где $x \in \Sigma$, γ — слово, являющееся палиндромом. По индукционному предположению $\vdash \gamma$. Поскольку $x = a$ или $x = b$, то по правилу (a) или (b) из γ получаем $x\gamma x$. Итак, $\vdash \delta$.

Доказательство завершено. ◁

Заметим, что каждое из двух правил вывода исчисления Pal является и отношением, и функцией. Однако, чтобы получить тот же результат о выводимости палиндромов, мы могли вместо этих двух правил сформулировать одно правило вывода, являющееся отношением, но не являющееся функцией:

$$\frac{\gamma}{c\gamma c},$$

где γ — произвольное слово языка L, c — произвольный символ алфавита Σ. Этому отношению принадлежит, например, как пара слов $\langle a,\, aaa\rangle$, так и $\langle a,\, bab\rangle$. (Конец примера 1.2.2.) ◁

Упражнение 1.2.3. Задайте исчисление, в котором будут выводимы все слова, являющиеся пропозициональными формулами, и только такие слова. Постройте выводы нескольких пропозициональных формул в этом исчислении. (Не предлагайте исчисление с бесконечным числом аксиом, например, исчисление, в котором аксиомами объявлены все пропозициональные формулы. Хотя такое решение данной задачи в принципе возможно, оно неинтересно, поскольку не приближает нас к уточнению того, как конструируются формулы из конечного числа символов некоторого алфавита.) ◁

Язык-объект и метаязык

Следует отметить, что, строя выводы в каком угодно исчислении, мы пользуемся (формальным) языком этого исчисления. Например, последовательность слов $(*)$ формального языка L из примера 1.2.2 является выводом в исчислении Pal. Понятие вывода (или доказательства) в исчислении строго определено. Чтобы это подчеркнуть, говорят также о *формальном выводе* (или *формальном доказательстве*).

Утверждения о свойствах исчисления мы формулируем и доказываем на русском (естественном, неформальном) языке, используя убедительные математические способы доказательства; в таких случаях говорят, что проводится *содержательное* (или *неформальное*) *доказательство.*[7] Упомянутый в предыдущем предложении естественный язык называется *метаязыком* (или *языком исследователя*). А язык, который мы изучаем (в данном случае — язык исчисления Pal), называется *языком-объектом* (или *предметным языком*). Конечно, при необходимости метаязык можно было бы формализовать и сделать объектом исследования, которое могло бы проводиться на некотором метаязыке.

[7] О понятии содержательного доказательства и о способах доказательства, считающихся в математике убедительными (попросту говоря, принятых в математике), мы скажем подробнее в разделах 3.3.3, 3.5.5, а также в замечаниях 2.5.7, 2.6.23 и в конце раздела 5.1.5.

Индукцию по натуральному параметру и по построению формулы мы, несомненно, считаем убедительным способом доказательства. При этом, конечно, база такой индукции и индукционный переход должны быть доказаны математически убедительно.

Другим примером языка-объекта и метаязыка служит английский язык и русский язык соответственно, когда преподаватель объясняет смысл фраз английского языка на русском языке.

1.2.2. Вывод из гипотез

Для дальнейшего изложения нам понадобится понятие вывода из гипотез. Сейчас мы проиллюстрируем это понятие на доступном примере. Пусть мы желаем с помощью вывода в исчислении Pal получать не только палиндромы, но и слова, в каждом из которых в середине стоит слово ab, а в остальном такие слова строятся как палиндромы. Примерами таких слов являются слова ab, $babb$, $aaba$, $baabab$. Тогда, чтобы воспользоваться исчислением Pal, можно предположить, что слово ab является дополнительной аксиомой этого исчисления, иначе говоря, считать ab гипотезой. Уточним этот пример, дав следующие определения.

Пусть $\mathcal{C}$ — исчисление в языке L, Γ — произвольное множество слов языка L. *Выводом из* Γ *в исчислении* $\mathcal{C}$ называется конечная последовательность слов, каждое из которых является аксиомой, принадлежит Γ или получается из предшествующих слов этой последовательности по некоторому правилу вывода исчисления $\mathcal{C}$. Число членов этой последовательности называется *длиной этого вывода* из Γ. Элементы множества Γ называются *гипотезами вывода* из Γ.

Выводом слова α *из* Γ *в исчислении* $\mathcal{C}$ называется вывод $\alpha_1, \ldots, \alpha_k$ из Γ такой, что α_k совпадает с α.

Говорят, что слово α *выводимо из* Γ *в исчислении* $\mathcal{C}$ и обозначают это как $\Gamma \vdash_{\mathcal{C}} \alpha$, если существует вывод слова α из Γ в исчислении $\mathcal{C}$.

Если слово α не является выводимым из Γ в исчислении $\mathcal{C}$, то его называют *невыводимым из* Γ *в исчислении* $\mathcal{C}$ и обозначают это через $\Gamma \nvdash_{\mathcal{C}} \alpha$.

Обозначения $\Gamma \vdash_{\mathcal{C}} \alpha$ и $\Gamma \nvdash_{\mathcal{C}} \alpha$ могут сокращаться до $\Gamma \vdash \alpha$ и $\Gamma \nvdash \alpha$ соответственно, если понятно, о каком исчислении идёт речь.

Заметим, что в выводе из Γ может использоваться только конечное число формул из Γ. Если множество Γ пусто, то выводимость из него означает выводимость в исчислении $\mathcal{C}$.

$\Gamma \cup \{\gamma\} \vdash \alpha$ будем также записывать как $\Gamma, \gamma \vdash \alpha$. Если Γ состоит из конечного числа слов $\gamma_1, \ldots, \gamma_n$, то $\Gamma \vdash \alpha$ будем также записывать как $\gamma_1, \ldots, \gamma_n \vdash \alpha$.

Пример 1.2.4. Продолжим пример 1.2.2 и построим вывод слова $baabab$ из $\{ab,\ bbba\}$ в исчислении Pal:

1) ab — гипотеза;

2) $aaba$ получается из слова 1 по правилу (a);

3) $baabab$ получается из слова 2 по правилу (b). ◁

1.3. Исчисление высказываний гильбертовского типа

Исчисления оказываются весьма продуктивными для формализации, поиска и исследования доказательств. В данном разделе мы изучим исто-

рически первое исчисление, в котором выводимы все общезначимые пропозициональные формулы и только они.

1.3.1. Формулировка исчисления

Зададим так называемое *исчисление высказываний гильбертовского типа*, являющееся исчислением в языке PL логики высказываний. Мы будем обозначать это исчисление через $\mathcal{H}$.

Аксиомами исчисления $\mathcal{H}$ являются все формулы, имеющие один из следующих видов:

1) $A \supset (B \supset A)$,
2) $(A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C))$,
3) $A \supset (B \supset A \wedge B)$,
4) $A \wedge B \supset A$,
5) $A \wedge B \supset B$,
6) $(A \supset C) \supset ((B \supset C) \supset (A \vee B \supset C))$,
7) $A \supset A \vee B$,
8) $B \supset A \vee B$,
9) $(A \supset B) \supset ((A \supset \neg B) \supset \neg A)$,
10) $\neg\neg A \supset A$,
11) $\mathbf{F} \supset A$,
12) $A \supset \mathbf{T}$,

где A, B, C — произвольные формулы. Таким образом, каждый из пунктов 1–12 представляет схему аксиом; аксиома получается при подстановке конкретных формул в схему аксиом вместо букв A, B и C.

Единственным правилом вывода исчисления $\mathcal{H}$ является правило, которое называется *модусом поненс* (modus ponens, сокращённо MP), а также *правилом отделения*, и которое позволяет получить из двух посылок A и $A \supset B$ заключение B, где A и B — произвольные формулы. Символически это правило записывается в следующем виде:

$$\frac{A;\ A \supset B}{B}.$$

Данное исчисление задумано так, чтобы вывод в нём оказался приемлемым способом получения общезначимых формул. Аксиомы характеризуют семантику логических связок, например, аксиому вида 8 можно интуитивно понимать так: если имеет место высказывание B, то также имеет место высказывание «A или B». Правило вывода модус поненс символически представляет общеупотребительное умозаключение: если имеет место A и известно, что A влечёт B, то имеет место B. В дальнейшем мы докажем, что с помощью вывода в этом исчислении можно получить все общезначимые формулы и только их. Связь исчисления $\mathcal{H}$ с семантикой языка PL

мы начнём исследовать в следующем разделе, а сейчас рассмотрим пример вывода в этом исчислении.

Пример 1.3.1. Построим вывод формулы $A \supset A$ (где A — произвольная пропозициональная формула):

1) $A \supset ((A \supset A) \supset A)$ — аксиома вида 1;

2) $(A \supset ((A \supset A) \supset A)) \supset ((A \supset (A \supset A)) \supset (A \supset A))$ — аксиома вида 2;

3) $(A \supset (A \supset A)) \supset (A \supset A)$ получается из формул 1 и 2 по MP;

4) $A \supset (A \supset A)$ — аксиома вида 1;

5) $A \supset A$ получается из формул 4 и 3 по MP. ◁

1.3.2. Корректность исчисления

Семантика языка PL логики высказываний определена путём сопоставления формулам истинностных значений в интерпретациях. Среди всех формул мы выделили логические законы. Естественное желание при изучении языка PL — отвлечься от его семантики и механически получать результаты об этом языке и о его семантике, например, результаты об общезначимости. Таким механическим способом мог бы стать вывод в некотором исчислении, но при условии, что любая выводимая формула является общезначимой. Поэтому к любому исчислению, предназначенному для получения общезначимых формул, предъявляется естественное требование *корректности*: любая выводимая в этом исчислении формула должна быть общезначимой. В данном разделе мы докажем, что исчисление $\mathcal{H}$ удовлетворяет этому требованию, иначе говоря, является корректным.

Замечание 1.3.2. Заданная семантика языка PL слишком проста, чтобы почувствовать значительную пользу от исчисления (семантические вопросы об общезначимости формул можно решить с помощью таблиц истинности), но чем сложнее семантика, тем сильнее желание перейти от неё к исчислению. Однако мы знакомимся с исчислением высказываний для того, чтобы затем нам было легче перейти к языкам с более сложной семантикой и исчислениям в таких языках. ◁

Утверждение о корректности исчисления $\mathcal{H}$ будет вытекать из следующей теоремы. Также эта теорема выявит то, что вывод из гипотез формализует правильное рассуждение в предположении истинности всех гипотез.

Теорема 1.3.3 (обобщённая теорема о корректности исчисления высказываний $\mathcal{H}$)**.** *Пусть* Γ *— произвольное множество формул,* A *— произвольная формула. Тогда если* $\Gamma \vdash A$*, то* $\Gamma \vDash A$*.*

Доказательство. Применим индукцию по длине вывода $\Gamma \vdash A$.

База индукции. Вывод состоит из одной формулы A. Тогда A — аксиома или гипотеза.

Легко проверить, что каждая аксиома исчисления $\mathcal{H}$ является общезначимой формулой: для этого достаточно составить таблицу истинности для каждой схемы аксиом, считая символы, обозначающие формулы в этой

схеме, пропозициональными переменными, а затем воспользоваться теоремой 1.1.25 о подстановке в тавтологию. Таким образом, если A — аксиома, то $\vDash A$, следовательно, $\Gamma \vDash A$.

Если A — гипотеза, т. е. $A \in \Gamma$, то $\Gamma \vDash A$.

Индукционный переход. Предположим, что доказываемое утверждение верно для любой формулы, являющейся последней в выводе длины, меньшей n ($n > 1$). Рассмотрим вывод $\Gamma \vdash A$ длины n. Случаи, когда формула A является аксиомой или гипотезой, уже разобраны выше. Остаётся разобрать случай, когда A получена из некоторых предшествующих формул B и $B \supset A$ этого вывода по правилу MP. Применив индукционное предположение к $\Gamma \vdash B$ и к $\Gamma \vdash B \supset A$, получим $\Gamma \vDash B$ и $\Gamma \vDash B \supset A$. Отсюда, согласно таблице истинности импликации, $\Gamma \vDash A$. ◁

Следствие 1.3.4. *Пусть* Γ *— произвольное множество общезначимых формул,* A *— произвольное формула. Тогда если* $\Gamma \vdash A$*, то* $\vDash A$*.*

Доказательство. По теореме 1.3.3 из $\Gamma \vdash A$ следует $\Gamma \vDash A$. В силу общезначимости всех формул из Γ $\Gamma \vDash A$ влечёт $\vDash A$. ◁

Следствие 1.3.5 (корректность исчисления высказываний $\mathcal{H}$)**.** *Всякая выводимая в исчислении* $\mathcal{H}$ *формула общезначима, т. е. для любой формулы* A*, если* $\vdash A$*, то* $\vDash A$*.*

Доказательство. Возьмём в качестве Γ пустое множество и применим теорему 1.3.3, помня, что $\varnothing \vdash A$ равносильно $\vdash A$, и $\varnothing \vDash A$ равносильно $\vDash A$. ◁

Следствие 1.3.6. *Исчисление* $\mathcal{H}$ непротиворечиво*, т. е. не существует формулы* A *такой, что* $\vdash A$ *и* $\vdash \neg A$*.*

Доказательство. Если бы нашлась такая формула A, что $\vdash A$ и $\vdash \neg A$, то в силу корректности исчисления имели бы $\vDash A$ и $\vDash \neg A$, что невозможно. ◁

1.3.3. Теорема о дедукции. Допустимые правила

В математических рассуждениях для доказательства утверждения вида «A влечёт B» нередко прибегают к такому способу: предполагают, что верно A, и в этом предположении доказывают B. Для исчисления $\mathcal{H}$ этот способ обосновывается следующей теоремой.

Теорема 1.3.7 (теорема о дедукции для исчисления высказываний $\mathcal{H}$)**.** *Пусть* Γ *— произвольное множество формул,* A*,* B *— произвольные формулы. Тогда* $\Gamma, A \vdash B$*, если и только если* $\Gamma \vdash A \supset B$*.*

Доказательство. То, что $\Gamma \vdash A \supset B$ влечёт $\Gamma, A \vdash B$, обосновывается применением правила MP к данному $\Gamma \vdash A \supset B$ и очевидному $\Gamma, A \vdash A$.

Докажем, что $\Gamma, A \vdash B$ влечёт $\Gamma \vdash A \supset B$. Воспользуемся индукцией по длине вывода $\Gamma, A \vdash B$.

База индукции. Вывод состоит из одной формулы B. Тогда возможны лишь 3 случая: (a) B является аксиомой, (b) B совпадает с A, (c) B принадлежит Γ. Разберём каждый из этих случаев.

(a) По правилу MP из $\vdash B$ (B — аксиома) и $\vdash B \supset (A \supset B)$ (аксиома вида 1) получаем $\vdash A \supset B$. Следовательно, $\Gamma \vdash A \supset B$.

(b) Согласно примеру 1.3.1 имеем $\vdash A \supset A$, следовательно, $\Gamma \vdash A \supset A$, т. е. $\Gamma \vdash A \supset B$ (поскольку B совпадает с A в рассматриваемом случае).

(c) По правилу MP из $\Gamma \vdash B$ (B принадлежит Γ) и $\vdash B \supset (A \supset B)$ (аксиома вида 1) получаем $\Gamma \vdash A \supset B$.

Индукционный переход. Предположим, что доказываемое утверждение верно для любого вывода длины, меньшей n ($n > 1$). Рассмотрим вывод $\Gamma, A \vdash B$ длины n. Если не имеет место ни один из уже разобранных случаев (a), (b), (c), то формула B получена по правилу MP из некоторых предшествующих формул C и $C \supset B$ этого вывода. Применив индукционное предположение к $\Gamma, A \vdash C$ и к $\Gamma, A \vdash C \supset B$, получим $(*)$ $\Gamma \vdash A \supset C$ и $(**)$ $\Gamma \vdash A \supset (C \supset B)$. По правилу MP из $(**)$ и аксиомы вида 2 $(A \supset (C \supset B)) \supset ((A \supset C) \supset (A \supset B))$ получим $\Gamma \vdash (A \supset C) \supset (A \supset B)$. Отсюда и из $(*)$ по правилу MP получим $\Gamma \vdash A \supset B$. ◁

Условимся, что далее в разделе 1.3 символы Γ и Δ (возможно, с индексами) будут обозначать произвольные множества пропозициональных формул, если не указано иное.

Исчисление $\mathcal{H}$ удобно для некоторых теоретических исследований в силу того, что выводы в нём имеют простую структуру, поскольку в этом исчислении всего лишь одно правило вывода. Однако искать выводы в этом исчислении неудобно. Как мы могли видеть в примере 1.3.1, поиск вывода простой формулы потребовал бы некоторой изобретательности. Мы хотим иметь более удобные средства для доказательства выводимости формул. Кроме того, желательно формализовать с помощью исчисления $\mathcal{H}$ обычные способы математических рассуждений (т. е. способы рассуждений, которые применяются в математических доказательствах). Здесь мы приведём вспомогательные правила, которые соответствуют обычным способам математических рассуждений. Такие вспомогательные правила иначе называют *техникой естественного вывода*.

Теорему 1.3.7 о дедукции можно записать в виде следующих вспомогательных правил:

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B}, \qquad \frac{\Gamma \vdash A \supset B}{\Gamma, A \vdash B}.$$

Первое правило понимается так: если $\Gamma, A \vdash B$, то $\Gamma \vdash A \supset B$. Как мы отметили перед формулировкой теоремы о дедукции, это правило соответствует одному из обычных способов рассуждений в математике. Это вспомогательное правило называется *правилом введения импликации* (или, короче, $\supset$-*введением*). Второе правило понимается аналогичным образом: если $\Gamma \vdash A \supset B$, то $\Gamma, A \vdash B$.

Далее, если $\Gamma \vdash A$ и $\Gamma \vdash A \supset B$, то по MP получаем $\Gamma \vdash B$. Таким образом, имеем ещё одно вспомогательное правило

$$\frac{\Gamma \vdash A; \quad \Gamma \vdash A \supset B}{\Gamma \vdash B},$$

называемое *правилом удаления импликации* (или, короче, $\supset$-*удалением*).

С помощью приведённых вспомогательных правил можно устанавливать выводимость некоторых формул (в том числе и из гипотез), не строя выводы целиком. При этом для доказательства выводимости некоторой формулы обычно пользуются правилами снизу вверх: чтобы установить выводимость, указанную в правиле под чертой, достаточно установить все выводимости, указанные над чертой.

Пример 1.3.8. Докажем, что для любых формул A, B и C

$$\vdash (A \supset B) \supset ((B \supset C) \supset (A \supset C)).$$

По $\supset$-введению достаточно доказать

$$A \supset B \vdash (B \supset C) \supset (A \supset C).$$

Для этого, в свою очередь, по $\supset$-введению достаточно доказать

$$A \supset B,\ B \supset C \vdash A \supset C.$$

Опять по $\supset$-введению достаточно доказать

$$A \supset B,\ B \supset C,\ A \vdash C. \qquad (\star)$$

$(\star)$ получается по $\supset$-удалению из

$$A \supset B,\ \underline{B \supset C},\ A \vdash \underline{B \supset C},$$

что очевидно (см. подчёркнутую формулу и определение вывода из гипотез), и

$$A \supset B,\ B \supset C,\ A \vdash B, \qquad (\star\star)$$

что достаточно доказать для завершения доказательства исходного утверждения. $(\star\star)$ получается по $\supset$-удалению из

$$\underline{A \supset B},\ B \supset C,\ A \vdash \underline{A \supset B},$$

что очевидно, и так же очевидного

$$A \supset B,\ B \supset C,\ \underline{A} \vdash \underline{A}.$$

Итак, требуемая выводимость доказана.

Проведённое доказательство удобно записать в следующем виде:

1) $\vdash (A \supset B) \supset ((B \supset C) \supset (A \supset C))$ из 2 по $\supset$-введению;

2) $A \supset B \vdash (B \supset C) \supset (A \supset C)$ из 3 по $\supset$-введению;

3) $A \supset B,\ B \supset C \vdash A \supset C$ из 4 по $\supset$-введению;

4) $A \supset B,\ B \supset C,\ A \vdash C$ из 5, 6 по $\supset$-удалению;

5) $A \supset B,\ B \supset C,\ A \vdash B \supset C$ верно;

6) $A \supset B,\ B \supset C,\ A \vdash B$ из 7, 8 по $\supset$-удалению;

7) $A \supset B,\ B \supset C,\ A \vdash A \supset B$ верно;

Связка $\odot$	$\odot$-введение	$\odot$-удаление
$\supset$	$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B}$,	$\dfrac{\Gamma \vdash A; \quad \Gamma \vdash A \supset B}{\Gamma \vdash B}$,
$\neg$	$\dfrac{\Gamma, A \vdash B; \quad \Gamma, A \vdash \neg B}{\Gamma \vdash \neg A}$,	$\dfrac{\Gamma \vdash \neg\neg A}{\Gamma \vdash A}$,
$\wedge$	$\dfrac{\Gamma \vdash A; \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$,	$\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}$, $\quad \dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$,
$\vee$	$\dfrac{\Gamma \vdash A}{\Gamma \vdash A \vee B}$, $\quad \dfrac{\Gamma \vdash B}{\Gamma \vdash A \vee B}$,	$\dfrac{\Gamma, A \vdash C; \quad \Gamma, B \vdash C}{\Gamma, A \vee B \vdash C}$.

Таблица 1.6. Допустимые в исчислении $\mathcal{H}$ правила.

8) $A \supset B,\ B \supset C,\ A \vdash A$ верно. $\triangleleft$

Перечислим некоторые вспомогательные правила в таблице 1.5, для систематизации повторив некоторые уже известные нам.

Каждое из этих вспомогательных правил имеет вид

$$\frac{\Gamma_1 \vdash A_1; \quad \ldots; \quad \Gamma_n \vdash A_n}{\Gamma \vdash A}, \quad \text{где } n \in \mathbb{N}_+. \qquad (\bigstar)$$

Запись ($\bigstar$) назовём *допустимым* (*в исчислении* $\mathcal{H}$) *правилом*, если верно утверждение: $\Gamma_1 \vdash A_1;\ \ldots;\ \Gamma_n \vdash A_n$ влечёт $\Gamma \vdash A$. Подчеркнём, что, согласно данному определению, допустимые правила не являются правилами вывода рассматриваемого исчисления, а являются своеобразными записями утверждений указанного вида.

Докажем, что в таблице 1.5 действительно приведены только допустимые правила.

Д о к а з а т е л ь с т в о. Допустимость $\supset$-введения и $\supset$-удаления установлена ранее в этом разделе.

Для доказательства допустимости $\neg$-введения применим теорему о дедукции к $\Gamma, A \vdash B$ и к $\Gamma, A \vdash \neg B$. Получим $(*)$ $\Gamma \vdash A \supset B$ и $(**)$ $\Gamma \vdash A \supset \neg B$. По правилу MP из аксиомы вида 9 $(A \supset B) \supset ((A \supset \neg B) \supset \neg A)$ и $(*)$ получим $\Gamma \vdash (A \supset \neg B) \supset \neg A$. Отсюда и из $(**)$ по правилу MP получим $\Gamma \vdash \neg A$, что и требуется.

$\neg$-удаление также допустимо: из $\Gamma \vdash \neg\neg A$ и аксиомы вида 10 $\neg\neg A \supset A$ по правилу MP получим $\Gamma \vdash A$.

Далее, $\wedge$-введение допустимо: из $\Gamma \vdash A$, $\Gamma \vdash B$ и аксиомы вида 3 $A \supset (B \supset (A \wedge B))$, дважды применив правило MP, получим $\Gamma \vdash A \wedge B$.

Каждое правило, называемое $\wedge$-удалением, допустимо: из $\Gamma \vdash A \wedge B$ и аксиомы вида 4 $A \wedge B \supset A$ по правилу MP получим $\Gamma \vdash A$; аналогично, из $\Gamma \vdash A \wedge B$ и аксиомы вида 5 $A \wedge B \supset B$ по правилу MP получим $\Gamma \vdash B$.

Каждое правило, называемое $\vee$-введением, допустимо: из $\Gamma \vdash A$ и аксиомы вида 7 $A \supset A \vee B$ по правилу MP получим $\Gamma \vdash A \vee B$; аналогично, из $\Gamma \vdash B$ и аксиомы вида 8 $B \supset A \vee B$ по правилу MP получим $\Gamma \vdash A \vee B$.

Наконец, установим, что $\vee$-удаление допустимо. По теореме о дедукции из $\Gamma, A \vdash C$ и $\Gamma, B \vdash C$ следует $\Gamma \vdash A \supset C$ и $\Gamma \vdash B \supset C$. Отсюда и из аксиомы вида 6 $(A \supset C) \supset ((B \supset C) \supset (A \vee B \supset C))$, дважды применив правило MP, получим $\Gamma \vdash A \vee B \supset C$. Применив MP к $A \vee B \vdash A \vee B$ (что, очевидно, верно) и $\Gamma \vdash A \vee B \supset C$, получим $\Gamma, A \vee B \vdash C$. $\triangleleft$

Видно, что допустимые правила, перечисленные выше, соответствуют обычным способам математических рассуждений. Например, $\neg$-введение соответствует способу рассуждения, называемому *приведением к абсурду* (*нелепости*, *противоречию*): чтобы доказать $\neg A$, достаточно допустить A и получить из этого допущения противоречие, т. е. для некоторого B вывести одновременно B и $\neg B$. $\vee$-удаление соответствует способу доказательства *разбором случаев*: чтобы доказать, что из A или B следует C, достаточно доказать, что и из A следует C, и из B следует C.

Пример 1.3.9. Докажем, что для любых формул A и B

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

(В силу теоремы о дедукции эту выводимость можно переписать в виде $A, \neg A \vdash B$ и прочитать так: из противоречия выводима любая формула.)

Представим доказательство в таком же виде, как и в конце примера 1.3.8:

1) $\vdash \neg A \supset (A \supset B)$ из 2 по $\supset$-введению;

2) $\neg A \vdash A \supset B$ из 3 по $\supset$-введению;

3) $\neg A, A \vdash B$ из 4 по $\neg$-удалению;

4) $\neg A, A \vdash \neg\neg B$ из 5, 6 по $\neg$-введению;

5) $\neg A, A, \neg B \vdash A$ верно;

6) $\neg A, A, \neg B \vdash \neg A$ верно. $\triangleleft$

Упражнение 1.3.10. Докажите, что следующее правило, называемое правилом сечения, допустимо:

$$\frac{\Gamma_1 \vdash A; \quad \Gamma_2, A \vdash B}{\Gamma_1 \cup \Gamma_2 \vdash B}.$$

$\triangleleft$

Упражнение 1.3.11. Докажите, что следующее правило, являющееся одним из вариантов $\vee$-удаления, допустимо:

$$\frac{\Gamma \vdash A \vee B; \quad \Gamma, A \vdash C; \quad \Gamma, B \vdash C}{\Gamma \vdash C}.$$

$\triangleleft$

Упражнение 1.3.12. Докажите, что следующее правило, являющееся одним из вариантов $\wedge$-удаления, допустимо:

$$\frac{\Gamma, A, B \vdash C}{\Gamma, A \wedge B \vdash C}.$$

$\triangleleft$

Упражнение 1.3.13. Докажите, что следующие правила допустимы:

$$\frac{\Gamma, A \vdash B; \quad \Gamma, B \vdash A}{\Gamma \vdash A \equiv B}, \quad \frac{\Gamma \vdash A \equiv B}{\Gamma, A \vdash B}, \quad \frac{\Gamma \vdash A \equiv B}{\Gamma, B \vdash A}.$$

◁

Упражнение 1.3.14. Докажите выводимость всех логических законов, перечисленных в разделе 1.1.3. ◁

1.3.4. Полнота исчисления

В разделе 1.3.2 мы установили корректность исчисления $\mathcal{H}$: всякая выводимая в исчислении $\mathcal{H}$ формула общезначима. В этом разделе мы докажем обратное утверждение — так называемую теорему о *полноте* этого исчисления: всякая общезначимая формула выводима в исчислении $\mathcal{H}$. Сначала дадим несколько определений, которые будут использоваться в доказательстве.

Определение 1.3.15. Множество формул Γ называется *непротиворечивым*, если не существует формулы A такой, что $\Gamma \vdash A$ и $\Gamma \vdash \neg A$. Иначе Γ называется *противоречивым*. ◁

Определение 1.3.16. Множество формул Γ называется *полным*, если для любой формулы A $\Gamma \vdash A$ или $\Gamma \vdash \neg A$. Иначе Γ называется *неполным*. ◁

Определение 1.3.17. Пусть Γ — произвольное множество формул. Интерпретацию языка PL логики высказываний, в которой истинны все формулы множества Γ, называют *моделью множества* Γ. Если существует модель множества Γ, то говорят, что Γ *имеет модель* (или Γ *совместно*). ◁

Теорема 1.3.18. *Всякое множество* Γ *замкнутых формул, имеющее модель, непротиворечиво.*

Доказательство. Пусть множество Γ имеет модель M. Тогда для любой формулы $G \in \Gamma$ верно $M \vDash G$. Если бы множество Γ было противоречивым, то для некоторой формулы A выполнялось бы $\Gamma \vdash A$ и $\Gamma \vdash \neg A$. Тогда по обобщённой теореме 1.3.3 о корректности исчисления $\mathcal{H}$ мы имели бы $M \vDash A$ и $M \vDash \neg A$, что невозможно. Поэтому множество Γ непротиворечиво. ◁

Теорему о полноте исчисления высказываний мы получим как следствие теоремы 1.3.19, обратной к только что доказанной теореме и формулируемой ниже.

Теорема 1.3.19. *Всякое непротиворечивое множество формул* Γ *имеет модель.*

Доказательству теоремы 1.3.19 мы предпошлём две леммы, но сначала изложим идею всего доказательства. Для непротиворечивого множества формул Γ и любой переменной p имеет место ровно одно из трёх: (1) $\Gamma \vdash p$, (2) $\Gamma \vdash \neg p$ или (3) $\Gamma \nvdash p$ и $\Gamma \nvdash \neg p$. Мы ищем модель множества Γ, т. е. интерпретацию M, в которой истинны все формулы этого множества. По обобщённой теореме 1.3.3 о корректности исчисления $\mathcal{H}$ в случае (1) должно

быть $M \vDash p$, а в случае (2) — $M \vDash \neg p$ и, значит, $M \nvDash p$. В случае (3) пока непонятно, какое истинностное значение сопоставить переменной p. Чтобы до конца определить искомую интерпретацию M, достаточно найти полное и непротиворечивое множество Δ, содержащее Γ: для такого множества Δ случай (3) не может иметь места. Наконец, показав, что в заданной интерпретации M действительно истинны все формулы из Δ, получим, что то же самое свойство имеет место и для подмножества Γ множества Δ.

Лемма 1.3.20. *Пусть Γ — непротиворечивое множество формул, A — формула. Тогда хотя бы одно из множеств $\Gamma \cup \{A\}$ или $\Gamma \cup \{\neg A\}$ непротиворечиво.*

Доказательство. Предположим, что оба множества $\Gamma \cup \{A\}$ и $\Gamma \cup \{\neg A\}$ противоречивы. Тогда найдутся такие формулы B и C такие, что $\Gamma, A \vdash B$, $\Gamma, A \vdash \neg B$ и $\Gamma, \neg A \vdash C$, $\Gamma, \neg A \vdash \neg C$.

По $\neg$-введению (см. с. 41) из $\Gamma, A \vdash B$ и $\Gamma, A \vdash \neg B$ получаем $\Gamma \vdash \neg A$. Аналогично из $\Gamma, \neg A \vdash C$ и $\Gamma, \neg A \vdash \neg C$ получаем $\Gamma \vdash \neg\neg A$.

Итак, имеем $\Gamma \vdash \neg A$ и $\Gamma \vdash \neg\neg A$, что противоречит непротиворечивости Γ. Поэтому сделанное предположение неверно. ◁

Лемма 1.3.21 (лемма Линденбаума)**.** *Всякое непротиворечивое множество формул Γ содержится в некотором непротиворечивом полном множестве формул.*

Доказательство. Очевидно, язык PL счётен. Пусть $A_1, A_2, \ldots$ — все пропозициональные формулы, перечисленные в некотором порядке.

Положим $\Delta_0 \leftrightharpoons \Gamma$. Для каждого $i \in \mathbb{N}_+$ определим множество Δ_i. Если $\Delta_{i-1} \cup \{A_i\}$ непротиворечиво, то положим $\Delta_i \leftrightharpoons \Delta_{i-1} \cup \{A_i\}$. Иначе (в этом случае $\Delta_{i-1} \cup \{\neg A_i\}$ непротиворечиво по лемме 1.3.20) положим $\Delta_i \leftrightharpoons \Delta_{i-1} \cup \{\neg A_i\}$. Для каждого $i \in \mathbb{N}$ множество Δ_i непротиворечиво по построению.

Пусть $\Delta \leftrightharpoons \cup_{i\in\mathbb{N}}\Delta_i$. По определению множество Δ содержит Γ. Для каждой формулы A множество Δ содержит либо A, либо $\neg A$, и потому Δ полно.

Чтобы утверждать, что Δ годится в качестве искомого множества, осталось лишь доказать непротиворечивость Δ. Если бы Δ было противоречивым, то нашлась бы формула A такая, что $\Delta \vdash A$ и $\Delta \vdash \neg A$. Тогда, поскольку в выводе может использоваться только конечное число формул из Δ, найдётся j, при котором $\Delta_j \vdash A$ и $\Delta_j \vdash \neg A$, что противоречит непротиворечивости Δ_j. Значит, Δ непротиворечиво. ◁

Доказательство теоремы 1.3.19. В силу леммы 1.3.21 достаточно доказать существование модели множества Γ, считая, что Γ непротиворечиво и полно. Тогда для каждой пропозициональной переменной p верно либо $\Gamma \vdash p$, либо $\Gamma \vdash \neg p$, но не то и другое одновременно. Мы ищем интерпретацию, в которой все формулы из Γ истинны, поэтому, согласно обобщённой теореме 1.3.3 о корректности исчисления $\mathcal{H}$, в первом случае переменной p мы должны сопоставить истину, а во втором — ложь. Так и определим интерпретацию M языка PL, сопоставив каждой переменной p истинностное значение 1, если $\Gamma \vdash p$, и значение 0 иначе (т. е. если $\Gamma \vdash \neg p$).

Чтобы доказать, что любая формула из Γ истинна в интерпретации M, достаточно доказать, что для любой формулы A $M \vDash A$ тогда и только тогда, когда $\Gamma \vdash A$. Воспользуемся индукцией по построению формулы A.

База индукции. A — переменная или истинностная константа. Если A — переменная, то доказываемое утверждение следует из определения интерпретации M. Если A — истинностная константа $\mathbf{F}$, то $M \nvDash \mathbf{F}$ и $\Gamma \nvdash \mathbf{F}$ (иначе из $\Gamma \vdash \mathbf{F}$ и аксиомы вида 11 по правилу MP получили бы, что из Γ выводима любая формула). Если A — истинностная константа $\mathbf{T}$, то $M \vDash \mathbf{T}$ и $\Gamma \vdash \mathbf{T}$ ($\Gamma \vdash \mathbf{T}$ получается по правилу MP из произвольной выводимой из Γ формулы и аксиомы вида 12).

Индукционный переход. Предположим, что доказываемое утверждение верно для формул B и C.

1. Пусть A имеет вид $\neg B$. $M \vDash A$ тогда и только тогда, когда $M \nvDash B$. По индукционному предположению $M \nvDash B$ равносильно $\Gamma \nvdash B$. В силу непротиворечивости и полноты Γ имеем: $\Gamma \nvdash B$ тогда и только тогда, когда $\Gamma \vdash \neg B$, т. е. когда $\Gamma \vdash A$.

2. Пусть A имеет вид $B \wedge C$. $M \vDash A$ тогда и только тогда, когда $M \vDash B$ и $M \vDash C$. По индукционному предположению $M \vDash B$ и $M \vDash C$, если и только если $\Gamma \vdash B$ и $\Gamma \vdash C$.

Из $\Gamma \vdash B$, $\Gamma \vdash C$ по $\wedge$-введению получаем $\Gamma \vdash B \wedge C$. С другой стороны, из $\Gamma \vdash B \wedge C$ по $\wedge$-удалению получаем $\Gamma \vdash B$ и $\Gamma \vdash C$. Таким образом, $\Gamma \vdash B$ и $\Gamma \vdash C$ тогда и только тогда, когда $\Gamma \vdash B \wedge C$, т. е. когда $\Gamma \vdash A$.

Оставшиеся случаи — 3 (A имеет вид $B \vee C$) и 4 (A имеет вид $B \supset C$) — рассматриваются аналогично случаю 2 и остаются в качестве упражнения. ◁

Упражнение 1.3.22. Разберите случаи 3 и 4 в доказательстве предыдущей теоремы. ◁

Теорема 1.3.23 (теорема о компактности для логики высказываний)**.** *Пусть* Γ *— множество формул. Тогда если всякое конечное подмножество множества* Γ *имеет модель, то* Γ *имеет модель.*

Доказательство. Пусть всякое конечное подмножество множества Γ имеет модель. Однако предположим, что Γ не имеет модели.

Тогда по теореме 1.3.19 множество Γ противоречиво, т. е. существует формула A такая, что $\Gamma \vdash A$ и $\Gamma \vdash \neg A$. Найдётся конечное множество $\Gamma_0 \subseteq \Gamma$ такое, что $\Gamma_0 \vdash A$ и $\Gamma_0 \vdash \neg A$. Таким образом, Γ_0 противоречиво.

По теореме 1.3.18 из противоречивости Γ_0 следует, что Γ_0 не имеет модели. Это противоречит условию доказываемой теоремы, поэтому предположение неверно, а Γ имеет модель. ◁

Теорема 1.3.24 (обобщённая теорема о полноте исчисления высказываний $\mathcal{H}$)**.** *Пусть* Γ *— множество формул,* A *— формула. Тогда если* $\Gamma \vDash A$*, то* $\Gamma \vdash A$*.*

Доказательство. Так как $\Gamma \vDash A$, то множество $\Gamma \cup \{\neg A\}$ не имеет модели, следовательно, по теореме 1.3.19 это множество противоречиво. Значит, найдётся формула B такая, что $\Gamma, \neg A \vdash B$ и $\Gamma, \neg A \vdash \neg B$. Тогда по $\neg$-введению $\Gamma \vdash \neg\neg A$, отсюда по $\neg$-удалению получаем $\Gamma \vdash A$. ◁

Теорема 1.3.25 (полнота исчисления высказываний $\mathcal{H}$)**.** *Всякая общезначимая пропозициональная формула выводима в исчислении $\mathcal{H}$, т. е. для любой формулы A, если $\models A$, то $\vdash A$.*

Доказательство. Возьмём в качестве Γ пустое множество и применим предыдущую теорему. ◁

Итак, теоремы о корректности и полноте исчисления высказываний $\mathcal{H}$ показывают, что вывод в этом исчислении является полностью адекватным способом получения общезначимых формул.

Упражнение 1.3.26. Докажите полноту исчисления $\mathcal{H}$ по следующей схеме.

1. Докажите, что для любых формул A и B имеет место

$$\begin{array}{ccc} A \vdash \neg\neg A, & \neg A \vdash \neg A, & \\ A, B \vdash A \wedge B, & A, B \vdash A \vee B, & A, B \vdash A \supset B, \\ A, \neg B \vdash \neg(A \wedge B), & A, \neg B \vdash A \vee B, & A, \neg B \vdash \neg(A \supset B), \\ \neg A, B \vdash \neg(A \wedge B), & \neg A, B \vdash A \vee B, & \neg A, B \vdash A \supset B, \\ \neg A, \neg B \vdash \neg(A \wedge B), & \neg A, \neg B \vdash \neg(A \vee B), & \neg A, \neg B \vdash A \supset B. \end{array}$$

2. Пусть A — формула, в которую входят лишь переменные из списка $p_1, \ldots, p_n$ попарно различных переменных. Докажите, что тогда для любой интерпретации M $\overline{p}_1, \ldots, \overline{p}_n \vdash \overline{A}$, где $\overline{B}$ есть B, если $M \models B$, и $\neg B$ иначе. (Сравните это утверждение с утверждением пункта 1.)

3. Пусть A — общезначимая формула, в которую входят лишь переменные из списка $p_1, \ldots, p_n$ попарно различных переменных. Докажите, что тогда $p_1 \vee \neg p_1,\ \ldots,\ p_n \vee \neg p_n \vdash A$. Отсюда получите теорему о полноте исчисления $\mathcal{H}$. ◁

1.3.5. Поиск вывода и алгоритм Британского музея

Исчисление высказываний гильбертовского типа $\mathcal{H}$ удобно для прямого способа рассуждения: из некоторых формул по правилу вывода модус поненс получаем заключения. Если таким способом, исходя из аксиом, механически получать всё новые и новые теоремы исчисления $\mathcal{H}$, то многие из этих теорем будут попросту неинтересными для нас. Часто требуется найти вывод заранее заданной формулы или выяснить, является ли заданная формула выводимой.

При поиске вывода заданной формулы A в исчислении $\mathcal{H}$ можно действовать следующим образом: если A не является аксиомой, то найти формулы, из которых A получается по правилу вывода MP, и для каждой найденной формулы, если она не является аксиомой, найти формулы, из которых она получается по MP, и т. д. Однако нахождение формул, из которых данная формула получается по правилу MP, по сути является угадыванием, и по крайней мере одна из посылок правила MP является более длинной, чем заключение. По этой причине исчисление $\mathcal{H}$ считают неудобным для поиска вывода заранее заданной формулы.

Указанный способ поиска вывода в исчислении $\mathcal{H}$ распространяется и на другие исчисления. Опишем в общих чертах этот способ поиска вывода

в произвольном исчислении: если заданное слово не является аксиомой, то найти слова, из которых оно получается по одному из правил вывода, и для каждого найденного слова, если оно не является аксиомой, найти слова, из которых оно получается по одному из правил вывода, и т. д. Такой способ поиска вывода называется поиском вывода *снизу вверх*.

Пусть Σ — какой угодно алфавит, $\mathcal{C}$ — произвольное исчисление в алфавите Σ. Ставя задачу поиска вывода в исчислении $\mathcal{C}$ любого слова в алфавите Σ, желают иметь алгоритм (*алгоритм поиска вывода*), который по любому слову A в алфавите Σ выдаёт вывод слова A, если и только если A выводимо в $\mathcal{C}$ (такой алгоритм может не заканчивать работу, если A невыводимо в $\mathcal{C}$). Также рассматривают задачу проверки выводимости (в исчислении $\mathcal{C}$) любого слова в алфавите Σ: желают иметь алгоритм (*алгоритм проверки выводимости*), который по любому слову A в алфавите Σ выдаёт 1, если и только если A выводимо в $\mathcal{C}$. Для каждой из этих двух задач наличие алгоритма, который вдобавок к упомянутым требованиям для любого невыводимого слова выдаёт 0, рассматривается как очень полезное, но не всегда имеющее место свойство рассматриваемого исчисления. (Для задачи поиска вывода, конечно, предполагается, что 0 не является выводом.)

Отметим, что требование к упомянутым алгоритмам выдавать в качестве ответа 1 или 0 несущественно; вместо 1 может выдаваться другое слово — индикатор того, что заданное слово выводимо, например, «выводимо», а вместо 0 может выдаваться слово, отличное от первого индикатора, например, «невыводимо». В неформальных описаниях алгоритмов мы будем применять подобные соглашения.

Для задачи проверки выводимости в исчислении $\mathcal{H}$ можно использовать следующий алгоритм: по поданной на вход формуле A строится таблица истинности этой формулы и выдаётся ответ «выводима», если обнаружена общезначимость A, иначе выдаётся ответ «невыводима». Этот алгоритм всегда завершает работу и выдаёт правильный ответ в силу корректности и полноты исчисления $\mathcal{H}$.

Мы уже говорили о том, что знакомимся с исчислением высказываний с целью подготовки к изучению языков с более сложной семантикой и исчислений в них. Поэтому мы стремимся отвлечься от семантики. Так мы и поступим сейчас, описав алгоритм поиска вывода в исчислении $\mathcal{H}$, который не использует семантических понятий.

Для поиска вывода любой заданной формулы в исчислении $\mathcal{H}$ (и во многих других исчислениях) существует *алгоритм Британского музея*: последовательно порождать все (возможно, с точностью до некоторой эквивалентности)[8] выводы до тех пор, пока не встретится вывод заданной формулы. Такой алгоритм порождает много лишних формул и поэтому вычислительно крайне неэффективен; но его теоретическая ценность именно в том, что он существует для многих исчислений. Также отметим, что такой алгоритм работает бесконечно, если заданная формула невыводима.

Легко строится алгоритм поиска вывода в исчислении $\mathcal{H}$, который для любой заданной формулы завершает работу и выдаёт вывод заданной формулы, если она выводима в $\mathcal{H}$, а иначе выдаёт «невыводима». Действитель-

[8] К исчислению $\mathcal{H}$ не относится фраза в скобках, отмеченная номером данного подстрочного примечания.

но, пусть этот алгоритм строит таблицу истинности заданной формулы и выдаёт «невыводима», если обнаружена её необщезначимость; иначе выдаёт результат работы алгоритма Британского музея над этой формулой.

Упражнение 1.3.27. Детально разработайте алгоритм Британского музея для исчисления $\mathcal{H}$. (По сути, надо разработать такой алгоритм порождения выводимых формул, который гарантирует, что рано или поздно любая выводимая формула будет порождена.) $\triangleleft$

1.4. Секвенциальное исчисление высказываний

Построим исчисление, которое, в отличие от исчисления высказываний гильбертовского типа, позволяет целенаправленно искать вывод заданной пропозициональной формулы, раскладывая задачу поиска вывода на аналогичные, но более простые подзадачи.

1.4.1. Поиск контрпримера для пропозициональной формулы

Пропозициональная формула общезначима тогда и только тогда, когда не существует интерпретации языка PL логики высказываний, в которой эта формула ложна. Интерпретация M называется *контрпримером для формулы* A, если $M \not\models A$. Продемонстрируем, как можно систематически искать контрпримеры для пропозициональных формул.

Пример 1.4.1. Попробуем найти какой-нибудь контрпример для формулы

$$A \leftrightharpoons \neg(p \wedge q) \supset (\neg p \vee r).$$

Предположим, что контрпример для A существует, и обозначим его через M. Формула A ложна в M тогда и только тогда, когда $\neg(p \wedge q)$ истинна в M и $(\neg p \vee r)$ ложна в M.

Станем записывать формулы, которые должны быть ложны в искомой интерпретации M справа от знака $\to$, а формулы, которые должны быть истинны в M — слева. В этих обозначениях имеем: $\to \neg(p \wedge q) \supset (\neg p \vee r)$ равносильно $\neg(p \wedge q) \to (\neg p \vee r)$.

Далее имеем: для $\neg(p \wedge q) \to (\neg p \vee r)$ необходимо и достаточно $\to (\neg p \vee r), (p \wedge q)$. В свою очередь, $\to (\neg p \vee r), (p \wedge q)$, если и только если (a) $\to (\neg p \vee r), p$ или (b) $\to (\neg p \vee r), q$; таким образом, поиск контрпримера разделился на два пути: один начинается с выражения (a), а другой — с (b). Искомый контрпример будет получен, если мы найдём контрпример хотя бы на одном из этих путей поиска.

Исследуем путь, начинающийся с выражения (a). $\to (\neg p \vee r), p$ равносильно $\to \neg p, r, p$, что в свою очередь равносильно $p \to r, p$. Видно, что на этом пути контрпримера не найти: p должна быть одновременно истинной и ложной в искомой интерпретации M.

На пути (b) имеем: $\to (\neg p \vee r), q$ равносильно $\to \neg p, r, q$, что в свою очередь равносильно $p \to r, q$. Теперь по выражению $p \to r, q$ можно задать контрпример M: $M(p) = 1$, $M(r) = 0$, $M(q) = 0$, остальным переменным интерпретация M сопоставляет какие угодно истинностные значения, для

определённости 0. Итак, найден контрпример для исходной формулы A, поэтому она необщезначима.

Только что проведённый поиск контрпримера можно изобразить в виде растущего вверх дерева, корнем которого является формула A с приписанным к ней слева символом $\to$:

$$\dfrac{\dfrac{\dfrac{\dfrac{p \to r, p}{\to \neg p, r, p}}{\to (\neg p \vee r), p} \quad \dfrac{\dfrac{p \to r, q}{\to \neg p, r, q}}{\to (\neg p \vee r), q}}{\to (\neg p \vee r), (p \wedge q)}}{\dfrac{\neg(p \wedge q) \to (\neg p \vee r)}{\to \neg(p \wedge q) \supset (\neg p \vee r)}}.$$

◁

Пример 1.4.2. Попробуем найти контрпример для формулы

$$\neg(p \wedge q) \supset (\neg p \vee \neg q)$$

и сразу представим поиск контрпримера в виде дерева:

$$\dfrac{\dfrac{\dfrac{\dfrac{p \to \neg q, p}{\to \neg p, \neg q, p}}{\to (\neg p \vee \neg q), p} \quad \dfrac{\dfrac{q \to \neg p, q}{\to \neg p, \neg q, q}}{\to (\neg p \vee \neg q), q}}{\to (\neg p \vee \neg q), (p \wedge q)}}{\dfrac{\neg(p \wedge q) \to (\neg p \vee \neg q)}{\to \neg(p \wedge q) \supset (\neg p \vee \neg q)}}.$$

Здесь один путь поиска контрпримера привёл к выражению $p \to \neg q, p$, которое предписывает, что p должна быть истинна и ложна одновременно (в искомой интерпретации-контрпримере), а другой путь — к выражению $q \to \neg p, q$, которое предписывает, что q должна быть истинна и ложна одновременно. Таким образом, для исходной формулы не существует контрпримера, что равносильно общезначимости этой формулы. ◁

Можно заметить, что поиск контрпримера для формулы осуществляется по нескольким правилам, точнее, по двум правилам для каждой логической связки: одно правило для связки, стоящей слева от $\to$, а другое — для связки, стоящей справа от $\to$. Такие правила формализуются в исчислении, которое определено в следующем разделе 1.4.2.

1.4.2. Формулировка исчисления

Сформулируем исчисление, называемое *секвенциальным исчислением высказываний генценовского типа*, а также, короче, *секвенциальным исчислением высказываний* или *исчислением высказываний генценовского типа*. Мы обозначим это исчисление через $\mathcal{G}$.

Секвенцией называется слово вида

$$A_1, \ldots, A_m \to B_1, \ldots, B_n,$$

где $A_1, \ldots, A_m, B_1, \ldots, B_n$ — формулы, $m, n \in \mathbb{N}$. Список формул $A_1, \ldots, A_m$ (который пуст при $m = 0$) называется *антецедентом* этой секвенции. Список формул $B_1, \ldots, B_n$ (который пуст при $n = 0$) называется *сукцедентом*

этой секвенции. Каждая формула A_i $(i = 1, \ldots, m)$ называется *членом антецедента*. Каждая формула B_j $(j = 1, \ldots, n)$ называется *членом сукцедента*. Любой член антецедента и любой член сукцедента секвенции называется *членом этой секвенции*.

Интерпретация языка PL называется *контрпримером для секвенции* S, если в этой интерпретации истинны все члены антецедента S и ложны все члены сукцедента S.

Для секвенции

$$S \leftrightharpoons A_1, \ldots, A_m \to B_1, \ldots, B_n$$

определим формулу $\Phi(S)$ так, что для любой интерпретации M верно: M является контрпримером для S, если и только если M является контрпримером для $\Phi(S)$. *Представлением секвенции S в виде формулы* (или *формульным образом секвенции S*) назовём формулу

$$\Phi(S) \leftrightharpoons A_1 \wedge \ldots \wedge A_m \supset B_1 \vee \ldots \vee B_n,$$

причём мы считаем, что конъюнкция (соответственно, дизъюнкция) нуля формул есть $\mathbf{T}$ (соответственно, $\mathbf{F}$).

Секвенцию S можно содержательно понимать как формулу $\Phi(S)$. Подробнее, при $m > 0$ и $n > 0$ S можно понимать таким образом: в любой интерпретации истинность всех формул $A_1, \ldots, A_m$ влечёт истинность хотя бы одной из формул $B_1, \ldots, B_n$. При $m = 0$ и $n > 0$ — так: в любой интерпретации истинна хотя бы одна из формул $B_1, \ldots, B_n$. А при $m > 0$ и $n = 0$ — так: в любой интерпретации ложна хотя бы одна из формул $A_1, \ldots, A_m$.

Назовём секвенцию S *общезначимой*, если формула $\Phi(S)$ общезначима. Очевидно, секвенция общезначима тогда и только тогда, когда не существует контрпримера для этой секвенции.

Для формулировки исчисления $\mathcal{G}$ примем следующие обозначения: пусть Γ, Δ обозначают любые конечные списки формул, A, B — любые формулы.

Правила вывода исчисления $\mathcal{G}$ таковы:

$$\frac{\Gamma \to \Delta, A}{\Gamma, \neg A \to \Delta}\ (\neg\to), \qquad \frac{\Gamma, A \to \Delta}{\Gamma \to \Delta, \neg A}\ (\to\neg),$$

$$\frac{\Gamma, A, B \to \Delta}{\Gamma, A \wedge B \to \Delta}\ (\wedge\to), \qquad \frac{\Gamma \to \Delta, A; \quad \Gamma \to \Delta, B}{\Gamma \to \Delta, A \wedge B}\ (\to\wedge),$$

$$\frac{\Gamma, A \to \Delta; \quad \Gamma, B \to \Delta}{\Gamma, A \vee B \to \Delta}\ (\vee\to), \qquad \frac{\Gamma \to \Delta, A, B}{\Gamma \to \Delta, A \vee B}\ (\to\vee),$$

$$\frac{\Gamma \to \Delta, A; \quad \Gamma, B \to \Delta}{\Gamma, A \supset B \to \Delta}\ (\supset\to), \qquad \frac{\Gamma, A \to \Delta, B}{\Gamma \to \Delta, A \supset B}\ (\to\supset).$$

Справа от каждого правила вывода указано его обозначение. В левом столбце приведены правила введения логических связок в антецедент секвенции: правило введения отрицания в антецедент, правило введения конъюнкции в антецедент и т. д. В правом столбце приведены правила введения логических связок в сукцедент секвенции: правило введения отрицания в сукцедент, правило введения конъюнкции в сукцедент и т. д. Будем

считать, что для применения правил вывода порядок, в котором записаны члены антецедента и члены сукцедента, не имеет значения.

Поиск контрпримера для формулы в разделе 1.4.1 осуществлялся как раз по вышеприведённым правилам вывода. Подробнее, для каждого правила вывода и любой интерпретации M верно следующее: M является контрпримером для секвенции-заключения этого правила вывода тогда и только тогда, когда M является контрпримером хотя бы для одной секвенции-посылки этого правила. Действительно, рассмотрим, например, правило

$$\frac{\Gamma \to \Delta, A; \quad \Gamma, B \to \Delta}{\Gamma, A \supset B \to \Delta}$$

и произвольную интерпретацию M. По определению контрпримера для секвенции, M является контрпримером для $\Gamma, A \supset B \to \Delta$ тогда и только тогда, когда в интерпретации M все члены антецедента $\Gamma, A \supset B$ истинны, и все члены сукцедента Δ ложны. В интерпретации M имеем: $A \supset B$ истинна, если и только если A ложна или B истинна. Поэтому M является контрпримером для $\Gamma, A \supset B \to \Delta$ тогда и только тогда, когда M является контрпримером для $\Gamma \to \Delta, A$ или контрпримером для $\Gamma, B \to \Delta$.

Аксиомами исчисления $\mathcal{G}$ являются секвенции следующих видов:

$$\Gamma, A \to \Delta, A;$$
$$\Gamma, \mathbf{F} \to \Delta;$$
$$\Gamma \to \Delta, \mathbf{T}.$$

Будем считать, что при решении вопроса, является ли любая заданная секвенция аксиомой, порядок, в котором записаны члены антецедента и члены сукцедента, не имеет значения. Очевидно, ни для какой аксиомы исчисления $\mathcal{G}$ нет контрпримера.

Замечание 1.4.3. Соглашение о несущественности порядка членов антецедента и членов сукцедента для применения правил вывода и распознавания аксиом позволило нам компактно записать правила вывода и аксиомы, выделив в этой записи существенные черты. Если бы мы не ввели это соглашение, то мы могли бы, например, записать правило $(\supset\!\to)$ в таком, немного более громоздком виде:

$$\frac{\Gamma_1, \Gamma_2 \to \Delta_1, A, \Delta_2; \quad \Gamma_1, B, \Gamma_2 \to \Delta_1, \Delta_2}{\Gamma_1, A \supset B, \Gamma_2 \to \Delta_1, \Delta_2},$$

а одну из схем аксиом — так: $\Gamma_1, A, \Gamma_2 \to \Delta_1, A, \Delta_2$, где Γ_i, Δ_i $(i = 1, 2)$ обозначают любые конечные списки формул, A, B — любые формулы. Другим из возможных подходов без введения этого соглашения могло бы быть добавление в исчисление правил, которые позволяют переставлять члены антецедента и члены сукцедента. ◁

Язык исчисления $\mathcal{G}$ составляют всевозможные секвенции; поэтому все определения, связанные с общим понятием вывода слова в исчислении (см. раздел 1.2), применяются к секвенциям и исчислению $\mathcal{G}$. На этом формулировка исчисления $\mathcal{G}$ завершена.

Под *выводом формулы* A (*в исчислении* $\mathcal{G}$) будем понимать вывод секвенции $\to A$. Формулу A назовём *выводимой* (*в исчислении* $\mathcal{G}$), если секвенция $\to A$ выводима.

Вывод заданной секвенции удобно искать, переходя от заключений правил к посылкам, аналогично тому, как производился поиск контрпримера для формулы, только теперь можно не учитывать семантику и действовать формально, в соответствии с правилами вывода. Напомним, что такой способ поиска вывода является поиском вывода снизу вверх (см. раздел 1.3.5). Нахождение посылок некоторого применения правила вывода по известному заключению S (говоря менее точно, «применение» этого правила от заключения к посылкам) называется *контрприменением* этого правила к S, а найденные посылки — *посылками* этого контрприменения.

Пример 1.4.4. Попробуем найти вывод формулы $\neg(p \wedge q) \supset (\neg p \vee \neg q)$ в исчислении $\mathcal{G}$. (Эта формула уже рассматривалась в примере 1.4.2.) Для этого будем искать вывод секвенции $\to \neg(p \wedge q) \supset (\neg p \vee \neg q)$.

1) $\to \neg(p \wedge q) \supset (\neg p \vee \neg q)$ получается из секвенции 2 по правилу $(\to\supset)$;

2) $\neg(p \wedge q) \to (\neg p \vee \neg q)$ получается из секвенции 3 по правилу $(\neg \to)$;

3) $\to (\neg p \vee \neg q), (p \wedge q)$ получается из секвенций 4 и 5 по правилу $(\to \wedge)$;

4) $\to (\neg p \vee \neg q), p$ получается из секвенции 6 по правилу $(\to \vee)$;

5) $\to (\neg p \vee \neg q), q$ получается из секвенции 8 по правилу $(\to \vee)$;

6) $\to \neg p, \neg q, p$ получается из секвенции 7 по правилу $(\to \neg)$;

7) $p \to \neg q, p$ — аксиома;

8) $\to \neg p, \neg q, q$ получается из секвенции 9 по правилу $(\to \neg)$;

9) $q \to \neg p, q$ — аксиома.

Итак, секвенция 1 (а также исходная формула) выводима, и последовательность секвенций 9, 8, ..., 2, 1 является выводом секвенции 1. ◁

Поиск вывода какой угодно секвенции S естественно представляется в виде дерева, корнем которого является S, сыновними узлами корня являются секвенции-посылки некоторого контрприменения правила вывода к S и т. д. Дадим определение такого дерева.

Дерево с конечным числом узлов, каждый из которых является (или помечен) секвенцией, называется *деревом поиска вывода* секвенции S, если (a) корнем этого дерева является секвенция S, и (b) для каждого внутреннего узла S' этого дерева сыновними узлами S' являются секвенции-посылки некоторого контрприменения правила вывода к S'. Условие (b) можно перефразировать и так: для каждого внутреннего узла S' этого дерева сыновними узлами S' являются секвенции, из которых по некоторому правилу вывода получается S'. Дерево поиска вывода традиционно изображается растущим снизу вверх.

Если все листья дерева поиска вывода секвенции S оказались аксиомами (это означает, что для секвенции S получен вывод в виде дерева), то это дерево называется *деревом вывода* секвенции S.

Пример 1.4.5. Дерево вывода секвенции $\to \neg(p \wedge q) \supset (\neg p \vee \neg q)$ приведено в примере 1.4.2. Если из этого дерева удалить хотя бы один лист, то получится дерево поиска вывода этой секвенции, не являющееся деревом вывода. ◁

Ясно, что по дереву вывода произвольной заданной секвенции можно записать вывод этой секвенции в виде последовательности и наоборот (как в примерах 1.4.2 и 1.4.4).

Замечание 1.4.6. Определения дерева поиска вывода и дерева вывода не специфичны для рассматриваемого секвенциального исчисления. Аналогичные определения можно дать и для произвольного исчисления. ◁

Упражнение 1.4.7. Постройте выводы (в секвенциальном исчислении высказываний) всех логических законов, перечисленных в разделе 1.1.3, и всех аксиом исчисления высказываний гильбертовского типа, указанных в разделе 1.3.1. ◁

1.4.3. Корректность и полнота исчисления

Лемма 1.4.8. *В исчислении $\mathcal{G}$ каждая аксиома общезначима; для каждого правила вывода заключение этого правила общезначимо, если и только если все посылки этого правила общезначимы.*

Доказательство. Это утверждение легко проверить, непосредственно пользуясь определением общезначимой секвенции.

Вместо такой проверки можно вспомнить следующее. Каждая аксиома не имеет контрпримера, следовательно, каждая аксиома общезначима. Правила вывода сформулированы так, что для каждого правила вывода заключение этого правила имеет контрпример тогда и только тогда, когда хотя бы одна посылка этого правила имеет контрпример. Поэтому общезначимость заключения этого правила равносильна общезначимости всех посылок этого правила. ◁

Теорема 1.4.9 (корректность исчисления $\mathcal{G}$)**.** *Любая выводимая в исчислении $\mathcal{G}$ секвенция общезначима.*

Доказательство. Действительно, если секвенция выводима, то имеется дерево вывода этой секвенции. Поэтому все возможности по поиску контрпримера для этой секвенции исчерпаны, значит, эта секвенция общезначима.

Приведём и другое, более подробное доказательство данной теоремы — индукцией по длине вывода секвенции.

База индукции. Вывод состоит из одной секвенции. Тогда эта секвенция является аксиомой, которая общезначима по лемме 1.4.8.

Индукционный переход. Предположим, что любая секвенция, являющаяся последней секвенцией в выводе длины, меньшей n ($n > 1$), общезначима. Рассмотрим вывод длины n. Последняя секвенция S этого вывода либо является аксиомой (и тогда S общезначима), либо получена из одной или двух предшествующих секвенций по некоторому правилу вывода $\mathcal{R}$. По индукционному предположению посылки правила $\mathcal{R}$, из которых получена S, общезначимы. По лемме 1.4.8 общезначимость этих посылок влечёт общезначимость заключения S правила $\mathcal{R}$. Таким образом, секвенция S общезначима. ◁

Упражнение 1.4.10. Докажите, что любая выводимая в исчислении $\mathcal{G}$ формула общезначима. ◁

Следствие 1.4.11. *Исчисление $\mathcal{G}$ непротиворечиво, т. е. не существует формулы A такой, что A выводима и $\neg A$ выводима.*

Доказательство. Если бы нашлась такая формула A, что A выводима и $\neg A$ выводима, то в силу корректности исчисления имели бы $\models A$ и $\models \neg A$, что невозможно. ◁

Лемма 1.4.12. *Пусть S — секвенция, ни в какой член которой не входит ни одна логическая связка. Тогда если S общезначима, то S является аксиомой.*

Доказательство. Предположим, что S не является аксиомой. Тогда в интерпретации, сопоставляющей каждой переменной из антецедента этой секвенции истинностное значение 1 и каждой переменной из сукцедента — 0, $\Phi(S)$ будет ложна, что противоречит общезначимости S. ◁

Теорема 1.4.13 (полнота исчисления $\mathcal{G}$)**.** *Любая общезначимая секвенция выводима в исчислении $\mathcal{G}$.*

Доказательство. Пусть дана произвольная общезначимая секвенция S. Осуществляя контрприменения правил вывода (в каком угодно порядке), построим дерево поиска вывода секвенции S, каждый лист которого не содержит ни одной логической связки. Так как S общезначима, то в силу леммы 1.4.8 каждый лист этого дерева является общезначимой секвенцией (более подробное доказательство этого утверждения можно провести индукцией по числу узлов дерева поиска вывода). Тогда по лемме 1.4.12 каждый лист является аксиомой. Значит, это дерево поиска вывода является деревом вывода. Таким образом, S выводима. ◁

Упражнение 1.4.14. Докажите, что любая общезначимая формула выводима в исчислении $\mathcal{G}$. (См. также упражнение 1.4.10.) ◁

Наконец, приведём лемму, которая будет полезна для установления невыводимости некоторых секвенций.

Лемма 1.4.15. *Пусть дано дерево поиска вывода секвенции S. Тогда S невыводима, если найдётся секвенция-лист S_l этого дерева такая, что S_l не является аксиомой, и ни в какой член секвенции S_l не входит ни одна логическая связка.*

Доказательство. По лемме 1.4.12 секвенция S_l необщезначима. Тогда в силу леммы 1.4.8 секвенция S необщезначима. Поэтому по теореме о корректности исчисления $\mathcal{G}$ секвенция S невыводима. ◁

Пример 1.4.16. Попробуем найти вывод формулы $\neg(p \wedge q) \supset (\neg p \vee r)$ из примера 1.4.1.

1) $\to \neg(p \wedge q) \supset (\neg p \vee r)$ получается из секвенции 2 по правилу $(\to\supset)$;

2) $\neg(p \wedge q) \to (\neg p \vee r)$ получается из секвенции 3 по правилу $(\neg\to)$;

3) $\to (\neg p \vee r), (p \wedge q)$ получается из секвенции 4 по правилу $(\to\vee)$;

4) $\to \neg p, r, (p \wedge q)$ получается из секвенции 5 по правилу $(\to \neg)$;

5) $p \to r, (p \wedge q)$ получается из секвенций 6 и 7 по правилу $(\to \wedge)$;

6) $p \to r, p$ — аксиома;

7) $p \to r, q$ невыводима.

Секвенция 7 не является аксиомой и не содержит ни одной логической связки, поэтому по лемме 1.4.15 секвенция 1 (а вместе с ней и исходная формула) невыводима. Дерево поиска вывода секвенции 1 на последнем шаге поиска имеет следующий вид:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p \to r, p \qquad p \to r, q}{p \to r, (p \wedge q)}}{\to \neg p, r, (p \wedge q)}}{\to (\neg p \vee r), (p \wedge q)}}{\neg(p \wedge q) \to (\neg p \vee r)}}{\to \neg(p \wedge q) \supset (\neg p \vee r)}.$$

Отметим также, что мы сократили число секвенций в дереве поиска вывода исходной секвенции (по сравнению с другим деревом поиска вывода этой секвенции в примере 1.4.1), в первую очередь осуществляя контрприменения однопосылочных правил. ◁

Упражнение 1.4.17. Опишите алгоритм, выясняющий по любой секвенции, является ли она выводимой в исчислении $\mathcal{G}$ или нет. Докажите, что этот алгоритм для любой поданной на вход секвенции завершается и выдаёт правильный ответ. (В силу корректности и полноты исчисления $\mathcal{G}$ один из таких алгоритмов по входной секвенции S может строить таблицу истинности формулы $\Phi(S)$ и выдавать ответ «выводима», если $\Phi(S)$ общезначима, иначе выдавать ответ «невыводима». Опишите другой алгоритм, основанный непосредственно на выводе в исчислении $\mathcal{G}$ и не использующий семантических понятий, хотя обоснование правильности такого алгоритма может опираться на семантические понятия.) ◁

Заметим, что исчисление высказываний генценовского типа $\mathcal{G}$ имеет следующую особенность: каждое правило исчисления $\mathcal{G}$ таково, что в посылки входят лишь подформулы формул, входящих в заключение. Эта особенность делает исчисление $\mathcal{G}$ более удобным для поиска вывода снизу вверх по сравнению с исчислением высказываний гильбертовского типа $\mathcal{H}$. В исчислении $\mathcal{G}$ поиск вывода заданной формулы снизу вверх производится почти однозначно: есть только возможность выбирать формулу в секвенции, к которой осуществляется контрприменение правила.

1.4.4. Допустимые правила

Для вывода секвенций, содержащих связку $\equiv$, было бы удобно использовать вспомогательные правила введения эквивалентности в антецедент и сукцедент. Вспомним, что $(A \equiv B)$ служит сокращённой записью формулы $((A \supset B) \wedge (B \supset A))$, и получим правило введения эквивалентности в сукцедент. Для этого найдём секвенции (по возможности простого вида), из которых секвенция $\Gamma \to \Delta, (A \supset B) \wedge (B \supset A)$ получается по правилам вывода исчисления $\mathcal{G}$:

1) $\Gamma \to \Delta, (A \supset B) \wedge (B \supset A)$ получается из секвенций 2 и 3 по правилу $(\to \wedge)$;

2) $\Gamma \to \Delta, A \supset B$ получается из секвенции 4 по правилу $(\to\supset)$;

3) $\Gamma \to \Delta, B \supset A$ получается из секвенции 5 по правилу $(\to\supset)$;

4) $\Gamma, A \to \Delta, B$;

5) $\Gamma, B \to \Delta, A$.

Таким образом, если выводимы секвенции $\Gamma, A \to \Delta, B$ и $\Gamma, B \to \Delta, A$, то выводима секвенция $\Gamma \to \Delta, A \equiv B$. Это утверждение можно записать в виде одного из так называемых допустимых (в исчислении $\mathcal{G}$) правил

$$\frac{\Gamma, A \to \Delta, B; \quad \Gamma, B \to \Delta, A}{\Gamma \to \Delta, A \equiv B} \ (\to\equiv),$$

обозначенного нами через $(\to\equiv)$.

В контексте секвенциального исчисления $\mathcal{G}$ *правилом* мы называем не менее чем двухместное отношение на множестве всех секвенций (т. е. на языке этого исчисления).

Под *допустимым* (*в исчислении* $\mathcal{G}$) *правилом* мы понимаем такое правило $\mathcal{R}$ (являющееся отношением), что если $\langle S_1, \ldots, S_n, S\rangle \in \mathcal{R}$, то выводимость секвенций $S_1, \ldots, S_n$ влечёт выводимость секвенции S.

Обычно правила записываются в виде, указанном в разделе 1.2.1. Допустимое правило $(\to\equiv)$ записано именно так. Определения посылки и заключения правила даются таким же образом, как и в разделе 1.2.1.

Вышеприведённые секвенции $5, 4, 3, 2, 1$ являются выводом заключения правила $(\to\equiv)$ из его посылок.

Аналогично тому, как это было сделано для правила $(\to\equiv)$, можно доказать допустимость правила введения эквивалентности в антецедент:

$$\frac{\Gamma, A, B \to \Delta; \quad \Gamma \to \Delta, A, B}{\Gamma, A \equiv B \to \Delta} \ (\equiv\to).$$

Отметим, что каждое из допустимых правил $(\to\equiv)$ и $(\equiv\to)$, хоть и не является правилом вывода исчисления $\mathcal{G}$, но обладает свойством, указанным в лемме 1.4.8.

Упражнение 1.4.18. Сформулируйте допустимые правила, вводящие штрих Шеффера и стрелку Пирса. ◁

1.5. Метод резолюций для логики высказываний

Данный раздел посвящён одному методу, который позволяет устанавливать невыполнимость любой КНФ, если она действительно невыполнима. Рассматриваемый метод называется *методом резолюций* (*для логики высказываний*). Мы увидим, что задача установления общезначимости любой формулы сводится к задаче установления невыполнимости некоторой КНФ, построенной по исходной формуле. Сначала покажем идею метода резолюций, который применяется к КНФ.

Пусть дана КНФ $A \leftrightharpoons (\neg p \vee q) \wedge p \wedge \neg q$. Предположим, что A выполнима, т. е. существует такая интерпретация M, что $M \vDash A$. Метод резолюций основан на простом наблюдении: $(\neg p \vee q), p \vDash q$. Отсюда, поскольку дизъюнкты $(\neg p \vee q)$ и p являются членами конъюнкции A, немедленно следует $A \vDash q$ и потому $M \vDash q$. С другой стороны, поскольку дизъюнкт $\neg q$ является членом конъюнкции A, то $A \vDash \neg q$ и потому $M \vDash \neg q$. Получили противоречие: $M \vDash q$ и $M \vDash \neg q$. Поэтому исходная КНФ A невыполнима.

Метод резолюций базируется на обобщении указанного наблюдения. Он основан на следующем правиле образования нового дизъюнкта из двух дизъюнктов: из одного дизъюнкта вычёркиваем переменную, из другого — её отрицание, и из всех остальных литер двух исходных дизъюнктов составляем новый дизъюнкт. Полученный дизъюнкт, как мы докажем, является логическим следствием этих двух дизъюнктов. Теперь перейдём к описанию метода резолюций в общем случае.

Так как для общезначимости произвольной формулы A необходимо и достаточно невыполнимости формулы $\neg A$, то проверку общезначимости формулы A можно заменить на проверку невыполнимости формулы $\neg A$.

По теореме 1.1.39 любую формулу можно выразить формулой, находящейся в КНФ вида $A_1 \wedge \ldots \wedge A_n$, где каждая формула A_i $(i = 1, \ldots, n)$ является дизъюнктом. При этом можно считать, что в каждый дизъюнкт A_i никакая литера не входит в качестве члена дизъюнкции более одного раза, и все дизъюнкты A_i попарно различны. Мы будем нередко рассматривать такую КНФ как множество дизъюнктов $\{A_1, \ldots, A_n\}$, а каждый дизъюнкт — как множество литер, являющихся членами этого дизъюнкта. Тогда можно говорить о невыполнимости множества дизъюнктов, понимая под этим невыполнимость соответствующей КНФ. Таким образом, проверка общезначимости произвольной формулы сводится к проверке невыполнимости некоторого множества дизъюнктов.

Далее в текущем разделе 1.5 под множеством дизъюнктов мы будем понимать непустое конечное множество дизъюнктов.

Метод резолюций служит для доказательства невыполнимости множества дизъюнктов. Этот метод основан на правиле образования дизъюнкта из двух данных дизъюнктов, как мы наметили выше. Сейчас мы сформулируем это правило.

Определение 1.5.1. Две литеры называются *контрарными*, если одна из них является отрицанием другой. ◁

Определение 1.5.2. Пусть элементом дизъюнкта C_1 (рассматриваемого как множество литер) является литера L_1, а элементом дизъюнкта C_2 — литера L_2, причём L_1 и L_2 — контрарные литеры. Тогда *резольвентой дизъюнктов* C_1 и C_2 называется множество литер $(C_1 \setminus \{L_1\}) \cup (C_2 \setminus \{L_2\})$. ◁

Если два дизъюнкта являются контрарными литерами, то их резольвента есть пустое множество литер. Удобно считать дизъюнктом и пустое множество литер[9], которое называется *пустым дизъюнктом* и обозначается $\square$. Таким образом, резольвента всегда является дизъюнктом (рассматриваемым и как множество литер). Будем считать пустой дизъюнкт невыполнимым, что естественно, поскольку лишь резольвента p и $\neg p$ (где p —

[9] Определение 1.1.35 на с. 27 требовало вхождения в дизъюнкт хотя бы одной литеры.

переменная) является пустым дизъюнктом, а конъюнкция p и $\neg p$ невыполнима.

Пример 1.5.3. Резольвентой дизъюнктов $\neg p \vee q$ и p является q. Резольвентой дизъюнктов $\neg q$ и q является $\square$. Резольвентой дизъюнктов $p \vee \neg q \vee \neg r$ и $q \vee r$ является как $p \vee \neg r \vee r$, так и $p \vee \neg q \vee q$. Резольвенты дизъюнктов $\neg p \vee q$ и $\neg p \vee r$ не существует. $\triangleleft$

Теорема 1.5.4. *Резольвента дизъюнктов есть их логическое следствие.*

Доказательство. Пусть дизъюнкт C_1 есть множество литер $C_1' \cup \{p\}$, дизъюнкт C_2 — множество литер $C_2' \cup \{\neg p\}$, а $C_1' \cup C_2'$ — резольвента этих дизъюнктов. Достаточно доказать, что для произвольной интерпретации M, если $M \vDash C_1$ и $M \vDash C_2$, то $M \vDash C_1' \cup C_2'$.

Ясно, что $M \vDash p$ или $M \vDash \neg p$. Разберём два случая.

1. Пусть $M \vDash p$. Тогда $M \nvDash \neg p$, поэтому из $M \vDash C_2$ получаем $M \vDash C_2'$, откуда $M \vDash C_1' \cup C_2'$.

2. Пусть $M \vDash \neg p$. Тогда $M \nvDash p$, поэтому из $M \vDash C_1$ получаем $M \vDash C_1'$, откуда $M \vDash C_1' \cup C_2'$.

Объединяя два разобранных случая, получаем $M \vDash C_1' \cup C_2'$. $\triangleleft$

Определение 1.5.5. Пусть C — дизъюнкт, S — множество дизъюнктов. *Резолютивным выводом* дизъюнкта C из множества S называется конечная последовательность дизъюнктов, каждый из которых принадлежит S или является резольвентой предшествующих дизъюнктов этой последовательности, и последним дизъюнктом этой последовательности является C. $\triangleleft$

Отметим, что резолютивный вывод дизъюнкта C из множества S дизъюнктов является выводом в исчислении, аксиомами которого являются все элементы множества S, а единственным правилом вывода — правило образования резольвенты двух дизъюнктов, называемое также *правилом резолюции*. Указанное исчисление называют *резолютивным исчислением*.

Следующая теорема показывает, что для доказательства невыполнимости множества дизъюнктов достаточно вывести из него пустой дизъюнкт.

Теорема 1.5.6 (корректность метода резолюций для логики высказываний)**.** *Пусть S — множество дизъюнктов. Если существует резолютивный вывод пустого дизъюнкта из S, то S невыполнимо.*

Доказательство. Пусть имеется следующий резолютивный вывод из S: $C_1, \ldots, C_k$, где C_k есть пустой дизъюнкт. Предположим, что S выполнимо, т. е. в некоторой интерпретации M все элементы множества S истинны. Тогда по теореме 1.5.4 для каждого $i = 1, \ldots, k$ дизъюнкт C_i истинен в интерпретации M, что противоречит невыполнимости пустого дизъюнкта C_k. Поэтому S невыполнимо. $\triangleleft$

Пример 1.5.7. С помощью резолютивного вывода докажем невыполнимость такого множества дизъюнктов:

1) $p \vee q$,

2) $\neg p \vee q$,

3) $p \vee \neg q$,

4) $\neg p \vee \neg q$.

Породим следующие резольвенты:

5) q — резольвента дизъюнктов 1 и 2,

6) $\neg q$ — резольвента дизъюнктов 3 и 4,

7) $\square$ — резольвента дизъюнктов 5 и 6.

Резольвента 7 есть пустой дизъюнкт, поэтому по предыдущей теореме данное множество дизъюнктов невыполнимо. $\triangleleft$

Вывод пустого дизъюнкта из множества дизъюнктов S также называют *резолютивным опровержением* множества S. Это название естественно: как видно из доказательства теоремы о корректности метода резолюций, предположив, что множество S выполнимо, мы опровергаем сделанное предположение, выведя из S невыполнимый пустой дизъюнкт.

Справедливо утверждение о полноте метода резолюций для логики высказываний, т. е. о том, что из любого невыполнимого множества дизъюнктов с помошью некоторого резолютивного вывода можно получить пустой дизьюнкт. Мы будем иметь это утверждение как следствие теоремы о полноте метода резолюций для логики предикатов (см. главу 4 и теорему 4.4.17).

На этом мы бы хотели завершить данную главу, посвящённую логике (и исчислениям) высказываний. Но что же означает термин «логика», до сих пор использовавшийся интуитивно?

Об использовании термина «логика»

Поясним использование термина «логика» теперь, когда у нас есть примеры для такого пояснения. Когда говорят о какой-нибудь конкретной *логике* (но не о логике как области человеческих знаний), то обычно под *логикой* понимают некоторый формальный язык вместе с семантикой этого языка или некоторым исчислением в этом языке. В настоящей книге мы придерживаемся следующего понимания: *логика* — это язык и его семантика. Например, под логикой высказываний мы понимаем язык PL и способ сопоставления истинностных значений формулам путём задания интерпретаций этого языка. Для логики высказываний мы ввели несколько исчислений: исчисление гильбертовского типа, секвенциальное исчисление и резолютивное исчисление (причём последнее зависит от конкретной формулы).

Обычно смысл термина «логика», используемого в литературе по математической логике, угадывается по контексту, и для точного изложения результатов о конкретном формальном языке, его семантике и об исчислении этот термин не требуется. Однако, особенно при изучении более чем одной логики, полезно осознавать значение этого термина.

Глава 2.

Исчисления предикатов

В математических рассуждениях наряду с логическими связками встречаются кванторы $\forall$ (читается как «для любого», «для всякого», «для каждого») и $\exists$ («существует», «для некоторого»). Например, утверждение «существует x такое, что для любого y x не превосходит y», в котором переменные берут свои значения из множества $\mathbb{N}$ всех натуральных чисел, можно символически записать как формулу $\exists x \forall y R(x, y)$, понимая под $R(x, y)$ отношение «x не превосходит y». Формула $\exists x \forall y R(x, y)$ истинна, если переменные пробегают множество всех натуральных чисел, а $R(x, y)$ интерпретируется как отношение «x не превосходит y». Если же мы интерпретируем $R(x, y)$ как отношение «y не превосходит x», то эта формула окажется ложной.

С другой стороны, формула $\forall y R(x, y)$, рассматриваемая на множестве $\mathbb{N}$ и в которой под $R(x, y)$ понимается отношение «x не превосходит y», не получает истинностного значения до тех пор, пока не задано значение переменной x. Если в качестве x взять 0, то эта формула будет истинна, а если в качестве x взять 2, то ложна.

Рассмотрим теперь утверждение, включающее не только отношение, но и функцию: «существует x такое, что для любого y удвоенное x не превосходит y» на множестве $\mathbb{N}$. Это утверждение можно записать в виде формулы как $\exists x \forall y R(f(x), y)$, интерпретируя $R(z, y)$ как отношение «z не превосходит y», а $f(x)$ — как функцию удвоения натурального числа x. При заданной интерпретации эта формула истинна.

Некоторые формулы символически представляют способы рассуждений, принятые в математике. Рассмотрим одну из таких формул. Пусть $P(z)$ означает, что объект z обладает свойством P. Рассуждение «если для любого x верно $P(x)$, то существует y такой, что верно $P(y)$», считается правильным вне зависимости от того, какое непустое множество объектов и свойство P рассматриваются. Это рассуждение можно символически записать в виде формулы $\forall x P(x) \supset \exists y P(y)$. Она истинна при любом выборе непустого множества объектов и свойства P. Про такую формулу говорят, что она является логическим законом.

Логика первого порядка, также называемая *логикой предикатов первого порядка* или просто *логикой предикатов*, изучает способы математических рассуждений, включающих в себя кванторы по объектам. В данной главе мы определим понятия формулы с кванторами, логического закона,

а также изучим то, как точно описать все логические законы.

2.1. Язык первого порядка

В данном разделе мы определим так называемый язык первого порядка, на котором записываются формулы с кванторами.

2.1.1. Формулы языка первого порядка

Сначала введём понятие так называемой сигнатуры; затем формулы любого языка первого порядка определим однотипно по соответствующей этому языку сигнатуре.

Определение 2.1.1. *Сигнатурой* назовём упорядоченную тройку $\langle PS, FS, \# \rangle$, где PS — непустой конечный или счётный язык, FS — конечный (возможно, пустой) или счётный язык, $\#$ — отображение из $PS \cup FS$ в $\mathbb{N}$, $PS \cap FS = \varnothing$.

Каждое слово языка PS[1] будем называть *предикатным символом*. Предикатный символ будем обозначать словом из букв латинского алфавита, которое начинается с одной из букв P, p, Q, q, R, r, причём такое слово может иметь индекс, являющийся натуральным числом. (Например, обозначениями предикатных символов являются следующие слова: $P, Padd, p_1, Q_0, r, Rxyz_{2007}$.) С каждым предикатным символом P связано натуральное число $\#(P)$ — его *местность* (*арность* или *число аргументов*). 0-местный предикатный символ иначе называют *пропозициональной переменной* или *пропозициональным символом*.

Каждое слово языка FS[2] будем называть *функциональным символом*. Функциональный символ будем обозначать словом из букв латинского алфавита, которое начинается с одной из букв a, b, c, f, g, h, причём такое слово может иметь индекс. (Например, обозначениями функциональных символов являются следующие слова: a, f, fy, fy_1, g_{2007}.) С каждым функциональным символом f связано натуральное число $\#(f)$ — его *местность* (*арность* или *число аргументов*). 0-местный функциональный символ иначе называют *предметной* (или *индивидной*) *константой*. ◁

Замечание 2.1.2. Иногда рассматривают сигнатуры с несчётным множеством предикатных или функциональных символов, но мы не будем этого делать. ◁

Через IV обозначим (счётный) язык $Vars$, который задан в определении 1.1.3. Каждое слово языка IV[3] будем называть *предметной* (или *индивидной*) *переменной*, а сам язык IV — множеством всех предметных переменных. Предметную переменную будем обозначать словом из букв латинского алфавита, которое начинается с одной из букв u, v, w, x, y, z, причём такое слово может иметь индекс. (Например, обозначениями предметных переменных являются следующие слова: $var, x, z_0, x_1, y_{2007}, z_{2008}$.)

[1] PS — сокращение «Predicate Symbols».
[2] FS — сокращение «Function Symbols».
[3] IV — сокращение «Individual Variables».

Замечание 2.1.3. Словосочетания «предикатный символ» и «функциональный символ» мы употребляем по традиции. В них слово «символ» не имеет самостоятельного значения. Как предикатный, так и функциональный символ может быть словом, а не только символом. ◁

Определим *язык первого порядка в сигнатуре* $\langle PS, FS, \# \rangle$, который мы будем обозначать через $FOL(PS, FS, \#)$.[4] Каждое слово этого языка является либо *термом этого языка*, либо *формулой этого языка*.

Пусть $\Omega \leftrightharpoons FOL(PS, FS, \#)$.

Определение 2.1.4. *Терм языка* Ω задаётся следующим индуктивным определением:

1) любая предметная константа из множества FS и любая предметная переменная является термом этого языка;

2) если f — n-местный функциональный символ из множества FS, $n > 0$, $t_1, \ldots, t_n$ — термы языка Ω, то $f(t_1, \ldots, t_n)$ является термом этого языка.

Каждый из термов $t_1, \ldots, t_n$ в терме $f(t_1, \ldots, t_n)$ называют *аргументом* функционального символа f. ◁

Определение 2.1.5. *Атомарная формула языка* Ω задаётся следующим определением:

1) любая пропозициональная переменная из множества PS и каждая истинностная константа $\mathbf{F}$ и $\mathbf{T}$ является атомарной формулой этого языка;

2) если P — n-местный предикатный символ из множества PS, $n > 0$, $t_1, \ldots, t_n$ — термы языка Ω, то $P(t_1, \ldots, t_n)$ является атомарной формулой этого языка.

Каждый из термов $t_1, \ldots, t_n$ в атомарной формуле $P(t_1, \ldots, t_n)$ называют *аргументом* предикатного символа P. ◁

Определение 2.1.6. *Формула языка* Ω задаётся следующим индуктивным определением:

1) любая атомарная формула языка Ω является формулой этого языка;

2) если A и B — формулы языка Ω, x — предметная переменная, то $\neg A$, $(A \wedge B)$, $(A \vee B)$, $(A \supset B)$, $\forall x A$ и $\exists x A$ являются формулами этого языка. ◁

Когда из контекста ясно, какой язык первого порядка рассматривается, термы этого языка часто называют просто *термами*, а формулы этого языка часто называют *формулами первого порядка*, *предикатными формулами* или просто *формулами*.

Логические связки и символы $\forall$ (*квантор всеобщности*) и $\exists$ (*квантор существования*) называют *логическими символами*.

Назовём *главным* логический символ, подчёркнутый в формулах видов $\underline{\forall} x A$, $\underline{\exists} x A$, $\underline{\neg} A$, $(A \underline{\odot} B)$, где $\odot$ — бинарная логическая связка.

[4] FOL — сокращение «First-Order Language».

Пример 2.1.7. Пусть сигнатура $\langle PS, FS, \# \rangle$ языка первого порядка $\Omega \leftrightharpoons FOL(PS, FS, \#)$ такова, что

1) $PS = \{P\}$, $\#(P) = 2$, т. е. P — двухместный предикатный символ,

2) $FS = \{a,\ f,\ g\}$, $\#(a) = 0$, $\#(f) = 1$, $\#(g) = 2$, т. е. a — нульместный функциональный символ (предметная константа), f и g — одноместный и двухместный функциональные символы соответственно.

Термами языка Ω являются, например,

1) предметные переменные x, y, z,

2) предметная константа a,

3) функциональные символы с термами-аргументами: $f(a)$, $g(x, y)$, $g(f(y),\, a)$.

Формулами языка Ω являются, например,

1) атомарные формулы: $\mathbf{T}$, $P(a, a)$, $P(x, y)$, $P(f(a),\ g(f(y),\, a))$,

2) формулы, не являющиеся атомарными: $\neg P(x, y)$, $\exists z P(y,\, g(x, z))$, $\neg P(f(x),\, y) \wedge \exists z P(y,\, g(a, z))$. $\triangleleft$

Индуктивный характер определения формул (соответственно, термов) какого угодно языка первого порядка позволяет использовать индукцию по построению множества всех формул (соответственно, термов) этого языка как метод доказательства и задавать функции индукцией по построению множества всех формул (соответственно, термов) этого языка аналогично тому, как мы это делали для множества всех пропозициональных формул (см. раздел 1.1.2).

При записи предикатных формул будем применять соглашения об опускании некоторых скобок, аналогичные принятым в конце раздела 1.1.2 соглашениям о записи пропозициональных формул.

Ниже, говоря о языке первого порядка, под «переменной» мы будем понимать предметную переменную, а для термина «пропозициональная переменная» не будем использовать никакое сокращение названия.

В дальнейшем изложении, говоря о термах и формулах, будем считать, что мы заранее зафиксировали произвольный язык первого порядка и рассматриваем этот язык, его термы и формулы, если явно не указано другое. Когда мы будем записывать термы и формулы, то будем предполагать, что в сигнатуре рассматриваемого языка имеются используемые в них предикатные и функциональные символы. Наконец, условимся, что далее буквы s и t (возможно, с индексами) будут обозначать произвольные термы, а буквы A, B, C, D, E и G (возможно, с индексами) — произвольные формулы рассматриваемого языка первого порядка, если иное не оговорено явно.

2.1.2. Вхождения предметных переменных в формулы

Дадим несколько определений, которые в основном касаются вхождений предметных переменных в формулы.

Подтермом терма t называется слово, входящее в t и являющееся термом.

Подформулой формулы A называется слово, входящее в A и являющееся формулой.

Пусть x — какая угодно предметная переменная; тогда слово $\forall x$ (и слово $\exists x$) называется *кванторной приставкой*, а x — *переменной этой кванторной приставки*.

Пусть зафиксировано какое угодно вхождение кванторной приставки (соответственно, вхождение квантора) в формулу A. Тогда *областью действия этого вхождения кванторной приставки* (соответственно, *областью действия этого вхождения квантора*) называется подформула формулы A, начинающаяся с этого вхождения кванторной приставки (соответственно, с этого вхождения квантора). (Согласно определению предикатной формулы, такая подформула находится однозначно.)

Пример 2.1.8. В формуле

$$\neg\overline{\forall x}\underline{P(x,y)} \wedge \exists z Q(f(x),z,x)$$

областью действия надчёркнутого вхождения кванторной приставки является подчёркнутая подформула. (Отметим, что, так же как и вхождение логической связки $\neg$, вхождение кванторной приставки относится к минимальной по длине подформуле, стоящей сразу за этой кванторной приставкой.)

В формуле

$$\neg\overline{\forall x}\underline{(P(x,y) \wedge \exists z Q(f(x),z,x))}$$

областью действия надчёркнутого вхождения кванторной приставки является подчёркнутая подформула. $\triangleleft$

Вхождение предметной переменной x в формулу A называется *связанным*, если это вхождение является вхождением в формулу вида $\forall x B$ или $\exists x B$, которая есть подформула формулы A (иначе говоря, если это вхождение является вхождением в область действия вхождения кванторной приставки $\forall x$ или $\exists x$).

Вхождение предметной переменной x в формулу называется *свободным*, если оно не является связанным.

Предметная переменная x называется *связанной переменной формулы* A, если x входит (хотя бы один раз) связанно в A. Множество всех связанных переменных формулы A обозначим через $BV(A)$.[5]

Предметная переменная x называется *свободной переменной формулы* A (или *параметром формулы* A), если x входит (хотя бы один раз) свободно в A. Множество всех парамертов формулы A обозначим через $FV(A)$.[6]

Пример 2.1.9. Если переменная x не совпадает ни с y, ни с z, то в формуле

$$A \leftrightharpoons \neg\forall x P(x,\underline{y}) \wedge \exists z Q(f(\underline{x}),z,\underline{x})$$

неподчёркнутые вхождения переменных являются связанными, а подчёркнутые — свободными, $FV(A) = \{y,x\}$, $BV(A) = \{x,z\}$. $\triangleleft$

Упражнение 2.1.10. Дайте индуктивные аналоги определений свободного и связанного вхождения переменной в формулу, множеств всех свободных и связанных переменных формулы. $\triangleleft$

[5] BV — сокращение «Bound Variables».
[6] FV — сокращение «Free Variables».

Пусть зафиксировано связанное k-вхождение переменной x в формулу A. Говорят, что это k-вхождение *связано тем вхождением кванторной приставки* с переменной x (а также *связано тем вхождением квантора*), которое имеет область действия минимальной длины и в области действия которого находится это k-вхождение.

Пример 2.1.11. Если переменная x отлична от z, то в формуле

$$\neg\forall x(P(x,y)\wedge\exists\underline{x}\,Q(f(\underline{x}),z,\underline{x}))$$

подчёркнутые вхождения переменной x связаны вхождением кванторной приставки $\exists x$, а неподчёркнутые вхождения переменной x связаны вхождением кванторной приставки $\forall x$. ◁

Формула, в которой нет свободных вхождений предметных переменных, называется *замкнутой формулой* (или *предложением*).

Терм, в который не входит ни одна предметная переменная, называется *замкнутым* (или *основным*).

Пусть A — формула, $x_1,\ldots,x_n$ — все попарно различные параметры A, перечисленные (для определённости) в порядке их первых вхождений в A. Тогда (замкнутая и при $n=0$ совпадающая с A) формула $\forall x_1\ldots\forall x_n A$ называется *замыканием формулы* A (*универсальным замыканием формулы* A или *замыканием формулы* A *кванторами всеобщности*) и обозначается $\mathbb{\forall}A$.

Аналогично определяется *замыкание формулы* A *кванторами существования* (или *экзистенциальное замыкание формулы* A), которое обозначается $\mathbb{\exists}A$.

Формула, в которую не входит ни один квантор, называется *бескванторной формулой*.

2.2. Семантика языка первого порядка

В этом разделе мы определим, как словам (термам и формулам) языка первого порядка придают содержательные значения. Терм получает в качестве такого значения элемент из некоторого множества, а формула — одно из двух истинностных значений. Также мы покажем, как формулы могут служить для записи утверждений.

2.2.1. Интерпретация языка первого порядка

Термы и формулы языка первого порядка — это определённым образом построенные последовательности букв. Чтобы придать любому терму и любой формуле языка некоторый содержательный смысл, задают так называемую интерпретацию этого языка, которая указывает, какое множество $\mathcal{D}$ пробегают предметные переменные, и как трактовать функциональные и предикатные символы. Функциональные символы естественно трактовать как функции на множестве $\mathcal{D}$, а предикатные символы — как свойства упорядоченных n-ок элементов множества $\mathcal{D}$. Для любой данной n-ки свойство либо выполняется, либо не выполняется. В связи с такой трактовкой предикатных символов введём следующее определение.

Определение 2.2.1. Пусть даны произвольное множество X и какое угодно число $n \in \mathbb{N}_+$. Любую функцию из множества X^n в множество всех истинностных значений назовём n-местным *предикатом* на множестве X.

Более общо, пусть $n \in \mathbb{N}_+$ и $X_1, \ldots, X_n$ — произвольные множества. Тогда любую функцию из множества $X_1 \times \ldots \times X_n$ в множество всех истинностных значений назовём n-местным *предикатом* с областью определения $X_1 \times \ldots \times X_n$. ◁

Замечание 2.2.2. Любое n-местное отношение $\mathcal{R}$, являющееся подмножеством множества $X_1 \times \ldots \times X_n$, задаёт такой n-местный предикат $\widetilde{\mathcal{R}}$, что $\widetilde{\mathcal{R}}(x_1, \ldots, x_n)$ истинно, если $\langle x_1, \ldots, x_n \rangle \in \mathcal{R}$, и ложно в противном случае.

С другой стороны, любой n-местный предикат $\widetilde{\mathcal{R}}$ с областью определения $X_1 \times \ldots \times X_n$ задаёт n-местное отношение

$$\{\langle x_1, \ldots, x_n \rangle \in X_1 \times \ldots \times X_n \mid \widetilde{\mathcal{R}}(x_1, \ldots, x_n) \text{ истинно}\}.$$

Имея в виду это соответствие между предикатами и отношениями, термины «предикат» и «отношение» часто используют как взаимозаменяемые, при необходимости различая точный смысл того или другого слова по контексту. ◁

Определение 2.2.3. *Интерпретацией* (или *моделью*) *языка первого порядка* $FOL(PS, FS, \#)$ называется упорядоченная пара $\langle \mathcal{D}, \mu \rangle$, где

1) $\mathcal{D}$ — непустое множество, называемое *носителем этой интерпретации*;

2) μ — отображение, которое

   а) каждому n-местному предикатному символу из PS при $n > 0$ сопоставляет n-местный предикат на $\mathcal{D}$, а при $n = 0$ пропозициональной переменной сопоставляет истинностное значение;

   б) каждому n-местному функциональному символу из FS при $n > 0$ сопоставляет функцию из $\mathcal{D}^n$ в $\mathcal{D}$, а при $n = 0$ предметной константе сопоставляет элемент множества $\mathcal{D}$.

Интерпретацию данного языка также называют *алгебраической системой сигнатуры* $\langle PS, FS, \# \rangle$. ◁

Определение 2.2.4. Интерпретация языка первого порядка называется *конечной* (соответственно, *счётной*), если её носитель является конечным (соответственно, счётным) множеством. ◁

Пример 2.2.5. Рассмотрим язык первого порядка, в сигнатуре которого есть только двухместный предикатный символ R, и формулу $A \leftrightharpoons \forall y R(x, y)$, где переменные x и y различны. Зададим интерпретацию этого языка: в качестве носителя возьмём множество $\mathbb{N}$, предикатному символу R сопоставим предикат $\widetilde{R}(x, y)$, определённый как «x не превосходит y». Понятно, что нельзя однозначно сказать, истинна или нет формула A в этой интерпретации, поскольку неизвестно значение параметра x этой формулы.

Таким образом, чтобы найти истинностное значение формулы в заданной интерпретации, требуется задать также значения всем параметрам этой формулы. Мы зададим значения параметрам формул с помощью отображения, называемого оценкой. ◁

Зафиксируем произвольный язык первого порядка Ω и любую его интерпретацию $M \leftrightharpoons \langle \mathcal{D}, \mu \rangle$ и будем рассматривать их до конца текущего раздела 2.2.1.

Определение 2.2.6. *Оценкой* языка Ω в интерпретации M называется какое угодно отображение из множества IV всех предметных переменных в носитель $\mathcal{D}$ этой интерпретации. $\triangleleft$

Пусть ν — оценка языка Ω в интерпретации M.

Определение 2.2.7. Для произвольного терма t определим *значение терма t в интерпретации M при оценке ν*. Это значение будем обозначать $|t|_{M,\nu}$. Зададим $|t|_{M,\nu}$ с помощью следующего индуктивного определения:

1) если t есть предметная переменная, то $|t|_{M,\nu} = \nu(t)$;

2) если t есть предметная константа, то $|t|_{M,\nu} = \mu(t)$;

3) если t имеет вид $f(t_1, \ldots, t_n)$, то $|t|_{M,\nu} = \mu(f)(|t_1|_{M,\nu}, \ldots, |t_n|_{M,\nu})$. $\triangleleft$

Если терм t замкнут, то, говоря о его значении, мы иногда не будем упоминать оценку, а также наряду с $|t|_{M,\nu}$ будем писать $|t|_M$.

Определение 2.2.8. Пусть X и Y — множества, ξ — функция из X в Y, x и α — элементы множеств X и Y соответственно. Через ξ^x_α будем обозначать такую функцию из X в Y, что $\xi^x_\alpha(z) = \xi(z)$ для любого $z \in X \setminus \{x\}$ и $\xi^x_\alpha(x) = \alpha$. (Таким образом, функция ξ^x_α имеет значение α в точке x и совпадает с ξ во всех остальных точках.) $\triangleleft$

Определение 2.2.9. Для произвольной формулы A определим *истинностное значение формулы A в интерпретации M при оценке ν*. Это значение будем обозначать $|A|_{M,\nu}$. Зададим $|A|_{M,\nu}$ с помощью следующего индуктивного определения:

1) если A есть истинностная константа $\mathbf{F}$, то $|A|_{M,\nu} = 0$;

2) если A есть истинностная константа $\mathbf{T}$, то $|A|_{M,\nu} = 1$;

3) если A есть пропозициональная переменная, то $|A|_{M,\nu} = \mu(A)$;

4) если A — атомарная формула вида $P(t_1, \ldots, t_n)$, то $|A|_{M,\nu} = \mu(P)(|t_1|_{M,\nu}, \ldots, |t_n|_{M,\nu})$;

5) если A имеет вид $\neg B$, то $|A|_{M,\nu} = F_\neg(|B|_{M,\nu})$ ($F_\neg$ — функция истинности отрицания, см. раздел 1.1.3);

6) если A имеет вид $(B \odot C)$, то $|A|_{M,\nu} = F_\odot(|B|_{M,\nu},\ |C|_{M,\nu})$, где $\odot$ — одна из связок $\wedge$, $\vee$, $\supset$ ($F_\odot$ — функция истинности соответствующей связки, см. раздел 1.1.3);

7) если A имеет вид $\forall x B$, то $|A|_{M,\nu} = 1$ тогда и только тогда, когда для любого $\alpha \in \mathcal{D}$ выполняется $|B|_{M,\nu^x_\alpha} = 1$;

8) если A имеет вид $\exists x B$, то $|A|_{M,\nu} = 1$ тогда и только тогда, когда существует $\alpha \in \mathcal{D}$ такой, что $|B|_{M,\nu^x_\alpha} = 1$.

Если $|A|_{M,\nu} = 1$, то будем говорить, что формула A *истинна в интерпретации M при оценке ν* (или *истинна при (в) интерпретации M и оценке ν*), и иначе записывать этот факт как $M, \nu \vDash A$.

Если $|A|_{M,\nu} = 0$, то будем говорить, что формула A *ложна в интерпретации M при оценке ν* (или *ложна при (в) интерпретации M и оценке ν*), и иначе записывать этот факт как $M, \nu \nvDash A$. ◁

Из этого определения видно, что оценка служит только для задания значений параметрам формул, и оценка не влияет на истинностное значение замкнутой формулы.

Если формула A замкнута, то, говоря о её истинностном значении, мы иногда не будем упоминать оценку, а также наряду с $|A|_{M,\nu}$ будем писать $|A|_M$, наряду с $M, \nu \vDash A$ — $M \vDash A$, а наряду с $M, \nu \nvDash A$ — $M \nvDash A$.

Формулу A назовём *истинной в интерпретации M*, если $M, \nu \vDash A$ при любой оценке ν. Этот факт будем обозначать через $M \vDash A$.

2.2.2. Примеры языков первого порядка и их интерпретаций

Иногда атомарную формулу, состоящую из двухместного предикатного символа P с аргументами t_1 и t_2, наряду с $P(t_1, t_2)$ записывают как $(t_1\, P\, t_2)$ или $t_1\, P\, t_2$. Аналогично, терм $f(t_1, t_2)$ иногда записывают как $(t_1\, f\, t_2)$ или, когда не возникает разночтений, как $t_1\, f\, t_2$. В дальнейшем мы нередко будем пользоваться такой записью.

В данном разделе 2.2.2 мы будем считать различными предметные переменные, обозначенные по-разному.

Язык элементарной арифметики

Опишем один из языков первого порядка — *язык Ar элементарной арифметики*. В сигнатуре этого языка имеются только предметная константа 0, одноместный функциональный символ S, двухместные функциональные символы $+$ и $\cdot$ и двухместный предикатный символ $=$.

Например, формулой языка Ar является $\exists w = (x, +(\cdot(y, z), w))$, что можно записать и как $\exists w(x = (y \cdot z) + w)$.

Там, где не может быть разночтений, будем использовать сокращение $St \leftrightharpoons S(t)$. Также для экономии скобок в записи формул примем обычное соглашение о том, что приоритет $\cdot$ выше, чем приоритет $+$. Тогда вышеприведённая формула может быть записана в виде $\exists w(x = y \cdot z + w)$.

Зададим интерпретацию $\omega \leftrightharpoons \langle \mathcal{D}, \mu \rangle$ языка Ar: в качестве носителя $\mathcal{D}$ возьмём множество всех натуральных чисел $\{0, 1, 2, \ldots\}$, обозначаемое через $\mathbb{N}$; предметной константе 0 сопоставим натуральное число 0 (точнее, $\mu(0)$ есть натуральное число 0); функциональному символу S — одноместную функцию прибавления единицы к натуральному числу (точнее, $\mu(S)$ есть одноместная функция прибавления единицы к натуральному числу); функциональным символам $+$ и $\cdot$ — двухместные функции сложения и умножения натуральных чисел соответственно ($\mu(+)$ и $\mu(\cdot)$ являются указанными функциями); предикатному символу $=$ — предикат равенства, выражающий совпадение двух натуральных чисел ($\mu(=)$ есть указанный предикат равенства). Интерпретацию ω называют *стандартной интерпретацией языка Ar*.

Введём сокращённые обозначения для некоторых формул языка Ar:

$$x \leqslant y \leftrightharpoons \exists z(y = x + z),$$
$$y|x \leftrightharpoons \exists z(x = y \cdot z),$$
$$\text{Простое}(x) \leftrightharpoons \neg(x = S0) \wedge \forall y(y|x \supset y = S0 \vee y = x).$$

При любой оценке ν такой, что $\nu(x) = 6$ и $\nu(y) = 2$, имеем $\omega, \nu \vDash y|x$. При любой оценке ν такой, что $\nu(x) = 6$ и $\nu(y) = 5$, имеем $\omega, \nu \nvDash y|x$.

Укажем другую интерпретацию языка Ar: её носитель — множество всех вещественных чисел; оставшаяся часть определения этой интерпретации получается из соответствующей части определения интерпретации ω заменой слов «натуральные числа» на слова «вещественные числа» (с соответствующими изменениями окончаний этих слов). В этой интерпретации истинна формула $\forall x \exists y(x + y = 0)$, которая ложна в интерпретации ω.

Язык теории множеств Цермело-Френкеля

Сигнатура *языка теории множеств Цермело-Френкеля* содержит только два двухместных предикатных символа $=$ и $\in$. К примеру, формулами этого языка являются $\forall z(z \in x \supset z \in y)$ и $\forall y \neg(y \in x)$. Для них естественно ввести обозначения $x \subseteq y$ и $x = \varnothing$ соответственно.

Зададим интерпретацию этого языка. Пусть её носителем будет объединение множества $\mathbb{N}$ и множества всех подмножеств множества $\mathbb{N}$. Предикатным символам $=$ и $\in$ сопоставлены предикаты равенства и принадлежности, определённые как обычно. Формула $x = y \wedge x \in z \supset y \in z$ истинна в этой интерпретации (при любой оценке), а формула $\exists x(x \in x)$ — ложна.

Язык элементарной теории полугрупп и элементарной теории групп

Сигнатура *языка элементарной теории полугрупп* (и *языка элементарной теории групп*) состоит лишь из двухместного функционального символа $\cdot$ и двухместного предикатного символа $=$.

Формула

$$\forall x \forall y \forall z((x \cdot y) \cdot z = x \cdot (y \cdot z)) \qquad (*)$$

выражает ассоциативность операции $\cdot$.

Следующая формула выражает существование нейтрального элемента, а также для каждого элемента группы существование обратного к нему элемента:

$$\exists z(\forall x(x \cdot z = x \wedge z \cdot x = x) \wedge \forall x \exists y(x \cdot y = z \wedge y \cdot x = z)). \qquad (**)$$

Рассмотрим следующие интерпретации этого языка:

(1) множество всех натуральных чисел с операцией сложения и обычным равенством;

(2) множество всех натуральных чисел с операцией умножения и обычным равенством;

(3) множество всех слов в некотором алфавите с операцией конкатенации и равенством слов, означающим их совпадение;

(4) множество всех целых чисел с операцией сложения и обычным равенством;

(5) множество всех положительных рациональных чисел с операцией умножения и обычным равенством.

В интерпретациях (1), (2), (3) истинна формула $(*)$ и ложна формула $(**)$. В интерпретациях (4), (5) истинны формулы $(*)$ и $(**)$.

2.2.3. Формула, выражающая предикат

В текущем разделе 2.2.3 мы считаем различными предметные переменные, которые имеют различные обозначения.

Рассмотрим язык элементарной арифметики Ar, его стандартную интерпретацию ω и формулу $x \leqslant y \leftrightharpoons \exists z(y = x + z)$ этого языка. Данная формула имеет ровно два параметра x и y. Задавая различные оценки ν языка Ar в интерпретации ω, мы будем тем самым сопоставлять параметрам x и y натуральные числа $\nu(x)$ и $\nu(y)$ и получать истину или ложь в качестве истинностного значения формулы $x \leqslant y$. Эта формула будет истинна, если $\nu(x)$ не превосходит $\nu(y)$, и ложна в противном случае. Таким образом, формула $x \leqslant y$ задаёт (или выражает) предикат от двух переменных x и y, определённый на множестве $\mathbb{N}^2$ как «x не превосходит y». Обобщим эти наблюдения, введя следующее определение.

Определение 2.2.10. Пусть фиксированы язык первого порядка Ω и его интерпретация M с носителем $\mathcal{D}$. Пусть Π — n-местный предикат на $\mathcal{D}$; $x_1, \ldots, x_n$ — список попарно различных предметных переменных; A — формула языка Ω, все параметры которой входят в этот список. Тогда говорят, что *формула A выражает предикат $\Pi(x_1, \ldots, x_n)$ в интерпретации M*, если для любой оценки ν выполняется $|A|_{M,\nu} = \Pi(\nu(x_1), \ldots, \nu(x_n))$. ◁

Пример 2.2.11. Пусть рассматривается язык Ar и его стандартная интерпретация ω.

Каждая из формул $\exists z(y = x + Sz)$ и $\exists z(y = x + z) \wedge \neg(x = y)$ выражает предикат «x меньше y», но никакая из них не выражает предикат «y меньше x».

Формулы $y|x$ и Простое(x) (см. раздел 2.2.2) выражают предикаты «y является делителем числа x» и «x — простое» соответственно.

Пусть одноместный предикат $even_1(x)$ и двухместный предикат $even_2(x, y)$ определены как «x чётно». Тогда каждый из этих предикатов выразим любой из формул $\exists z(x = z \cdot SS0)$ и $\exists z(x = z + z)$. ◁

Упражнение 2.2.12. Запишите формулу языка Ar, выражающую в стандартной интерпретации этого языка следующие предикаты: 1) «z есть наибольший общий делитель x и y», 2) «z есть остаток от деления x на y, и y отлично от нуля», 3) «x есть степень некоторого простого числа». ◁

2.3. Свободные и правильные подстановки

В разделе 2.3 мы будем считать различными предметные переменные, обозначенные по-разному.

Рассмотрим, например, формулу $\exists z(y = x + z)$ языка Ar, которую мы сокращённо обозначаем через $x \leqslant y$.

Зафиксируем стандартную интерпретацию ω языка Ar. Попробуем выразить предикат $le2(u, z)$ «u не превосходит удвоенного z» формулой языка Ar, воспользовавшись тем, что формула $\exists z(y = x + z)$ выражает предикат $le(x, y)$ «x не превосходит y».

I. Сначала попытаемся выразить предикат $le2(u, z)$ формулой языка Ar, подставив терм u вместо переменной x и терм $z + z$ вместо переменной y в формулу $\exists z(y = x + z)$: получаем $\exists z(z + z = u + z)$. Но в интерпретации ω полученная формула выражает совсем не то, что мы хотели выразить! Эта неприятная ситуация произошла из-за того, что при подстановке увеличилось число вхождений связанных переменных, точнее, в область действия вхождения кванторной приставки $\exists z$ был подставлен терм $z + z$, в который входит переменная этой кванторной приставки.

Такую ситуацию, когда в область действия вхождения кванторной приставки вместо свободного вхождения переменной подставляется терм, в который входит переменная этой кванторной приставки, назовём *коллизией переменных*. Данный пример показывает, что коллизия переменных в некоторых случаях нежелательна.

Если же мы аналогичным образом попробуем выразить формулой языка Ar предикат $le2(u, v)$ «u не превосходит удвоенного v», то коллизии переменных не возникнет, и мы получим $\exists z(v + v = u + z)$, что и ожидали. Таким образом, можно избежать коллизии переменных, ограничившись подстановкой только таких термов, которые не вызывают коллизии переменных.

II. Рассмотрим второй способ избегания коллизии переменных при подстановке. Из определения 2.2.9 истинностного значения формулы следует, что истинностное значение рассматриваемой формулы не изменится, если все вхождения переменной z (связанные одним и тем же вхождением кванторной приставки) заменить на новую (т. е. не входящую в эту формулу) переменную w. Такую замену мы назовём (в общем случае определив в разделе 2.3.2) переименованием связанной переменной z.

Таким образом, для выражения предиката $le2(u, z)$ формулой языка Ar можно сначала переименовать связанную переменную z в формуле $\exists z(y = x + z)$ на новую переменную w. Получив формулу $\exists w(y = x + w)$, подставить в неё терм u вместо переменной x и терм $z + z$ вместо переменной y. Результатом будет формула $\exists w(z + z = u + w)$, выражающая предикат $le2(u, z)$.

2.3.1. Подстановка терма в формулу. Свободные подстановки

В данном разделе мы определим результат подстановки терма вместо переменной в формулу и уточним первый способ избегания коллизии переменных при такой подстановке.

Определение 2.3.1. Результат подстановки терма t вместо всех свободных вхождений переменной x в формулу A обозначим через $[A]_t^x$ и будем кратко называть *результатом подстановки t вместо x в A*. ◁

Пример 2.3.2. Пусть

$$E \leftrightharpoons \forall x P(x,y) \wedge \exists z Q(f(x),z,x).$$

Тогда $[E]^x_{g(w,y)}$ есть

$$\forall x P(x,y) \wedge \exists z Q(f(g(w,y)),z,g(w,y)).$$

$[\forall x P(x,y)]^x_w$ есть $\forall x P(x,y)$. ◁

Определение 2.3.3. Говорят, что *терм t свободен для переменной x в формуле A*, если никакое свободное вхождение переменной x в формулу A не находится в области действия вхождения кванторной приставки с переменной, входящей в терм t. В этом случае также говорят, что *подстановка терма t вместо переменной x в формулу A свободна*. ◁

Пример 2.3.4. Рассмотрим формулу E, определённую в предыдущем примере. Термы x и $g(w,y)$ свободны для переменной x в формуле E. Ни терм z, ни $g(z,y)$ не является свободным для x в E. ◁

Согласно проведённому в начале раздела 2.3 обсуждению подстановок термов вместо переменных, коллизия переменных при такой подстановке не возникает, если эта подстановка свободна.

Упражнение 2.3.5. Дайте индуктивные аналоги двух предыдущих определений. ◁

2.3.2. Конгруэнтные формулы. Правильные подстановки

В этом разделе мы уточним второй способ избегания коллизии переменных при подстановке. Этот способ заключается в переименовании некоторых связанных переменных. Сначала уточним, как понимать то, что одна формула получается из другой путём переименования связанных переменных; такие формулы мы назовём конгруэнтными.

Определение 2.3.6. *Лексемой* формулы A будем называть любое подслово слова A, являющееся предметной переменной, функциональным символом, предикатным символом, истинностной константой, логическим символом, открывающей круглой скобкой «(», закрывающей круглой скобкой «)» или запятой «,». ◁

Определение 2.3.7. Формулы A и B назовём *конгруэнтными* (и обозначим это как $A \approx B$), если формулы A и B состоят из одинакового числа k лексем, и для каждого $i = 1, \ldots, k$ выполняются все следующие условия:

1) если i-ая лексема формулы A не представляет вхождение предметной переменной, то эта же лексема является i-ой лексемой формулы B;

2) если i-ая лексема формулы A представляет свободное вхождение предметной переменной, то i-ая лексема формулы B представляет свободное вхождение той же предметной переменной;

3) если i-ая лексема формулы A представляет вхождение предметной переменной, связанное вхождением квантора, которое[7] представлено j-ой лексемой, то i-ая лексема формулы B представляет вхождение какой угодно предметной переменной, связанное вхождением квантора, которое представлено j-ой лексемой формулы B.

Если $A \approx B$, то будем также говорить, что для получения B мы *переименовали связанные переменные* в A. ◁

Пример 2.3.8. Любые две формулы из следующих трёх конгруэнтны:

$$\neg\forall x(P(x,y) \wedge \exists x Q(f(x),z,x)),$$
$$\neg\forall u(P(u,y) \wedge \exists x Q(f(x),z,x)),$$
$$\neg\forall u(P(u,y) \wedge \exists v Q(f(v),z,v)).$$

◁

Упражнение 2.3.9. Дайте индуктивное определение конгруэнтных формул. ◁

Замечание 2.3.10. Легко видеть, что $A \approx A$ (т. е. отношение $\approx$ рефлексивно); из $A \approx B$ следует $B \approx A$ (т. е. $\approx$ симметрично); из $A \approx B$ и $B \approx C$ следует $A \approx C$ (т. е. $\approx$ транзитивно); таким образом, отношение $\approx$ является отношением эквивалентности на множестве всех формул рассматриваемого языка первого порядка. ◁

Определение 2.3.11. Говорят, что формула A *обладает свойством чистоты переменных*, если $BV(A) \cap FV(A) = \varnothing$ и в A нет различных вхождений кванторных приставок с одной и той же переменной. ◁

Говоря очень грубо, в формуле, обладающей свойством чистоты переменных, ни одна переменная не используется в разных смыслах.

Пример 2.3.12. Формула $\forall x P(x,y) \vee \exists z Q(z)$ обладает свойством чистоты переменных, но ни одна из формул $\forall x P(x,y) \vee \exists x Q(x)$ и $\forall x P(x,y) \vee \exists y Q(y)$ не обладает свойством чистоты переменных. ◁

Лемма 2.3.13 (о чистоте переменных)**.** *Для любой формулы A и любого конечного множества V предметных переменных найдётся формула A' такая, что A' обладает свойством чистоты переменных, $A \approx A'$ и $BV(A') \cap V = \varnothing$.*

Доказательство. Применим индукцию по построению формулы A.

База индукции. A — атомарная формула. В качестве искомой формулы A' возьмём A.

Индукционный переход. Предположим, что доказываемое утверждение верно для формул C и D.

Пусть A имеет вид $\neg C$. По индукционному предположению найдётся формула C' такая, что она обладает свойством чистоты переменных, $C \approx C'$ и $BV(C') \cap V = \varnothing$. В качестве искомой формулы A' возьмём $\neg C'$.

Пусть A имеет вид $(C \odot D)$, где $\odot$ — одна из связок $\wedge, \vee, \supset$. Положим $V' \leftrightharpoons V \cup FV(D)$. По индукционному предположению найдётся

[7] Слово «которое» относится к ближайшему слева от него слову (точнее, вхождению слова) «вхождением», как и следующее вхождение слова «которое».

формула C' такая, что она обладает свойством чистоты переменных, $C \approx C'$ и $BV(C') \cap V' = \varnothing$. Положим $V'' \leftrightharpoons V' \cup FV(C') \cup BV(C')$. По индукционному предположению найдётся формула D' такая, что она обладает свойством чистоты переменных, $D \approx D'$ и $BV(D') \cap V'' = \varnothing$. В качестве искомой формулы A' подойдёт $(C' \odot D')$.

Наконец, пусть A имеет вид $\mathcal{Q}xC$, где $\mathcal{Q}$ — квантор. Выберем любую переменную y, не входящую в A и не принадлежащую V. По индукционному предположению найдётся формула C' такая, что она обладает свойством чистоты переменных, $[C]^x_y \approx C'$ и $BV(C') \cap (V \cup \{y\}) = \varnothing$. В качестве искомой формулы A' сгодится $\mathcal{Q}yC'$. ◁

Замечание 2.3.14. Изложенное (индуктивное) доказательство леммы о чистоте переменных по сути является описанием (рекурсивного) алгоритма построения формулы, обладающей указанными в лемме свойствами. ◁

Лемма о чистоте переменных позволяет ввести следующее определение подстановки терма вместо переменной в формулу. Это определение уточняет второй способ избегания коллизии переменных при подстановке.

Определение 2.3.15. Пусть A — формула, t – терм, x — предметная переменная. Найдём формулу A' такую, что $A \approx A'$ и t свободен для x в A'. Тогда $[A']^x_t$ назовём *результатом правильной подстановки* t *вместо* x *в* A и обозначим этот результат через $[[A]]^x_t$. ◁

Пример 2.3.16. Результатом правильной подстановки $[[\forall x P(f(x), y, x)]]^y_{f(x)}$ является $\forall z P(f(z), f(x), z)$, а также $\forall v P(f(v), f(x), v)$. ◁

По лемме 2.3.13 о чистоте переменных всегда можно найти требуемую в определении 2.3.15 формулу A' (например, можно построить такую формулу A', что $BV(A')$ не пересекается с множеством всех переменных, входящих в t). Однако $[[A]]^x_t$ зависит от выбора формулы A'. Для того, чтобы $[[A]]^x_t$ была определена однозначно, можно было бы выбирать переменные, связанные в A', некоторым детерминированным образом. Мы не будем определять такой выбор по следующей причине. Как мы отметили в замечании 2.3.10, $\approx$ является отношением эквивалентности на рассматриваемом множестве формул. Кроме того, легко показать, что если $A' \approx A''$ и t свободен для x в A' и в A'', то $[A']^x_t \approx [A'']^x_t$. Таким образом, результат правильной подстановки определён однозначно с точностью до конгруэнтности. И, как мы увидим в разделах 2.4 и 2.6.4, конгруэнтные формулы обычно взаимозаменяемы.

2.4. Логические законы

В данном разделе мы определим понятие логического закона (в логике предикатов), а также приведём некоторые логические законы и некоторые способы их получения. Напомним (см. конец раздела 2.1.1), что мы рассматриваем произвольный заранее зафиксированный язык первого порядка, если иное не оговорено явно.

Определение 2.4.1. Формула, истинная во всякой интерпретации при любой оценке, называется *общезначимой* формулой (*тождественно истинной* формулой, *тавтологией* или *логическим законом*). То, что формула A общезначима, обозначается через $\vDash A$.

Формула, не являющаяся общезначимой, называется *необщезначимой*. То, что формула A необщезначима, обозначается через $\not\models A$.

Формула, ложная во всякой интерпретации при любой оценке, называется *невыполнимой* (*противоречивой* или *тождественно ложной*).

Формула, не являющаяся противоречивой, называется *выполнимой* (или *непротиворечивой*). $\triangleleft$

Очевидно, что общезначимость формулы равносильна общезначимости её замыкания. Поэтому при исследовании вопроса об общезначимости формулы, можно предполагать, что эта формула замкнута.

Определение 2.4.2. Формула A называется *логическим следствием* (или *семантическим следствием*) множества формул Γ (это обозначается как $\Gamma \models A$), если для всякой интерпретации M и любой оценки ν выполняется: если для любой формулы $G \in \Gamma$ верно $M, \nu \models G$, то $M, \nu \models A$.

Если неверно, что $\Gamma \models A$, то пишем $\Gamma \not\models A$. $\triangleleft$

Если множество Γ пусто, то $\Gamma \models A$ в точности означает, что $\models A$. Будем записывать $G_1, \ldots, G_n \models A$ наряду с $\{G_1, \ldots, G_n\} \models A$, а также писать $G_1, \ldots, G_n \not\models A$ наряду с $\{G_1, \ldots, G_n\} \not\models A$.

Следующая теорема доказывается аналогично теореме 1.1.21 о логическом следствии для логики высказываний.

Теорема 2.4.3 (о логическом следствии)**.** *Пусть $G_1, \ldots, G_n$ ($n \in \mathbb{N}_+$), A — формулы. Тогда $G_1, \ldots, G_n \models A$, если и только если $\models G_1 \wedge \ldots \wedge G_n \supset A$.*

Бинарная логическая связка $\equiv$ (эквивалентность) вводится так же, как и для логики высказываний: $(A \equiv B)$ служит сокращённой записью для $((A \supset B) \wedge (B \supset A))$.

Определение 2.4.4. Формулы A и B называются *равносильными* (*логически эквивалентными* или *семантически эквивалентными*), если во всякой интерпретации M при любой оценке ν выполняется: $M, \nu \models A$ тогда и только тогда, когда $M, \nu \models B$. То, что формулы A и B равносильны, будем обозначать через $A \sim B$. $\triangleleft$

Очевидно, что формулы A и B равносильны, если и только если формула $(A \equiv B)$ является общезначимой. Также очевидно, что, как и для логики высказываний, отношение $\sim$ является отношением эквивалентности на множестве всех формул рассматриваемого языка.

Пример 2.4.5. Докажем, что для любой формулы A и любой переменной x справедливо

$$\neg\forall x A \sim \exists x \neg A. \qquad (*)$$

Согласно определению равносильных формул, для $(*)$ необходимо и достаточно, чтобы во всякой интерпретации $M \leftrightharpoons \langle \mathcal{D}, \mu \rangle$ при любой оценке ν имело место следующее:

(1) если $M, \nu \models \neg\forall x A$, то $M, \nu \models \exists x \neg A$; и

(2) если $M, \nu \models \exists x \neg A$, то $M, \nu \models \neg\forall x A$.

(1) Пусть $M, \nu \vDash \neg\forall x A$. Тогда по определению истинности формулы с главной связкой $\neg$ неверно, что $M, \nu \vDash \forall x A$. По определению истинности формулы с главным квантором $\forall$ имеем: неверно, что для любого $\alpha \in \mathcal{D}$ $M, \nu^x_\alpha \vDash A$. Следовательно, существует $\alpha \in \mathcal{D}$ такой, что $M, \nu^x_\alpha \nvDash A$. Тогда по определению истинности формулы с главной связкой $\neg$ существует $\alpha \in \mathcal{D}$ такой, что $M, \nu^x_\alpha \vDash \neg A$, что, согласно определению истинности формулы с главным квантором $\exists$, означает $M, \nu \vDash \exists x \neg A$.

(2) Доказательство того, что $M, \nu \vDash \exists x \neg A$ влечёт $M, \nu \vDash \neg\forall x A$, проводится аналогично и может быть получено путём записи предложений доказательства (1) в обратном порядке с очевидными изменениями. ◁

Пример 2.4.6. Докажем, что для любых формул A и B и какой угодно переменной x общезначима формула $\forall x A \vee \forall x B \supset \forall x (A \vee B)$.

Согласно определению истинности формулы с главной связкой $\supset$, для общезначимости этой формулы необходимо и достаточно, чтобы во всякой интерпретации $M \leftrightharpoons \langle \mathcal{D}, \mu \rangle$ при любой оценке ν было верно: если $M, \nu \vDash \forall x A \vee \forall x B$, то $M, \nu \vDash \forall x (A \vee B)$.

Пусть $M, \nu \vDash \forall x A \vee \forall x B$. Тогда $M, \nu \vDash \forall x A$ или $M, \nu \vDash \forall x B$. По определению истинности формулы с главным квантором $\forall$ имеем: для любого $\alpha \in \mathcal{D}$ выполняется $M, \nu^x_\alpha \vDash A$ или для любого $\beta \in \mathcal{D}$ выполняется $M, \nu^x_\beta \vDash B$. Следовательно, для любого $\gamma \in \mathcal{D}$ выполняется $M, \nu^x_\gamma \vDash A$ или $M, \nu^x_\gamma \vDash B$. Поэтому, согласно определению истинности формулы с главной связкой $\vee$, для любого $\gamma \in \mathcal{D}$ выполняется $M, \nu^x_\gamma \vDash A \vee B$. Отсюда по определению истинности формулы с главным квантором $\forall$ получаем требуемое $M, \nu \vDash \forall x (A \vee B)$. ◁

Пример 2.4.7. Докажем, что формула $\forall x (P(x) \vee Q(x)) \supset \forall x P(x) \vee \forall x Q(x)$ необщезначима, где P и Q — различные одноместные предикатные символы.

Для этого укажем интерпретацию $M \leftrightharpoons \langle \mathcal{D}, \mu \rangle$ и оценку ν, при которых эта формула ложна, т. е. при которых $M, \nu \vDash \forall x (P(x) \vee Q(x))$ и $M, \nu \nvDash \forall x P(x) \vee \forall x Q(x)$. Легко проверить, что искомые интерпретация M и оценка ν могут быть определены следующим образом. $\mathcal{D}$ — двухэлементное множество $\{d, e\}$. Предикат $\mu(P)$ таков, что $\mu(P)(d) = 1$, $\mu(P)(e) = 0$. Предикат $\mu(Q)$ таков, что $\mu(Q)(d) = 0$, $\mu(Q)(e) = 1$. Оценка ν может быть произвольной, поскольку рассматриваемые формулы замкнуты.

С другой стороны, исходная формула выполнима: она истинна в любой интерпретации, в которой предикатным символам P и Q сопоставлены тождественно истинные предикаты. ◁

Упражнение 2.4.8. Убедитесь, что формула из предыдущего примера истинна в любой интерпретации с одноэлементным носителем. ◁

Перечислим некоторые общезначимые и равносильные формулы, а также некоторые способы получения таких формул. Ниже в текущем разделе 2.4 A и B обозначают любые формулы рассматриваемого языка первого порядка (как мы и условились в конце раздела 2.1.1), x и y — произвольные переменные; дополнительные ограничения на A, B, x и y специально оговариваются.

Следующие 2 пункта представляют так называемые законы Де Моргана для кванторов.

1. $\neg\forall x A \sim \exists x \neg A$.

2. $\neg\exists x A \sim \forall x \neg A$.

В следующих пунктах 3–10 требуется, чтобы переменная x не входила свободно в A. Эти пункты представляют так называемые законы одностороннего пронесения кванторов.

3. $A \wedge \forall x B \sim \forall x (A \wedge B)$.
4. $A \vee \forall x B \sim \forall x (A \vee B)$.
5. $A \wedge \exists x B \sim \exists x (A \wedge B)$.
6. $A \vee \exists x B \sim \exists x (A \vee B)$.
7. $A \supset \forall x B \sim \forall x (A \supset B)$.
8. $A \supset \exists x B \sim \exists x (A \supset B)$.
9. $\forall x B \supset A \sim \exists x (B \supset A)$.
10. $\exists x B \supset A \sim \forall x (B \supset A)$.

В следующих пунктах переменная x уже может входить свободно в A.

11. $\forall x A \wedge \forall x B \sim \forall x (A \wedge B)$.
12. $\exists x A \vee \exists x B \sim \exists x (A \vee B)$.
13. $\vDash \forall x A \vee \forall x B \supset \forall x (A \vee B)$.
14. $\vDash \exists x (A \wedge B) \supset \exists x A \wedge \exists x B$.

В пунктах 15–16 переменная y не входит в A.

15. $\forall x A \sim \forall y [A]^x_y$.
16. $\exists x A \sim \exists y [A]^x_y$.
17. Если $A \approx B$, то $A \sim B$. (Т. е. конгруэнтные формулы равносильны.)
18. Если терм t свободен для переменной x в A, то $\vDash \forall x A \supset [A]^x_t$.
19. Если терм t свободен для переменной x в A, то $\vDash [A]^x_t \supset \exists x A$.
20. Если $A \sim B$, терм t свободен для переменной x в A и в B, то $[A]^x_t \sim [B]^x_t$.

Определение 2.4.9. Пусть C — пропозициональная формула, $p_1, \ldots, p_n$ — все входящие в C попарно различные пропозициональные переменные, $B_1, \ldots, B_n$ — предикатные формулы. Тогда формула, полученная в результате одновременной замены всех вхождений $p_1, \ldots, p_n$ в C на $B_1, \ldots, B_n$ соответственно, называется *частным случаем пропозициональной формулы* C. ◁

Теорема 2.4.10 (о подстановке предикатных формул в пропозициональную тавтологию)**.** *Любой частный случай пропозициональной тавтологии является общезначимой формулой.*

Пример 2.4.11. Так как $\models \underline{q} \supset \underline{\underline{p}} \vee \underline{q}$, то по предыдущей теореме

$$\models \underline{\neg\exists x Q(x,y)} \supset \underline{\underline{R(y) \wedge R(z)}} \vee \underline{\neg\exists x Q(x,y)}.$$

Здесь одним количеством черт подчёркнуты пропозициональная переменная и подставленная вместо неё предикатная формула. ◁

Следующая теорема позволяет выполнять равносильные преобразования формул таким образом: можно заменить в формуле вхождение подформулы на формулу, семантически эквивалентную (равносильную) этой подформуле, и утверждать, что семантически эквивалентны исходная формула и формула, полученная в результате этой замены.

Теорема 2.4.12 (о семантически эквивалентной замене)**.** *Пусть A, B и B' — формулы, A' есть результат замены некоторого вхождения B в A на B'. Тогда если $B \sim B'$, то $A \sim A'$.*

Пример 2.4.13. Проиллюстрируем эту теорему:

$$\underbrace{\neg\exists x Q(x,y)}_{B} \sim \underbrace{\forall x \neg Q(x,y)}_{B'} \quad \Rightarrow$$

$$\overbrace{\forall y(P(x,y) \wedge \underbrace{\neg\exists x Q(x,y)}_{B})}^{A} \sim \overbrace{\forall y(P(x,y) \wedge \underbrace{\forall x \neg Q(x,y)}_{B'})}^{A'}.$$

◁

Упражнение 2.4.14. Докажите утверждения, приведённые выше в пунктах 2–12, 14–18 и 20, а также теоремы 2.4.10 и 2.4.12. Можно ли ослабить какое-либо из ограничений типа «переменная не входит в формулу», «переменная не входит свободно в формулу» и «терм свободен для переменной в формуле»? *Указание.* Большинство из требуемых доказательств можно провести таким же образом, как и доказательства в примерах текущего раздела. Остальные доказательства можно провести с помощью индукции по построению формул и термов. ◁

Докажем утверждение пункта 19, сформулировав его в виде следующей теоремы.

Теорема 2.4.15. *Если терм t свободен для переменной x в формуле A, то $\models [A]^x_t \supset \exists x A$.*

Для доказательства этой теоремы введём обозначение и докажем две леммы.

Результат подстановки терма t вместо всех вхождений переменной x в терм s обозначим через $[s]^x_t$.

Лемма 2.4.16. *Пусть s, t — термы, x — переменная, $M \leftrightharpoons \langle \mathcal{D}, \mu \rangle$ — интерпретация, ν — оценка. Тогда $|[s]^x_t|_{M,\nu} = |s|_{M,\nu^x_{|t|_{M,\nu}}}$.*

Доказательство. Проведём индукцию по построению терма s.

База индукции. Терм s является предметной константой или переменной.

Если s является предметной константой или переменной, отличной от x, то $|[s]^x_t|_{M,\nu} = |s|_{M,\nu} = |s|_{M,\nu^x_{|t|_{M,\nu}}}$.

Если же s есть x, то $|[s]^x_t|_{M,\nu} = |t|_{M,\nu} = |x|_{M,\nu^x_{|t|_{M,\nu}}} = |s|_{M,\nu^x_{|t|_{M,\nu}}}$.

Индукционный переход. Предположим, что доказываемое утверждение верно для термов $s_1, \ldots, s_n$, взятых в качестве s. Рассмотрим в качестве s терм $f(s_1, \ldots, s_n)$.

По индукционному предположению $|[s_i]^x_t|_{M,\nu} = |s_i|_{M,\nu^x_{|t|_{M,\nu}}}$ верно для каждого $i = 1, \ldots, n$. Тогда

$$|[s]^x_t|_{M,\nu} = |[f(s_1, \ldots, s_n)]^x_t|_{M,\nu} = |f([s_1]^x_t, \ldots, [s_n]^x_t)|_{M,\nu} =$$
$$\mu(f)(|[s_1]^x_t|_{M,\nu}, \ldots, |[s_n]^x_t|_{M,\nu}) = \mu(f)(|s_1|_{M,\nu^x_{|t|_{M,\nu}}}, \ldots, |s_n|_{M,\nu^x_{|t|_{M,\nu}}}) =$$
$$|f(s_1, \ldots, s_n)|_{M,\nu^x_{|t|_{M,\nu}}} = |s|_{M,\nu^x_{|t|_{M,\nu}}}.$$

Это завершает доказательство индукционного перехода и данной леммы. ◁

Лемма 2.4.17. *Пусть A — формула, x — переменная, t — терм, свободный для x в A, $M \leftrightharpoons \langle \mathcal{D}, \mu \rangle$ — интерпретация, ν — оценка. Тогда* $|[A]^x_t|_{M,\nu} = |A|_{M,\nu^x_{|t|_{M,\nu}}}$.

Д о к а з а т е л ь с т в о. Воспользуемся индукцией по построению формулы A.

База индукции. Формула A атомарна.

Если A есть истинностная константа или пропозициональная переменная, то $|[A]^x_t|_{M,\nu} = |A|_{M,\nu} = |A|_{M,\nu^x_{|t|_{M,\nu}}}$.

В противном случае A имеет вид $P(s_1, \ldots, s_n)$. По лемме 2.4.16 для каждого $i = 1, \ldots, n$ верно $|[s_i]^x_t|_{M,\nu} = |s_i|_{M,\nu^x_{|t|_{M,\nu}}}$. Тогда требуемое равенство $|[A]^x_t|_{M,\nu} = |A|_{M,\nu^x_{|t|_{M,\nu}}}$ получается с помощью цепочки равенств, которая аналогична цепочке равенств, приведённой в доказательстве индукционного перехода леммы 2.4.16.

Индукционный переход. Предположим, что доказываемое утверждение верно для формул B и C, взятых в качестве A.

1. Пусть A имеет вид $\neg B$. Тогда, учитывая, что по индукционному предположению $|[B]^x_t|_{M,\nu} = |B|_{M,\nu^x_{|t|_{M,\nu}}}$, имеем

$$|[A]^x_t|_{M,\nu} = |[\neg B]^x_t|_{M,\nu} = |\neg[B]^x_t|_{M,\nu} = F_\neg(|[B]^x_t|_{M,\nu}) =$$
$$F_\neg(|B|_{M,\nu^x_{|t|_{M,\nu}}}) = |\neg B|_{M,\nu^x_{|t|_{M,\nu}}} = |A|_{M,\nu^x_{|t|_{M,\nu}}},$$

где $F_\neg$ — функция истинности отрицания (см. раздел 1.1.3).

2–4. Случаи 2–4, когда A имеет вид $B \wedge C$, $B \vee C$ и $B \supset C$ соответственно, рассматриваются аналогично случаю 1.

5. Пусть A имеет вид $\forall y B$.

Если x не является параметром формулы A, то требуемое равенство очевидно.

Если же x является параметром формулы A, то x не совпадает с y, и мы имеем

$$|[A]^x_t|_{M,\nu} = |[\forall y B]^x_t|_{M,\nu} = |\forall y [B]^x_t|_{M,\nu}. \qquad (\star)$$

Далее, $|\forall y [B]^x_t|_{M,\nu} = 1$, если и только если для любого $\alpha \in \mathcal{D}$ выполняется $|[B]^x_t|_{M,\nu^y_\alpha} = 1$. По индукционному предположению

$$|[B]^x_t|_{M,\nu^y_\alpha} = |B|_{M,(\nu^y_\alpha)^x_{|t|_{M,\nu^y_\alpha}}}. \qquad (\star\star)$$

Так как t свободен для x в формуле A (имеющей вид $\forall y B$), то y не входит в t, поэтому $|t|_{M,\nu^y_\alpha} = |t|_{M,\nu}$. Кроме того, изменения оценки ν в различных точках x и y можно переставить местами; таким образом, мы получаем $(\nu^y_\alpha)^x_{|t|_{M,\nu^y_\alpha}} = (\nu^y_\alpha)^x_{|t|_{M,\nu}} = \left(\nu^x_{|t|_{M,\nu}}\right)^y_\alpha$. Тогда из $(\star\star)$ следует, что

$$|[B]^x_t|_{M,\nu^y_\alpha} = |B|_{M,\left(\nu^x_{|t|_{M,\nu}}\right)^y_\alpha}.$$

Теперь $|\forall y [B]^x_t|_{M,\nu} = 1$, если и только если для любого $\alpha \in \mathcal{D}$ выполняется $|B|_{M,\left(\nu^x_{|t|_{M,\nu}}\right)^y_\alpha} = 1$. Отсюда

$$|\forall y [B]^x_t|_{M,\nu} = |\forall y B|_{M,\nu^x_{|t|_{M,\nu}}} = |A|_{M,\nu^x_{|t|_{M,\nu}}},$$

что вместе с $(\star)$ даёт нам требуемое равенство.

6. Последний случай, когда A имеет вид $\exists y B$, рассматривается аналогично случаю 5. ◁

Д о к а з а т е л ь с т в о теоремы 2.4.15. Чтобы доказать общезначимость формулы $[A]^x_t \supset \exists x A$, возьмём произвольные интерпретацию $M \leftrightharpoons \langle \mathcal{D}, \mu \rangle$ и оценку ν, предположим, что $M, \nu \vDash [A]^x_t$, и покажем, что $M, \nu \vDash \exists x A$.

Для $M, \nu \vDash \exists x A$ необходимо и достаточно, чтобы существовал $\alpha \in \mathcal{D}$ такой, что $|A|_{M,\nu^x_\alpha} = 1$. По лемме 2.4.17 $|[A]^x_t|_{M,\nu} = |A|_{M,\nu^x_{|t|_{M,\nu}}}$, а по предположению $|[A]^x_t|_{M,\nu} = 1$, поэтому в качестве искомого α подойдёт $|t|_{M,\nu}$. ◁

Упражнение 2.4.18. Докажите следующее утверждение.

Пусть A — бескванторная предикатная формула; $B_1, \ldots, B_n$ — все входящие в A попарно различные атомарные формулы, ни одна из которых не является истинностной константой; $p_1, \ldots, p_n$ — попарно различные пропозициональные переменные; A' — пропозициональная формула, полученная из A заменой каждого вхождения атомарной формулы B_i на p_i $(i = 1, \ldots, n)$. Тогда A общезначима, если и только если A' общезначима. (Это утверждение можно обобщить и сформулировать короче: бескванторная предикатная формула общезначима тогда и только тогда, когда она является частным случаем пропозициональной тавтологии. См. также теорему 2.4.10.)

Проиллюстрируем это утверждение примером предикатной формулы и соответствующей пропозициональной формулы:

$$(P(x) \supset P(y)) \wedge (P(y) \supset R(z)) \supset (P(x) \supset R(z)),$$
$$(px \supset py) \wedge (py \supset r) \supset (px \supset r),$$

где различны предметные переменные и предикатные символы, которые имеют различные обозначения. ◁

2.5. Предварённая нормальная форма

Определение 2.5.1. Говорят, что формула A *находится в предварённой* (или *пренексной*) *нормальной форме*, если A имеет вид $\mathcal{Q}_1 x_1 \dots \mathcal{Q}_n x_n B$, где $n \geqslant 0$, $\mathcal{Q}_i$ — квантор для каждого $i = 1, \dots, n$, B — бескванторная формула. Также такая формула A называется *предварённой нормальной формой* (сокращённо — *ПНФ*). Формула B называется *матрицей* предварённой нормальной формы A.

Предварённой нормальной формой формулы A (короче — *ПНФ формулы* A) называется любая формула B, находящаяся в ПНФ и такая, что $A \sim B$. ◁

Пример 2.5.2. Найдём ПНФ формулы $A \leftrightharpoons \forall x P(x) \supset \forall x Q(x)$. Воспользовавшись одним из законов одностороннего пронесения кванторов (см. раздел 2.4), получаем формулу $A_1 \leftrightharpoons \exists x (P(x) \supset \forall x Q(x))$, равносильную формуле A. Прежде чем применить к A_1 другой из законов одностороннего пронесения кванторов, переименуем связанную кванторной приставкой $\forall x$ переменную x на отличную от неё переменную y: имеем формулу $A_2 \leftrightharpoons \exists x (P(x) \supset \forall y Q(y))$, равносильную формуле A_1. Теперь получаем формулу $\exists x \forall y (P(x) \supset Q(y))$, находящуюся в ПНФ и равносильную формуле A_2, а значит, и исходной формуле A. Итак, $\exists x \forall y (P(x) \supset Q(y))$ является ПНФ формулы A. ◁

Теорема 2.5.3. *Для любой формулы A существует предварённая нормальная форма B формулы A.*

Доказательство. Используем индукцию по построению формулы A.

База индукции. A — атомарная формула. Тогда в качестве B подходит сама формула A.

Индукционный переход. Предположим, что доказываемое утверждение верно для формул C и D.

Пусть A имеет вид $\neg C$. По индукционному предположению для формулы C найдётся её ПНФ вида $\mathcal{Q}_1 x_1 \dots \mathcal{Q}_n x_n E$, где $\mathcal{Q}_i$ — квантор для каждого $i = 1, \dots, n$, E — матрица. В качестве искомой формулы B возьмём $\overline{\mathcal{Q}}_1 x_1 \dots \overline{\mathcal{Q}}_n x_n \neg E$, где $\overline{\forall} \leftrightharpoons \exists$ и $\overline{\exists} \leftrightharpoons \forall$.

Пусть A имеет вид $(C \odot D)$, где $\odot$ — логическая связка $\wedge$, $\vee$ или $\supset$. По индукционному предположению найдутся формулы E и F — ПНФ формул C и D соответственно. В силу леммы 2.3.13 о чистоте переменных можно считать, что $BV(E) \cap FV(F) = \varnothing$ и $BV(F) \cap (FV(E) \cup BV(E)) = \varnothing$. Тогда, применяя законы одностороннего пронесения кванторов, из $(E \odot F)$ получим искомую ПНФ формулы A.

Наконец, пусть A имеет вид $\mathcal{Q} x C$, где $\mathcal{Q}$ — квантор. По индукционному предположению найдётся формула E — ПНФ формулы C. В качестве искомой формулы B возьмём $\mathcal{Q} x E$. ◁

Замечание 2.5.4. Из доказательства предыдущей теоремы легко видеть, что для замкнутой формулы может быть построена замкнутая ПНФ этой формулы. ◁

Упражнение 2.5.5. Опишите алгоритм построения ПНФ произвольной заданной формулы. Этот алгоритм может быть оптимизированным по сравнению с алгоритмом, действующим в соответствии с доказательством предыдущей теоремы. ◁

Пример 2.5.6. Построим ПНФ формулы

$$A \leftrightharpoons \exists x(\exists y P(y) \supset (\neg\exists z Q(z) \wedge \exists y R(x,y))),$$

причём во всём этом примере мы считаем различными предметные переменные, которые имеют различные обозначения. Имеем следующую цепочку равносильных формул:[8]

$$\begin{gathered}
A \sim \exists x\forall y(P(y) \supset (\neg\exists z Q(z) \wedge \exists y R(x,y))) \sim \\
\exists x\forall y(P(y) \supset (\forall z\neg Q(z) \wedge \exists y R(x,y))) \sim \\
\exists x\forall y(P(y) \supset \forall z(\neg Q(z) \wedge \exists y R(x,y))) \sim \\
\exists x\forall y\forall z(P(y) \supset (\neg Q(z) \wedge \exists y R(x,y))) \sim \\
\exists x\forall y\forall z(P(y) \supset \exists y(\neg Q(z) \wedge R(x,y))) \sim \\
\exists x\forall y\forall z(P(y) \supset \exists y_1(\neg Q(z) \wedge R(x,y_1))) \sim \\
\exists x\forall y\forall z\exists y_1(P(y) \supset (\neg Q(z) \wedge R(x,y_1))).
\end{gathered}$$

◁

Замечание 2.5.7. В доказательстве теоремы 2.5.3 рассматриваются лишь конечные объекты (слова), строящиеся из символов некоторого (конечного) алфавита, и даётся явный способ построения всех вводимых в рассмотрение объектов из ранее построенных. Про такие доказательства говорят, что они являются конструктивными[9]. Характерным примером конструктивного доказательства является также доказательство леммы 2.3.13 о чистоте переменных; в этом доказательстве (индуктивно) строится объект, существование которого утверждает эта лемма, так что это доказательство по сути представляет собой (рекурсивный) алгоритм нахождения требуемой формулы.

Ценность конструктивных доказательств заключается как раз в том, что если в них утверждается существование объекта, то они дают явный способ (алгоритм) построения такого объекта, в отличие, скажем, от приведённого нами в примере 2.4.5 доказательства. В нём мы совершили переход вида: неверно, что для любого $\alpha \in \mathcal{D}$ имеет место $\boldsymbol{A}$; следовательно, существует $\alpha \in \mathcal{D}$ такой, что неверно $\boldsymbol{A}$. При этом мы не предложили способ построения α, существование которого утверждается. Кроме того, понятие множества, вообще говоря, не предполагает алгоритма построения множества. Так что это доказательство неконструктивно. ◁

[8] Цепочка $A \sim B \sim C$ (где A, B и C — формулы) понимается так: $A \sim B$ и $B \sim C$. Аналогично понимаются и цепочки с бóльшим числом вхождений $\sim$.

[9] Мы даём лишь некоторое представление о конструктивных доказательствах, а не их точное определение.

2.6. Исчисление предикатов гильбертовского типа

В данном разделе для произвольного языка первого порядка мы определим и изучим исчисление, в котором выводимы все общезначимые формулы этого языка и только они.

2.6.1. Формулировка исчисления

Изучение исчисления высказываний гильбертовского типа было вызвано в первую очередь методическими причинами (см. замечание 1.3.2). Для доказательства общезначимости пропозициональной формулы достаточно вычислить истинностное значение этой формулы в конечном числе интерпретаций. С доказательством общезначимости предикатной формулы дело, вообще говоря, обстоит иначе: требуется учитывать бесконечное число интерпретаций, в том числе и с бесконечными носителями. Доказательства общезначимости, приведённые в разделе 2.4, неформальны: они опираются на неуточнённые семантические представления о естественном языке и о множествах. Поэтому исчисление, формализующее доказательства общезначимости предикатных формул, имеет принципиальное значение: такое уточнение понятия доказательства позволит, во-первых, сделать доказательства объектами математического исследования и, во-вторых, алгоритмизировать проверку и поиск доказательств.

Пусть задан произвольный язык первого порядка Ω. Сформулируем так называемое *исчисление предикатов гильбертовского типа в языке* Ω, которое будем обозначать через $\mathcal{H}\forall(\Omega)$.

Аксиомами исчисления $\mathcal{H}\forall(\Omega)$ являются все формулы языка Ω, имеющие один из следующих видов:

все формулы, имеющие вид аксиом исчисления высказываний гильбертовского типа $\mathcal{H}$ (см. схемы аксиом 1–12 в разделе 1.3.1), где A, B и C теперь обозначают произвольные формулы языка Ω;

13) $\forall x A \supset [A]^x_t$;

14) $[A]^x_t \supset \exists x A$;

15) $\forall x(B \supset A) \supset (B \supset \forall x A)$;

16) $\forall x(A \supset B) \supset (\exists x A \supset B)$.

В схемах аксиом 13–16 A и B — произвольные формулы языка Ω; x — произвольная предметная переменная, не входящая свободно в B; t — произвольный терм, свободный для x в A.

Правилами вывода исчисления $\mathcal{H}\forall(\Omega)$ являются *модус поненс* (обозначается, как и раньше, MP) и *правило обобщения* (обозначается Gen)[10]:

$$\frac{A;\ A \supset B}{B}\ (MP), \qquad \frac{A}{\forall x A}\ (Gen),$$

где A, B — произвольные формулы языка Ω, x — произвольная предметная переменная.

[10] Gen — сокращение «*Generalization*».

Формулировка исчисления $\mathcal{H}\forall(\Omega)$ закончена. Содержательный смысл правила вывода модус поненс был раскрыт в разделе 1.3.1. Содержательный смысл правила обобщения естественен: если доказано утверждение A, которое, возможно, зависит от произвольного x, то тем самым доказано, что для любого x имеет место A.

Как мы скоро докажем, связь исчисления предикатов $\mathcal{H}\forall(\Omega)$ с семантикой языка Ω подобна связи исчисления высказываний $\mathcal{H}$ с семантикой языка логики высказываний: в исчислении $\mathcal{H}\forall(\Omega)$ выводимы все общезначимые формулы языка Ω и только они.

Как и в исчислении высказываний $\mathcal{H}$, в исчислении $\mathcal{H}\forall(\Omega)$ выводима формула $A \supset A$ для любой формулы A языка Ω. Действительно, вывод формулы $A \supset A$ в $\mathcal{H}\forall(\Omega)$ по форме совпадает с выводом в $\mathcal{H}$ (см. пример 1.3.1).

Пример 2.6.1. Построим вывод формулы $\exists x A \supset \exists y [A]^x_y$, где A — произвольная формула, x и y — произвольные предметные переменные, причём y не входит в A.

1) $[[A]^x_y]^y_x \supset \exists y [A]^x_y$ — аксиома вида 14 (все указанные подстановки свободны, и $[[A]^x_y]^y_x$ совпадает с A, так как y не входит в A);

2) $\forall x (A \supset \exists y [A]^x_y)$ получается из формулы 1 по Gen;

3) $\forall x (A \supset \exists y [A]^x_y) \supset (\exists x A \supset \exists y [A]^x_y)$ — аксиома вида 16 (x не входит свободно в $\exists y [A]^x_y$);

4) $\exists x A \supset \exists y [A]^x_y$ получается из формул 2 и 3 по MP. ◁

Упражнение 2.6.2. Докажите, что

$$\vdash \exists y [A]^x_y \supset \exists x A, \qquad \vdash \forall x A \supset \forall y [A]^x_y, \qquad \vdash \forall y [A]^x_y \supset \forall x A,$$

где A — произвольная формула, x и y — произвольные предметные переменные, причём y не входит в A. Как можно ослабить ограничение «y не входит в A»? ◁

Упражнение 2.6.3. Рассмотрим исчисление, которое получается из исчисления $\mathcal{H}\forall(\Omega)$ заменой схем аксиом 15 и 16 и правила обобщения на так называемые правила Бернайса:

$$\frac{B \supset A}{B \supset \forall x A}, \qquad \frac{A \supset B}{\exists x A \supset B},$$

где A, B — произвольные формулы языка Ω; x — произвольная предметная переменная, не входящая в B свободно. Докажите, что в этом исчислении и в исчислении $\mathcal{H}\forall(\Omega)$ выводимы одни и те же формулы. ◁

2.6.2. Вывод из гипотез. Корректность исчисления. Теорема о дедукции. Допустимые правила

Нам известны теоремы о корректности и о дедукции для исчисления высказываний гильбертовского типа. Мы докажем аналоги этих теорем для исчисления предикатов гильбертовского типа, но перед этим нам потребуется особым образом определить понятие вывода из гипотез в этом исчислении.

Пусть в текущем разделе 2.6.2 рассматривается произвольный язык первого порядка Ω.

Предположим, что одноместный предикатный символ P входит в сигнатуру этого языка. Из формулы $P(x)$ по правилу Gen получается формула $\forall x P(x)$, но $P(x) \not\models \forall x P(x)$ (подберите интерпретацию и оценку, в которых истинна $P(x)$ и ложна $\forall x P(x)$). Поэтому обобщённая теорема 1.3.3 о корректности исчисления высказываний $\mathcal{H}$ не переносится (без каких-либо изменений) на исчисление предикатов $\mathcal{H}\forall(\Omega)$.

Указанная проблема возникла из-за того, что в выводе из гипотезы мы, применив правило Gen, сделали связанным параметр гипотезы, тогда как в обычных математических рассуждениях это не допускается. Дадим для вывода из гипотез в исчислении $\mathcal{H}\forall(\Omega)$ определение, накладывающее дополнительное ограничение на общее понятие вывода из гипотез в произвольном исчислении (см. раздел 1.2.2).

Пусть дано произвольное множество Γ формул языка Ω. *Выводом из* Γ *в исчислении* $\mathcal{H}\forall(\Omega)$ называется конечная последовательность формул, каждая из которых либо является аксиомой, либо принадлежит Γ, либо получается из предшествующих формул этой последовательности по правилу MP, либо имеет вид $\forall x A$ и получается из некоторой предшествующей формулы A этой последовательности по правилу Gen, причём переменная x не входит свободно ни в одну из формул-элементов множества Γ.

Таким образом, дополнительное ограничение, накладываемое на общее понятие вывода из гипотез, состоит в том, что при получении формулы $\forall x A$ из формулы A по правилу Gen в качестве x нельзя брать никакой параметр гипотезы. Заметим, что если все гипотезы замкнуты, то это ограничение будет выполнено, какую бы переменную ни взять в качестве x.

Понятия вывода формулы из Γ в исчислении $\mathcal{H}\forall(\Omega)$ и формулы, выводимой (соответственно, невыводимой) из Γ в исчислении $\mathcal{H}\forall(\Omega)$, определяются на основе данного определения вывода из Γ в исчислении $\mathcal{H}\forall(\Omega)$ так же, как в разделе 1.2.2. То, что формула A выводима (соответственно, невыводима) из Γ в $\mathcal{H}\forall(\Omega)$ (в определённом в текущем разделе 2.6.2 смысле), мы будем обозначать через $\Gamma \vdash_{\mathcal{H}\forall(\Omega)} A$ или, короче, через $\Gamma \vdash A$ (соответственно, через $\Gamma \nvdash_{\mathcal{H}\forall(\Omega)} A$ или $\Gamma \nvdash A$).

Теорема 2.6.4 (обобщённая теорема о корректности исчисления предикатов $\mathcal{H}\forall(\Omega)$)**.** *Пусть* Γ *— произвольное множество формул,* A *— произвольная формула. Тогда если* $\Gamma \vdash A$*, то* $\Gamma \models A$.

Доказательство проводится аналогично доказательству теоремы 1.3.3 индукцией по длине вывода $\Gamma \vdash A$. Факты из раздела 2.4 позволяют легко установить, что каждая аксиома исчисления $\mathcal{H}\forall(\Omega)$ общезначима. Остаётся лишь добавить в индукционный переход рассмотрение случая, когда формула A имеет вид $\forall x B$ и получается в выводе из формулы B по правилу Gen, причём переменная x не входит свободно ни в одну из формул множества Γ.

Пусть $M \leftrightharpoons \langle \mathcal{D}, \mu \rangle$ и ν — произвольные интерпретация и оценка такие, что для любой формулы $G \in \Gamma$ верно $M, \nu \models G$. Для завершения доказательства достаточно установить, что $M, \nu \models \forall x B$. Зафиксируем произвольный элемент $\alpha \in \mathcal{D}$. Так как x не входит свободно ни в одну из формул множества Γ, то для любой формулы $G \in \Gamma$ верно $M, \nu^x_\alpha \models G$. По индукци-

онному предположению $\Gamma \vdash B$ влечёт $\Gamma \models B$. Тогда $M, \nu^x_\alpha \models B$. Поскольку α — произвольный элемент $\mathcal{D}$, имеем $M, \nu \models \forall x B$. $\triangleleft$

Приводимые ниже 3 следствия теоремы 2.6.4 доказываются так же, как и их пропозициональные аналоги (см. раздел 1.3.2).

Следствие 2.6.5. *Пусть* Γ *— произвольное множество общезначимых формул,* A *— произвольная формула. Тогда если* $\Gamma \vdash A$*, то* $\models A$*.*

Следствие 2.6.6 (корректность исчисления предикатов $\mathcal{H}\forall(\Omega)$)**.** *Всякая выводимая в исчислении предикатов* $\mathcal{H}\forall(\Omega)$ *формула общезначима, т. е. для любой формулы* A*, если* $\vdash A$*, то* $\models A$*.*

Следствие 2.6.7. *Исчисление предикатов* $\mathcal{H}\forall(\Omega)$ *непротиворечиво, т. е. не существует формулы* A *такой, что* $\vdash A$ *и* $\vdash \neg A$*.*

Перейдём к теореме о дедукции. Заметим, что без обсуждавшегося ограничения на вывод из гипотез теорема 1.3.7 о дедукции для исчисления высказываний $\mathcal{H}$ не могла бы быть перенесена на исчисление предикатов $\mathcal{H}\forall(\Omega)$: из $P(x)$ по правилу Gen получается $\forall x P(x)$, но формула $P(x) \supset \forall x P(x)$ не должна быть выводимой. Эта формула не должна быть выводимой, поскольку она необщезначима, а мы требуем, чтобы исчисление $\mathcal{H}\forall(\Omega)$ было корректным, т. е. чтобы в нём были выводимы только общезначимые формулы языка Ω. Однако данное нами определение вывода из гипотез в исчислении предикатов $\mathcal{H}\forall(\Omega)$ обеспечивает теорему о дедукции.

Теорема 2.6.8 (теорема о дедукции для исчисления предикатов $\mathcal{H}\forall(\Omega)$)**.** *Пусть* Γ *— произвольное множество формул,* A *и* B *— произвольные формулы. Тогда* $\Gamma, A \vdash B$*, если и только если* $\Gamma \vdash A \supset B$*.*

Доказательство проводится аналогично доказательству теоремы 1.3.7 индукцией по длине вывода $\Gamma, A \vdash B$. Остаётся лишь добавить в индукционный переход рассмотрение случая, когда формула B имеет вид $\forall x C$ и получается в выводе из формулы C по правилу Gen, причём переменная x не входит свободно ни в одну из формул множества $\Gamma \cup \{A\}$.

По индукционному предположению из $\Gamma, A \vdash C$ следует $\Gamma \vdash A \supset C$, откуда по правилу Gen получаем $\Gamma \vdash \forall x(A \supset C)$. Отсюда и из аксиомы вида 15 $\forall x(A \supset C) \supset (A \supset \forall x C)$ по правилу MP получаем $\Gamma \vdash A \supset \forall x C$. $\triangleleft$

Условимся, что далее в текущем разделе 2.6.2 символ Γ будет обозначать произвольное множество формул рассматриваемого языка первого порядка Ω.

Для исчисления предикатов $\mathcal{H}\forall(\Omega)$ можно дополнить *технику естественного вывода*, которой был посвящён раздел 1.3.3. Определение допустимого в исчислении $\mathcal{H}\forall(\Omega)$ правила аналогично определению допустимого в исчислении $\mathcal{H}$ правила, данному в разделе 1.3.3.

Будем считать, что в правилах, приведённых в таблице 1.5 на с. 41, теперь вместо пропозициональных формул используются формулы языка Ω. Тогда ясно, что эти правила являются допустимыми в исчислении $\mathcal{H}\forall(\Omega)$. Действительно, как только доказана теорема о дедукции для исчисления $\mathcal{H}\forall(\Omega)$, доказательства допустимости в $\mathcal{H}$ всех правил из указанной таблицы (кроме $\supset$-введения, являющегося иной формой записи теоремы о дедукции) переносятся на исчисление $\mathcal{H}\forall(\Omega)$.

Приведём допустимые в исчислении $\mathcal{H}\forall(\Omega)$ правила, которые соответствуют некоторым обычным способам математических рассуждений со словами «для любого» и «существует»:

$$\frac{\Gamma \vdash [A]_y^x}{\Gamma \vdash \forall x A}\ (\forall\text{-введение}), \qquad \frac{\Gamma \vdash \forall x A}{\Gamma \vdash [A]_t^x}\ (\forall\text{-удаление}),$$

$$\frac{\Gamma \vdash [A]_t^x}{\Gamma \vdash \exists x A}\ (\exists\text{-введение}), \qquad \frac{\Gamma, [A]_y^x \vdash B}{\Gamma, \exists x A \vdash B}\ (\exists\text{-удаление}),$$

причём 1) в $\forall$-введении переменная y (являющаяся термом) свободна для переменной x в формуле A и не входит свободно ни в одну из формул множества $\Gamma \cup \{\forall x A\}$, 2) в $\forall$-удалении и $\exists$-введении терм t свободен для переменной x в формуле A, 3) в $\exists$-удалении переменная y свободна для переменной x в формуле A и не входит свободно ни в одну из формул множества $\Gamma \cup \{\exists x A,\, B\}$.

Скажем, $\forall$-введение формализует следующее рассуждение: для доказательства утверждения вида $\forall x A$ достаточно выбрать новую переменную y и доказать, что имеет место $[A]_y^x$. $\exists$-удаление соответствует так называемому *правилу единичного выбора* (или *правилу* C — от слова «Choice»): чтобы доказать, что из $\exists x A$ следует B, обозначают через новую переменную y такой x, для которого A верно в силу $\exists x A$, и доказывают, что $[A]_y^x$ влечёт B.

Установим, что $\forall$-введение действительно допустимо: из $\Gamma \vdash [A]_y^x$ по правилу *Gen* получаем $\Gamma \vdash \forall y [A]_y^x$. В силу упражнения 2.6.2 выводима формула $\forall y [A]_y^x \supset \forall x A$. Отсюда и из $\Gamma \vdash \forall y [A]_y^x$ по правилу MP получаем $\Gamma \vdash \forall x A$.

$\forall$-удаление также допустимо: из $\Gamma \vdash \forall x A$ и аксиомы вида 13 $\forall x A \supset [A]_t^x$ по MP получаем $\Gamma \vdash [A]_t^x$.

Упражнение 2.6.9. Докажите допустимость $\exists$-введения и $\exists$-удаления. ◁

Упражнение 2.6.10. Докажите, что следующее правило, являющееся одним из вариантов $\exists$-удаления, допустимо:

$$\frac{\Gamma \vdash \exists x A; \quad \Gamma, [A]_y^x \vdash B}{\Gamma \vdash B},$$

где переменная y свободна для переменной x в формуле A и не входит свободно ни в одну из формул множества $\Gamma \cup \{\exists x A,\, B\}$. ◁

Пример 2.6.11. С помощью допустимых правил докажем, что выводима формула

$$\exists x \neg A \supset \neg \forall x A,$$

где A — произвольная формула, x — произвольная переменная. Как и для исчисления высказываний (см. раздел 1.3.3), такое доказательство удобно вести, пользуясь правилами снизу вверх:

1) $\vdash \exists x \neg A \supset \neg \forall x A$ из 2 по $\supset$-введению;

2) $\exists x \neg A \vdash \neg \forall x A$ из 3 по $\exists$-удалению (в качестве переменной y из формулировки этого допустимого правила мы берём переменную x);

3) $\neg A \vdash \neg \forall x A$ из 4, 5 по $\neg$-введению;

4) $\neg A, \forall x A \vdash A$ из 6 по $\forall$-удалению;

5) $\neg A, \forall x A \vdash \neg A$ верно;

6) $\neg A, \forall x A \vdash \forall x A$ верно. $\triangleleft$

Следующая теорема оказывается полезной для доказательства выводимости в исчислении предикатов.

Теорема 2.6.12. *Любой частный случай пропозициональной тавтологии является выводимой в исчислении $\mathcal{H}\forall(\Omega)$ формулой.*

Доказательство. Пусть предикатная формула B получена в результате одновременной замены всех вхождений пропозициональных переменных $p_1, \ldots, p_n$ в пропозициональную тавтологию C на предикатные формулы $B_1, \ldots, B_n$ соответственно (см. определение 2.4.9 на с. 77). В силу теоремы 1.3.25 о полноте исчисления $\mathcal{H}$, имеем вывод пропозициональной тавтологии C в исчислении $\mathcal{H}$. Заменим в этом выводе все вхождения $p_1, \ldots, p_n$ на $B_1, \ldots, B_n$ соответственно, а также заменим вхождения всех других пропозициональных переменных в этот вывод на любую заранее фиксированную предикатную формулу. Очевидно, в результате этих замен вывод формулы C в исчислении $\mathcal{H}$ преобразуется в вывод формулы B в исчислении $\mathcal{H}\forall(\Omega)$. $\triangleleft$

Теорема 2.6.12 даёт нам целый ряд верных утверждений о выводимости. Например, если $\Gamma \vdash \neg A \supset B$, то $\Gamma \vdash \neg B \supset A$ (где A и B — любые формулы языка Ω). Действительно, формула $(\neg A \supset B) \supset (\neg B \supset A)$ является частным случаем пропозициональной тавтологии, следовательно, по теореме 2.6.12 эта формула выводима. Из выводимости этой формулы и $\Gamma \vdash \neg A \supset B$ по правилу MP получаем $\Gamma \vdash \neg B \supset A$, что и требовалось. Только что доказанное утверждение о выводимости можно записать в виде допустимого (в исчислении $\mathcal{H}\forall(\Omega)$) правила

$$\frac{\Gamma \vdash \neg A \supset B}{\Gamma \vdash \neg B \supset A}, \qquad (*)$$

которое мы назовём *правилом контрапозиции.*

По теореме 2.6.8 о дедукции $\Gamma \vdash \neg A \supset B$ равносильно $\Gamma, \neg A \vdash B$, а $\Gamma \vdash \neg B \supset A$ равносильно $\Gamma, \neg B \vdash A$, поэтому $(*)$ можно переписать в виде допустимого правила

$$\frac{\Gamma, \neg A \vdash B}{\Gamma, \neg B \vdash A},$$

которое мы будем называть *правилом контрапозиции.*

Аналогично устанавливается, что допустимы правила

$$\frac{\Gamma, A \vdash \neg B}{\Gamma, B \vdash \neg A}, \quad \frac{\Gamma, A \vdash B}{\Gamma, \neg B \vdash \neg A}, \quad \frac{\Gamma, \neg A \vdash \neg B}{\Gamma, B \vdash A},$$

которые мы будем называть так же — *правилами контрапозиции.*

Пример 2.6.13. Докажем, что для любой формулы A и какой угодно переменной x имеет место

$$\vdash \neg\forall x A \supset \exists x \neg A.$$

1) $\vdash \neg\forall x A \supset \exists x \neg A$ из 2 по $\supset$-введению;

2) $\neg\forall x A \vdash \exists x \neg A$ из 3 по правилу контрапозиции;

3) $\neg\exists x \neg A \vdash \forall x A$ из 4 по $\forall$-введению;

4) $\neg\exists x \neg A \vdash A$ из 5 по правилу контрапозиции;

5) $\neg A \vdash \exists x \neg A$ из 6 по $\exists$-введению;

6) $\neg A \vdash \neg A$ верно. $\triangleleft$

Упражнение 2.6.14. В разделе 2.4 в пунктах 1–12 приведены пары равносильных формул. Из каждой пары равносильных формул, соединённых знаком $\sim$, получается общезначимая формула путём замены $\sim$ на $\equiv$. Докажите выводимость формул, полученных таким образом, а также докажите выводимость формул из пунктов 13, 14 того же раздела. $\triangleleft$

Замечание 2.6.15. (Будет использовано нами в следующем разделе 2.6.3.) Очевидно, $\Gamma \vdash A$ равносильно тому, что существует конечное число формул $G_1, \ldots, G_n$ из множества Γ таких, что

$$G_1, \ldots, G_n \vdash A. \tag{a}$$

По теореме 2.6.8 о дедукции (a) равносильно

$$\vdash G_1 \supset (G_2 \supset (\ldots (G_n \supset A) \ldots)), \tag{b}$$

что в свою очередь равносильно (см. следующее упражнение)

$$\vdash G_1 \wedge \ldots \wedge G_n \supset A. \tag{c}$$

По теореме 2.6.8 о дедукции (c) равносильно

$$G_1 \wedge \ldots \wedge G_n \vdash A.$$

$\triangleleft$

Упражнение 2.6.16. Докажите равносильность (b) и (c) из предыдущего замечания. *Указание.* Воспользуйтесь теоремой 2.6.12. $\triangleleft$

2.6.3. Полнота исчисления

Пусть Ω — произвольный язык первого порядка, который мы будем рассматривать в разделе 2.6.3, если не оговорено иначе. В данном разделе мы докажем теорему о полноте исчисления предикатов $\mathcal{H}\forall(\Omega)$: всякая общезначимая формула языка Ω выводима в этом исчислении. Эту теорему мы получим как следствие так называемой теоремы о существовании модели (см. теорему 2.6.21 ниже). В целом ход доказательства теоремы 2.6.21 будет напоминать ход доказательства её пропозиционального аналога — теоремы 1.3.19. Предварительно дадим несколько определений.

Определение 2.6.17. Множество Γ замкнутых формул языка Ω называется *непротиворечивым*, если не существует формулы A языка Ω такой, что $\Gamma \vdash A$ и $\Gamma \vdash \neg A$. Иначе Γ называется *противоречивым*. $\triangleleft$

Определение 2.6.18. Множество Γ замкнутых формул языка Ω называется *полным*, если для любой замкнутой формулы A языка Ω имеем $\Gamma \vdash A$ или $\Gamma \vdash \neg A$. Иначе Γ называется *неполным*. $\triangleleft$

Определение 2.6.19. Пусть Γ — произвольное множество замкнутых формул языка Ω. Интерпретацию языка Ω, в которой истинны все формулы множества Γ, называют *моделью множества* Γ. Если существует модель множества Γ, то говорят, что Γ *имеет модель* (или Γ *совместно*). $\triangleleft$

Следующая теорема доказывается аналогично теореме 1.3.18 с использованием обобщённой теоремы 2.6.4 о корректности исчисления $\mathcal{H}\forall(\Omega)$.

Теорема 2.6.20. *Всякое множество замкнутых формул, имеющее модель, непротиворечиво.*

Наша цель — доказать теорему 2.6.21, обратную к теореме 2.6.20.

Теорема 2.6.21 (о существовании модели)**.** *Всякое непротиворечивое множество* Γ *замкнутых формул имеет модель.*

Мы докажем эту теорему, опираясь на следующие леммы.

Лемма 2.6.22. *Пусть* Γ *— непротиворечивое множество формул языка* Ω*,* A *— замкнутая формула языка* Ω*. Тогда хотя бы одно из множеств* $\Gamma \cup \{A\}$ *или* $\Gamma \cup \{\neg A\}$ *непротиворечиво.*

Доказательство полностью аналогично доказательству леммы 1.3.20, но мы приведём его здесь для удобства ссылок на него из следующего замечания 2.6.23.

Предположим, что оба множества $\Gamma \cup \{A\}$ и $\Gamma \cup \{\neg A\}$ противоречивы. Тогда найдутся такие формулы B и C такие, что $\Gamma, A \vdash B$, $\Gamma, A \vdash \neg B$ и $\Gamma, \neg A \vdash C$, $\Gamma, \neg A \vdash \neg C$.

По $\neg$-введению из $\Gamma, A \vdash B$ и $\Gamma, A \vdash \neg B$ получаем $\Gamma \vdash \neg A$. Аналогично из $\Gamma, \neg A \vdash C$ и $\Gamma, \neg A \vdash \neg C$ получаем $\Gamma \vdash \neg\neg A$.

Итак, имеем $\Gamma \vdash \neg A$ и $\Gamma \vdash \neg\neg A$, что противоречит непротиворечивости Γ. Поэтому сделанное предположение неверно. $\triangleleft$

Замечание 2.6.23. (См. также замечание 2.5.7 о конструктивных доказательствах.) Для конструктивного доказательсльта дизъюнкции требуется указать явный способ (алгоритм) выяснения верного члена этой дизъюнкции. (Поэтому в конструктивном доказательстве, вообще говоря, неприемлем закон исключённого третьего $B \vee \neg B$.)

Покажем, что доказательство леммы 2.6.22 неконструктивно: оно не даёт способа выяснить, какое из множеств $\Gamma \cup \{A\}$ или $\Gamma \cup \{\neg A\}$ непротиворечиво.

Предположим, что данное в условии леммы 2.6.22 множество формул Γ пусто. Тогда формулировка этой леммы говорит только о словах, являющихся формулами. Но всё равно доказательство леммы 2.6.22 не отвечает на вопрос, какое же из множеств $\{A\}$ или $\{\neg A\}$ непротиворечиво. Может, причина этого в том, что мы предложили не самое удачное доказательство данной леммы? В этом доказательстве устанавливается (лишь с помощью $\neg$-введения), что противоречивость $\{A\}$ влечёт $\vdash \neg A$. Очевидно, что $\vdash \neg A$

влечёт противоречивость $\{A\}$. Следовательно, проверка того, является ли $\{A\}$ противоречивым или нет, равносильным образом сводится к проверке того, является ли $\neg A$ выводимой или нет соответственно. Поэтому, возможно, причина в том, что мы не можем предложить алгоритм, который бы для любой предикатной формулы определял, выводима она или нет. В разделе 5.5.2 мы увидим (при надлежащем уточнении постановки задачи), что на самом деле такого алгоритма не существует. ◁

Лемма 2.6.24 (лемма Линденбаума)**.** *Всякое непротиворечивое множество замкнутых формул языка Ω содержится в некотором непротиворечивом полном множестве замкнутых формул того же языка.*

Доказательство. Аналогично доказательству леммы 1.3.21 по очереди просматриваем все замкнутые формулы языка Ω и на каждом шаге, используя лемму 2.6.22, добавляем в строящееся множество либо саму замкнутую формулу, либо её отрицание. ◁

Лемма 2.6.25. *Пусть $\Omega \leftrightharpoons FOL(PS, FS, \#)$; Γ — непротиворечивое множество замкнутых формул языка Ω; $\exists x A$ — замкнутая формула языка Ω; $\Gamma \vdash \exists x A$; c — слово, не принадлежащее множеству $PS \cup FS$; $\Omega_c \leftrightharpoons FOL(PS, FS \cup \{c\}, \#_c)$, причём $\#_c(c) = 0$ (т. е. c есть предметная константа языка Ω_c) и отображения $\#_c$ и $\#$ совпадают на $PS \cup FS$. Тогда непротиворечиво множество $\Gamma \cup \{[A]_c^x\}$ формул языка Ω_c.*

Доказательство. Предположим, что множество $\Gamma \cup \{[A]_c^x\}$ противоречиво, т. е. найдётся формула B языка Ω_c такая, что $\Gamma, [A]_c^x \vdash B$ и $\Gamma, [A]_c^x \vdash \neg B$. Отсюда по $\neg$-введению получаем $\Gamma \vdash \neg[A]_c^x$, причём имеем вывод формулы $\neg[A]_c^x$ из Γ в исчислении $\mathcal{H}\forall(\Omega_c)$. Заменив c всюду в этом выводе на новую (не входящую в этот вывод) предметную переменную z, получим вывод формулы $\neg[A]_z^x$ из Γ в исчислении $\mathcal{H}\forall(\Omega)$. Для $\Gamma \vdash \neg[A]_z^x$ необходимо и достаточно (см. замечание 2.6.15 на с. 89) существования конечного множества $\Gamma_0 \subseteq \Gamma$ такого, что $\gamma \vdash \neg[A]_z^x$, где γ — конъюнкция всех формул множества Γ_0, переменная z не входит в γ. По правилу контрапозиции из $\gamma \vdash \neg[A]_z^x$ следует $[A]_z^x \vdash \neg\gamma$. По $\exists$-удалению имеем $\exists z[A]_z^x \vdash \neg\gamma$. Отсюда по правилу контрапозиции получаем $\gamma \vdash \neg\exists z[A]_z^x$. Следовательно, $\Gamma \vdash \neg\exists z[A]_z^x$.

По условию $\Gamma \vdash \exists x A$, откуда (см. пример 2.6.1 на с. 84) $\Gamma \vdash \exists z[A]_z^x$. Итак, мы получили $\Gamma \vdash \neg\exists z[A]_z^x$ и $\Gamma \vdash \exists z[A]_z^x$, что противоречит непротиворечивости Γ. Поэтому предположение неверно, а множество $\Gamma \cup \{[A]_c^x\}$ непротиворечиво. ◁

Определение 2.6.26. Множество Γ замкнутых формул языка Ω назовём *экзистенциально полным*, если для всякой замкнутой формулы $\exists x A$ этого языка такой, что $\Gamma \vdash \exists x A$, найдётся замкнутый терм t, для которого $\Gamma \vdash [A]_t^x$. ◁

Лемма 2.6.27. *Пусть $\Omega \leftrightharpoons FOL(PS, FS, \#)$; Γ — непротиворечивое множество замкнутых формул языка Ω. Тогда существует счётный язык CS, не пересекающийся с множеством $PS \cup FS$, и существует непротиворечивое, полное, экзистенциально полное и содержащее Γ множество*

Γ' замкнутых формул языка $\Omega_{CS} \rightleftharpoons FOL(PS, FS \cup CS, \#_{CS})$, причём $\#_{CS}$ равно 0 на CS (т. е. каждый элемент множества CS есть предметная константа языка Ω_{CS}) и отображения $\#_{CS}$ и $\#$ совпадают на $PS \cup FS$.

Д о к а з а т е л ь с т в о. Поскольку язык Ω конечен или счётен, то и множество всех замкнутых формул языка Ω вида $\exists x A$, которые выводимы из Γ, является конечным или счётным. Пусть $\exists x_i A_i$ $(i \in I)$ — все такие формулы, где $I = \{1, 2, \ldots, k\}$ для некоторого $k \in \mathbb{N}$ или $I = \mathbb{N}_+$. Пусть также $CS^{(1)} \rightleftharpoons \{c_1, c_2, \ldots\}$ — счётный язык, не пересекающийся с множеством $PS \cup FS$. (Слова этого языка будут предметными константами, которыми мы расширим сигнатуру языка Ω и которые войдут в множество CS.)

Положим $\Gamma_0^{(1)} \rightleftharpoons \Gamma$ и для каждого $i \in I$

$$\Gamma_i^{(1)} \rightleftharpoons \Gamma_{i-1}^{(1)} \cup \{[A_i]_{c_i}^{x_i}\}.$$

По лемме 2.6.25 для каждого $i \in I$ $\Gamma_i^{(1)}$ является непротиворечивым множеством замкнутых формул языка $\Omega_i^{(1)} \rightleftharpoons FOL(PS, FS \cup CS_i^{(1)}, \#_{CS_i^{(1)}})$, где $CS_i^{(1)} \rightleftharpoons \{c_1, \ldots, c_i\}$, отображение $\#_{CS_i^{(1)}}$ определено аналогично $\#_{CS}$, но применительно к множеству $CS_i^{(1)}$ вместо CS. Далее, положим

$$\Gamma_I^{(1)} \rightleftharpoons \Gamma_0^{(1)} \cup (\cup_{i \in I} \Gamma_i^{(1)}).$$

$\Gamma_I^{(1)}$ является множеством замкнутых формул языка

$$\Omega^{(1)} \rightleftharpoons FOL(PS, FS \cup CS^{(1)}, \#^{(1)}),$$

где отображение $\#^{(1)}$ определено аналогично $\#_{CS}$, но применительно к множеству $CS^{(1)}$ вместо CS.

Если бы $\Gamma_I^{(1)}$ было противоречивым, то нашлась бы такая формула A языка $\Omega^{(1)}$, что $\Gamma_I^{(1)} \vdash_{\mathcal{H}\forall(\Omega^{(1)})} A$ и $\Gamma_I^{(1)} \vdash_{\mathcal{H}\forall(\Omega^{(1)})} \neg A$. Тогда, поскольку в вывод входит лишь конечное число формул из $\Gamma_I^{(1)}$, нашлось бы i, при котором мы бы имели выводы $\Gamma_i^{(1)} \vdash_{\mathcal{H}\forall(\Omega^{(1)})} A$ и $\Gamma_i^{(1)} \vdash_{\mathcal{H}\forall(\Omega^{(1)})} \neg A$. Заменив в двух последних выводах каждую предметную константу из $CS^{(1)} \setminus CS_i^{(1)}$ на новую предметную переменную, для некоторой формулы A' мы бы получили выводы $\Gamma_i^{(1)} \vdash_{\mathcal{H}\forall(\Omega_i^{(1)})} A'$ и $\Gamma_i^{(1)} \vdash_{\mathcal{H}\forall(\Omega_i^{(1)})} \neg A'$, что противоречило бы непротиворечивости $\Gamma_i^{(1)}$. Поэтому $\Gamma_I^{(1)}$ непротиворечиво.

По лемме 2.6.24 существует полное непротиворечивое множество $\Gamma^{(1)}$ замкнутых формул языка $\Omega^{(1)}$, содержащее $\Gamma_I^{(1)}$. Вспомним, что $\Gamma \subseteq \Gamma_I^{(1)}$, откуда $\Gamma \subseteq \Gamma^{(1)}$.

Взяв язык $\Omega^{(1)}$ и множество $\Gamma^{(1)}$ замкнутых формул этого языка в качестве Ω и Γ соответственно и повторив вышеописанный процесс, получим полное непротиворечивое множество $\Gamma^{(2)}$ замкнутых формул языка $\Omega^{(2)}$, содержащее $\Gamma^{(1)}$. Действуя так и далее, для каждого $j \in \mathbb{N}_+$ получим полное непротиворечивое множество $\Gamma^{(j)}$ замкнутых формул языка $\Omega^{(j)}$, содержащее Γ.

Положим $\Gamma' \rightleftharpoons \cup_{j \in \mathbb{N}_+} \Gamma^{(j)}$. Γ' является множеством замкнутых формул языка Ω_{CS}, где $CS \rightleftharpoons \cup_{j \in \mathbb{N}_+} CS^{(j)}$ — множество предметных констант, которыми была расширена сигнатура языка Ω при определении языка Ω_{CS}.

CS счётно как объединение счётного числа счётных множеств. По построению Γ' содержит Γ. Γ' непротиворечиво, что устанавливается с помощью рассуждений, аналогичных рассуждениям о непротиворечивости $\Gamma_I^{(1)}$, приведённым в этом доказательстве выше. Γ' полно, так как любая замкнутая формула языка Ω_{CS} есть формула языка $\Omega^{(j)}$ при некотором j, а множество $\Gamma^{(j)}$ формул языка $\Omega^{(j)}$ содержится в Γ' и является полным. Γ' экзистенциально полно, поскольку любая замкнутая формула $\exists x A$ языка Ω_{CS} такая, что $\Gamma' \vdash_{\mathcal{H}\forall(\Omega_{CS})} \exists x A$, есть формула языка $\Omega^{(j)}$ при некотором j, а при некотором $c \in CS$ формула $[A]_c^x$ принадлежит множеству $\Gamma^{(j+1)} \subseteq \Gamma'$. $\triangleleft$

Доказательство теоремы 2.6.21. В силу предыдущей леммы достаточно доказать существование искомой модели, считая, что множество Γ замкнутых формул языка $\Omega \leftrightharpoons FOL(PS, FS, \#)$ непротиворечиво, полно и экзистенциально полно, а множество FS счётно.

Определим интерпретацию $M \leftrightharpoons \langle \mathcal{D}, \mu \rangle$ языка Ω. Пусть носителем $\mathcal{D}$ будет множество всех замкнутых термов языка Ω (это множество, очевидно, счётно). Для каждого n-местного функционального символа $f \in FS$ $(n > 0)$ определим функцию $\mu(f)$ так, что для любых $t_1, \ldots, t_n \in \mathcal{D}$ $\mu(f)(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$; для 0-местного функционального символа f положим $\mu(f) = f$. Для каждого n-местного предикатного символа $P \in PS$ $(n > 0)$ определим предикат $\mu(P)$ так, что для любых $t_1, \ldots, t_n \in \mathcal{D}$ $\mu(P)(t_1, \ldots, t_n) = 1$, если $\Gamma \vdash P(t_1, \ldots, t_n)$, и $\mu(P)(t_1, \ldots, t_n) = 0$ иначе (т. е. если $\Gamma \vdash \neg P(t_1, \ldots, t_n)$). Определение $\mu(P)$ для 0-местного предикатного символа P аналогично предыдущему: $\mu(P) = 1$, если $\Gamma \vdash P$, и $\mu(P) = 0$ иначе.

Чтобы доказать, что любая (замкнутая) формула из Γ истинна в интерпретации M, достаточно показать, что для любой замкнутой формулы A языка Ω верно: $M \vDash A$ тогда и только тогда, когда $\Gamma \vdash A$. Воспользуемся индукцией по числу вхождений логических символов в формулу A.

База индукции (формула A атомарна) и индукционный переход в случаях 1–4, когда формула A в качестве главного логического символа имеет одну из четырёх связок ($\neg$, $\wedge$, $\vee$, $\supset$), доказываются аналогично тому, как это было сделано в доказательстве теоремы 1.3.19. Докажем лишь индукционный переход в случаях 5–6, когда формула A в качестве главного логического символа имеет квантор.

5. Пусть A имеет вид $\exists x B$. Если $\Gamma \vdash \exists x B$, то в силу экзистенциальной полноты множества Γ найдётся замкнутый терм t такой, что $\Gamma \vdash [B]_t^x$. Тогда по индукционному предположению $M \vDash [B]_t^x$, откуда следует $M \vDash \exists x B$.

Если $M \vDash \exists x B$, то существует замкнутый терм t такой, что $M \vDash [B]_t^x$. Тогда по индукционному предположению $\Gamma \vdash [B]_t^x$. Отсюда и из аксиомы вида 14 $[B]_t^x \supset \exists x B$ по правилу вывода MP получаем $\Gamma \vdash \exists x B$.

6. Пусть, наконец, A имеет вид $\forall x B$. Если $\Gamma \vdash \forall x B$, то для любого замкнутого терма t из аксиомы вида 13 $\forall x B \supset [B]_t^x$ и $\Gamma \vdash \forall x B$ по правилу MP получаем $\Gamma \vdash [B]_t^x$. Тогда по индукционному предположению $M \vDash [B]_t^x$ для любого замкнутого терма t, что и означает $M \vDash \forall x B$.

Пусть теперь $M \vDash \forall x B$. Предположим, что $\Gamma \nvdash \forall x B$. Тогда в силу полноты Γ имеем $\Gamma \vdash \neg \forall x B$. Отсюда (см. пример 2.6.13 на с. 88) $\Gamma \vdash \exists x \neg B$. В силу экзистенциальной полноты множества Γ найдётся замкнутый терм t такой, что $\Gamma \vdash [\neg B]_t^x$. Согласно уже разобранному случаю, когда формула имеет отрицание в качестве главного логического символа, получаем

$M \models [\neg B]^x_t$. Следовательно, $M \not\models [B]^x_t$, что противоречит $M \models \forall x B$. Таким образом, предположение $\Gamma \not\vdash \forall x B$ неверно, и мы имеем $\Gamma \vdash \forall x B$. ◁

В доказательстве теоремы 2.6.21 о существовании модели построена модель со счётным носителем, поэтому имеет место следующее усиление этой теоремы.

Теорема 2.6.28 (о существовании модели)**.** *Всякое непротиворечивое множество замкнутых формул имеет счётную модель.*

Теорема 2.6.29 (теорема Лёвенгейма-Скулема)**.** *Если множество замкнутых формул имеет какую-нибудь модель, то оно имеет счётную модель.*

Доказательство. По теореме 2.6.20 множество, имеющее модель, непротиворечиво. Тогда по предыдущей теореме 2.6.28 оно имеет счётную модель. ◁

Приведённые ниже 3 следствия теоремы 2.6.21 о существовании модели доказываются аналогично их пропозициональным вариантам — теоремам 1.3.23, 1.3.24 и 1.3.25.

Теорема 2.6.30 (теорема о компактности (для логики предикатов))**.** *Пусть* Γ *— множество замкнутых формул. Тогда если всякое конечное подмножество множества* Γ *имеет модель, то* Γ *имеет модель.*

Теорема 2.6.31 (обобщённая теорема о полноте исчисления предикатов $\mathcal{H}\forall(\Omega)$)**.** *Пусть* Γ *— множество замкнутых формул,* A *— формула. Тогда если* $\Gamma \models A$*, то* $\Gamma \vdash A$*.*

Чтобы доказательство предыдущей теоремы прошло аналогично доказательству её пропозиционального варианта (теоремы 1.3.24), рассматриваемая формула A должна быть замкнута. Действительно, формулу A можно считать замкнутой в силу утверждения, сформулированного в следующем упражнении.

Упражнение 2.6.32. Пусть Γ — множество замкнутых формул, A — формула. Докажите, что тогда а) $\Gamma \models A$ равносильно $\Gamma \models \mathbb{\forall} A$; и б) $\Gamma \vdash A$ равносильно $\Gamma \vdash \mathbb{\forall} A$. ◁

Теорема 2.6.33 (теорема Гёделя о полноте исчисления предикатов $\mathcal{H}\forall(\Omega)$)**.** *Всякая общезначимая формула выводима в исчислении* $\mathcal{H}\forall(\Omega)$*, т. е. для любой формулы* A*, если* $\models A$*, то* $\vdash A$*.*

Упражнение 2.6.34. Всякая ли формула, истинная в любой счётной интерпретации, общезначима? ◁

Упражнение 2.6.35. Всякая ли формула, истинная в любой конечной интерпретации, общезначима? *Указание.* Рассмотрите формулу

$$A \leftrightharpoons \neg(A_1 \wedge A_2 \wedge A_3),$$

где

$$A_1 \leftrightharpoons \forall x \forall y \forall z (x < y \wedge y < z \supset x < z),$$
$$A_2 \leftrightharpoons \forall x \neg (x < x), \qquad A_3 \leftrightharpoons \forall x \exists y (x < y),$$

$<$ — двухместный предикатный символ. ◁

2.6.4. Синтаксическая эквивалентность формул

В начале раздела 2.6.1 мы отметили недостатки неформальных доказательств общезначимости предикатных формул. Конечно, то же самое относится и к неформальным доказательствам семантической эквивалентности (равносильности) формул. В настоящем разделе мы введём другое понятие эквивалентности формул, которое основано на понятии формального доказательства (вывода). Также это новое понятие эквивалентности формул удовлетворит следующую потребность. Мы встречались с ситуациями, когда можно рассматривать формулы с точностью до некоторого отношения эквивалентности, например, отношения конгруэнтности или отношения семантической эквивалентности. В дальнейшем[11] нам иногда будет удобно рассматривать формулы с точностью до эквивалентности с точки зрения выводимости.

Формулы A и B называются *синтаксически эквивалентными* (*выводимо* или *доказуемо эквивалентными*), если $\vdash A \equiv B$. Факт синтаксической эквивалентности формул A и B обозначим через $A \simeq B$.

$A \equiv B$ является сокращением формулы, согласно которому $A \simeq B$ означает $\vdash (A \supset B) \wedge (B \supset A)$. В силу упражнения 1.3.13 на с. 43, для установления $A \simeq B$ необходимо и достаточно установить, что $A \vdash B$ и $B \vdash A$. Поэтому синтаксически эквивалентные формулы выводимы или невыводимы одновременно.

Упражнение 2.6.36. Докажите, что отношение $\simeq$ является отношением эквивалентности на множестве всех формул рассматриваемого языка первого порядка. ◁

Вспомним, что для любых формул A и B справедливо: $A \sim B$, если и только если $\vDash A \equiv B$. В силу теорем о корректности и полноте исчисления предикатов имеем: $A \sim B$, если и только если $A \simeq B$. Этот факт позволяет переформулировать многие утверждения о семантической эквивалентности формул в виде утверждений о синтаксической эквивалентности формул. Например, теорема 2.4.12 о семантически эквивалентной замене вместе с этим фактом влечёт следующую теорему.

Теорема 2.6.37 (о синтаксически эквивалентной замене)**.** *Пусть A, B и B' — формулы, A' есть результат замены некоторого вхождения B в A на B'. Тогда если $B \simeq B'$, то $A \simeq A'$.*

Дадим доказательство теоремы о синтаксически эквивалентной замене, которое не опирается на семантику языка первого порядка, в частности, на понятие интерпретации языка первого порядка, а значит, и на понятие семантической эквивалентности формул.

Лемма 2.6.38. *Пусть A, B и C — произвольные формулы, x — произвольная предметная переменная. Если $A \simeq B$, то*

(1) $\neg A \simeq \neg B$,

(2) $A \wedge C \simeq B \wedge C$, $C \wedge A \simeq C \wedge B$,

(3) $A \vee C \simeq B \vee C$, $C \vee A \simeq C \vee B$,

[11] См. в текущем разделе 2.6.4 подраздел, посвящённый сокращённым обозначениям формул, и главу 3, в которой такие сокращения интенсивно используются.

(4) $A \supset C \simeq B \supset C$, $C \supset A \simeq C \supset B$,

(5) $\forall x A \simeq \forall x B$,

(6) $\exists x A \simeq \exists x B$.

Доказательство. (1) Формула $(A \equiv B) \supset (\neg A \equiv \neg B)$ является частным случаем пропозициональной тавтологии, следовательно, по теореме 2.6.12 она выводима. Применив правило MP к этой формуле и к выводимой по условию формуле $A \equiv B$, получим, что выводима $\neg A \equiv \neg B$. Выводимость последней формулы и означает, что $\neg A \simeq \neg B$. (Заметим, что мы использовали понятие пропозициональной тавтологии, относящееся к семантике языка логики высказываний. Можно избежать опоры и на семантику языка логики высказываний, доказав выводимость $\neg A \equiv \neg B$ из $A \equiv B$ с помощью техники естественного вывода, см. разделы 1.3.3 и 2.6.2.)

Доказательство пунктов (2)–(6) остаётся в качестве упражнения. ◁

Упражнение 2.6.39. Докажите пункты (2)–(6) из леммы 2.6.38. ◁

Доказательство теоремы 2.6.37. Пусть $B \simeq B'$. Доказательство того, что $A \simeq A'$, проведём индукцией по построению формулы A.

База индукции. A является атомарной формулой. Тогда B совпадает с A, и B' совпадает с A', поэтому $A \simeq A'$.

Индукционный переход. Предположим, что утверждение теоремы справедливо для формул C и D, взятых в качестве A.

Рассмотрим формулу A, имеющую вид $\neg C$. Если B совпадает с A, то доказываемое утверждение очевидно (как и при установлении базы индукции). Иначе B входит в C. Тогда A' есть $\neg C'$, где C' есть результат замены некоторого вхождения B в C на B'. По индукционному предположению $C \simeq C'$. Следовательно, по лемме 2.6.38 $A \simeq A'$.

Рассмотрим формулу A, имеющую вид $C \wedge D$. Если B совпадает с A, то доказываемое утверждение очевидно. Иначе при получении A' из A происходит замена некоторого вхождения B в C или в D. Пусть для определённости происходит замена некоторого вхождения B в C. Тогда A' есть $C' \wedge D$, где C' есть результат замены некоторого вхождения B в C на B'. По индукционному предположению $C \simeq C'$. Следовательно, по лемме 2.6.38 $A \simeq A'$.

Случаи, когда A имеет вид $C \vee D$, $C \supset D$, $\forall x C$ и $\exists x C$, рассматриваются аналогично предыдущим случаям. ◁

Отметим, что из теоремы о синтаксически эквивалентной замене и теорем о корректности и полноте исчисления предикатов следует теорема о семантически эквивалентной замене.

Упражнение 2.6.40. Не опираясь на семантику языка первого порядка, докажите, что конгруэнтные формулы синтаксически эквивалентны. *Указание.* Используйте пример 2.6.1, следующее за ним упражнение 2.6.2 и теорему 2.6.37. ◁

Упражнение 2.6.41. Не опираясь на семантику языка первого порядка, докажите, что для любой формулы A существует формула, находящаяся в ПНФ и синтаксически эквивалентная формуле A. ◁

Сокращённые обозначения формул

Рассмотрим введённое ранее сокращение формулы языка элементарной арифметики: $x \leqslant y \leftrightharpoons \exists z(y = x + z)$. Предположим, что мы используем это сокращение с какими-то термами вместо x и y, например, так: $u \leqslant z + z$. Чтобы не возникла коллизия переменных (см. раздел 2.3), иногда требуется переименовать связанную переменную z в формуле $\exists z(y = x + z)$ прежде, чем производить подстановку термов вместо x и y. Переименование связанных переменных в формуле было определено нами как нахождение формулы, конгруэнтной данной (см. раздел 2.3.2).

В тех случаях, когда нас интересует семантика формул, мы можем переименовать связанные переменные, и это не повлияет на истинностные значения формул в силу того, что конгруэнтные формулы семантически эквивалентны (см. пункт 17 в разделе 2.4). А в тех случаях, когда нас интересует выводимость формул, мы также можем переименовать связанные переменные, и это не повлияет на выводимость формул в силу того, что конгруэнтные формулы синтаксически эквивалентны (см. упражнение 2.6.40). Оперируя сокращениями формул, мы будем подразумевать, что при необходимости происходит переименование связанных переменных.

2.7. Секвенциальное исчисление предикатов

Сказанное в разделе 1.3.5 о поиске вывода в исчислении $\mathcal{H}$ высказываний гильбертовского типа относится и к исчислению $\mathcal{H}\forall(\Omega)$ предикатов гильбертовского типа (где Ω — произвольный язык первого порядка). Для поиска вывода в исчислении $\mathcal{H}\forall(\Omega)$ можно использовать алгоритм Британского музея, если предположить, что имеется вспомогательный алгоритм, который по любому слову в алфавите языка Ω проверяет, принадлежит ли это слово языку Ω или нет. Исчисления $\mathcal{H}$ и $\mathcal{H}\forall(\Omega)$ хорошо подходят для получения заключений из аксиом и ранее полученных формул, но неудобны для поиска вывода заранее заданной формулы. Поэтому сохраняется та же мотивировка для рассмотрения исчисления, ориентированного на целенаправленный поиск вывода заданной предикатной формулы.

В данном разделе мы рассмотрим секвенциальное исчисление предикатов. В этом исчислении правила вывода, вводящие логические связки, основываются на таких же семантических соображениях, что и правила введения связок в секвенциальном исчислении высказываний (см. раздел 1.4). Сначала мы покажем, как семантические соображения приводят нас к правилам вывода для кванторов, а затем сформулируем секвенциальное исчисление предикатов.

2.7.1. Поиск контрпримера для формулы с кванторами

Предикатная формула A общезначима тогда и только тогда, когда не существует интерпретации и оценки, в которых A ложна. Упорядоченную пару, состоящую из интерпретации M и оценки ν, назовём *контрпримером для формулы* A, если $M, \nu \nvDash A$. Покажем, как можно искать контрпримеры для предикатных формул с кванторами.

Пример 2.7.1. Попробуем найти какой-нибудь контрпример, состоящий из интерпретации $M \leftrightharpoons \langle \mathcal{D}, \mu \rangle$ и оценки ν, для формулы

$$\exists x P(x) \vee Q(z) \supset \forall x (P(x) \vee Q(z)).$$

Как и в пропозициональном случае, будем записывать формулы, которые должны быть ложны в искомом контрпримере, справа от знака $\to$, а формулы, которые должны быть истинны — слева. Изначально имеем

$$\to \exists x P(x) \vee Q(z) \supset \forall x (P(x) \vee Q(z)),$$

что равносильно

$$\exists x P(x) \vee Q(z) \to \forall x (P(x) \vee Q(z)).$$

Формула $\forall x (P(x) \vee Q(z))$ должна быть ложна при искомых интерпретации M и оценке ν; для этого необходимо и достаточно, чтобы формула $P(y_1) \vee Q(z)$, где y_1 — новая (т. е. не встречавшаяся ранее в текущем рассуждении) предметная переменная, была ложна при интерпретации M и некоторой оценке ν', отличающейся от оценки ν разве лишь значением переменной y_1. Поэтому вместо контрпримера, в котором ложна $\forall x (P(x) \vee Q(z))$, подойдёт контрпример, в котором ложна $P(y_1) \vee Q(z)$, причём оба таких контрпримера существуют или не существуют одновременно. Интуитивно говоря, формула $\forall x (P(x) \vee Q(z))$ ложна, если и только если ложна формула $P(y_1) \vee Q(z)$, где y_1 — новая предметная переменная, обозначающая ещё неопределённый элемент носителя искомой интерпретации. Итак, на данном шаге поиска имеем

$$\exists x P(x) \vee Q(z) \to P(y_1) \vee Q(z),$$

что равносильно

$$\exists x P(x) \vee Q(z) \to P(y_1),\ Q(z).$$

В свою очередь, это равносильно

$$\exists x P(x) \to P(y_1),\ Q(z) \quad \text{или} \quad Q(z) \to P(y_1),\ Q(z),$$

здесь поиск контрпримера разделился на два пути.

На втором пути

$$Q(z) \to P(y_1),\ Q(z)$$

контрпримера не найти, поскольку формула $Q(z)$, стоящая слева и справа от $\to$, ни в каком контрпримере не может быть истинной и ложной одновременно.

Исследуем первый путь

$$\exists x P(x) \to P(y_1),\ Q(z). \qquad (*)$$

Рассуждая аналогично тому, как мы рассуждали о стоящей справа от $\to$ формуле с главным квантором $\forall$, получаем: в качестве контрпримера, в котором истинна формула $\exists x P(x)$, необходимо и достаточно найти контрпример, в котором истинна формула $P(y_2)$, где y_2 — новая предметная переменная. Таким образом, получаем

$$P(y_2) \to P(y_1),\ Q(z). \qquad (**)$$

Отметим, что было бы неправомерно взять вместо y_2 переменную, входящую свободно в какую-либо формулу выражения $(*)$: интуитивно говоря, тот элемент носителя, который делает формулу $\exists x P(x)$ истинной, вовсе не обязательно совпадает с элементом носителя, сопоставленным какой-либо из переменных, входящих свободно хотя бы в одну формулу выражения $(*)$. В частности, если бы мы взяли y_1 в качестве y_2, то получили бы $P(y_1) \to P(y_1),\ Q(z)$ и не нашли бы контрпримера. Однако вместо новизны переменной y_2 достаточно потребовать лишь того, чтобы y_2 не входила свободно ни в одну из формул выражения $(*)$.

Наконец, по выражению $(**)$ мы можем определить контрпример для исходной формулы, состоящий из интерпретации $M = \langle \mathcal{D}, \mu \rangle$ и оценки ν. В качестве носителя $\mathcal{D}$ возьмём множество термов $\{y_1,\ y_2,\ z\}$. Оценка ν такова, что $\nu(\alpha) = \alpha$ для каждого $\alpha \in \mathcal{D}$. Предикаты $\mu(P)$ и $\mu(Q)$ определим согласно $(**)$ (помня, что формулы слева от знака $\to$ должны быть истинны, а справа — ложны в этом контрпримере):

$$\mu(P)(y_2) = 1, \quad \mu(P)(y_1) = 0, \quad \mu(Q)(z) = 0.$$

В остальном интерпретация M и оценка ν могут быть заданы как угодно.

Проведённый поиск контрпримера наглядно изображается в виде следующего дерева:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(y_2) \to P(y_1),\ Q(z)}{\exists x P(x) \to P(y_1),\ Q(z)} \qquad Q(z) \to P(y_1),\ Q(z)}{\exists x P(x) \vee Q(z) \to P(y_1),\ Q(z)}}{\exists x P(x) \vee Q(z) \to P(y_1) \vee Q(z)}}{\exists x P(x) \vee Q(z) \to \forall x (P(x) \vee Q(z))}}{\to \exists x P(x) \vee Q(z) \supset \forall x (P(x) \vee Q(z))}.$$

$\triangleleft$

Пример 2.7.2. Попробуем найти какой-нибудь контрпример для формулы

$$\forall x P(f(x)) \vee Q(z) \supset \exists x (P(x) \vee Q(x)).$$

Для сокращения записи введём обозначение: $E \leftrightharpoons \exists x (P(x) \vee Q(x))$. Вначале имеем

$$\to \forall x P(f(x)) \vee Q(z) \supset E,$$

что равносильно

$$\forall x P(f(x)) \vee Q(z) \to E,$$

и, в свою очередь, равносильно

$$\forall x P(f(x)) \to E \quad \text{или} \quad Q(z) \to E.$$

Теперь поиск контрпримера может следовать по любому из двух появившихся путей.

Сначала последуем по пути

$$Q(z) \to E.$$

Для ложности (в искомом контрпримере) формулы $E \leftrightharpoons \exists x (P(x) \vee Q(x))$ необходима ложность формулы $P(t_1) \vee Q(t_1)$, где t_1 — произвольным образом выбранный терм (не исключается, что t_1 совпадает с одним из термов,

входящих в выражение $Q(z) \to E$). Ложность $P(t_1) \vee Q(t_1)$ при конкретном выбранном терме t_1, вообще говоря, недостаточна для ложности E, поэтому мы сохраним требование ложности E. Формула E ложна, если и только если E ложна и $P(t_1) \vee Q(t_1)$ ложна. Таким образом, на рассматриваемом пути получаем

$$Q(z) \to E,\ P(t_1) \vee Q(t_1),$$

причём мы пока не фиксируем конкретный терм, который выступит в качестве t_1. Мы бы могли выбрать конкретный терм в качестве t_1 сейчас, но нам будет удобнее сделать это позже. Затем на данном пути получаем

$$Q(z) \to E,\ P(t_1),\ Q(t_1).$$

Теперь видно, что при выборе в качестве t_1 терма z мы получим выражение $Q(z) \to E,\ P(z),\ Q(z)$, которое свидетельствует о том, что на этом пути контрпримера не найти ($Q(z)$ должна быть истинна и ложна одновременно).

Последуем по другому пути поиска:

$$\forall x P(f(x)) \to E.$$

Для истинности $\forall x P(f(x))$ необходимо и достаточно истинности $\forall x P(f(x))$ и истинности $P(f(t_2))$, где t_2 — произвольным образом выбранный терм. Опять отложим выбор конкретного терма в качестве t_2. Таким образом, получаем

$$\forall x P(f(x)),\ P(f(t_2)) \to E.$$

Отсюда, повторяя рассуждение о необходимом и достаточном условии для ложности формулы E, получаем

$$\forall x P(f(x)),\ P(f(t_2)) \to E,\ P(t_3) \vee Q(t_3),$$

где t_3 обозначает как угодно выбранный терм, который мы зафиксируем позже. Далее, очевидно, имеем

$$\forall x P(f(x)),\ P(f(t_2)) \to E,\ P(t_3),\ Q(t_3). \qquad (\star)$$

Попытаемся подобрать конкретные термы в качестве t_2 и t_3 так, чтобы в выражении $(\star)$ слева и справа от $\to$ стояла одна и та же формула. Так будет, если в качестве t_2 взять, например, предметную переменную u, а в качестве t_3 — терм $f(u)$. Значит, контрпримера на данном пути не найти. Поскольку мы исследовали все возможности для поиска контрпримера, то исходная формула общезначима.

Проведённый поиск наглядно представим в виде дерева:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\forall x P(f(x)),\ P(f(t_2)) \to E,\ P(t_3),\ Q(t_3)}{\forall x P(f(x)),\ P(f(t_2)) \to E,\ P(t_3) \vee Q(t_3)}}{\forall x P(f(x)),\ P(f(t_2)) \to E}}{\forall x P(f(x)) \to E} \qquad \dfrac{\dfrac{Q(z) \to E,\ P(t_1),\ Q(t_1)}{Q(z) \to E,\ P(t_1) \vee Q(t_1)}}{Q(z) \to E}}{\dfrac{\forall x P(f(x)) \vee Q(z) \to E}{\to \forall x P(f(x)) \vee Q(z) \supset E}}.$$

$\triangleleft$

Рассмотренные примеры подсказывают, что поиск контрпримера для формулы с кванторами может быть формализован в виде правил вывода некоторого исчисления: к известными правилам для логических связок добавляются правила для кванторов слева и справа от $\to$. Такое исчисление мы изложим в следующем разделе.

2.7.2. Формулировка исчисления

Зафиксируем произвольный язык первого порядка Ω, всюду ниже в разделе 2.7 все зависящие от языка первого порядка понятия относятся к языку Ω.

Сформулируем исчисление $\mathcal{G}\forall(\Omega)$, называемое *секвенциальным исчислением предикатов генценовского типа для языка* Ω.

Определения секвенции, представления секвенции S в виде формулы $\Phi(S)$, общезначимой секвенции и др. аналогичны определениям из раздела 1.4.2, только теперь вместо пропозициональных формул рассматриваются формулы языка Ω. Отметим лишь, что *контрпримером для секвенции* S теперь называется упорядоченная пара, состоящая из интерпретации языка Ω и оценки, в которых истинны все члены антецедента S и ложны все члены сукцедента S.

Будем говорить, что предметная переменная входит в секвенцию *свободно* (соответственно, *связанно*), если эта переменная входит свободно (соответственно, связанно) хотя бы в один член данной секвенции.

Правила вывода исчисления $\mathcal{G}\forall(\Omega)$ подразделяются на *пропозициональные* и *кванторные*. Пропозициональные правила вывода имеют тот же вид, что и правила вывода секвенциального исчисления высказываний (см. раздел 1.4.2), только теперь Γ, Δ обозначают любые конечные списки формул языка Ω, A, B — любые формулы языка Ω. Кванторные правила вывода формализуют шаги поиска контрпримера для предикатной формулы, на которых анализируется формула с квантором в качестве главного логического символа. Кванторные правила вывода исчисления $\mathcal{G}\forall(\Omega)$ таковы:

$$\frac{\Gamma,\ \forall x A,\ [A]^x_t \to \Delta}{\Gamma,\ \forall x A \to \Delta}\ (\forall \to), \qquad \frac{\Gamma \to \Delta,\ [A]^x_y}{\Gamma \to \Delta,\ \forall x A}\ (\to \forall),$$

$$\frac{\Gamma,\ [A]^x_y \to \Delta}{\Gamma,\ \exists x A \to \Delta}\ (\exists \to), \qquad \frac{\Gamma \to \Delta,\ \exists x A,\ [A]^x_t}{\Gamma \to \Delta,\ \exists x A}\ (\to \exists).$$

В этих кванторных правилах x — любая предметная переменная; t — любой терм, свободный для x в A; y — любая предметная переменная, свободная для x в A и не входящая свободно в заключения правил $(\to \forall)$ и $(\exists \to)$. Описанные в этом абзаце условия назовём *ограничениями на кванторные правила вывода исчисления* $\mathcal{G}\forall(\Omega)$.

В левом столбце приведены правила введения кванторов в антецедент секвенции: правило введения квантора всеобщности в антецедент и правило введения квантора существования в антецедент. В правом столбце приведены правила введения кванторов в сукцедент секвенции: правило введения квантора всеобщности в сукцедент и правило введения квантора существования в сукцедент.

Для какого угодно применения правила вывода исчисления $\mathcal{G}\forall(\Omega)$ член секвенции-заключения, явно выписанный в этом правиле, назовём *главным*, а каждый член секвенции-заключения из списков Γ и Δ — *параметрическим*. Для всякого применения правил $(\to \forall)$ и $(\exists \to)$ переменная y называется *собственной* переменной этого применения.

Аксиомы исчисления $\mathcal{G}\forall(\Omega)$ имеют тот же вид, что и аксиомы секвенциального исчисления высказываний, только теперь Γ, Δ обозначают любые конечные списки формул языка Ω, A — любую формулу языка Ω.

Как и в секвенциальном исчислении высказываний, вывод заданной секвенции в секвенциальном исчислении предикатов удобно искать, начиная с этой секвенции и осуществляя контрприменения правил (см. раздел 1.4.2).

Пример 2.7.3. Попытаемся найти вывод формулы из примера 2.7.2:

$$A \leftrightharpoons \forall x P(f(x)) \vee Q(z) \supset E, \quad \text{где} \quad E \leftrightharpoons \exists x (P(x) \vee Q(x)).$$

1) $\to \forall x P(f(x)) \vee Q(z) \supset E$ получается из секвенции 2 по правилу $(\to\supset)$;

2) $\forall x P(f(x)) \vee Q(z) \to E$ получается из секвенций 3 и 4 по правилу $(\vee \to)$;

3) $\forall x P(f(x)) \to E$ получается из секвенции 7 по правилу $(\forall \to)$;

4) $Q(z) \to E$ получается из секвенции 5 по правилу $(\to \exists)$;

5) $Q(z) \to E,\ P(z) \vee Q(z)$ получается из секвенции 6 по правилу $(\to \vee)$;

6) $Q(z) \to E,\ P(z),\ Q(z)$ — аксиома;

7) $\forall x P(f(x)),\ P(f(u)) \to E$ получается из секвенции 8 по правилу $(\to \exists)$;

8) $\forall x P(f(x)),\ P(f(u)) \to E,\ P(f(u)) \vee Q(f(u))$ получается из секвенции 9 по правилу $(\to \vee)$;

9) $\forall x P(f(x)),\ P(f(u)) \to E,\ P(f(u)),\ Q(f(u))$ — аксиома.

Итак, формула A выводима, и последовательность секвенций 9, 8, ..., 1 является выводом этой формулы. Этот вывод естественным образом представляется в виде дерева. Дерево, приведённое в примере 2.7.2, строго говоря, не является деревом вывода секвенции $\to A$, поскольку в нём $t_1,\ t_2,\ t_3$ служат для обозначения термов, которые не конкретизированы. Но заменив в этом дереве $t_1,\ t_2,\ t_3$ на $z,\ u,\ f(u)$ соответственно, мы получим дерево вывода этой секвенции. ◁

В предыдущем примере при контрприменениях правил $(\forall \to)$ и $(\to \exists)$ мы удачно подобрали термы для подстановки (в этих правилах терм для подстановки обозначен через t) и быстро закончили поиск вывода. Далеко не всегда легко подобрать термы так, чтобы поскорее прийти к аксиомам и закончить поиск вывода (в предположении, что вывод существует). Мы коснёмся этой проблемы в разделе 2.7.5.

Упражнение 2.7.4. Постройте выводы (в секвенциальном исчислении предикатов) всех формул, указанных в упражнении 2.6.14 на с. 89. ◁

2.7.3. Корректность и полнота исчисления

Следующие лемма, теорема и следствие этой теоремы доказываются аналогично соответствующим утверждениям о секвенциальном исчислении высказываний (см. раздел 1.4.3); дополнительно требуется лишь разобрать кванторные правила вывода в доказательстве леммы 2.7.5, что оставлено в качестве упражнения 2.7.6.

Лемма 2.7.5. *В исчислении $\mathcal{G}\forall(\Omega)$ каждая аксиома общезначима; для каждого правила вывода заключение этого правила общезначимо, если и только если все посылки этого правила общезначимы.*

Упражнение 2.7.6. Докажите утверждение леммы 2.7.5 для кванторных правил вывода. *Указание.* Используйте лемму 2.4.17. ◁

Теорема 2.7.7 (корректность исчисления $\mathcal{G}\forall(\Omega)$)**.** *Любая выводимая в исчислении $\mathcal{G}\forall(\Omega)$ секвенция общезначима.*

Следствие 2.7.8. *Исчисление $\mathcal{G}\forall(\Omega)$ непротиворечиво, т. е. не существует формулы A такой, что A выводима и $\neg A$ выводима.*

Упражнение 2.7.9. Рассмотрим исчисление, которое отличается от сформулированного секвенциального исчисления лишь тем, что допускается свободное вхождение собственной переменной в заключения правил $(\to \forall)$ и $(\exists \to)$. Приведите пример вывода необщезначимой формулы в этом новом исчислении. (Это новое исчисление мы рассматриваем только в данном упражнении.) ◁

Формулу (соответственно, секвенцию) назовём *чистой*, если никакая предметная переменная не входит в неё одновременно свободно и связанно.

Теорема 2.7.10 (полнота исчисления $\mathcal{G}\forall(\Omega)$)**.** *Пусть S_0 — произвольная чистая секвенция, все члены которой являются формулами языка Ω. Тогда если S_0 общезначима, то S_0 выводима в исчислении $\mathcal{G}\forall(\Omega)$.*

Замечание 2.7.11. Чистота формулы слабее обладания свойством чистоты переменных (см. определение 2.3.11). Например, формула $\forall x P(x,y) \vee \exists x Q(x)$ является чистой, но не обладает свойством чистоты переменных. ◁

Замечание 2.7.12. Секвенции S_1 и S_2 назовём *конгруэнтными*, если между членами антецедентов S_1 и S_2 и между членами сукцедентов S_1 и S_2 существует такая биекция, что соответствующие друг другу при этой биекции формулы-члены секвенций являются конгруэнтными.

Например, секвенции

$$\forall x P(x) \to Q(x),\ \exists y R(y) \quad \text{и} \quad \forall u P(u) \to \exists v R(v),\ Q(x),$$

очевидно, являются конгруэнтными.

С помощью леммы 2.3.13 о чистоте переменных для любой формулы (соответственно, секвенции) можно найти конгруэнтную ей чистую формулу (соответственно, секвенцию). С семантической точки зрения всё равно, какую из конгруэнтных формул (секвенций) рассматривать, поэтому ограничение чистыми секвенциями в теореме о полноте исчисления $\mathcal{G}\forall(\Omega)$ несущественно с семантической точки зрения. ◁

Доказательство теоремы 2.7.10. Опишем алгоритм, который определённым образом строит дерево поиска вывода произвольной чистой секвенции S_0, и покажем, что либо этот алгоритм построит дерево вывода секвенции S_0, либо существует контрпример для S_0. Отсюда, очевидно, будет следовать доказываемая теорема.

Для простоты изложения будем считать, что язык Ω не содержит более чем 0-местные функциональные символы. Описываемый ниже алгоритм построения дерева поиска вывода секвенции S_0 постепенно достраивает дерево, которое мы называем *текущим деревом*, подчеркивая то, что мы ссылаемся на дерево, построенное к текущему моменту исполнения алгоритма. Этот алгоритм таков:

(1) Строится текущее дерево из одного узла — исходной секвенции S_0.

(2) Строится множество $Terms$ всех свободных предметных переменных и предметных констант, входящих в S_0. Если $Terms = \varnothing$, то в $Terms$ добавляется какая угодно предметная переменная, не входящая в S_0.

(3) Повторяются следующие действия.

(3.1) Если каждый лист текущего дерева является аксиомой, то алгоритм завершается, выдавая в качестве ответа пару, состоящую из слова «выводима» и текущего дерева.

(3.2) Если в каждом листе текущего дерева имеются только члены, являющиеся атомарными формулами, то алгоритм завершается, выдавая в качестве ответа слово «невыводима».

(3.3) Если ни в одном листе текущего дерева нет помеченного члена (изначально так и будет), то в каждом листе S текущего дерева помечается каждый неатомарный член секвенции S.

(3.4) Как угодно выбирается лист S текущего дерева и помеченный член C секвенции S (для детерминированности способ такого выбора можно зафиксировать). Снимается пометка с члена C секвенции S.

(3.5) Если главным логическим символом формулы C является логическая связка, то осуществляется контрприменение соответствующего правила вывода к секвенции S с главным членом C, и (новыми) сыновними узлами рассматриваемого узла S текущего дерева становятся секвенции-посылки этого контрприменения. Здесь и всюду в этом алгоритме пометка каждого параметрического члена секвенции сохраняется в посылках произведённого контрприменения правила к этой секвенции.

(3.6) Если C является членом антецедента секвенции S и главным логическим символом C является $\exists$, или C является членом сукцедента секвенции S и главным логическим символом C является $\forall$, то

(3.6.1) выбирается предметная переменная y такая, что $y \notin Terms$ и y не входит ни в одну секвенцию текущего дерева;

(3.6.2) осуществляется контрприменение правила вывода $(\exists \to)$ или, соответственно, $(\to \forall)$ к S с главным членом C, причём в качестве собственной переменной используется y;

(3.6.3) сыновним узлом рассматриваемого узла S текущего дерева становится секвенция-посылка этого контрприменения;

(3.6.4) в множество $Terms$ добавляется переменная y.

(3.7) Иначе (т. е. если C является членом антецедента секвенции S и главным логическим символом C является $\forall$, или C является членом сукцедента секвенции S и главным логическим символом C является $\exists$) для каждого терма $t' \in Terms$ выполняются следующие действия:

(3.7.1) осуществляется контрприменение правила вывода $(\forall \rightarrow)$ или, соответственно, $(\rightarrow \exists)$ к S с главным членом C, причём в качестве подставляемого терма, обозначенного в этих правилах через t, берётся t';

(3.7.2) сыновним узлом S_l рассматриваемого узла S текущего дерева становится секвенция-посылка этого контрприменения;

(3.7.3) полагается, что обозначение C теперь относится к тому члену секвенции S_l, который был главным в секвенции S в (3.7.1), а обозначение S — к только что созданному узлу S_l.

Легко видеть, что в силу чистоты исходной секвенции S_0 на каждом шаге работы описанного алгоритма при контрприменении правил выполняются ограничения на кванторные правила. Поэтому на каждом шаге работы этого алгоритма построенное дерево действительно является деревом поиска вывода секвенции S_0.

Если алгоритм завершается при исполнении (3.1), то полученное в результате его работы дерево является деревом вывода секвенции S_0. Дальше разберём два оставшихся случая: алгоритм завершается при исполнении (3.2) и алгоритм не завершается.

Если алгоритм завершается при исполнении (3.2), то существует конечная ветвь[12] построенного к моменту завершения дерева, каждая секвенция которой не является аксиомой.

Если алгоритм не завершается, то рассмотрим дерево с бесконечным числом узлов, которое определено в соответствии с описанным выше построением. Так как алгоритм не завершается, то существует сколь угодно длинная ветвь этого дерева, каждая секвенция которой не является аксиомой. Тогда в этом дереве существует бесконечная ветвь, каждая секвенция которой не является аксиомой (иначе, очевидно, не существовало бы и сколь угодно длинной ветви).

Итак, в обоих случаях имеется ветвь Br, каждая секвенция которой не является аксиомой. Для завершения доказательства теоремы достаточно определить контрпример для S_0.

Обозначим через $\Gamma_\cup$ (соответственно, через $\Delta_\cup$) множество всех членов антецедентов (соответственно, сукцедентов) секвенций ветви Br. Множества $\Gamma_\cup$ и $\Delta_\cup$ не содержат общих атомарных формул, а также $\Gamma_\cup$ не содержит $\mathbf{F}$, и $\Delta_\cup$ не содержит $\mathbf{T}$: иначе на ветви Br нашлась бы секвенция, являющаяся аксиомой. Определим интерпретацию $M \leftrightharpoons \langle \mathcal{D}, \mu \rangle$ языка Ω. Носителем $\mathcal{D}$ объявим множество, состоящее из всех термов, которые когда-либо попадают в множество $Terms$ при неограниченном продолжении описанного выше построения дерева. Для каждой предметной константы $a \in \mathcal{D}$ положим $\mu(a) = a$; для любой другой предметной константы b языка Ω $\mu(b)$ можно определить как угодно. Для каждого n-местного предикатного символа P языка Ω $(n > 0)$ определим предикат $\mu(P)$ так, что для любых $t_1, \ldots, t_n \in \mathcal{D}$ $\mu(P)(t_1, \ldots, t_n) = 1$, если атомарная формула $P(t_1, \ldots, t_n)$ принадлежит $\Gamma_\cup$, иначе $\mu(P)(t_1, \ldots, t_n) = 0$. Опреде-

[12] Пусть T — дерево с конечным или бесконечным числом узлов. *Ветвью* дерева T мы называем конечную (или бесконечную) последовательность узлов $S_0, S_1, \ldots, S_k$ (соответственно, $S_0, S_1, \ldots$) этого дерева, где S_0 — корень дерева T, и для каждого $i = 0, 1, \ldots, k-1$ (соответственно, для каждого $i = 0, 1, \ldots$) узел S_{i+1} является сыном узла S_i, причём в случае конечной последовательности S_k является листом дерева T.

ление $\mu(P)$ для 0-местного предикатного символа P аналогично предыдущему: $\mu(P) = 1$, если $P \in \Gamma_\cup$, иначе $\mu(P) = 0$. Определим оценку ν так, что $\nu(x) = x$ для каждой предметной переменной $x \in \mathcal{D}$; для любой другой предметной переменной z $\mu(z)$ можно определить как угодно.

Для доказательства того, что интерпретация M и оценка ν составляют контрпример для S_0, достаточно доказать следующее: для любой формулы F, если $F \in \Gamma_\cup$, то $M, \nu \vDash F$, а если $F \in \Delta_\cup$, то $M, \nu \nvDash F$. Воспользуемся индукцией по числу вхождений логических символов в формулу F. База индукции (формула F атомарна) очевидна по определению M и ν.

Индукционный переход. Предположим, что доказываемое утверждение справедливо для любой формулы F с числом вхождений логических символов, равным n. Рассмотрим формулу F с ровно $n + 1$ вхождением логических символов.

Пусть F имеет вид $\exists x A$.

Если $F \in \Gamma_\cup$, то по построению дерева $[A]^x_y \in \Gamma_\cup$ для некоторой переменной $y \in \mathcal{D}$. Тогда по индукционному предположению $M, \nu \vDash [A]^x_y$. Следовательно, $M, \nu \vDash \exists x A$.

Если $F \in \Delta_\cup$, то по построению дерева $[A]^x_t \in \Delta_\cup$ для любого терма $t \in \mathcal{D}$. Тогда по индукционному предположению $M, \nu \nvDash [A]^x_t$ для любого терма $t \in \mathcal{D}$. Следовательно, $M, \nu \nvDash \exists x A$.

Рассмотрение случаев, когда F имеет вид $\neg A$, $A \wedge B$, $A \vee B$, $A \supset B$ и $\forall x A$ остаётся в качестве упражнения. ◁

Упражнение 2.7.13. Рассмотрите случаи, упомянутые в последнем абзаце доказательства теоремы 2.7.10. ◁

Упражнение 2.7.14. В соответствии с приведённым в доказательстве теоремы 2.7.10 алгоритмом постройте дерево вывода или контрпример для исходной секвенции из примера 2.7.1 и из примера 2.7.2, а также для секвенции

$$\to \exists x(P(x) \vee \neg P(x)),\ \exists x Q(x).$$

◁

Упражнение 2.7.15. Пусть рассматривается формула A из упражнения 2.6.35 на с. 94. Начните строить дерево поиска вывода секвенции $\to A$ согласно приведённому в доказательстве теоремы 2.7.10 алгоритму. Найдите контрпример для этой секвенции. ◁

Доказательство теоремы 2.7.10 даёт нам алгоритм поиска вывода в исчислении $\mathcal{G}\forall(\Omega)$. Для любой чистой секвенции S_0 этот алгоритм выдаёт «выводима» и вместе с этим выдаёт дерево вывода S_0, если и только если S_0 выводима; если же S_0 невыводима, то алгоритм выдаёт «невыводима» или не завершает свою работу.

Упражнение 2.7.16. Предложите алгоритм поиска вывода, более экономный, чем алгоритм из доказательства теоремы 2.7.10. *Указание.* По крайней мере обратите внимание на следующие вопросы: можно ли в большем числе случаев выдавать ответ «невыводима»; можно ли сократить число контрприменений правил вывода (в случае завершаемости алгоритма), изменив порядок контрприменений правил; можно ли уменьшить число контрприменений правил $(\forall \to)$ и $(\to \exists)$. ◁

Упражнение 2.7.17. Проведите доказательство теоремы 2.7.10 допустив, что язык Ω может содержать какие угодно функциональные символы. *Указание.* Модифицируйте приведённый в доказательстве теоремы 2.7.10 алгоритм построения дерева поиска вывода так, чтобы на любой ветви дерева, если её построение не заканчивается за конечное число шагов, рано или поздно каждый терм из некоторого множества термов использовался в качестве подставляемого при контрпримененении правил $(\forall \to)$ и $(\to \exists)$ с каждым возможным главным членом. Упомянутое множество термов (которое требуется определить при выполнении данного упражнения) можно объявить носителем интерпретации при задании контрпримера. $\triangleleft$

Упражнение 2.7.18. Найдите общезначимую формулу (не являющуюся чистой), которая невыводима в секвенциальном исчислении предикатов. $\triangleleft$

2.7.4. Соотношение с исчислением предикатов гильбертовского типа

Теоремы о корректности и полноте исчисления $\mathcal{G}\forall(\Omega)$ показывают, что это секвенциальное исчисление и исчисление $\mathcal{H}\forall(\Omega)$ гильбертовского типа в некотором смысле эквивалентны, точнее, для любой чистой формулы A языка Ω имеем: A выводима в $\mathcal{H}\forall(\Omega)$ тогда и только тогда, когда A выводима в $\mathcal{G}\forall(\Omega)$. Действительно, выводимость A как в $\mathcal{H}\forall(\Omega)$, так и в $\mathcal{G}\forall(\Omega)$ равносильна общезначимости A.

С другой стороны, эту эквивалентность исчислений можно доказать без обращения к семантике, конструктивно, с помощью так называемого *правила сечения*, которое допустимо в исчислении $\mathcal{G}\forall(\Omega)$. (Определение допустимого в исчислении $\mathcal{G}\forall(\Omega)$ правила аналогично определению допустимого в исчислении $\mathcal{G}$ правила, данному в разделе 1.4.4.) Правило сечения таково:

$$\frac{\Gamma,\ A \to \Delta; \quad \Gamma \to \Delta,\ A}{\Gamma \to \Delta},$$

где Γ, Δ обозначают любые конечные списки формул языка Ω, A — любую формулу языка Ω, и заключение является чистой секвенцией.

Мы не будем давать конструктивное доказательство допустимости правила сечения, такое доказательство довольно длинное (хотя оно ценно тем, что даёт алгоритм преобразования вывода, использующего правило сечения, в вывод, использующий только правила вывода секвенциального исчисления). Приведём неконструктивное, основанное на семантических понятиях доказательство допустимости правила сечения. Пусть выводимы обе секвенции-посылки этого правила. Тогда по теореме о корректности исчисления $\mathcal{G}\forall(\Omega)$ обе эти посылки общезначимы. Предположив, что для секвенции-заключения существует контрпример, немедленно получим противоречие с общезначимостью посылок (более подробное рассуждение остаётся в качестве лёгкого упражнения). Следовательно, для секвенции-заключения не существует контрпримера, т. е. она общезначима, значит, по теореме о полноте исчисления $\mathcal{G}\forall(\Omega)$ эта секвенция выводима, что и требовалось доказать.

Правило сечения по сути является аналогом правила модус поненс, которое затрудняет поиск вывода снизу вверх (см. раздел 1.3.5). Для поиска вывода снизу вверх секвенциальное исчисление предикатов (без правила

сечения) более удобно, чем исчисление предикатов гильбертовского типа. Мы продолжим обсуждение поиска вывода в следующем разделе 2.7.5.

Упражнение 2.7.19. Используя допустимость правила сечения, конструктивно докажите теорему 2.7.8 (ранее полученную как следствие теоремы 2.7.7). ◁

2.7.5. О поиске вывода в секвенциальном исчислении предикатов

Сказанное в конце раздела 1.4.3 об удобстве секвенциального исчислении высказываний для поиска вывода снизу вверх требует дополнения для секвенциального исчислении предикатов. При контрприменении любого правила вывода, кроме $(\forall \to)$ и $(\to \exists)$, если выбран главный член секвенции-заключения, то секвенции-посылки подбираются детерминированным образом. Точнее, для пропозициональных правил вывода секвенции-посылки определяются однозначно, а для правил $(\to \forall)$ и $(\exists \to)$, хоть и можно варьировать собственную переменную, достаточно выбрать какую угодно, удовлетворяющую ограничениям на кванторные правила. Однако при контрприменении правил $(\forall \to)$ и $(\to \exists)$ желательно так подобрать подставляемый терм (обозначенный в этих правилах через t), чтобы по возможности скорее получить аксиомы в качестве листьев строящегося дерева поиска вывода.

Рассмотрим, например, дерево вывода секвенции $P(f(z)) \to \exists x P(x)$, полученное в результате поиска вывода снизу вверх:

$$\dfrac{\dfrac{P(f(z)) \to \exists x P(x),\ P(f(z)),\ P(z)}{P(f(z)) \to \exists x P(x),\ P(z)}}{P(f(z)) \to \exists x P(x)}.$$

При первом контрприменении правила $(\to \exists)$ подставляемый терм z был выбран неудачно, но при втором контрприменении этого правила подставляемый терм $f(z)$ был выбран удачно, что позволило получить аксиому и завершить поиск вывода. Конечно, если имеется сильно разветвлённое дерево поиска вывода, то совсем неочевидно, какой выбор терма считать удачным, ведь мы стремимся, чтобы каждый лист такого дерева стал аксиомой.

Итак, существенной проблемой при поиске вывода снизу вверх в секвенциальном исчислении предикатов является нахождение термов для подстановки при контрприменении правил $(\forall \to)$ и $(\to \exists)$. Детальное описание подходов к решению этой проблемы выходит за рамки вводного курса математической логики. Наметим лишь идею одного из подходов. Вместо того, чтобы при контрприменении правила $(\forall \to)$ или $(\to \exists)$ подставлять конкретный терм, отложим его выбор, подставив вместо него новую переменную. Такую переменную называют метапеременной, она служит для обозначения некоторого терма. Позднее в ходе поиска вывода, когда, скорее всего, будет больше информации для удачного выбора термов (если удачный выбор вообще возможен), мы попытаемся подобрать термы-значения для всех метапеременных так, чтобы превратить текущее дерево в дерево вывода. В небольших упражнениях на поиск вывода подбор термов можно осуществлять «вручную», а чтобы автоматизировать такой подбор, можно воспользоваться алгоритмом унификации, который мы рассмотрим в разделе 4.3.2.

Заметим, что мы по сути использовали только что обрисованный подход в примере 2.7.2. В заключение приведём упражнение, которое рекомендуется выполнить, используя намеченный подход.

Упражнение 2.7.20. Найдите вывод каждой из формул:

$$\exists x(P(a) \vee P(f(b)) \supset P(x)), \qquad \exists x \forall y(P(x) \supset P(y)).$$

◁

Завершая текущую главу, носящую название «Исчисления предикатов», скажем, что одному из исчислений предикатов — резолютивному — мы посвятим главу 4.

Глава 3.

Формальные аксиоматические теории

Мы знаем, что в исчислении предикатов $\mathcal{H}\forall(Ar)$ в языке элементарной арифметики Ar выводимы в точности все формулы, истинные в любой интерпретации этого языка. Естественным образом встаёт задача вывода формул языка Ar, которые выражают свойства натуральных чисел, т. е. которые истинны в стандартной интерпретации ω языка Ar. Тогда лишь формулами, истинными в любой интерпретации этого языка, не обойтись. Рассмотрим, например, формулу $A \leftrightharpoons (S(0) + 0 = S(0))$ языка Ar. В интерпретации ω формула A говорит о том, что сумма натуральных чисел 1 и 0 равна 1. Формула A невыводима в исчислении $\mathcal{H}\forall(Ar)$, поскольку необщезначима. Вспомним (см. теорему 2.6.4), что если формулы, используемые как гипотезы вывода, истинны в интерпретации ω, то из таких гипотез выводимы только формулы, истинные в этой интерпретации. Поэтому для того, чтобы выводить формулы, истинные в интерпретации ω, но необязательно общезначимые, можно воспользоваться выводом из гипотез. При этом подходе к гипотезам надлежит отнести истинные в ω формулы, которые описывают базовые свойства натуральных чисел. Например, формулу A можно вывести из гипотезы $\forall x(x + 0 = x)$, которая описывает одно из свойств натуральных чисел с операцией сложения.

Аналогичные соображения применимы и для формального доказательства свойств других математических объектов. Для того, чтобы формально доказывать формулы, выражающие свойства рассматриваемых объектов, базовые свойства этих объектов записывают в виде некоторого набора формул Γ и применяют вывод из гипотез Γ. Язык первого порядка для записи свойств и множество формул Γ составляют формальную аксиоматическую теорию первого порядка.

В данной главе мы точно определим понятие формальной аксиоматической теории и рассмотрим конкретные важные теории: элементарную арифметику, формализующую свойства натуральных чисел, и теорию множеств Цермело-Френкеля, служащую основанием современной математики (точнее, её классического направления,[1] которого мы придерживаемся в

[1] Классическое направление в математике также называют классической математикой.

данной книге и в рамках которого обычно читаются базовые курсы алгебры, математического анализа, геометрии).

3.1. Основные определения

Определение 3.1.1. *Формальной аксиоматической теорией первого порядка* (короче — *теорией*) *в языке* Ω называется упорядоченная пара $\langle \Omega, \Gamma \rangle$, где Ω — язык первого порядка, Γ — множество замкнутых формул этого языка. Каждый элемент множества Γ называется *собственной* (или *нелогической*) *аксиомой* этой теории. $\triangleleft$

Определение 3.1.2. Пусть $T \leftrightharpoons \langle \Omega, \Gamma \rangle$ — теория, A — формула языка Ω.

Формула A называется *теоремой теории* T (а также *выводимой в теории* T), если $\Gamma \vdash_{\mathcal{H}\forall(\Omega)} A$.

То, что A — теорема теории T, будем обозначать через $T \vdash A$. В противном случае будем писать $T \nvdash A$.

Вывод из Γ будем называть *выводом в теории* T; вывод формулы A из Γ будем называть *выводом формулы* A *в теории* T. $\triangleleft$

Определение 3.1.3. Теория $\langle \Omega, \Gamma \rangle$ называется *непротиворечивой* (соответственно, *противоречивой*, *полной*, *неполной*), если множество Γ непротиворечиво (соответственно, противоречиво, полно, неполно). $\triangleleft$

Определение 3.1.4. Пусть $T \leftrightharpoons \langle \Omega, \Gamma \rangle$ — теория. *Интерпретацией теории* T называется любая интерпретация языка Ω. *Моделью теории* T называется любая модель множества Γ (т. е. интерпретация языка Ω, в которой истинны все собственные аксиомы этой теории). $\triangleleft$

Отметим, что исчисление предикатов $\mathcal{H}\forall(\Omega)$ можно рассматривать как теорию с пустым множеством всех собственных аксиом; любая интерпретация языка Ω является моделью этой теории.

Произвольное множество Γ замкнутых формул языка Ω можно отождествить с теорией $\langle \Omega, \Gamma \rangle$. Так что теоремы 2.6.20, 2.6.21, 2.6.28–2.6.30 о множествах замкнутых формул легко сформулировать и в терминах теорий. Например, теорему 2.6.21 о существовании модели можно переформулировать так: *любая непротиворечивая теория имеет модель.*

Сформулируем и докажем аналог обобщённых теорем о корректности и полноте исчисления предикатов для теорий.

Теорема 3.1.5. *Пусть* $T \leftrightharpoons \langle \Omega, \Gamma \rangle$ *— теория,* A *— формула языка* Ω*. Тогда* $T \vdash A$*, если и только если* A *истинна в любой модели теории* T*.*

Доказательство. По обобщённым теоремам о корректности и полноте исчисления $\mathcal{H}\forall(\Omega)$ $\Gamma \vdash A$ равносильно $\Gamma \vDash A$. $\Gamma \vDash A$ означает, что в любой интерпретации языка Ω при любой оценке выполняется: если все формулы из Γ истинны, то A истинна. Поскольку все формулы из Γ замкнуты, имеем: $\Gamma \vDash A$, если и только если в любой модели теории T истинна A. $\triangleleft$

Теорема 3.1.5 показывает, что вывод в теории является подходящим способом получения формул, истинных в любой модели этой теории.

Непротиворечивость какой бы то ни было теории имеет первостепенную важность. Как показывает следующая лемма, в противоречивой теории T выводима любая формула языка этой теории, так что T бесполезна для получения теорем теории T. Если же обратиться к семантике, то по теореме 2.6.20 противоречивая теория не имеет модели.

Лемма 3.1.6. *Пусть T — теория в языке Ω. Тогда T непротиворечива, если и только если существует невыводимая в T формула языка Ω.*

Доказательство. Очевидно, что если T непротиворечива, то существует невыводимая в T формула языка Ω. Чтобы доказать обратное утверждение, докажем утверждение, равносильное ему: если T противоречива, то любая формула языка Ω выводима в T.

Пусть теория T противоречива, т. е. существует формула A языка Ω такая, что $T \vdash A$ и $T \vdash \neg A$. Для любой формулы B языка Ω имеем $\vdash \neg A \supset (A \supset B)$. Следовательно, $T \vdash B$. Таким образом, в противоречивой теории T выводима любая формула языка Ω. ◁

Упражнение 3.1.7. Докажите, что каждое из двух следующих утверждений равносильно противоречивости теории T: а) некоторая формула вида $A \wedge \neg A$ выводима в T, б) некоторая формула вида $A \equiv \neg A$ выводима в T. Обобщите установленные результаты. ◁

Требование полноты теории вовсе не является обязательным для успешного применения этой теории. Приведём наглядный пример, иллюстрирующий свойство полноты теории. Теорию можно образно представлять себе как эксперта в какой-нибудь области знаний. Этому эксперту можно задавать лишь вопросы, требующие ответа «да» или «нет», впрочем, эксперт иногда может ответить «не знаю». Полная теория соответствует эксперту, который на любой вопрос из области знаний этого эксперта отвечает «да» или «нет». Неполная теория соответствует эксперту, который в аналогичной ситуации может также ответить «не знаю». Непротиворечивая теория подобна эксперту, который всегда отвечает правдиво, точнее, для любого вопроса, если эксперт отвечает на него «да», то на отрицание этого вопроса не отвечает «да». Вряд ли можно сыскать эксперта в сколь-нибудь серьёзной области знаний, который бы всегда отвечал правдиво и никогда не говорил «не знаю».

Следующее упражнение выявляет связь понятия полноты теории с семантикой языка этой теории.

Упражнение 3.1.8. Пусть T — теория в языке Ω, M — модель этой теории, $[T]$ — множество всех замкнутых формул, выводимых в теории T, $Th(M)$ — множество всех истинных в M замкнутых формул языка Ω. Теория T называется *полной по отношению к модели* M, если $[T] = Th(M)$.

Докажите, что

1) если теория T полна по отношению к некоторой модели, то T является полной теорией;

2) если T является полной теорией, то для любой модели M теории T эта теория полна по отношению к M. ◁

Замечание 3.1.9. Мы определили теорию на основе исчисления предикатов гильбертовского типа, дав определение 3.1.2 выводимой в теории формулы. Можно определить теорию и на основе секвенциального исчисления предикатов генценовского типа, дав следующее определение выводимой в теории формулы.

Пусть $T \leftrightharpoons \langle \Omega, \Gamma \rangle$ — теория, A — формула языка Ω. Формула A называется теоремой теории T (или выводимой в теории T), если существует такое множество $\{G_1, \ldots, G_n\} \subseteq \Gamma$, что в исчислении $\mathcal{G}\forall(\Omega)$ выводима секвенция $G'_1, \ldots, G'_n \to A$, где $G'_1, \ldots, G'_n$ — формулы, конгруэнтные $G_1, \ldots, G_n$ соответственно и такие, что данная секвенция является чистой. ◁

Упражнение 3.1.10. Зачем в определении из предыдущего замечания понадобилось обеспечить чистоту указанной секвенции? Докажите, что это определение не зависит от выбора формул $G'_1, \ldots, G'_n$, конгруэнтных формулам $G_1, \ldots, G_n$ соответственно. Докажите, что это определение выводимой в теории формулы равносильно определению 3.1.2 выводимой в теории формулы. ◁

Упражнение 3.1.11. Как изменить формулировку исчисления $\mathcal{G}\forall(\Omega)$, чтобы определения выводимой в теории формулы, данные в следующем абзаце и в определении 3.1.2, были равносильны?

Пусть $T \leftrightharpoons \langle \Omega, \Gamma \rangle$ — теория, A — формула языка Ω. Формула A называется теоремой теории T (или выводимой в теории T), если существует такое множество $\{G_1, \ldots, G_n\} \subseteq \Gamma$, что в исчислении $\mathcal{G}\forall(\Omega)$ выводима секвенция $G_1, \ldots, G_n \to A$.

Указание. См. приложение А в настоящей книге. ◁

3.2. Теории с равенством

Многие математические теории используют равенство, точнее, предикат равенства, выражающий совпадение своих аргументов. Рассматриваемое в данном разделе понятие теории с равенством формализует ключевые свойства равенства.

Примем соглашение, что в каждой формуле текущей главы различны предметные переменные, которые имеют различные обозначения.

Определение 3.2.1. Пусть $T \leftrightharpoons \langle \Omega, \Gamma \rangle$ — теория, и в сигнатуре языка Ω имеется двухместный предикатный символ $=$.

Будем называть $=$ *предикатным символом равенства* и использовать следующие сокращения (см. также соглашение о записи двухместных предикатных символов в начале раздела 2.2.2):

$$t_1 = t_2 \leftrightharpoons (t_1 = t_2) \leftrightharpoons = (t_1, t_2),$$
$$t_1 \neq t_2 \leftrightharpoons (t_1 \neq t_2) \leftrightharpoons \neg(t_1 = t_2).$$

Тогда теория T называется *теорией с равенством*, если все следующие формулы являются теоремами (или, в частном случае, собственными аксиомами) этой теории:

1) $\forall x(x = x)$;

2) $\forall x \forall y(x = y \supset y = x)$;

3) $\forall x \forall y \forall z (x = y \wedge y = z \supset x = z)$;

4) формулы вида

$$\forall x_1 \ldots \forall x_n \forall y_1 \ldots \forall y_n (x_1 = y_1 \wedge \ldots \wedge x_n = y_n \supset f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n))$$

для любого $n \in \mathbb{N}_+$ и любого n-местного функционального символа f языка Ω;

5) формулы вида

$$\forall x_1 \ldots \forall x_n \forall y_1 \ldots \forall y_n (x_1 = y_1 \wedge \ldots \wedge x_n = y_n \supset (P(x_1, \ldots, x_n) \equiv P(y_1, \ldots, y_n)))$$

для любого $n \in \mathbb{N}_+$ и любого n-местного предикатного символа P языка Ω, кроме предикатного символа $=$.

Формулы 1–3 называют *аксиомой рефлексивности, симметричности* и *транзитивности равенства* соответственно, а формулы видов 4–5 — *аксиомами подстановочности равенства*. Все формулы видов 1–5 называют *аксиомами равенства*. ◁

Определение 3.2.2. Пусть Ω — язык первого порядка, в сигнатуре которого имеется двухместный предикатный символ равенства $=$, $M \leftrightharpoons \langle \mathcal{D}, \mu \rangle$ — интерпретация языка Ω. Тогда M называется *нормальной интерпретацией*, если отношение $\mu(=)$ является отношением тождества (совпадения элементов) на множестве $\mathcal{D}$. ◁

Теперь мы можем употреблять прилагательное «нормальная» применительно к интерпретации, удовлетворяющей дополнительным требованиям, в частности, можем говорить о нормальной модели множества замкнутых формул (языка первого порядка с предикатным символом равенства) и о нормальной модели теории с равенством.

Так как в любой модели теории с равенством истинны аксиомы равенства, то отношение, соответствующее в этой модели предикатному символу $=$, рефлексивно, симметрично и транзитивно, т. е. является отношением эквивалентности на носителе этой модели. В нормальной модели теории с равенством такое отношение эквивалентности является отношением тождества.

Заметим также, что в любой нормальной интерпретации теории с равенством (т. е. нормальной интерпретации языка этой теории, необязательно являющейся моделью этой теории) истинны аксиомы равенства.

Содержательно (в модели теории с равенством) аксиомы подстановочности равенства выражают то, что замена аргументов функции (соответственно, предиката) на равные не изменяет значение функции (соответственно, предиката).

Упражнение 3.2.3. Выведите из аксиом равенства формулу

$$\forall x_1 \forall x_2 \forall y_1 \forall y_2 (x_1 = y_1 \wedge x_2 = y_2 \supset (x_1 = x_2 \equiv y_1 = y_2)).$$

Именно из-за этой выводимости данная формула («аксиома подстановочности равенства» для предикатного символа $=$) не упоминается среди

аксиом подстановочности равенства в определении 3.2.1 теории с равенством. ◁

Пример 3.2.4. *Элементарная теория полугрупп* есть теория в языке, сигнатура которого состоит лишь из двухместного функционального символа $\cdot$ и двухместного предикатного символа $=$ (см. также раздел 2.2.2). Перечислим собственные аксиомы этой теории.

Аксиомы равенства:

1) $\forall x(x = x)$,

2) $\forall x \forall y(x = y \supset y = x)$,

3) $\forall x \forall y \forall z(x = y \wedge y = z \supset x = z)$,

4) $\forall x_1 \forall x_2 \forall y_1 \forall y_2(x_1 = y_1 \wedge x_2 = y_2 \supset x_1 \cdot x_2 = y_1 \cdot y_2)$.

Аксиома, выражающая ассоциативность операции $\cdot$:

5) $\forall x \forall y \forall z((x \cdot y) \cdot z = x \cdot (y \cdot z))$.

На этом перечисление собственных аксиом элементарной теории полугрупп закончено. Любая нормальная модель этой теории называется *полугруппой*. (Это определение полугруппы является уточнением определения, известного из курса алгебры.)

Элементарная теория групп получается из элементарной теории полугрупп лишь добавлением собственной аксиомы, которая выражает существование нейтрального элемента, а также для каждого элемента группы существование обратного к нему элемента:

6) $\exists z(\forall x(x \cdot z = x \wedge z \cdot x = x) \wedge \forall x \exists y(x \cdot y = z \wedge y \cdot x = z))$.

Любая нормальная модель этой теории называется *группой*.

Добавив ещё одну собственную аксиому

7) $\forall x \forall y(x \cdot y = y \cdot x)$,

получим так называемую *элементарную теорию абелевых* (или *коммутативных*) *групп*.

Интерпретация, носитель которой есть множество всех слов в некотором алфавите, и которая функциональному символу $\cdot$ сопоставляет функцию конкатенации, а предикатному символу $=$ — предикат, выражающий совпадение аргументов, является нормальной моделью элементарной теории полугрупп, но не является моделью элементарной теории групп.

Интерпретация, носитель которой есть множество всех целых чисел, и которая функциональному символу $\cdot$ сопоставляет функцию сложения, а предикатному символу $=$ — предикат, выражающий совпадение аргументов, является нормальной моделью каждой из трёх данных теорий. ◁

Упражнение 3.2.5. Предложите для рассмотренных в предыдущем примере теорий модели, не являющиеся нормальными. ◁

Теорема 3.2.6 (о существовании нормальной модели)**.** *Любая непротиворечивая теория T с равенством имеет нормальную модель, притом конечную или счётную.*

Доказательство. В силу теоремы 2.6.28 теория T имеет модель $M \leftrightharpoons \langle \mathcal{D}, \mu \rangle$ со счётным носителем $\mathcal{D}$. Построим интерпретацию $M' \leftrightharpoons \langle \mathcal{D}', \mu' \rangle$ этой теории и докажем, что M' является нормальной моделью теории T.

В качестве носителя $\mathcal{D}'$ возьмём множество всех классов эквивалентности, определяемых отношением эквивалентности $\mu(=)$ на множестве $\mathcal{D}$. (Грубо говоря, один элемент множества $\mathcal{D}'$ получается «склеиванием» всех тех элементов множества $\mathcal{D}$, которые находятся друг с другом в отношении $\mu(=)$.) Так как $\mathcal{D}$ счётно, то $\mathcal{D}'$ конечно или счётно. Будем обозначать класс эквивалентности с представителем $\alpha \in \mathcal{D}$ через $[\alpha]$.

Для каждого n-местного функционального символа f при $n > 0$ определим функцию

$$\mu'(f) : [\alpha_1], \ldots, [\alpha_n] \mapsto [\mu(f)(\alpha_1, \ldots, \alpha_n)],$$

а при $n = 0$ положим $\mu'(f) = [\mu(f)]$.

Для каждого n-местного предикатного символа P при $n > 0$ определим предикат

$$\mu'(P) : [\alpha_1], \ldots, [\alpha_n] \mapsto \mu(P)(\alpha_1, \ldots, \alpha_n),$$

а при $n = 0$ положим $\mu'(P) = \mu(P)$.

Так как аксиомы подстановочности равенства истинны в модели M, то определение μ' не зависит от выбора представителей $\alpha_1, \ldots, \alpha_n$ классов эквивалентности.

По определению $\mathcal{D}'$ и μ' отношение $\mu'(=)$ является отношением тождества на $\mathcal{D}'$. Осталось показать, что M' является моделью теории T. Для этого достаточно доказать, что для любой формулы A и для любой оценки $\nu : IV \to \mathcal{D}$ выполняется: $M, \nu \vDash A$, если и только если $M', \nu' \vDash A$, где оценка $\nu' : IV \to \mathcal{D}'$ такова, что для любой предметной переменной $x \in IV$ имеет место $\nu'(x) = [\nu(x)]$. Несложное доказательство этого утверждения индукцией по построению формулы A остаётся в качестве упражнения. ◁

Упражнение 3.2.7. Докажите утверждение, сформулированное в предыдущем абзаце. ◁

Упражнение 3.2.8. Приведите пример непротиворечивой теории с равенством, не имеющей счётной нормальной модели. ◁

Упражнение 3.2.9. Приведите пример непротиворечивой теории с равенством, не имеющей конечной нормальной модели. *Указание.* См. упражнение 2.6.35 на с. 94. ◁

Из теоремы 3.2.6 о существовании нормальной модели получаем следствия, аналогичные теоремам 2.6.30 и 2.6.31 — следствиям теоремы 2.6.21 о существовании модели.

Теорема 3.2.10 (теорема о компактности для нормальных моделей)**.** *Пусть $T \leftrightharpoons \langle \Omega, \Gamma \rangle$ — теория с равенством. Тогда если всякое конечное подмножество множества Γ имеет нормальную модель, то теория T имеет нормальную модель.*

Теорема 3.2.11 (полнота исчисления предикатов для теорий с равенством)**.** *Пусть T — теория с равенством в языке Ω, A — формула языка Ω. Тогда если в любой нормальной модели теории T формула A истинна, то $T \vdash A$.*

Замечание 3.2.12. Иногда рассматривают только теории с равенством (которые называют просто теориями) и только нормальные интерпретации таких теорий. При этом в сигнатуру каждой рассматриваемой теории входит предикатный символ равенства, и к аксиомам исчисления предикатов гильбертовского типа добавляют все формулы, указанные в определении 3.2.1. Получившееся исчисление называют *исчислением предикатов гильбертовского типа с равенством*. ◁

Выражение существования и единственности формулой

Пусть Ω — язык первого порядка с предикатным символом $=$, A — формула этого языка, возможно, со свободным вхождением переменной x (это и есть наиболее интересный случай, хотя мы не исключаем, что x не является параметром A, а также не исключаем, что кроме x в A имеются другие параметры). Введём следующее сокращение:

$$\exists! x A \leftrightharpoons \exists x A \wedge \forall x \forall y (A \wedge [A]^x_y \supset x = y),$$

где y — переменная, не входящая в A. Мы будем читать сокращение $\exists! x A$ как «существует единственный x такой, что A» или «существует единственный x, удовлетворяющий A».

Поясним, как семантически понимается формула $\exists! x A$. Пусть заданы произвольная нормальная интерпретация $M \leftrightharpoons \langle \mathcal{D}, \mu \rangle$ языка Ω и произвольная оценка ν. Тогда $M, \nu \vDash \exists! x A$, если и только если существует единственный элемент $\alpha \in \mathcal{D}$ такой, что $M, \nu^x_\alpha \vDash A$. Интуитивно говоря, в формуле $\exists! x A$ подформула $\exists x A$ утверждает существование x, удовлетворяющего A, а подформула $\forall x \forall y (A \wedge [A]^x_y \supset x = y)$ говорит о том, что любые два x и y, удовлетворяющие A, равны.

Упражнение 3.2.13. Можно ли отменить или ослабить ограничение «y не входит в A» для формулы, обозначенной через $\exists! x A$? При этом, конечно, требуется, чтобы $\exists! x A$ продолжала выражать семантически понимаемые существование и единственность x, удовлетворяющего A. ◁

Упражнение 3.2.14. Докажите, что в любой теории с равенством выводимы формулы:

$$\exists! x A \equiv \exists x (A \wedge \forall y ([A]^x_y \supset x = y)) \quad \text{и} \quad \exists! x A \equiv \exists y \forall x (A \equiv (x = y)),$$

если переменная y не входит в A. (Таким образом, вместо формулы, обозначенной через $\exists! x A$ до этого упражнения, можно использовать любую из двух указанных формул, стоящих справа от «$\exists! x A \equiv$».) ◁

3.3. Элементарная арифметика

В данном разделе мы определим теорию с равенством, служащую для формального доказательства свойств натуральных чисел, и продемонстрируем содержательные и формальные доказательства в этой теории.

3.3.1. Формулировка теории

Элементарная арифметика, иначе называемая *арифметикой Пеано*, есть теория в языке Ar (см. раздел 2.2.2). Эту теорию мы будем обозначать через PA.[2] Перечислим собственные аксиомы теории PA.

Аксиомы равенства:

$E1)$ $\forall x(x = x)$,

$E2)$ $\forall x \forall y(x = y \supset y = x)$,

$E3)$ $\forall x \forall y \forall z(x = y \wedge y = z \supset x = z)$.

$E4)$ $\forall x \forall y(x = y \supset Sx = Sy)$,

$E5)$ $\forall x_1 \forall x_2 \forall y_1 \forall y_2(x_1 = y_1 \wedge x_2 = y_2 \supset x_1 + x_2 = y_1 + y_2)$,

$E6)$ $\forall x_1 \forall x_2 \forall y_1 \forall y_2(x_1 = y_1 \wedge x_2 = y_2 \supset x_1 \cdot x_2 = y_1 \cdot y_2)$.

Аксиомы Пеано:

$P1)$ $\forall x(Sx \neq 0)$,

$P2)$ $\forall x \forall y(Sx = Sy \supset x = y)$,

$P3)$ формулы вида

$$\Bbb{W}([A]_0^x \wedge \forall x(A \supset [A]_{Sx}^x) \supset \forall x A)$$

для любой формулы A языка Ar и любой переменной x, так что пункт $P3$ представляет схему аксиом, задающую бесконечное число аксиом, каждая из которых называется *аксиомой индукции*.

Аксиомы, определяющие сложение и умножение:

$A1)$ $\forall x(x + 0 = x)$,

$A2)$ $\forall x \forall y(x + Sy = S(x + y))$,

$M1)$ $\forall x(x \cdot 0 = 0)$,

$M2)$ $\forall x \forall y(x \cdot Sy = x \cdot y + x)$.

На этом определение теории PA закончено.

Отметим, что формула пункта $P3$ формализует метод математической индукции для доказательства утверждения, записанного в виде формулы $\forall x A$ языка Ar.

Очевидно, стандартная интерпретация ω языка Ar (ω определена в разделе 2.2.2) является нормальной моделью теории PA, следовательно, по теореме 3.1.5 любая формула, выводимая в PA, истинна в ω. Собственные аксиомы теории PA подобраны с тем, чтобы в PA выводились практически все верные утверждения о натуральных числах, записанные в виде формул языка Ar (иначе говоря, чтобы в PA выводились практически все формулы языка Ar, истинные в интерпретации ω). Однако, как мы увидим в разделе 5.5.4, существует истинная в ω формула, невыводимая в PA, но построить такую формулу совсем непросто.

[2] PA — сокращение «Peano Arithmetic».

3.3.2. Нестандартная модель арифметики

Элементарная арифметика сформулирована так, что стандартная интерпретация ω языка Ar является нормальной моделью этой теории. Однако, как мы покажем в данном разделе, теория PA имеет также модель, значительно отличающуюся от стандартной модели ω.

Для любого натурального числа m положим

$$\widetilde{m} \leftrightharpoons \underbrace{SS\ldots S}_{m}0$$

(в терме $SS\ldots S0$, стоящем справа от $\leftrightharpoons$, имеется ровно m вхождений символа S). Пусть

$$x < y \leftrightharpoons \exists z(y = x + z) \wedge (x \neq y). \qquad (*)$$

Расширим сигнатуру языка Ar новой предметной константой c и для каждого $m \in \mathbb{N}$ добавим к теории PA собственную аксиому $\widetilde{m} < c$. Полученную теорию обозначим через T. Любое конечное подмножество Γ_0 множества всех собственных аксиом теории T имеет нормальную модель. Действительно, найдётся такое натуральное число n, что Γ_0 содержит аксиомы вида $\widetilde{m} < c$ лишь с m, меньшим n. Тогда в качестве нормальной модели множества Γ_0 можно взять ω, дополнительно сопоставив предметной константе c число n.

По теореме 3.2.10 о компактности для нормальных моделей существует нормальная модель ω_c теории T. Очевидно, ω_c является и нормальной моделью теории PA. Однако модель ω_c существенно отличается от стандартной модели ω. Уточним, что значит две модели теории существенно отличаются друг от друг; такие модели мы назовём неизоморфными.

Следующее определение обобщает известное из алгебры понятие изоморфизма групп. (Напомним, что группа — это интерпретация некоторого языка первого порядка, см. пример 3.2.4.)

Определение 3.3.1. Пусть Ω — язык первого порядка, $M_1 \leftrightharpoons \langle \mathcal{D}_1, \mu_1 \rangle$ и $M_2 \leftrightharpoons \langle \mathcal{D}_2, \mu_2 \rangle$ — интерпретации этого языка. Для более наглядной записи в текущем определении введём сокращения: $P_i \leftrightharpoons \mu_i(P)$, $f_i \leftrightharpoons \mu_i(f)$, где P — предикатный символ, f — функциональный символ, $i = 1, 2$.

Биекция $\varphi : \mathcal{D}_1 \to \mathcal{D}_2$ называется *изоморфизмом из M_1 в M_2*, если для любого натурального n, любых n-местных предикатного символа P и функционального символа f языка Ω и любых $\alpha_1, \ldots, \alpha_n \in \mathcal{D}_1$ выполняется

1) $P_1(\alpha_1, \ldots, \alpha_n) = P_2(\varphi(\alpha_1), \ldots, \varphi(\alpha_n))$ при $n > 0$ и
   $P_1 = P_2$ при $n = 0$, и

2) $\varphi(f_1(\alpha_1, \ldots, \alpha_n)) = f_2(\varphi(\alpha_1), \ldots, \varphi(\alpha_n))$ при $n > 0$ и
   $\varphi(f_1) = f_2$ при $n = 0$.

Интерпретации M_1 и M_2 называются *изоморфными*, если существует изоморфизм из M_1 в M_2, иначе они называются *неизоморфными*. ◁

Упражнение 3.3.2. Докажите, что если интерпретации M_1 и M_2 языка Ω изоморфны, то для любой формулы A языка Ω верно следующее: $M_1 \vDash A$ тогда и только тогда, когда $M_2 \vDash A$. (Следовательно, в изоморфных интерпретациях истинны одни и те же формулы.) ◁

Покажем, что интерпретации $\omega \leftrightharpoons \langle \mathbb{N}, \mu_1 \rangle$ и $\omega_c \leftrightharpoons \langle \mathcal{D}_2, \mu_2 \rangle$ неизоморфны. Предположим, что существует изоморфизм $\varphi : \mathbb{N} \to \mathcal{D}_2$ из ω в ω_c. Тогда для некоторого натурального m верно $\varphi(m) = \mu_2(c)$. С другой стороны, введя сокращения $S_i \leftrightharpoons \mu_i(S)$, $0_i \leftrightharpoons \mu_i(0)$ $(i = 1, 2)$ и иногда опуская скобки вокруг единственного аргумента функци, имеем

$$\varphi(m) = \varphi(|\underbrace{SS\ldots S}_{m}0|_\omega) = \varphi(\underbrace{S_1S_1\ldots S_1}_{m}0_1) = \underbrace{S_2S_2\ldots S_2}_{m}0_2 = |\underbrace{SS\ldots S}_{m}0|_{\omega_c} = \mu_2(\widetilde{m}).$$

Следовательно, $\mu_2(\widetilde{m}) = \mu_2(c)$, но это невозможно, поскольку $\omega_c \vDash \widetilde{m} \neq c$ в силу того, что в модели ω_c теории T истинна собственная аксиома $\widetilde{m} < c$ этой теории. Полученное противоречие показывает, что изоморфизма из ω в ω_c не существует.

Любую нормальную модель теории PA, неизоморфную стандартной модели ω, называют *нестандартной* моделью этой теории. Таким образом, ω_c является нестандартной моделью теории PA.

3.3.3. Содержательные и формальные доказательства

Докажем, например, что в теории PA имеет место $SS0 + S0 = SSS0$.

По аксиоме $A2$ получаем $(\star)$ $SS0 + S0 = S(SS0 + 0)$. Далее, в силу аксиомы $A1$ имеем $SS0 + 0 = SS0$. Следовательно, $S(SS0 + 0) = S(SS0)$. Из последнего равенства и $(\star)$ получаем $SS0 + S0 = SSS0$, что и требовалось доказать. Это пример так называемого содержательного (или неформального) доказательства.

Отметим, что в этом доказательстве мы рассматриваем термы вида $SS\ldots S0$ как некоторые объекты, удовлетворяющие определённым условиям, выраженным в виде аксиом $P1$–$P3$, $A1$, $A2$, $M1$, $M2$ на формальном языке. Для построения такого доказательства было бы достаточно выразить эти условия на естественном (русском) языке, с обычными математическими обозначениями. Например, аксиому $A1$ достаточно было записать так: «для всякого x выполняется $x + 0 = x$». При любом из указанных способов выражения этих условий, последние мы понимаем содержательно, как истинные утверждения, обычным для математической практики образом. Некоторые факты, например, свойства равенства, мы считаем очевидными и применяем их без явного упоминания. Такие доказательства характерны для так называемого *содержательно аксиоматического метода*, широко используемого в математике.

Математическая логика призвана предоставить основания для содержательных доказательств, выявить и формализовать способы рассуждений, применимых в таких доказательствах. С этой целью условия, которым должны удовлетворять рассматриваемые объекты, записывают на формальном языке, например, языке первого порядка; принятые способы рассуждений формализуют с помощью исчисления; таким образом, определяют формальную аксиоматическую теорию. Для получения записанных на формальном языке утверждений об объектах используют формальный вывод. В этом и заключается так называемый *формально аксиоматический метод*.

Повсеместно в математике применять формально аксиоматический метод нецелесообразно хотя бы из-за его громоздкости. Даже в исследованиях по математической логике нередко удовлетворяются содержательным доказательством того, что существует формальное доказательство требуемого утверждения. Однако, применяя содержательно аксиоматический метод, следует понимать, как содержательное доказательство может быть формализовано, т. е. как, имея содержательное доказательство утверждения, можно построить формальное доказательство этого утверждения, записанного на формальном языке. Необходимость в формализации содержательного доказательства возникает, например, при попытке тщательно проверить некоторые шаги доказательства. Если такую проверку или же поиск доказательства поручают компьютеру, то необходимость в формализации очевидна. Мы продолжим обсуждение связи между содержательными и формальными доказательствами в разделе 3.5.5.

Приведём формальное доказательство того, что в теории PA имеет место $SS0 + S0 = SSS0$, т. е. построим вывод формулы $SS0 + S0 = SSS0$ в теории PA. Наметим этапы этого вывода: сначала мы, используя аксиому $A2$, выведем $(\star)$ $SS0 + S0 = S(SS0 + 0)$ (см. формулы 1–5 ниже); далее, с помощью аксиомы $A1$ выведем $SS0 + 0 = SS0$ (см. формулы 6–8); затем, используя аксиому $E4$, выведем $(\star\star)$ $S(SS0 + 0) = SSS0$ (см. формулы 9–14); наконец, с помощью аксиомы $E3$ из $(\star)$ и $(\star\star)$ получим $SS0 + S0 = SSS0$ (см. формулы 15–25).

1) $\forall x \forall y(x + Sy = S(x + y))$ — собственная аксиома $A2$ теории PA;

2) $\forall x \forall y(x + Sy = S(x + y)) \supset \forall y(SS0 + Sy = S(SS0 + y))$ — аксиома вида 13 исчисления предикатов $\mathcal{H}\forall(Ar)$;

3) $\forall y(SS0 + Sy = S(SS0 + y))$ из 1 и 2 по MP;

4) $\forall y(SS0 + Sy = S(SS0 + y)) \supset SS0 + S0 = S(SS0 + 0)$ — аксиома вида 13;

5) $SS0 + S0 = S(SS0 + 0)$ из 3 и 4 по MP;

6) $\forall x(x + 0 = x)$ — аксиома $A1$;

7) $\forall x(x + 0 = x) \supset SS0 + 0 = SS0$ — аксиома вида 13;

8) $SS0 + 0 = SS0$ из 6 и 7 по MP;

9) $\forall x \forall y(x = y \supset Sx = Sy)$ — аксиома $E4$;

10) $\forall x \forall y(x = y \supset Sx = Sy) \supset \forall y(SS0 + 0 = y \supset S(SS0 + 0) = Sy)$ — аксиома вида 13;

11) $\forall y(SS0 + 0 = y \supset S(SS0 + 0) = Sy)$ из 9 и 10 по MP;

12) $\forall y(SS0 + 0 = y \supset S(SS0 + 0) = Sy) \supset$
$(SS0 + 0 = SS0 \supset S(SS0 + 0) = SSS0)$ — аксиома вида 13;

13) $SS0 + 0 = SS0 \supset S(SS0 + 0) = SSS0$ из 11 и 12 по MP;

14) $S(SS0 + 0) = SSS0$ из 8 и 13 по MP;

15) $\forall x \forall y \forall z(x = y \wedge y = z \supset x = z)$ — аксиома $E3$;

16) $\forall x\forall y\forall z(x = y \wedge y = z \supset x = z) \supset$
$\forall y\forall z(SS0 + S0 = y \wedge y = z \supset SS0 + S0 = z)$ — аксиома вида 13;

17) $\forall y\forall z(SS0 + S0 = y \wedge y = z \supset SS0 + S0 = z)$ из 15 и 16 по MP;

18) $\forall y\forall z(SS0 + S0 = y \wedge y = z \supset SS0 + S0 = z) \supset$
$\forall z(SS0 + S0 = S(SS0 + 0) \wedge S(SS0 + 0) = z \supset SS0 + S0 = z)$ — аксиома вида 13;

19) $\forall z(SS0 + S0 = S(SS0 + 0) \wedge S(SS0 + 0) = z \supset SS0 + S0 = z)$ из 17 и 18 по MP;

20) $\forall z(SS0 + S0 = S(SS0 + 0) \wedge S(SS0 + 0) = z \supset SS0 + S0 = z) \supset$
$(SS0 + S0 = S(SS0 + 0) \wedge S(SS0 + 0) = SSS0 \supset SS0 + S0 = SSS0)$ — аксиома вида 13;

21) $SS0 + S0 = S(SS0 + 0) \wedge S(SS0 + 0) = SSS0 \supset SS0 + S0 = SSS0$ из 19 и 20 по MP;

22) $SS0 + S0 = S(SS0 + 0) \supset (S(SS0 + 0) = SSS0 \supset$
$SS0 + S0 = S(SS0 + 0) \wedge S(SS0 + 0) = SSS0)$ — аксиома вида 3;

23) $S(SS0 + 0) = SSS0 \supset SS0 + S0 = S(SS0 + 0) \wedge S(SS0 + 0) = SSS0$ из 5 и 22 по MP;

24) $SS0 + S0 = S(SS0 + 0) \wedge S(SS0 + 0) = SSS0$ из 14 и 23 по MP;

25) $SS0 + S0 = SSS0$ из 21 и 24 по MP.

Упражнение 3.3.3. Докажите как содержательно, так и формально, что $PA \vdash S0 \neq 0$ и $PA \vdash S0 \neq SS0$. ◁

Упражнение 3.3.4. Используя допустимые правила, докажите, что $PA \vdash SS0 + S0 = SSS0$ и $PA \vdash SS0 \cdot SS0 = SSSS0$. (Тем самым будут получены содержательные, но довольно подробные доказательства выводимости в теории PA данных формул.) ◁

Как мы уже отмечали, теория PA определена так, чтобы в ней были выводимы практически все верные утверждения о натуральных числах, сформулированные на языке Ar. Приведём содержательное доказательство коммутативности сложения, т. е. того, что в PA верно $x + y = y + x$ для любых x и y. Это содержательное доказательство можно формализовать, построив вывод данной формулы в теории PA.

Обозначим через A формулу $x + y = y + x$. Для того, чтобы с помощью аксиомы индукции $P3$ получить A, докажем $[A]^x_0$ (базу индукции) и $A \supset [A]^x_{Sx}$ (индукционный переход).

Доказательство верности $B \leftrightharpoons [A]^x_0$, т. е. $0 + y = y + 0$, проведём так же — с помощью аксиомы индукции. Для этого установим $[B]^y_0$ и $B \supset [B]^y_{Sy}$. $[B]^y_0$ есть $0 + 0 = 0 + 0$ и, очевидно, верно по аксиоме $A1$. $B \supset [B]^y_{Sy}$ есть

$$0 + y = y + 0 \supset 0 + Sy = Sy + 0$$

и верно в силу равенств

$$0 + Sy = S(0 + y) = S(y + 0) = Sy = Sy + 0,^3$$

[3]Напомним, что цепочка равенств $t_1 = t_2 = t_3$ в содержательных доказательствах понимается так: $t_1 = t_2$ и $t_2 = t_3$. Аналогично понимаются и цепочки с бóльшим числом равенств.

где первое равенство имеет место по аксиоме $A2$, второе является следствием посылки $0 + y = y + 0$ вышеприведённой импликации, а третье и четвёртое выполняются по аксиоме $A1$.

Обратимся к $A \supset [A]^x_{Sx}$, т. е. к

$$x + y = y + x \supset Sx + y = y + Sx.$$

Из аксиомы $A2$ и посылки $x + y = y + x$ этой импликации следует

$$y + Sx = S(y + x) = S(x + y),$$

поэтому для завершения доказательства достаточно показать, что

$$Sx + y = S(x + y).$$

Чтобы по аксиоме индукции получить $C \leftrightharpoons Sx + y = S(x + y)$, установим $[C]^y_0$ и $C \supset [C]^y_{Sy}$. $[C]^y_0$ есть $Sx + 0 = S(x + 0)$ и верно в силу аксиомы $A1$. $C \supset [C]^y_{Sy}$ есть

$$Sx + y = S(x + y) \supset Sx + Sy = S(x + Sy)$$

и верно в силу равенств

$$Sx + Sy = S(Sx + y) = SS(x + y) = S(x + Sy),$$

первое и третье из которых имеют место по аксиоме $A2$, а второе — следствие посылки $Sx + y = S(x + y)$ предыдущей импликации. Этим завершается доказательство коммутативности сложения.

Упражнение 3.3.5. Приведите содержательное доказательство ассоциативности сложения ($(x + y) + z = x + (y + z)$), коммутативности и ассоциативности умножения ($x \cdot y = y \cdot x$ и $(x \cdot y) \cdot z = x \cdot (y \cdot z)$), закона дистрибутивности ($x \cdot (y + z) = x \cdot y + x \cdot z$). ◁

Упражнение 3.3.6. Приведите содержательное доказательство выводимости в PA формулы

$$\forall x(\forall y(y < x \supset [A]^x_y) \supset A) \supset \forall x A, \qquad (**)$$

где A — любая формула языка Ar, x и y — любые переменные, причём y не входит в A, сокращение $x < y$ определено в разделе 3.3.2 (см. $(*)$). (Отметим, что $(**)$ формализует метод возвратной индукции для доказательства утверждения, записанного в виде формулы $\forall x A$ языка Ar.) ◁

3.4. Наивная теория множеств

Наша цель — описать формальную аксиоматическую теорию первого порядка, которая формализует широко используемую в математике «теорию множеств» и служит основанием классического направления в математике. Такая формальная теория будет описана в разделе 3.5, а в данном разделе мы изложим некоторые предпосылки её создания.

3.4.1. Язык и аксиомы наивной теории множеств

Начнём с напоминания о *наивной теории множеств*. В рамках наивной теории множеств под *множеством* понимают совокупность объектов, мыслимую как единое целое. Каждый объект этой совокупности называют *элементом* данного множества. То, что s является элементом множества t, обозначают через $s \in t$. Если $s \in t$, то говорят также, что s *принадлежит* t и что t *содержит элемент* s. Явно отметим, что наивная теория множеств даёт лишь интуитивное понятие множества.

Зададим язык наивной теории множеств, состоящий из определённых ниже термов и формул:

1) предметная переменная является термом;

2) если s и t — термы, то $(s = t)$ и $(s \in t)$ являются формулами;

3) если A и B — формулы, x — предметная переменная, то $\neg A$, $(A \wedge B)$, $(A \vee B)$, $(A \supset B)$, $\forall x A$ и $\exists x A$ являются формулами;

4) если A — формула, x — предметная переменная, то $\{x \mid A\}$ является термом.

Например, $\{x \mid \neg(x \in y)\} = z$ является формулой этого языка.

Предметные переменые языка наивной теории множеств служат для обозначения множеств. Этот язык не является языком первого порядка, потому что согласно пункту 4 предыдущего определения формула может входить в терм. И *наивная теория множеств не является формальной аксиоматической теорией первого порядка* в смысле данного нами определения 3.1.1. Однако мы, руководствуясь очевидной аналогией, будем применять некоторые термины, введённые для языка первого порядка, и к языку наивной теории множеств.

Терм $\{x \mid A\}$ называется *свёрткой по формуле* A; все вхождения переменной x в этот терм считаются связанными: «$x \mid$» играет роль кванторной приставки.

К собственным аксиомам наивной теории множеств, кроме аксиом равенства, относятся следующие:

1) аксиома объёмности (экстенсиональности):

$$\forall x \forall y (\forall z (z \in x \equiv z \in y) \supset x = y).$$

Выражаясь иначе, любые два множества x и y, содержащие одни и те же элементы, равны;

2) схема аксиом свёртывания:

$$\forall\!\!\forall y ((y \in \{x \mid A\}) \equiv [A]^x_y),$$

где A — любая формула языка наивной теории множеств, x и y — любые переменные, причём y не входит в A. Иначе говоря, $\{x \mid A\}$ есть множество всех x, для которых истинна A.

Эти аксиомы отражают интуитивные свойства множеств.

Теперь можно ввести все обычные теоретико-множественные определения: определения пустого множества, множества всех подмножеств, объединения, пересечения и декартова произведения множеств, отношения, функции и т. д. Например, пустое множество есть по определению $\{x \mid \neg(x = x)\}$, а объединение двух множеств x и y есть по определению $\{z \mid z \in x \vee z \in y\}$. Также в рамках этой теории можно определить множества всех натуральных, целых, рациональных и вещественных чисел.

В конце девятнадцатого века Г. Кантор создал «учение о множествах», которое также называли *канторовской теорией множеств* и позже стали называть наивной теорией множеств. (Эту неформальную теорию мы частично изложили подобно формальной теории.) Представлялось, что вся математика может быть построена на основе канторовской теории множеств, но к началу двадцатого века в этой теории были обнаружены противоречия, иначе называемые парадоксами. Противоречие есть пара утверждений (быть может, записанных в виде формул), одно из которых является отрицанием другого. Противоречие опасно тем, что из него путём правильных математических рассуждений можно получить любое утверждение (что для формальной теории уточнено в лемме 3.1.6). Один из парадоксов канторовской теории множеств мы сейчас рассмотрим.

3.4.2. Парадокс Рассела

Рассмотрим так называемое множество Рассела $u \leftrightharpoons \{x \mid \neg(x \in x)\}$, т. е. множество всех множеств, которые не являются элементами самих себя. По аксиоме свёртывания для любого y верно $y \in u \equiv \neg(y \in y)$. Отсюда, взяв в качестве y множество u, получаем

$$u \in u \equiv \neg(u \in u). \qquad (*)$$

Предположив, что $u \in u$, из $(*)$ получим $\neg(u \in u)$, что противоречит предположению; поэтому имеем $\neg(u \in u)$ (отметим, что мы использовали способ доказательства «приведение к абсурду», см. также раздел 1.3.3). Теперь уже доказанные $\neg(u \in u)$ и $(*)$ дают $u \in u$. Таким образом, мы получили противоречие $u \in u$ и $\neg(u \in u)$, называемое парадоксом Рассела. Кроме того, и проведённое рассуждение называют парадоксом Рассела.

Популярный вариант этого парадокса таков: «Парикмахер, живущий в деревне, бреет всех тех и только тех жителей этой деревни, которые не бреют себя сами. Бреет ли этот парикмахер себя сам?» Очевидно, что из предположения о том, что этот парикмахер бреет себя сам, следует, что он не бреет себя сам, и наоборот. Полученное противоречие показывает, что такого парикмахера не существует. Аналогичным образом решается проблема с множеством Рассела: это множество не должно существовать. Однако в канторовской теории множеств существует множество Рассела, поэтому следует отказаться от этой теории (или пересмотреть её, тем самым создав другую теорию).

3.5. Теория множеств Цермело-Френкеля ZF

Парадокс Рассела и другие обнаруженные в канторовской теории множеств парадоксы выявили необходимость в пересмотре этой теории и вообще оснований математики для того, чтобы избежать известных парадоксов и, по возможности, создать надёжные основания математики. Было предложено несколько подходов к решению этой проблемы, среди которых так называемые теория типов, интуиционизм, конструктивизм, но наибольшее признание получил подход, называемый теорией множеств Цермело-Френкеля. Идея этого подхода состоит в том, что ограничиваются рассмотрением множеств, существование которых обеспечивается некоторыми аксиомами, причём не должно быть видно, как известные парадоксы могли бы следовать из этих аксиом, а сами эти аксиомы должны быть достаточны для доказательства всех верных математичеких фактов.

3.5.1. Формулировка теории ZF

Опишем так называемую *теорию множеств Цермело-Френкеля*, являющуюся формальной аксиоматической теорией первого порядка с равенством. Эту теорию обозначают через ZF.[4] Сигнатура языка этой теории содержит только двухместные предикатные символы равенства $=$ и принадлежности $\in$. Будем сокращённо записывать $\neg(s \in t)$ как $(s \notin t)$ или как $s \notin t$.

Собственные аксиомы теории ZF с пояснениями будут перечислены ниже. Подчеркнём, что аксиомы являются формулами, а сопутствующие формулировки аксиом на русском языке предназначены лишь для облегчения восприятия. Встречающиеся далее в разделе 3.5 выражения типа «имеет место (верно) A» и просто «A», где A — формула или сокращённая запись формулы языка теории ZF, строго говоря, надлежит понимать как «A выводима в ZF», если не указано иное. Однако удобно подразумевать, что предметные переменные в формулах служат для обозначения множеств из некоторого семейства множеств[5]. Тогда такие выражения можно понимать и буквально как утверждения об истинности A, обычным для содержательных рассуждений образом.

Сейчас мы перечислим собственные аксиомы теории ZF (кроме легко записываемых аксиом равенства) и введём сопутствующие определения и обозначения.

1. Аксиома объёмности (экстенсиональности). Любые два множества x и y, содержащие одни и те же элементы, равны:

$$\forall x \forall y (\forall z (z \in x \equiv z \in y) \supset x = y).$$

Легко проверить, что импликация в обратную сторону, точнее, формула $\forall x \forall y (x = y \supset \forall z (z \in x \equiv z \in y))$ выводима из аксиом равенства.

[4] Аббревиатура ZF образована из первых букв фамилий создателей этой теории — Zermelo и Fraenkel.

[5] Для благозвучия удобно употреблять «семейство множеств» вместо «множество множеств», что мы и делаем.

2. Аксиома пустого множества. Существует множество x, не содержащее ни одного элемента:

$$\exists x \forall y (y \notin x).$$

Множество x, не содержащее ни одного элемента (т. е. для которого $\forall y(y \notin x)$), называется *пустым множеством*.

Докажем, что пустое множество, существование которого утверждает эта аксиома, единственно, т. е. докажем[6] $\exists! x \forall y (y \notin x)$. Если бы нашлись два множества x и x_1 такие, что $\forall y(y \notin x)$ и $\forall y(y \notin x_1)$, то имело бы место $\forall y(y \notin x \equiv y \notin x_1)$. Следовательно, $\forall y(y \in x \equiv y \in x_1)$, и тогда по аксиоме объёмности $x = x_1$, что и требовалось доказать.

Можно добавить в сигнатуру языка теории ZF нульместный функциональный символ $\varnothing$, обозначающий пустое множество. Тогда аксиому пустого множества можно записать в виде

$$\forall y (y \notin \varnothing).$$

Далее для удобства мы будем вводить и другие функциональные символы, но формулы, в которые входят новые функциональные символы, будут являться просто сокращениями формул языка теории ZF. Например, запись $w = \varnothing$ понимается как $\forall y(y \notin w)$.

3. Аксиома пары. Для любых множеств x и y существует множество z, содержащее в точности элементы x и y:

$$\forall x \forall y \exists z \forall w (w \in z \equiv (w = x \vee w = y)).$$

Упражнение 3.5.1. Используя аксиомы, сформулированные выше в текущем разделе 3.5.1 докажите, что такое множество z единственно, точнее, $\forall x \forall y \exists! z \forall w (w \in z \equiv (w = x \vee w = y))$. ◁

Это множество z называется *неупорядоченной парой* x и y и обозначается $\{x, y\}$. Данное обозначение может рассматриваться как сокращённая запись нового двухместного функционального символа h с аргументами: $h(x, y)$. С использованием обозначения неупорядоченной пары аксиому пары можно записать так:

$$\forall x \forall y \forall w (w \in \{x, y\} \equiv (w = x \vee w = y)).$$

Для любого множества x множество $\{x\} \leftrightharpoons \{x, x\}$ называется *одноэлементным множеством*. Очевидно, что единственным элементом множества $\{x\}$ является x.

Множество

$$\langle x, y \rangle \leftrightharpoons \{\{x\}, \{x, y\}\}$$

будем называть *упорядоченной парой* x и y. x и y назовём *первым* и, соответственно, *вторым членом упорядоченной пары* $\langle x, y \rangle$. Основное свойство упорядоченной пары сформулировано в следующей теореме.

Теорема 3.5.2. $\forall x \forall y \forall u \forall v (\langle x, y \rangle = \langle u, v \rangle \equiv (x = u \wedge y = v))$.

[6] Здесь и далее в разделе 3.5 мы приводим содержательные, а не формальные доказательства с целью сокращения записи доказательств и облегчения их восприятия.

Д о к а з а т е л ь с т в о. Импликация $(x = u \wedge y = v) \supset \langle x, y\rangle = \langle u, v\rangle$, очевидно, следует из аксиомы пары и аксиомы объёмности.

Пусть теперь $\langle x, y\rangle = \langle u, v\rangle$, т. е.

$$\{\{x\}, \{x, y\}\} = \{\{u\}, \{u, v\}\}.$$

С помощью аксиомы пары получаем (I) $\{u\} \in \langle x, y\rangle$ и (II) $\{u, v\} \in \langle x, y\rangle$, поэтому по той же аксиоме имеем:

(I) (1) $\{u\} = \{x\}$ или (2) $\{u\} = \{x, y\}$

и

(II) (3) $\{u, v\} = \{x\}$ или (4) $\{u, v\} = \{x, y\}$.

Из (2) следует, что $x = y = u$. Тогда и (3), и (4) принимают вид $\{x, v\} = \{x\}$, отсюда $x = v$. Таким образом, в случае, когда выполняется (2), имеем $x = y = u = v$, следовательно, верно $(x = u \wedge y = v)$.

Аналогично рассматривается случай, когда выполняется (3).

Для завершения доказательства осталось рассмотреть случай, когда одновременно выполняются (1) и (4). (1) влечёт $u = x$, а (4) влечёт $v = x$ или $v = y$. Если $u = x$ и $v = x$, то (4) принимает вид $\{x\} = \{x, y\}$, поэтому $u = v = x = y$, следовательно, имеет место $(x = u \wedge y = v)$. Иначе, т. е. если $u = x$ и $v = y$, требуемое $(x = u \wedge y = v)$ очевидно. $\triangleleft$

Упорядоченные тройки, четвёрки и вообще *упорядоченные* $(n + 1)$*-ки* $(n = 2, 3, \ldots)$ определим таким образом:

$$\langle x_1, \ldots, x_n, x_{n+1}\rangle \leftrightharpoons \langle\langle x_1, \ldots, x_n\rangle, x_{n+1}\rangle,$$

причём для удобства упорядоченной n-кой при $n = 1$ считаем $\langle x_1\rangle \leftrightharpoons x_1$. Упорядоченные m-ки $(m = 1, 2, 3, \ldots)$ обладают свойством, аналогичным только что установленному свойству упорядоченной пары.

4. Аксиома объединения (суммы). Для любого семейства множеств x существует множество u, элементами которого являются в точности элементы множеств, принадлежащих x:

$$\forall x \exists u \forall y (y \in u \equiv \exists z (z \in x \wedge y \in z)).$$

Упражнение 3.5.3. Докажите, что такое множество u единственно, используя сформулированные выше аксиомы. $\triangleleft$

Это множество u называется *объединением элементов множества* x и обозначается $\cup x$. Другим традиционным обозначением такого объединения является $\cup_{z \in x} z$. Используя обозначение $\cup x$, аксиому объединения можно записать в виде

$$\forall x \forall y (y \in \cup x \equiv \exists z (z \in x \wedge y \in z)).$$

В силу аксиомы пары и аксиомы объединения для любых множеств x_1 и x_2 существует множество

$$x_1 \cup x_2 \leftrightharpoons \cup\{x_1, x_2\},$$

называемое *объединением множеств* x_1 и x_2. (Сравните:

$$\cup\{\{x_1\},\{x_2\}\} = \{x_1, x_2\}\ .)$$

Для любых множеств x_1, x_2 и x_3 существует множество

$$x_1 \cup x_2 \cup x_3 \leftrightharpoons (x_1 \cup x_2) \cup x_3,$$

называемое объединением множеств x_1, x_2 и x_3. Аналогично можно определить объединение большего числа множеств. Также можно определить

$$\{x_1, x_2, x_3\} \leftrightharpoons \cup\{\{x_1, x_2\}, \{x_3\}\}, \quad \{x_1, x_2, x_3, x_4\} \leftrightharpoons \cup\{\{x_1, x_2, x_3\}, \{x_4\}\}$$

и т. д.

Упражнение 3.5.4. Докажите, что

$$\forall x \forall y \forall z (z \in x \cup y \equiv z \in x \vee z \in y),$$
$$\forall x \forall y \forall z ((x \cup y) \cup z = x \cup (y \cup z)),$$
$$\forall x \forall y \forall z (\{x, y, z\} = \{z, x, y, y\}).$$

◁

Прежде чем сформулировать следующую аксиому, введём обозначение:

$$x \subseteq y \leftrightharpoons \forall z (z \in x \supset z \in y).$$

Будем говорить, что множество x есть *подмножество* множества y (x *включено в* y, x *содержится в* y или y *содержит подмножество* x), если $x \subseteq y$, т. е. если любой элемент множества x является элементом множества y.

5. Аксиома степени (множества всех подмножеств). Для любого множества x существует множество y, элементами которого являются все подмножества множества x и только они:

$$\forall x \exists y \forall z (z \in y \equiv z \subseteq x).$$

Упражнение 3.5.5. Докажите, что такое множество y единственно, используя сформулированные выше аксиомы. ◁

Это множество y называется *степенью* (или *множеством всех подмножеств*) *множества* x и обозначается 2^x. Используя это обозначение, аксиому степени можно записать в виде

$$\forall x \forall z (z \in 2^x \equiv z \subseteq x).$$

6. Аксиома бесконечности. Существует множество x, которое имеет своим элементом пустое множество, и для каждого своего элемента $y \in x$ содержит также элемент $y \cup \{y\}$:

$$\exists x\, Ind(x), \quad \text{где} \quad Ind(x) \leftrightharpoons (\varnothing \in x \wedge \forall y (y \in x \supset y \cup \{y\} \in x)),$$

с использованием вышеприведённых обозначений, а непосредственно на языке теории ZF $Ind(x)$ записывается так:

$$\forall w (\forall z (z \notin w) \supset w \in x) \wedge \forall y (y \in x \supset \forall u (\forall v (v \in u \equiv v \in y \vee v = y) \supset u \in x)).$$

Если для множества x имеет место $Ind(x)$, то x называется *индуктивным*.

7. Схема аксиом выделения. Для любого множества x существует множество y, содержащее в точности элементы множества x, которые выполняют условие (т. е. формулу) A:

$$\forall\forall x\exists y\forall z(z \in y \equiv (z \in x \wedge A)),$$

где A — любая формула языка теории ZF, x, y, z — любые (попарно различные[7]) переменные, причём y не входит свободно в A.

Если A имеет параметры, отличные от z, то множество y зависит от этих параметров. (Сама переменная z может как быть, так и не быть параметром формулы A.) Легко видеть, что множество y, существование которого утверждает аксиома выделения, единственно, точнее, $\forall\forall x\exists! y\forall z(z \in y \equiv (z \in x \wedge A))$. Будем обозначать это множество y как $\{z \in x \mid A\}$. С использованием этого обозначения схему аксиом выделения можно записать так:

$$\forall\forall x\forall w(w \in \{z \in x \mid A\} \equiv w \in x \wedge [A]^z_w),$$

где A — любая формула языка теории ZF, w, x, z — любые (попарно различные) переменные, причём w не входит в A.

По аксиоме выделения для любых множеств x_1 и x_2 существует множество

$$x_1 \cap x_2 \leftrightharpoons \{z \in x_1 \mid z \in x_2\},$$

называемое *пересечением множеств* x_1 и x_2. Для любых множеств x_1, x_2 и x_3 существует множество

$$x_1 \cap x_2 \cap x_3 \leftrightharpoons (x_1 \cap x_2) \cap x_3,$$

называемое пересечением множеств x_1, x_2 и x_3. Аналогично определяется пересечение большего числа множеств. Кроме того, для любого множества x существует множество

$$\cap x \leftrightharpoons \{y \in \cup x \mid \forall z(z \in x \supset y \in z)\},$$

называемое *пересечением элементов множества* x. Другим традиционным обозначением такого пересечения является $\cap_{z\in x} z$.

Разность множеств x и y есть по определению множество

$$x \setminus y \leftrightharpoons \{z \in x \mid z \notin y\}.$$

Упражнение 3.5.6. Докажите несколько общеизвестных законов теории множеств, в частности,

$$\forall x\forall y\forall z((x \cap y) \cap z = x \cap (y \cap z)), \qquad \forall x\forall y\forall z((x \cap y) \cup z = (x \cup z) \cap (y \cup z)),$$
$$\forall x\forall y\forall z((x \cup y) \setminus z = (x \setminus z) \cup (y \setminus z)), \quad \forall x\forall y\forall z(x \setminus (y \cap z) = (x \setminus y) \cup (x \setminus z)).$$

$\triangleleft$

[7] См. соглашение о переменных в начале раздела 3.2.

8. Схема аксиом подстановки (замены). Если для каждого множества u существует единственное множество v такое, что выполняется условие $A(u,v)$ (т. е. формула A с параметрами u и v), то для любого множества x существует множество y, содержащее те и только те элементы v, которые при некотором $u \in x$ выполняют $A(u,v)$:

$$\mathbb{V}(\forall u \exists! v A \supset \forall x \exists y \forall v (v \in y \equiv \exists u (u \in x \wedge A))),$$

где A — любая формула языка теории ZF, u, v, x, y — любые (попарно различные) переменные, причём y не входит свободно в A, и переменные u и v являются параметрами A (не исключено, что A имеет и другие параметры).

Для облегчения первоначального восприятия аксиомы подстановки образование множества y из множества x можно представить более наглядно, но неточно: $x \overset{u\in x}{\dashrightarrow} A(u,v) \overset{v\in y}{\dashrightarrow} y$. Здесь $A(u,v)$ по элементу $u \in x$ «вырабатывает» элемент $v \in y$. Подчеркнём, что в текущем абзаце дано очень грубое пояснение аксиомы подстановки.

С помощью аксиомы объёмности легко показать, что множество y, существование которого утверждает аксиома подстановки при верности $\forall u \exists! v A$, единственно, точнее,

$$\mathbb{V}(\forall u \exists! v A \supset \forall x \exists! y \forall v (v \in y \equiv \exists u (u \in x \wedge A))).$$

Упражнение 3.5.7. Обозначим через ZF_8 теорию в языке ZF со всеми собственными аксиомами, приведёнными выше в текущем разделе 3.5.1. Через $ZF_{8'}$ обозначим теорию, которая получается из ZF_8 путём следующего изменения схемы аксиом подстановки: не потребуем, чтобы u и v были параметрами A. Покажите, что в ZF_8 и $ZF_{8'}$ выводимы одни и те же формулы. $\triangleleft$

9. Аксиома регулярности (фундирования). Всякое непустое множество x содержит такой элемент y, что x и y не пересекаются (т. е. x и y не имеют ни одного общего элемента):

$$\forall x (x \neq \varnothing \supset \exists y (y \in x \wedge x \cap y = \varnothing))$$

в вышеприведённых обозначениях, или непосредственно на языке теории ZF:

$$\forall x (\exists z (z \in x) \supset \exists y (y \in x \wedge \forall w \neg (w \in x \wedge w \in y))).$$

Из этой аксиомы следует, что не существует таких множеств, что

$$x \in x, \quad x \in y \wedge y \in x, \quad x \in y \wedge y \in z \wedge z \in x.$$

Покажем, например, что не существует множеств x и y таких, что $x \in y \wedge y \in x$. Действительно, предположим, что для некоторых множеств x и y верно $x \in y \wedge y \in x$. Тогда существует множество $v \leftrightharpoons \{x, y\}$. $v \cap x \neq \varnothing$, поскольку $y \in v$ и в силу предположения $y \in x$. Аналогично $v \cap y \neq \varnothing$. Итак, множество v непусто, но не содержит элемента, который не пересекается с v, что противоречит аксиоме регулярности. Полученное противоречие доказывает требуемое.

На этом перечисление собственных аксиом теории ZF закончено.

Замечание 3.5.8. В сформулированной выше теории ZF имеются собственные аксиомы, выводимые из остальных аксиом этой теории. Такова, например, аксиома пустого множества. Действительно, аксиома бесконечности[8] обеспечивает существование по крайней мере одного множества, из этого множества по аксиоме выделения можно выделить элементы, выполняющие формулу **F**. Таких элементов не существует, поэтому полученное этим выделением множество x будет обладать свойством пустого множества: $\forall y(y \notin x)$. Мы сформулировали аксиомы теории ZF несколько избыточно и таким образом, чтобы (на наш взгляд) их было легче воспринимать и использовать. $\triangleleft$

3.5.2. Отношения и функции в теории ZF

Теория множеств Цермело-Френкеля была создана с целью служить основанием математики. Не ставя здесь перед собой задачу построения всей математики на основе теории ZF, в данном разделе 3.5.2 и в следующем разделе 3.5.3 мы определим некоторые фундаментальные математические понятия в рамках этой теории.

Декартовым (или *прямым*) *произведением множеств* x и y называется множество

$$x \times y \leftrightharpoons \{z \in 2^{2^{x \cup y}} \mid \exists u \exists v(u \in x \wedge v \in y \wedge z = \langle u, v\rangle\}.$$

Упражнение 3.5.9. Объясните, почему так определённое множество $x \times y$ существует и единственно. $\triangleleft$

Ограничение $z \in 2^{2^{x \cup y}}$ в определении $x \times y$ фиктивно, т. е. упорядоченная пара $z = \langle u, v\rangle$ и так принадлежит $2^{2^{x \cup y}}$, точнее,

$$u \in x \wedge v \in y \supset \langle u, v\rangle \in 2^{2^{x \cup y}}.$$

Действительно, вспомним, что $\langle u, v\rangle \leftrightharpoons \{\{u\}, \{u, v\}\}$. Из $u \in x \wedge v \in y$ получаем $u \in x \cup y$ и $v \in x \cup y$. Тогда $\{u\} \subseteq x \cup y$ и $\{u, v\} \subseteq x \cup y$, поэтому $\{u\} \in 2^{x \cup y}$ и $\{u, v\} \in 2^{x \cup y}$. Отсюда $\{\{u\}, \{u, v\}\} \subseteq 2^{x \cup y}$, т. е. $\langle u, v\rangle \subseteq 2^{x \cup y}$, следовательно, $\langle u, v\rangle \in 2^{2^{x \cup y}}$.

Таким образом, данное определение согласуется с общеизвестным неформальным определением декартова произведения множеств x и y как множества всех пар $\langle u, v\rangle$ таких, что $u \in x$ и $v \in y$.

Декартово произведение $n + 1$ множеств $(n = 2, 3, \ldots)$ определяется следующим образом:

$$x_1 \times \ldots \times x_n \times x_{n+1} \leftrightharpoons (x_1 \times \ldots \times x_n) \times x_{n+1}.$$

Упражнение 3.5.10. Докажите несколько общеизвестных законов теории множеств, в которых фигурирует декартово произведение, в частности,

$$\forall x \forall y \forall z((x \cap y) \times z = (x \times z) \cap (y \times z)), \quad \forall x \forall y \forall z((x \setminus y) \times z = (x \times z) \setminus (y \times z)).$$

$\triangleleft$

[8] Эта аксиома, записанная нами на языке теории ZF, непосредственно не утверждает существование пустого множества.

Обозначим

$$Rel(z) \leftrightharpoons \exists x \exists y (z \subseteq x \times y).$$

Множество z называется *отношением*, если $Rel(z)$. Таким образом, отношение z является подмножеством некоторого декартова произведения, иначе говоря, z есть некоторое множество упорядоченных пар.

Областью определения отношения z называется множество

$$dom(z) \leftrightharpoons \{u \in \cup \cup z \mid \exists v (\langle u, v \rangle \in z)\}.$$

Аналогично тому, как это было сделано выше для декартова произведения, легко проверить, что ограничение $u \in \cup \cup z$ фиктивно. Поэтому $dom(z)$ есть просто множество первых членов всех пар из z.

Областью значений отношения z называется множество

$$rng(z) \leftrightharpoons \{v \in \cup \cup z \mid \exists u (\langle u, v \rangle \in z)\}.$$

И здесь ограничение $u \in \cup \cup z$ фиктивно, а $rng(z)$ есть множество вторых членов всех пар из z.

Любое отношение можно назвать *двухместным* (или *бинарным*) отношением. Если отношение является подмножеством декартова произведения некоторых трёх множеств, то это отношение называется *трёхместным*. Аналогичным образом можно определить отношение большей местности. Заметим, что трёхместное отношение является одновременно и двухместным отношением, но наоборот, — вообще говоря, нет. Любое подмножество множества x мы для удобства считаем *одноместным* отношением на x.

Обозначим

$$Fnc(z) \leftrightharpoons Rel(z) \wedge \forall u \forall v_1 \forall v_2 (\langle u, v_1 \rangle \in z \wedge \langle u, v_2 \rangle \in z \supset v_1 = v_2).$$

Множество z называется *функцией* (*отображением* или *преобразованием*), если $Fnc(z)$. Говоря неформально, функция — это отношение, однозначное по вторым членам пар.

Введём обозначение[9]

$$f : X \to Y \;\leftrightharpoons\; Fnc(f) \wedge dom(f) = X \wedge rng(f) \subseteq Y.$$

Говорят, что f является *функцией из множества* X *в множество* Y (или f *отображает* X *в* Y), если $f : X \to Y$, т. е. если f — функция с областью определения X и областью значений, являющейся подмножеством множества Y.

Определим также множество $X \to Y$ всех функций из множества X в множество Y как $\{f \in 2^{X \times Y} \mid f : X \to Y\}$. Легко показать, что ограничение $f \in 2^{X \times Y}$ фиктивно.

Часто используются следующие обозначения:

$$y = f(x) \;\leftrightharpoons\; f : x \mapsto y \;\leftrightharpoons\; Fnc(f) \wedge \langle x, y \rangle \in f.$$

[9] Здесь и ниже в разделе 3.5 символы f, X, Y обозначают предметные переменные. Мы позволим себе отступить от принятого при определении языка первого порядка соглашения об обозначениях предметных переменных (см. раздел 2.1.1) с тем, чтобы обозначения функций были более привычными.

Как следует из определения функции, для любой функции f и любого $x \in dom(f)$ существует единственное y такое, что $\langle x, y\rangle \in f$. Это y называется *значением функции f на x* (*в x*, *в точке x* или *при значении аргумента x*) и обозначается $f(x)$. Явное определение значения функции f на $x \in dom(f)$ таково:

$$f(x) \leftrightharpoons \cup\{y \in rng(f) \mid \langle x, y\rangle \in f\};$$

ясно, что для любого $x \in dom(f)$ множество $\{y \in rng(f) \mid \langle x, y\rangle \in f\}$ одноэлементное.

Если область определения функции f является подмножеством декартова произведения двух множеств, то f называется *функцией двух переменных* (*аргументов*), или *двухместной* функцией. Запись $f : \langle x, y\rangle \mapsto z$ обычно сокращают до $f : x, y \mapsto z$, а запись $f(\langle x, y\rangle)$ — до $f(x, y)$. Аналогичным образом определяют функцию большего числа переменных.

Упражнение 3.5.11. Дайте формальные определения а) отношения эквивалентности, б) множества всех классов эквивалентности, определяемых отношением эквивалентности, в) инъективной, сюръективной и биективной функций, г) композиции функций. ◁

3.5.3. Числа в теории ZF

Покажем, как в теории ZF можно определить множество всех натуральных чисел. По аксиоме бесконечности существует индуктивное множество x такое, что имеет место $Ind(x)$, т. е.

$$\varnothing \in x \wedge \forall y(y \in x \supset y \cup \{y\} \in x).$$

Обозначим $Sy \leftrightharpoons S(y) \leftrightharpoons y \cup \{y\}$ и рассмотрим множества

$$\varnothing, \quad S\varnothing, \quad SS\varnothing, \quad SSS\varnothing, \quad \ldots .$$

Выпишем эти множества и введём обозначения для них:

$$0 \leftrightharpoons \varnothing, \quad 1 \leftrightharpoons \{\varnothing\}, \quad 2 \leftrightharpoons \{\varnothing, \{\varnothing\}\}, \quad 3 \leftrightharpoons \{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}\}, \quad \ldots .$$

Эти множества являются элементами множества x, но x может содержать и другие элементы. Для того, чтобы исключить «лишние» элементы, мы определим множество $\mathbb{N}$ всех натуральных чисел как индуктивное множество, которое является подмножеством любого индуктивного множества. Тогда (в интуитивном понимании) множество $\mathbb{N}$ будет содержать в точности элементы $0, 1, 2, 3, \ldots$. В силу аксиомы выделения искомое множество (т. е. индуктивное множество, которое является подмножеством любого индуктивного множества)

$$\{z \in x \mid \forall u(Ind(u) \supset z \in u)\}$$

существует. Это множество и считают множеством $\mathbb{N}$ всех натуральных чисел. Для так определённого множества $\mathbb{N}$ можно доказать (в теории ZF) аксиомы Пеано, а также доказать существование функций сложения и умножения с их обычными свойствами.

Замечание 3.5.12. В теории ZF можно вывести существование несчётного множества, например, множества $2^{\mathbb{N}}$ всех подмножеств множества $\mathbb{N}$. Однако, если ZF непротиворечива, то по теореме 2.6.28 теория ZF имеет счётную модель. Эти два факта составляют так называемый парадокс Скулема. «Парадокс Скулема» на самом деле не является парадоксом в обычном смысле, т. е. не является противоречием, но носит указанное традиционное название. ◁

Множество всех целых чисел в теории ZF можно определить как множество всех классов эквивалентности упорядоченных пар натуральных чисел по следующему отношению эквивалентности: $\langle u, v\rangle$ эквивалентна $\langle x, y\rangle$, если $u + y = x + v$. Обозначим класс эквивалентности с представителем $\langle x, y\rangle$ через $[x, y]$. В силу определения классов эквивалентности имеем: $[u, v] = [x, y]$, если и только если $u + y = x + v$. Говоря неформально, класс эквивалентности $[x, y]$ состоит в точности из пар $\langle x, y\rangle$ с одинаковой разностью $(x - y)$ и является целым числом $(x - y)$. Например, класс эквивалентности $[0, 2]$, который принято сокращённо обозначать через -2, состоит в точности из пар $\langle x, x + 2\rangle$, где x пробегает множество $\mathbb{N}$. Сложение и умножение целых чисел определяются естественным образом, если учесть указанную трактовку пары как разности:

$$[u, v] + [x, y] = [u + x,\ v + y]$$
$$\text{(неформально: } (u - v) + (x - y) = (u + x) - (v + y)\text{)},$$

$$[u, v] \cdot [x, y] = [u \cdot x + v \cdot y,\ u \cdot y + v \cdot x]$$
$$\text{(неформально: } (u - v) \cdot (x - y) = (u \cdot x + v \cdot y) - (u \cdot y + v \cdot x)\text{)},$$

причём легко доказать, что эти определения не зависят от выбора конкретных представителей классов эквивалентности. На основе этих определений можно установить свойства целых чисел.

Множество всех рациональных чисел можно определить как множество всех классов эквивалентности упорядоченных пар целых чисел (причём в каждой такой паре второй член отличен от 0) по следующему отношению эквивалентности: $\langle u, v\rangle$ эквивалентна $\langle x, y\rangle$, если $u \cdot y = x \cdot v$.

Упражнение 3.5.13. Проведите детальное построение множества всех рациональных чисел. ◁

Множество $\mathbb{R}$ всех вещественных (или действительных) чисел можно определить как множество так называемых дедекиндовых сечений в области рациональных чисел. Дедекиндово сечение в области рациональных чисел является упорядоченной парой множеств рациональных чисел, причём эти множества должны удовлетворять определённым условиям. Затем можно определить все обычные отношения (например, двухместные отношения $<$ «меньше» и $>$ «больше») и операции над вещественными числами (например, сложение, умножение вещественных чисел, взятие модуля вещественного числа). Такое, но неявно использующее аксиомы теории ZF определение множества всех вещественных чисел даётся в некоторых курсах математического анализа (см., например, [45]).

Множество x называется (*бесконечной*) *последовательностью*, если $\exists u(x : \mathbb{N} \to u)$. Если x — последовательность, то значение $x(n)$ принято обозначать через x_n и называть *n-ым членом этой последовательности*.

Множество x называется *последовательностью вещественных чисел*, если $x : \mathbb{N} \to \mathbb{R}$.

На языке теории ZF, расширенном введёнными обозначениями, можно записать практически любое математическое утверждение. Запишем, например, определение предела последовательности вещественных чисел. Пусть x — последовательность вещественных чисел и $z \in \mathbb{R}$. Неформальное определение того, что z является пределом последовательности x, таково: для любого вещественного числа $\varepsilon > 0$ существует натуральное число k такое, что для любого натурального числа $m > k$ выполняется $|x_m - z| < \varepsilon$. Формально это определение может быть записано так:

$$\forall\varepsilon(\varepsilon \in \mathbb{R} \wedge \varepsilon > 0 \supset \exists k(k \in \mathbb{N} \wedge \forall m(m \in \mathbb{N} \wedge m > k \supset |x_m - z| < \varepsilon))).$$

Введём следующие обозначения:

$$\exists x \in y\ A \ \leftrightharpoons\ \exists x(x \in y \wedge A)$$

(«существует x, принадлежащее y, такое, что имеет место A») и

$$\forall x \in y\ A \ \leftrightharpoons\ \forall x(x \in y \supset A)$$

(«для любого x, принадлежащего y, имеет место A»).

Тогда определение предела можно переписать короче:

$$\forall\varepsilon \in \mathbb{R}\ (\varepsilon > 0 \supset \exists k \in \mathbb{N}\ \forall m \in \mathbb{N}\ (m > k \supset |x_m - z| < \varepsilon)).$$

Упражнение 3.5.14. На языке теории ZF, расширенном введёнными обозначениями, запишите определения а) предела функции вещественной переменной «на языке последовательностей» и «на языке $\varepsilon - \delta$», б) непрерывной функции, в) равномерно непрерывной функции, г) производной функции. (Неформальные определения этих понятий можно найти, например, в [45].) ◁

3.5.4. Аксиома выбора и теория ZFC

Добавив к собственным аксиомам теории ZF ещё одну — *аксиому выбора*, получим теорию, которую обозначают ZFC и называют *теорией множеств Цермело-Френкеля с аксиомой выбора.*[10]

Аксиома выбора. Для любого семейства множеств X существует функция f такая, что для любого непустого множества $x \in X$ значение функции f на x является элементом множества x:

$$\forall X \exists f(Fnc(f) \wedge dom(f) = X \wedge \forall x(x \in X \wedge x \neq \varnothing \supset f(x) \in x))$$

в вышеприведённых обозначениях. Говоря менее точно, функция f «выбирает» из каждого непустого множества $x \in X$ по одному элементу в качестве своего значения на x. Функцию f, существование которой для произвольного множества X утверждает аксиома выбора, называют *функцией выбора на* X.

[10] Аббревиатура ZFC образована из уже известной нам аббревиатуры ZF и буквы C из словосочетания «axiom of Choice».

В отличие от собственных аксиом теории ZF, аксиома выбора не даёт однозначного определения множества (являющегося функцией выбора), существование которого утверждает. Отсюда происходит критическое отношение некоторых математиков к аксиоме выбора. Иногда, когда это возможно, стараются приводить доказательства, не использующие эту аксиому. Однако во многих рассуждениях классической математики аксиома выбора применяется.

Пример использования (содержательно понимаемой) аксиомы выбора можно привести даже из базового курса математического анализа. Обычно в таком курсе вводятся два определения предела функции вещественной переменной: (a) определение «на языке последовательностей» и (b) определение «на языке $\varepsilon - \delta$». Затем доказывается равносильность этих определений. В типичном доказательстве того, что (a) влечёт (b), имеется следующий шаг (см., например, [45]): для каждого натурального числа n из (непустой) проколотой δ_n-окрестности точки a выбирается вещественное число x'_n, и из всех таких чисел x'_n составляется последовательность. Как раз здесь неявно используется аксиома выбора: по каждой окрестности из семейства окрестностей функция выбора выдаёт соответствующее число x'_n.

В данной книге мы используем аксиому выбора, например, в лемме 4.1.3.

Упражнение 3.5.15. Выявите несколько других применений аксиомы выбора в доказательствах каких-либо математических теорем. ◁

Использование аксиомы выбора иногда приводит к неожиданным и находящимся далеко за пределами интуиции фактам. Например, в ZFC имеет место теорема: шар в трёхмерном евклидовом пространстве можно разбить на конечное число непересекающихся множеств, из которых с помощью движений можно получить два шара того же радиуса. Эта теорема демонстрирует, что некоторые множества, существование которых доказуемо в теории ZFC, являются абстракциями, очень далёкими от реальных физических объектов. Однако один из доводов в пользу аксиомы выбора и всей теории ZFC состоит в том, что теория ZFC является мощным инструментом для получения результатов и об абстракциях, которые близки к реальным физическим объектам.

3.5.5. Теория ZFC как основание классической математики

На языке теории ZFC, расширенном введёнными обозначениями, можно записать практически любое математическое утверждение. Эти обозначения служат лишь для удобства и сокращения записи формул. Можно описать алгоритм преобразования формул, использующих такие обозначения, в формулы языка теории ZFC, но мы не будем этим заниматься.

Что касается теории ZFC, то в ней можно формализовать все общепринятые в современной классической математике способы рассуждений. В обычных математических доказательствах аксиомы (и теоремы) теории ZFC понимаются содержательно и зачастую используются неявно, что характерно для содержательно аксиоматического метода. Вместе с тем в современной классической математике формальный вывод в ZFC рассматривается как уточнение понятия математического доказательства.

Возможность изложить какое-либо содержательное математическое доказательство в виде формального вывода в теории ZFC позволяет судить о правильности и строгости этого доказательства. (Отметим, кстати, что существование всех множеств, упоминаемых в этой книге за исключением раздела 3.4 о наивной теории множеств, можно доказать в теории ZFC.) Кроме того, как мы уже отмечали в разделе 3.3.3, для проверки и поиска доказательств с помощью компьютера, просто необходимо придерживаться формально аксиоматического метода. А в таких задачах за основу формализации может браться и нередко берётся теория ZFC.

В теории ZFC (или ZF) можно, как говорят, *относительно интерпретировать* некоторые другие формальные аксиоматичские теории и сводить вопросы об их непротиворечивости к вопросу о непротиворечивости ZFC (или ZF) без обращения к семантике. Для примера обрисуем относительную интерпретацию элементарной арифметики PA в ZF и сведение вопроса о непротиворечивости PA к вопросу о непротиворечивости ZF. Можно описать алгоритм преобразования любой формулы A языка теории PA в формулу A' языка теории ZF и конструктивно доказать, что $PA \vdash A$ влечёт $ZF \vdash A'$, а также что $(\neg A)'$ совпадает с $\neg A'$. Тем самым получена названная выше относительная интерпретация. Теперь легко установить, что непротиворечивость PA следует из непротиворечивости ZF. Действительно, если бы PA была противоречива, то нашлась бы формула A такая, что $PA \vdash A$ и $PA \vdash \neg A$; следовательно, $ZF \vdash A'$ и $ZF \vdash \neg A'$, значит, ZF оказалась бы противоречивой.

В заключение скажем несколько слов о проблеме непротиворечивости теорий ZF и ZFC, а также о программе Гильберта обоснования математики. Установлено, что ZF непротиворечива тогда и только тогда, когда ZFC непротиворечива. Из так называемой второй теоремы Гёделя о неполноте (см. теорему 5.5.38 и комментарии к ней) следует, что если теория ZFC непротиворечива, то непротиворечивость ZFC невозможно установить средствами этой теории. Однако ZFC охватывает все общепринятые в современной классической математике способы рассуждений. Поэтому затруднительно подобрать убедительные средства для доказательства непротиворечивости ZFC. В связи с этими трудностями непротиворечивость ZFC не установлена до сих пор, но и противоречия в ZFC до сих пор не обнаружены. Таким образом, непротиворечивость широко используемых теорий ZF и ZFC остаётся делом веры. Подчеркнём, что классическое направление в математике — не единственное, например, существует конструктивное направление в математике (также называемое конструктивной математикой или конструктивизмом), которое основано не на понятии множества, а на понятии алгоритма.

Наряду с упомянутыми в начале раздела 3.5 подходами к спасению математики от парадоксов, Д. Гильберт предложил программу обоснования математики, суть которой заключается в следующем: математика представляется в виде формальной аксиоматической теории, и конструктивно (или финитно, как говорил Д. Гильберт) доказывается непротиворечивость этой теории. Вторая теорема Гёделя о неполноте обнаружила невыполнимость этой программы, однако работа над программой Гильберта привела к развитию формально аксиоматического метода и метаматематики (также называемой теорией доказательств) — раздела математической логики, в котором изучаются формальные доказательства. Несмотря на проблемы

доказательства непротиворечивости, развитый Д. Гильбертом формально аксиоматический метод широко применяется. Отметим, кстати, что теория множеств Цермело-Френкеля, сформулированная нами в виде формальной аксиоматической теории, изначально была сформулирована как содержательная математическая теория. Теперь нам видна историческая причина развития формально аксиоматического метода: точное определение доказательства в какой-либо теории потребовалось для установления недоказуемости некоторых утверждений в теории, формализующей математику.

Глава 4.

Метод резолюций для логики предикатов

Эту главу мы посвятим так называемому методу резолюций, который позволяет установить общезначимость любой предикатной формулы, если эта формула действительно общезначима. Этот метод на практике оказывается эффективнее методов поиска доказательства, основанных на исчислениях предикатов гильбертовского и генценовского типа. Кроме того, метод резолюций широко используется для автоматического (компьютерного) поиска доказательств, в частности, для реализации языков логического программирования.

Условимся, что в каждом примере данной главы различны предметные переменные, предикатные символы и функциональные символы, которые имеют различные обозначения.

4.1. Скулемовская стандартная форма

Подготовительным этапом в доказательстве методом резолюций является преобразование исходной предикатной формулы к некоторой стандартной форме. Затем метод резолюций применяется к формуле в стандартной форме. Под методом резолюций мы будем понимать и описываемый ниже метод доказательства общезначимости формулы, применяемый к любой формуле (назовём этот метод методом резолюций в широком смысле), и описываемый ниже метод доказательства невыполнимости формулы, являющийся этапом метода резолюций в широком смысле и применяемый лишь к формуле в стандартной форме (назовём этот метод методом резолюций в узком смысле). Стандартная форма предикатной формулы обобщает понятие конъюнктивной нормальной формы (КНФ), введённое нами ранее для логики высказываний.

Сначала распространим понятие КНФ на логику предикатов. Для этого лишь расширим понятие литеры: *литерой* назовём любую атомарную формулу, не являющуюся истинностной константой, и отрицание такой атомарной формулы. Тогда определения дизъюнкта и КНФ, данные нами для логики высказываний (см. определение 1.1.35 на с. 27), сохраняют свой вид и для логики предикатов.

Теперь перейдём к определению упомянутой стандартной формы, которую мы назовём скулемовской.

Пусть Ω — язык первого порядка с множеством функциональных символов FS. Рассмотрим произвольную *замкнутую* формулу A этого языка. Найдём какую-нибудь её замкнутую предварённую нормальную форму (ПНФ, см. раздел 2.5) $\mathcal{Q}_1 x_1 \dots \mathcal{Q}_n x_n B$, где $n \geqslant 0$, для каждого $i = 1, \dots, n$ $\mathcal{Q}_i$ — квантор, B — матрица этой ПНФ.

Для любой бесквантворной формулы можно построить равносильную ей КНФ (считая различные атомарные формулы, не являющиеся истинностными константами, различными пропозициональными переменными и строя КНФ как для пропозициональной формулы). Поэтому будем считать, что B находится в КНФ. Также будем считать, что переменные $x_1, \dots, x_n$ попарно различны: если x_i совпадает с x_j при $i < j$, то вычеркнем из этой ПНФ кванторную приставку $\mathcal{Q}_i x_i$ (при этом, очевидно, получим формулу, равносильную исходной: вычеркнутая кванторная приставка была фиктивна с семантической точки зрения).

Пусть Sko — такое счётное множество функциональных символов, что $FS \cap Sko = \varnothing$ и для любого $m \in \mathbb{N}$ множество Sko содержит бесконечное множество m-местных функциональных символов. Построим по ПНФ формулы A некоторую формулу языка первого порядка Ω', который получается из Ω добавлением в его сигнатуру всех функциональных символов из Sko.

Если формула $\mathcal{Q}_1 x_1 \dots \mathcal{Q}_n x_n B$ имеет вид

$$\exists x_1 \mathcal{Q}_2 x_2 \dots \mathcal{Q}_n x_n B,$$

то выберем из множества Sko нульместный функциональный символ c, не входящий в B, и положим

$$A_1 \leftrightharpoons \mathcal{Q}_2 x_2 \dots \mathcal{Q}_n x_n [B]_c^{x_1}.$$

Если формула $\mathcal{Q}_1 x_1 \dots \mathcal{Q}_n x_n B$ имеет вид

$$\forall x_1 \dots \forall x_{k-1} \exists x_k \mathcal{Q}_{k+1} x_{k+1} \dots \mathcal{Q}_n x_n B,$$

то выберем из множества Sko $(k-1)$-местный функциональный символ f, не входящий в B, и положим

$$A_1 \leftrightharpoons \forall x_1 \dots \forall x_{k-1} \mathcal{Q}_{k+1} x_{k+1} \dots \mathcal{Q}_n x_n [B]_{f(x_1,\dots,x_{k-1})}^{x_k}.$$

Затем повторяем эти действия с полученной формулой A_1 и т. д. до тех пор, пока не получится формула без кванторов существования. Полученная формула без кванторов существования (она имеет вид $\forall y_1 \dots \forall y_l B'$, где $l \geqslant 0$, B' — бесквантворная формула, находящаяся в КНФ) называется *ску́лемовской*[1] *стандартной формой* (*ССФ*) *формулы* A. Элементы множества Sko называют *скулемовскими функциональными символами*.

Пример 4.1.1. Построим ССФ замкнутой формулы

$$A \leftrightharpoons \exists x(\exists y P(y) \supset (\neg \exists z Q(z) \wedge \exists y R(x, y))).$$

[1] Нередко употребляют «сколемовская» вместо «скулемовская» (это вызвано различными вариантами написания в русском языке фамилии логика Skolem).

ПНФ этой формулы мы получили в примере 2.5.6 на с. 82:

$$A \sim \exists x \forall y \forall z \exists y_1 (P(y) \supset (\neg Q(z) \wedge R(x, y_1))).$$

Выразим матрицу этой ПНФ формулой, находящейся в КНФ:

$$\begin{gathered}(P(y) \supset (\neg Q(z) \wedge R(x, y_1))) \sim \\ (\neg P(y) \vee (\neg Q(z) \wedge R(x, y_1))) \sim \\ (\neg P(y) \vee \neg Q(z)) \wedge (\neg P(y) \vee R(x, y_1)).\end{gathered}$$

Таким образом, имеем формулу

$$\exists x \forall y \forall z \exists y_1 ((\neg P(y) \vee \neg Q(z)) \wedge (\neg P(y) \vee R(x, y_1))), \qquad (*)$$

равносильную A. Теперь исключим кванторную приставку $\exists x$, введя новый скулемовский нульместный функциональный символ cx и подставив cx в матрицу формулы $(*)$ вместо x:

$$\forall y \forall z \exists y_1 ((\neg P(y) \vee \neg Q(z)) \wedge (\neg P(y) \vee R(cx, y_1))). \qquad (**)$$

Наконец, исключим кванторную приставку $\exists y_1$, введя новый скулемовский двухместный функциональный символ fy_1 и подставив $fy_1(y, z)$ в матрицу формулы $(**)$ вместо y_1:

$$\forall y \forall z ((\neg P(y) \vee \neg Q(z)) \wedge (\neg P(y) \vee R(cx,\ fy_1(y, z)))).$$

Итак, последняя формула является ССФ формулы A. ◁

Пусть язык первого порядка Ω_1 получается из языка первого порядка Ω добавлением в сигнатуру языка Ω предикатных или функциональных символов. Ясно, что любая формула A языка Ω является формулой языка Ω_1, и A выполнима как формула языка Ω, если и только если A выполнима как формула языка Ω_1. Поэтому, говоря о выполнимости каких-либо формул, можно молчаливо предполагать, что рассматривается язык, сигнатура которого содержит каждый предикатный и функциональный символ, входящий хотя бы в одну из данных формул.

Теорема 4.1.2. *Замкнутая формула A невыполнима, если и только если её ССФ невыполнима.*

Доказательство. Докажем утверждение, равносильное этой теореме: замкнутая формула A выполнима, если и только если её ССФ выполнима. Формула A равносильна своей замкнутой ПНФ $\mathcal{Q}_1 x_1 \dots \mathcal{Q}_n x_n B$ такой, что матрица B находится в КНФ, и переменные $x_1, \dots, x_n$ попарно различны. Далее применяем индукцию по числу вхождений квантора $\exists$ в эту ПНФ, индукционный переход обосновывается следующей леммой. ◁

Лемма 4.1.3. *Пусть $A \leftrightharpoons \forall x_1 \dots \forall x_{k-1} \exists x_k C$ $(k \geqslant 1)$ является замкнутой формулой, переменные $x_1, \dots, x_k$ попарно различны; f — $(k-1)$-местный функциональный символ, не входящий в формулу C; терм $f(x_1, \dots, x_{k-1})$ свободен для x_k в C; $E \leftrightharpoons \forall x_1 \dots \forall x_{k-1} [C]^{x_k}_{f(x_1, \dots, x_{k-1})}$. Тогда A выполнима, если и только если E выполнима.*

Доказательство. Пусть A выполнима. Это означает, что существует интерпретация $M \leftrightharpoons \langle \mathcal{D}, \mu \rangle$ такая, что $M \vDash A$. Тогда для любых $\alpha_1, \ldots, \alpha_{k-1} \in \mathcal{D}$ существует $\alpha_k \in \mathcal{D}$ такой, что $M, \nu \vDash C$, где ν — любая оценка такая, что $\nu(x_1) = \alpha_1, \ldots, \nu(x_{k-1}) = \alpha_{k-1}$, $\nu(x_k) = \alpha_k$. По аксиоме выбора (см. раздел 3.5.4) существует функция $\widetilde{f} : \alpha_1, \ldots, \alpha_{k-1} \mapsto \alpha_k$.

Теперь определим интерпретацию $M' \leftrightharpoons \langle \mathcal{D}, \mu' \rangle$: положим $\mu'(f) = \widetilde{f}$ и $\mu'(s) = \mu(s)$ для любого предикатного и любого функционального символа s, отличного от f. Тогда для любых $\alpha_1, \ldots, \alpha_{k-1} \in \mathcal{D}$ справедливо $M', \nu' \vDash [C]^{x_k}_{f(x_1,\ldots,x_{k-1})}$, где ν' — любая оценка такая, что $\nu'(x_1) = \alpha_1, \ldots, \nu'(x_{k-1}) = \alpha_{k-1}$. Следовательно, $M' \vDash E$, поэтому E выполнима.

Обратно, пусть выполнима E. Это означает, что существует интерпретация $M \leftrightharpoons \langle \mathcal{D}, \mu \rangle$ такая, что $M \vDash E$. Тогда для любых $\alpha_1, \ldots, \alpha_{k-1} \in \mathcal{D}$ имеет место $M, \nu \vDash [C]^{x_k}_{f(x_1,\ldots,x_{k-1})}$, где ν — любая оценка такая, что $\nu(x_1) = \alpha_1, \ldots, \nu(x_{k-1}) = \alpha_{k-1}$. Поэтому для любых $\alpha_1, \ldots, \alpha_{k-1} \in \mathcal{D}$ существует $\alpha_k = \mu(f)(\alpha_1, \ldots, \alpha_{k-1})$ такой, что $M, \nu^{x_k}_{\alpha_k} \vDash C$. Следовательно, $M \vDash A$, так что A выполнима. ◁

Упражнение 4.1.4. Где в доказательстве леммы 4.1.3 неявно использовалось то, что терм $f(x_1, \ldots, x_{k-1})$ свободен для x_k в C? *Указание.* См. также лемму 2.4.17. ◁

Замечание 4.1.5. Если формула A невыполнима, то по теореме 4.1.2 A и её ССФ равносильны. Если же A выполнима, то её ССФ, вообще говоря, не равносильна A. Действительно, формула $\exists x P(x)$ и её ССФ $P(c)$ не равносильны (подберите интерпретацию, в которой истинностные значения этих формул различны). ◁

Упражнение 4.1.6. Пусть $A \leftrightharpoons A_1 \wedge \ldots \wedge A_n$ — замкнутая формула; для каждого $i = 1, \ldots, n$ A'_i есть матрица ССФ формулы A_i, причём для любых различных i и j множества скулемовских функциональных символов, использованных для построения A'_i и A'_j соответственно, не пересекаются. Докажите, что тогда формула A невыполнима, если и только если формула $\forall\!\!\forall(A'_1 \wedge \ldots \wedge A'_n)$ невыполнима. ◁

Для замкнутой формулы вида $A_1 \wedge \ldots \wedge A_n$ предыдущее упражнение даёт зачастую более экономный способ построения формулы, которая имеет вид $\forall\!\!\forall B'$ (где B' — бесквантворная формула, находящаяся в КНФ) и невыполнимость которой равносильна невыполнимости исходной формулы, чем описанный выше способ построения ССФ. Формулу, полученную и этим, более экономным, способом, будем называть ССФ исходной формулы.

Замечание 4.1.7. Мы определили ССФ какой угодно замкнутой формулы императивно, т. е. как результат описанной последовательности действий. ССФ замкнутой формулы A можно было бы определить декларативно — как формулу, а) которая имеет вид $\forall\!\!\forall B'$, где B' — бесквантворная формула, находящаяся в КНФ, и б) которая невыполнима, если и только если A невыполнима. (Сравните, например, с декларативным определением 2.5.1 ПНФ формулы на с. 81.) Тогда вместо теоремы 4.1.2 мы бы доказали, что для любой замкнутой формулы A существует ССФ формулы A (описав способ построения такой ССФ), а упражнение 4.1.6 дало бы ещё один способ построения ССФ формулы указанного вида.

Упражнение 4.1.8. Докажите, что а) формула A общезначима тогда и только тогда, когда формула $\neg\forall A$ невыполнима; б) формула A общезначима тогда и только тогда, когда формула $\neg A$ невыполнима; в) неверно, что для любой формулы B имеет место: B невыполнима тогда и только тогда, когда формула $\forall B$ невыполнима. $\triangleleft$

Метод резолюций и представление ССФ в виде множества дизъюнктов

Для того, чтобы доказать общезначимость формулы A' методом резолюций (в широком смысле) доказывают равносильное утверждение: невыполнимость формулы $A'' \leftrightharpoons \neg\forall A'$. Для этого строят ССФ формулы A''; как мы знаем, невыполнимость этой ССФ равносильна невыполнимости A''. Наконец, применяют метод резолюций (в узком смысле) к этой ССФ для доказательства её невыполнимости.

Пусть формула A является ССФ и, значит, имеет вид $\forall(A_1 \wedge \ldots \wedge A_n)$, где A_i $(i = 1, \ldots, n)$ является дизъюнктом, т. е. дизъюнкцией литер. При этом можно считать, что в каждый дизъюнкт A_i никакая литера не входит в качестве члена дизъюнкции более одного раза, и все эти дизъюнкты попарно различны. Как и в логике высказываний (см. раздел 1.5), мы будем, когда это удобно, представлять A как множество дизъюнктов $\{A_1, \ldots, A_n\}$, а каждый дизъюнкт — как множество литер, являющихся членами этого дизъюнкта.

С другой стороны, если дано множество дизъюнктов $S \leftrightharpoons \{A_1, \ldots, A_n\}$, то соответствующая множеству S формула (являющаяся ССФ) есть $\forall(A_1 \wedge \ldots \wedge A_n)$. Отметим также, что эта формула равносильна формуле $\forall A_1 \wedge \ldots \wedge \forall A_n$, а дизъюнкту A_i $(i = 1, \ldots, n)$ из множества S соответствует формула $\forall A_i$.

Ниже в текущей главе под множеством дизъюнктов мы будем понимать непустое конечное множество дизъюнктов. Имея в виду указанное соответствие между множеством дизъюнктов и ССФ, мы будем применять к множеству дизъюнктов термины, введённые ранее для формул. Например, под невыполнимостью множества дизъюнктов будем понимать невыполнимость соответствующей этому множеству формулы. Также мы будем говорить, что слово α входит в множество дизъюнктов, если α входит в соответствующую этому множеству ССФ.

В следующих разделах мы опишем метод резолюций (в узком смысле), который будет применяться к множеству дизъюнктов для доказательства невыполнимости этого множества.

4.2. Теорема Эрбрана

Метод резолюций основан на так называемой теореме Эрбрана, доказательству которой посвящён данный раздел. Сначала мы покажем, что при установлении невыполнимости произвольного множества дизъюнктов вместо учёта интерпретаций с любыми носителями можно ограничиться учётом интерпретаций с одним носителем.

4.2.1. Эрбрановская интерпретация множества дизъюнктов

Будем рассматривать произвольный язык первого порядка Ω, считая, что в сигнатуре этого языка есть хотя бы одна предметная константа (иначе добавим предметную константу в сигнатуру этого языка).

Определение 4.2.1. Пусть S — множество дизъюнктов языка Ω. Для каждого $i \in \mathbb{N}$ определим множество $H_i(S)$ замкнутых термов.

$H_0(S)$ есть множество всех предметных констант, входящих в S, если в S входит хотя бы одна предметная константа; иначе $H_0(S)$ есть $\{a\}$, где a — какая угодно предметная константа языка Ω.

$H_{i+1}(S)$ получается объединением уже построенного множества $H_i(S)$ и множества всех термов вида $f(t_1, \ldots, t_n)$, где f — n-местный функциональный символ, входящий в S, $t_1, \ldots, t_n \in H_i(S)$, $n \in \mathbb{N}_+$.

Эрбрановским универсумом множества дизъюнктов S называется множество $\cup_{i\in\mathbb{N}} H_i(S)$, которое обозначается $H(S)$. ◁

Пример 4.2.2. Пусть $S_1 \leftrightharpoons \{P(f(x)) \vee \neg Q(y),\ Q(g(a))\}$; тогда

$$H_0(S_1) = \{a\},$$
$$H_1(S_1) = \{a,\ f(a),\ g(a)\},$$
$$H_2(S_1) = \{a,\ f(a),\ g(a),\ f(f(a)),\ f(g(a)),\ g(f(a)),\ g(g(a))\}.$$

Пусть $S_2 \leftrightharpoons \{P(x) \vee \neg Q(y),\ Q(z)\}$; тогда $H_i(S_2) = \{a\} = H(S_2)$ для любого $i \in \mathbb{N}$. ◁

Определение 4.2.3. *Эрбрановским базисом* множества дизъюнктов S называется множество $\widehat{H}(S)$ всех атомарных формул вида $P(t_1, \ldots, t_n)$, где P — n-местный предикатный символ, входящий в S, $t_1, \ldots, t_n \in H(S)$, $n \in \mathbb{N}$ (мы отождествляем $P()$ с P, что используется при $n = 0$). ◁

Определение 4.2.4. *Основным примером* дизъюнкта C, принадлежащего множеству дизъюнктов S, называется любая замкнутая формула, которая получается из C заменой переменных в C на элементы множества $H(S)$, причём все вхождения одной и той же переменной в C заменяются на один и тот же элемент множества $H(S)$. ◁

Определение 4.2.5. Пусть S — множество дизъюнктов языка первого порядка Ω, $H(S)$ — эрбрановский универсум множества S, I — интерпретация языка Ω с носителем $H(S)$. Тогда I называется *эрбрановской интерпретацией* множества дизъюнктов S, если для любого $n \in \mathbb{N}$, каждого n-местного функционального символа f, входящего в S, и любых $t_1, \ldots, t_n \in H(S)$ имеет место следующее: при $n > 0$ верно $|f(t_1, \ldots, t_n)|_I = f(t_1, \ldots, t_n)$, а при $n = 0$ верно $|f|_I = f$. ◁

Заметим, что эрбрановские интерпретации одного и того же множества дизъюнктов S могут отличаться только интерпретацией предикатных символов или интерпретацией не входящих в S функциональных символов.

Пусть $\{P_0, P_1, \ldots, P_n, \ldots\}$ — эрбрановский базис множества дизъюнктов S. Эрбрановскую интерпретацию I множества S удобно представлять в виде множества литер $\{\overline{P}_0, \overline{P}_1, \ldots, \overline{P}_n, \ldots\}$, где $\overline{P}_n$ есть P_n, если $I \vDash P_n$,

и $\neg P_n$ иначе. В это представление не включены определения предикатов и функций, сопоставленных предикатным и функциональным символам, которые не входят в S, поскольку такие предикаты и функции не влияют на истинностное значение S. В дальнейшем мы для краткости будем использовать как взаимозаменяемые понятия эрбрановской интерпретации и представления эрбрановской интерпретации в виде множества литер.

Пример 4.2.6. Пусть

$$S \leftrightharpoons \{P(f(x)) \vee \neg Q(y),\ Q(a)\}.$$

Эрбрановский универсум множества S есть

$$H(S) = \{a,\ f(a),\ f(f(a)),\ f(f(f(a))),\ \ldots\}.$$

Эрбрановский базис множества S есть

$$\widehat{H}(S) = \{P(a),\ Q(a),\ P(f(a)),\ Q(f(a)),\ P(f(f(a))),\ Q(f(f(a))),\ \ldots\}.$$

Ниже перечислены некоторые эрбрановские интерпретации множества S, представленные множествами литер:

$\{P(a),\ Q(a),\ P(f(a)),\ Q(f(a)),\ P(f(f(a))),\ Q(f(f(a))),\ \ldots\}$,
$\{\neg P(a),\ \neg Q(a),\ \neg P(f(a)),\ \neg Q(f(a)),\ \neg P(f(f(a))),\ \neg Q(f(f(a))),\ \ldots\}$,
$\{\neg P(a),\ Q(a),\ \neg P(f(a)),\ Q(f(a)),\ \neg P(f(f(a))),\ Q(f(f(a))),\ \ldots\}$.

◁

Определение 4.2.7. Пусть S — множество дизъюнктов языка первого порядка Ω, M — интерпретация языка Ω, I — эрбрановская интерпретация множества S. Тогда говорят, что эрбрановская интерпретация I *соответствует* интерпретации M, если для любого $n \in \mathbb{N}$, каждого n-местного предикатного символа P, входящего в S, и любых $t_1, \ldots, t_n \in H(S)$ имеет место следующее: при $n > 0$ верно $|P(t_1, \ldots, t_n)|_I = |P(t_1, \ldots, t_n)|_M$, а при $n = 0$ верно $|P|_I = |P|_M$. ◁

Пусть S — множество дизъюнктов языка Ω. Легко видеть, что для любой интерпретации языка Ω можно определить соответствующую ей эрбрановскую интерпретацию множества дизъюнктов S. Проиллюстрируем это следующим примером.

Пример 4.2.8. Продолжим рассмотрение множества дизъюнктов S из примера 4.2.6. Пусть дана интерпретация $M \leftrightharpoons \langle \{d, e\},\ \mu\rangle$ такая, что

$$\mu(a) = d, \quad \mu(f)(d) = d, \quad \mu(f)(e) = d,$$
$$\mu(P)(d) = 0, \quad \mu(P)(e) = 1, \quad \mu(Q)(d) = 1, \quad \mu(Q)(e) = 0.$$

Построим эрбрановскую интерпретацию I множества S, соответствующую интерпретации M. Истинностные значения элементов $\widehat{H}(S)$ определяются следующим образом:

$$|P(a)|_I = |P(a)|_M = \mu(P)(|a|_M) = \mu(P)(d) = 0,$$
$$|Q(a)|_I = |Q(a)|_M = \mu(Q)(d) = 1,$$

$$|P(f(a))|_I = |P(f(a))|_M = \mu(P)(|f(a)|_M) = \mu(P)(d) = 0,$$
$$|Q(f(a))|_I = |Q(f(a))|_M = \mu(Q)(d) = 1,$$
$$|P(f(f(a)))|_I = |P(f(f(a)))|_M = \mu(P)(d) = 0,$$
$$|Q(f(f(a)))|_I = |Q(f(f(a)))|_M = \mu(Q)(d) = 1 \text{ и т. д.}$$

Таким образом, I представляется в виде

$$\{\neg P(a),\ Q(a),\ \neg P(f(a)),\ Q(f(a)),\ \neg P(f(f(a))),\ Q(f(f(a))),\ \ldots\}.$$

◁

Лемма 4.2.9. *Пусть S — множество дизъюнктов, M — интерпретация языка Ω. Тогда если S истинно в M, то S истинно в любой эрбрановской интерпретации, соответствующей интерпретации M.*

Упражнение 4.2.10. Докажите предыдущую лемму. ◁

Теорема 4.2.11. *Множество дизъюнктов S невыполнимо тогда и только тогда, когда S ложно во всех эрбрановских интерпретациях множества S.*

Доказательство. Если S невыполнимо, то S ложно во всех интерпретациях, в том числе и во всех эрбрановских интерпретациях множества S.

Обратно, пусть S ложно во всех эрбрановских интерпретациях множества S. Если бы S было выполнимым, то нашлась бы интерпретация M, в которой S истинно. Тогда по предыдущей лемме в эрбрановской интерпретации, соответствующей интерпретации M, S было бы истинно, что противоречило бы условию. Следовательно, S невыполнимо. ◁

Итак, мы доказали, что при установлении невыполнимости множества дизъюнктов, не умаляя общности, можно рассматривать только эрбрановские интерпретации этого множества.

4.2.2. Семантические деревья

Вводимые в этом разделе семантические деревья окажутся удобными для представления доказательств методом резолюций.

Определение 4.2.12. Пусть S — множество дизъюнктов, все элементы его эрбрановского базиса $\widehat{H}(S)$ выстроены в (конечную или бесконечную) последовательность $P_0, P_1, P_2, \ldots$ с попарно различными членами. *Семантическим деревом для S* называется такое дерево с непустым конечным или счётным множеством узлов, что для всякого $i \in \mathbb{N}$ из каждого внутреннего узла, находящегося на уровне[2] i (если такой узел имеется в этом дереве), выходят 2 ребра, помеченные P_i и $\neg P_i$ соответственно. Общий вид семантического дерева для S показан на рис. 4.1. ◁

[2]Считаем, что корень дерева находится на уровне 0.

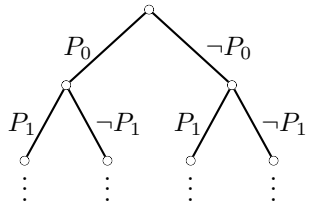


Рис. 4.1. Семантическое дерево для множества дизъюнктов S.

Определение 4.2.13. Семантическое дерево для множества дизъюнктов S называется *полным*, если для каждого $P \in \widehat{H}(S)$ на каждой ветви[3] этого дерева имеется ребро, помеченное P или $\neg P$. $\triangleleft$

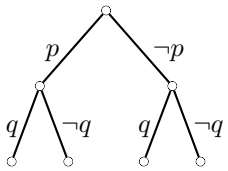


Рис. 4.2. Полное семантическое дерево для множества дизъюнктов S_1.

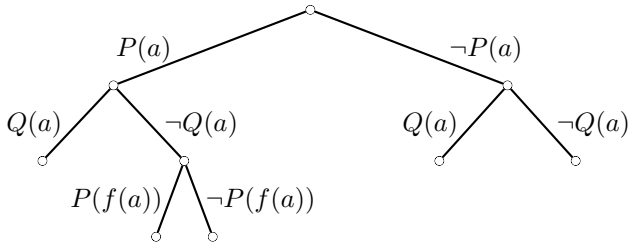


Рис. 4.3. Семантическое дерево для множества дизъюнктов S_2.

Пример 4.2.14. Пусть $S_1 \leftrightharpoons \{\neg p,\ p \vee \neg q,\ q\}$. Тогда $\widehat{H}(S_1) = \{p,\ q\}$. Полное семантическое дерево для S_1 изображено на рис. 4.2.

Пусть

$$S_2 \leftrightharpoons \{P(x),\ \neg P(f(y)) \vee Q(a),\ \neg Q(x)\}.$$

Тогда

$$\widehat{H}(S_2) = \{P(a),\ Q(a),\ P(f(a)),\ Q(f(a)),\ P(f(f(a))),\ Q(f(f(a))),\ \ldots\}.$$

Семантическое дерево для S_2 изображено на рис. 4.3. Полное семантическое дерево для S_2 бесконечно, поскольку множество $\widehat{H}(S_2)$ бесконечно; поэтому конечное семантическое дерево, изображённое на этом рисунке, не является полным. $\triangleleft$

[3]Определение ветви дерева дано в подстрочном примечании на с. 105.

Полное семантическое дерево для множества дизъюнктов S соответствует перебору всех возможных эрбрановских интерпретаций множества S в смысле, который уточнён следующей очевидной леммой.

Лемма 4.2.15. *Пусть S — множество дизъюнктов, T — полное семантическое дерево для S. Тогда*

(1) *для каждой ветви дерева T множество всех пометок рёбер этой ветви является представлением некоторой эрбрановской интерпретации множества S; и*

(2) *каждая эрбрановская интерпретация множества S может быть представлена множеством, состоящим в точности из всех пометок некоторой ветви дерева T.*

Упражнение 4.2.16. Докажите предыдущую лемму. ◁

Для каждого узла N семантического дерева для множества дизъюнктов S обозначим через $I(N)$ множество всех литер, которыми помечены рёбра, ведущие от корня этого дерева до узла N. Очевидно, $I(N)$ есть подмножество множества, представляющего некоторую эрбрановскую интерпретацию множества дизъюнктов S. По этой причине $I(N)$ называют *частичной эрбрановской интерпретацией* множества дизъюнктов S.

Определение 4.2.17. Пусть N — узел семантического дерева для множества дизъюнктов S. Будем говорить, что узел N (а также частичная эрбрановская интерпретация $I(N)$) *опровергает* дизъюнкт $C \in S$, если существует такой основной пример C' этого дизъюнкта, что $I(N) \models \neg C'$. (См. обозначение $\models$ в определении 2.4.2 логического следствия на с. 75.) ◁

Упражнение 4.2.18. Пусть даны замкнутый дизъюнкт

$$C' \leftrightharpoons P_1 \vee \ldots \vee P_l \vee \neg P_{l+1} \vee \ldots \vee \neg P_m$$

(где P_i — атомарная формула для каждого $i = 1, \ldots, m$) и множество $I(N)$ замкнутых литер. Докажите, что $I(N) \models \neg C'$ тогда и только тогда, когда

$$\{\neg P_1, \ldots, \neg P_l, P_{l+1}, \ldots, P_m\} \subseteq I(N).$$

◁

Определение 4.2.19. Пусть S — множество дизъюнктов, T — семантическое дерево для S. Узел N дерева T называется *опровергающим*, если N опровергает некоторый дизъюнкт из S, и никакой узел N', предшествующий узлу N, не опровергает никакой дизъюнкт из S. ◁

Определение 4.2.20. Семантическое дерево с конечным числом узлов называется *закрытым*, если каждый его листовой узел является опровергающим. ◁

Определение 4.2.21. Узел закрытого семантического дерева называется *выводящим*, если каждый сын этого узла является опровергающим. ◁

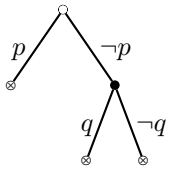


Рис. 4.4. Закрытое семантическое дерево для множества дизъюнктов S_1.

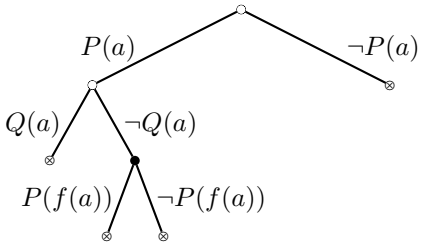


Рис. 4.5. Закрытое семантическое дерево для множества дизъюнктов S_2.

Пример 4.2.22. Как легко проверить, закрытое семантическое дерево для множества дизъюнктов S_1 (соответственно, S_2) из примера 4.2.14 изображено на рис. 4.4 (соответственно, на рис. 4.5). На этих рисунках опровергающие узлы помечены крестиками, а выводящие узлы — закрашенными кружками. ◁

Упражнение 4.2.23. Докажите, что множество дизъюнктов S ложно в эрбрановской интерпретации I этого множества, если и только если в I ложен некоторый основной пример некоторого дизъюнкта из S. ◁

Упражнение 4.2.24. Пусть S — множество дизъюнктов, дизъюнкт $C \in S$, I — эрбрановская интерпретация множества S. Докажите, что тогда некоторый основной пример дизъюнкта C ложен в I, если и только если некоторая частичная эрбрановская интерпретация $I(N) \subseteq I$ опровергает дизъюнкт C. ◁

Упражнение 4.2.25. Пусть S — множество дизъюнктов, $I(N)$ — частичная эрбрановская интерпретация множества S. Докажите, что тогда $I(N)$ опровергает некоторый дизъюнкт C из S, если и только если S ложно в любой эрбрановской интерпретации множества S, содержащей подмножество $I(N)$. ◁

4.2.3. Теорема Эрбрана

Теорема Эрбрана связывает понятие невыполнимого множества дизъюнктов с понятием закрытого (и потому конечного) семантического дерева.

Теорема 4.2.26 (теорема Эрбрана (в терминах семантических деревьев))**.** *Пусть S — множество дизъюнктов. Тогда S невыполнимо, если и только если существует закрытое семантическое дерево для S.*

Доказательство. Пусть S невыполнимо. Рассмотрим какое угодно полное семантическое дерево T для S и произвольную ветвь B этого дерева.

Обозначим через I_B эрбрановскую интерпретацию множества S, представленную множеством всех пометок рёбер ветви B. Поскольку S невыполнимо, то S ложно в интерпретации I_B. Следовательно, существует основной пример C' некоторого дизъюнкта C из S такой, что C' ложен в I_B. Поэтому для каждой литеры из C' контрарная ей литера[4] должна принадлежать I_B. Так как число литер в C' конечно, то существует частичная эрбрановская интерпретация, которая содержится в I_B и опровергает C. Поэтому на ветви B имеется опровергающий узел. Поскольку ветвь B была выбрана в дереве T произвольно, на любой ветви этого дерева имеется опровергающий узел. Искомое закрытое семантическое дерево для S получается из T объявлением каждого опровергающего узла листом.

Обратно, пусть существует закрытое семантическое дерево для S. Тогда S ложно в любой эрбрановской интерпретации множества S. Следовательно, по теореме 4.2.11 S невыполнимо. ◁

Теорема 4.2.27 (теорема Эрбрана (в терминах основных примеров дизъюнктов))**.** *Множество дизъюнктов S невыполнимо, если и только если существует конечное невыполнимое множество S' основных примеров дизъюнктов из S.*

Доказательство. Пусть S невыполнимо. По теореме Эрбрана (в терминах семантических деревьев) существует закрытое семантическое дерево T для S. Число листьев дерева T конечно, и для каждого листа L этого дерева существует основной пример C'_L некоторого дизъюнкта из S такой, что $I(L) \models \neg C'_L$. Элементами искомого множества S' объявим все такие C'_L по одному от каждого листа L дерева T. Очевидно, S' конечно и ложно в любой эрбрановской интерпретации множества S. Так как любая эрбрановская интерпретация множества S' содержится в некоторой эрбрановской интерпретации множества S, то S' ложно и в любой эрбрановской интерпретации множества S'. По теореме 4.2.11 S' невыполнимо.

Обратно, пусть существует конечное невыполнимое множество S' основных примеров дизъюнктов из S. Для доказательства невыполнимости S, в силу теоремы 4.2.11, достаточно доказать, что S ложно в произвольной эрбрановской интерпретации I множества S. Если бы S было истинно в I, то и все основные примеры дизъюнктов из S были бы истинны в I. Это противоречило бы невыполнимости множества S'. Поэтому S ложно в I. ◁

Упражнение 4.2.28. Приведите доказательство предыдущей теоремы, не использующее понятие семантического дерева. *Указание.* Воспользуйтесь теоремой о компактности. (Достаточно ли использовать теорему о компактности для логики высказываний и не обращаться к теореме о компактности для логики предикатов?) ◁

4.2.4. Алгоритм проверки невыполнимости множества дизъюнктов

Теорема Эрбрана (в терминах основных примеров дизъюнктов) служит обоснованием следующего алгоритма, устанавливающего факт невы-

[4] Определение 1.5.1 контрарных литер, данное на с. 57 для логики высказываний, сохраняет свой вид и для логики предикатов (см. также начало раздела 4.1).

полнимости произвольного заданного множества дизъюнктов S, если это множество действительно невыполнимо.

Этот алгоритм для $i = 0, 1, 2, \ldots$ порождает множество S'_i основных примеров дизъюнктов из S путём всевозможных замен переменных на элементы $H_i(S)$ (см. определение 4.2.1 на с. 145). Как только множество S'_i порождено, оно проверяется на невыполнимость; для этого применяется любой вспомогательный алгоритм проверки невыполнимости пропозициональной формулы (см. упражнение 2.4.18 на с. 80). Если невыполнимость множества S'_i зафиксирована, то алгоритм заканчивает работу, выдавая в качестве результата «невыполнимо».

По теореме Эрбрана (в терминах основных примеров дизъюнктов) этот алгоритм на входе S заканчивает свою работу с результатом «невыполнимо», если и только если S невыполнимо. Если S выполнимо, то этот алгоритм работает бесконечно.

Число элементов множества S'_i растёт очень быстро с увеличением i, если в S имеется несколько более чем нульместных функциональных символов. Очевидно, этому алгоритму зачастую потребуется породить слишком много основных примеров дизъюнктов, которые будут лишними в том смысле, что без них можно обойтись при формировании некоторого конечного невыполнимого множества основных примеров дизъюнктов. Описываемый ниже метод резолюций адресует эту проблему.

Упражнение 4.2.29. Детализируйте алгоритм, в общих чертах описанный выше. (Как порождать множество S'_i, как проверять его на невыполнимость?) ◁

4.3. Унификация

Правило образования резольвенты двух дизъюнктов было сформулировано выше (в разделе 1.5) лишь для логики высказываний. Мы бы хотели обобщить это правило для логики предикатов. Напомним, что для образования резольвенты двух пропозициональных дизъюнктов мы «отрезали» одну из контрарных литер от первого дизъюнкта, а другую — от второго, и из всех остальных литер этих дизъюнктов составляли их резольвенту.

Рассмотрим, например, дизъюнкты

$$\neg P(x) \vee Q(x) \quad \text{и} \quad P(f(y)) \vee R(y).$$

Литеры $\neg P(x)$ и $P(f(y))$ не контрарны, но если заменить y на предметную константу a и x на $f(a)$, то получится пара контрарных литер. Эта замена даёт дизъюнкты $\neg P(f(a)) \vee Q(f(a))$ и $P(f(a)) \vee R(a)$, резольвентой которых естественно считать $C_1 \leftrightharpoons Q(f(a)) \vee R(a)$.

Вместо указанных замен можно подставить $f(a)$ вместо y и $f(f(a))$ вместо x, тем самым сделав литеры $\neg P(x)$ и $P(f(y))$ контрарными. Тогда мы получим дизъюнкты $\neg P(f(f(a))) \vee Q(f(f(a)))$ и $P(f(f(a))) \vee R(f(a))$ и их резольвенту $C_2 \leftrightharpoons Q(f(f(a))) \vee R(f(a))$.

Можно действовать по-другому: для превращения литер $\neg P(x)$ и $P(f(y))$ в контрарные заменить в исходных дизъюнктах x на $f(y)$. В результате этой подстановки мы получим дизъюнкты $\neg P(f(y)) \vee Q(f(y))$ и

$P(f(y)) \vee R(y)$ и их резольвенту $C_3 \rightleftharpoons Q(f(y)) \vee R(y)$. Резольвента C_3 более общая, чем C_1 и C_2, в том смысле, что как C_1, так и C_2 получается подстановкой термов вместо предметных переменных в C_3.

Прежде чем определить понятие резольвенты предикатных дизъюнктов, мы изучим некоторые свойства подстановок термов вместо переменных и опишем алгоритм нахождения такой подстановки, которая делает две литеры контрарными.

4.3.1. Подстановки и унификаторы

Пусть зафиксирован произвольный язык первого порядка Ω. Далее в разделе 4.3 под термами понимаются термы языка Ω.

Определение 4.3.1. *Подстановкой* назовём любое такое отображение θ из множества IV всех предметных переменных в множество всех термов языка Ω, что множество $Dom(\theta) \rightleftharpoons \{x \in IV \mid \theta(x) \neq x\}$ конечно (иначе говоря, отображение θ тождественно всюду, кроме конечного множества переменных). Множество $Dom(\theta)$ назовём *носителем подстановки* θ. $\triangleleft$

Подстановку θ такую, что $Dom(\theta) \subseteq \{x_1, \ldots, x_n\}$ и $\theta(x_1) = t_1$, $\ldots$, $\theta(x_n) = t_n$, мы будем представлять в виде множества слов $\{t_1/x_1, \ldots, t_n/x_n\}$ (каждое из этих слов состоит из терма, символа-разделителя «/» и переменной). Множество, представляющее подстановку, может иметь элементы вида x/x; это соглашение будет удобно в дальнейший выкладках. Однако такие элементы можно исключить из этого множества и тем самым получить множество, представляющее ту же подстановку. Тождественная подстановка (её носитель пуст) представляется как $\{\}$. Мы условимся считать $\{\}$ иным обозначением пустого множества. (Заметим, что и $\{x/x\}$ представляет тождественную подстановку, но мы предпочтитаем $\{\}$.)

Имея в виду представление подстановки в виде множества, мы будем иногда применять к подстановкам теоретико-множественные операции (например, объединение), но только если множество-результат операции будет представлять подстановку (например, множество $\{a/x\} \cup \{b/x\}$ не представляет подстановку).

В разделе 4.3 под *выражением* мы будем понимать терм или литеру рассматриваемого языка первого порядка, а под множеством выражений — непустое конечное множество выражений.

Определение 4.3.2. Пусть $\theta \rightleftharpoons \{t_1/x_1, \ldots, t_n/x_n\}$ — подстановка и E — выражение. Через $E\theta$ обозначим результат одновременной замены всех вхождений переменных $x_1, \ldots, x_n$ в E на $t_1, \ldots, t_n$ соответственно. $E\theta$ назовём *результатом применения подстановки* θ *к* E, а также *примером* E.

Пусть также дано множество выражений $W \rightleftharpoons \{E_1, \ldots, E_n\}$. Через $W\theta$ обозначим множество $\{E_1\theta, \ldots, E_n\theta\}$, которое назовём *результатом применения подстановки* θ *к* W. $\triangleleft$

Таким образом, любое отображение θ, являющееся подстановкой, мы продолжили до отображения множества всех выражений в себя, а также до отображения множества всех (непустых конечных) множеств выражений в себя.

Пример 4.3.3. Пусть

$$\theta \leftrightharpoons \{a/u,\ f(z,w)/x,\ x/y,\ g(b,y,z)/z\},$$
$$E \leftrightharpoons P(v,\ h(x,z),\ y,\ z).$$

Тогда

$$E\theta = P(v,\ h(f(z,w),\ g(b,y,z)),\ x,\ g(b,y,z)).$$

◁

В соответствии с общеизвестным определением композиции отображений, композицией подстановок θ и η называют такую подстановку τ, что $\tau(x) = \eta(\theta(x))$ для любой переменной x; обычно композицию отображений θ и η обозначают через $\eta \circ \theta$. Но из-за того, что результат применения подстановки σ к выражению E традиционно обозначают через $E\sigma$ (как мы и обозначили выше), мы будем использовать обозначение композиции подстановок, отличающееся от обычного обозначения композиции отображений переменой мест θ и η. Итак, композицию подстановок θ и η мы будем обозначать через $\theta \circ \eta$, часто сокращая это обозначение до $\theta\eta$.

Опишем алгоритм представления композиции $\tau \leftrightharpoons \theta\eta$ подстановок $\theta \leftrightharpoons \{t_1/x_1, \ldots, t_n/x_n\}$ и $\eta \leftrightharpoons \{s_1/y_1, \ldots, s_m/y_m\}$ в виде $\{t'_1/x'_1, \ldots, t'_k/x'_k\}$:

1. Список $\theta_1 \leftrightharpoons (t_1\eta/x_1, \ldots, t_n\eta/x_n)$[5] получаем, применяя η к каждому терму t_i.

2. Из списка θ_1 исключаем все элементы вида x/x, где x — переменная, тем самым получаем список θ_2.

3. Из списка $s_1/y_1, \ldots, s_m/y_m$ исключаем все элементы вида s_j/y_j, где y_j совпадает с одной из переменных $x_1, \ldots, x_n$, тем самым получаем список η_1.

4. В искомое множество, представляющее подстановку $\theta\eta$, помещаем в точности все элементы списков θ_2 и η_1.

Пример 4.3.4. Пусть $\theta \leftrightharpoons \{f(y)/x,\ z/y\}$, $\eta \leftrightharpoons \{a/x,\ b/y,\ y/z\}$. Построим $\theta\eta$. Имеем $\theta_1 \leftrightharpoons (f(y)\eta/x,\ z\eta/y) = (f(b)/x,\ y/y)$; $\theta_2 \leftrightharpoons (f(b)/x)$; $\eta_1 \leftrightharpoons (y/z)$; и окончательно $\theta\eta = \{f(b)/x,\ y/z\}$. ◁

Теорема 4.3.5. *Данный алгоритм правильно строит композицию подстановок θ и η, т. е. полученная в результате работы этого алгоритма подстановка τ такова, что для любого выражения E верно $E\tau = (E\theta)\eta$.*

Доказательство. Достаточно доказать, что для любой переменной v имеет место $v\tau = (v\theta)\eta$. Рассмотрим 3 возможных случая.

1. Пусть $v \in Dom(\theta)$, т. е. $v = x_i$ для некоторого i. Тогда получаем $v\tau = t_i\eta = (v\theta)\eta$.

2. Пусть $v \in Dom(\eta) \setminus Dom(\theta)$. Тогда $v = y_j$ для некоторого j, и мы видим, что $v\tau = s_j = v\eta = (v\theta)\eta$.

3. Пусть, наконец, $v \notin Dom(\theta) \cup Dom(\eta)$. Тогда $v\tau = v = (v\theta)\eta$. ◁

Отметим, что имеют место тождества $\{\}\theta = \theta\{\} = \theta$, $(E\theta)\eta = E(\theta \circ \eta)$ и $(\theta\eta)\tau = \theta(\eta\tau)$ (последнее выражает ассоциативность композиции подстановок), где θ, η, τ — произвольные подстановки, E — произвольное выражение.

[5] Во избежание неоднозначности заключаем список в скобки.

Определение 4.3.6. Подстановка θ называется *унификатором* непустого множества выражений $\{E_1, \ldots, E_n\}$ (а также унификатором выражений $E_1, \ldots, E_n$), если $E_1\theta = E_2\theta = \ldots = E_n\theta$.

Говорят, что множество $\{E_1, \ldots, E_n\}$ *унифицируемо* (выражения $E_1, \ldots, E_n$ *унифицируемы*), если существует унификатор множества $\{E_1, \ldots, E_n\}$; иначе это множество называется *неунифицируемым* (и эти выражения называются *неунифицируемыми*). ◁

Пример 4.3.7. Унификатором множества $\{P(a, f(x)),\ P(x, y)\}$ является подстановка $\{a/x,\ f(a)/y\}$.

Унификаторами множества $\{P(x, y),\ P(y, x)\}$ являются подстановки $\{y/x\}$, $\{x/y\}$, $\{a/x,\ a/y\}$, $\{f(b)/x,\ f(b)/y,\ a/z\}$.

Множества $\{P(f(x), z),\ P(g(y), a)\}$ и $\{P(f(x)),\ P(x)\}$ неунифицируемы. ◁

Определение 4.3.8. Унификатор σ множества выражений W называется *наиболее общим унификатором* (*НОУ*) этого множества, если для любого унификатора θ множества W существует такая подстановка η, что $\theta = \sigma\eta$. ◁

Пример 4.3.9. Унификатор $\{y/x\}$ множества $\{P(x, y),\ P(y, x)\}$ является НОУ этого множества. Действительно, любой унификатор θ этого множества, очевидно, должен иметь вид $\{t/x,\ t/y,\ L\}$, где t — терм, L — список из элементов вида s/z, где s — терм, z — переменная, отличная от x и от y. Тогда $\theta = \{y/x\} \circ \{t/y,\ L\}$ (а также $\theta = \{y/x\} \circ \theta$), следовательно, $\{y/x\}$ является НОУ рассматриваемого множества. ◁

Упражнение 4.3.10. Являются ли подстановки $\{f(x)/y\}$, $\{f(x)/y,\ a/z\}$, $\{f(x)/y,\ u/v\}$, $\{f(z)/y,\ z/x\}$ НОУ множества $\{P(y),\ P(f(x))\}$? ◁

Упражнение 4.3.11. Найдите все НОУ одноэлементного множества выражений. ◁

Упражнение 4.3.12. Для подстановки $\theta \leftrightharpoons \{t_1/x_1, \ldots, t_n/x_n\}$ через $Rng(\theta)$ обозначим множество $\{t_1, \ldots, t_n\}$. Подстановка θ называется *перестановкой переменных*, если $Dom(\theta) = Rng(\theta)$.

Докажите следующие утверждения.

1. Пусть σ является НОУ множества выражений W. Тогда для любой перестановки переменных ρ подстановка $\sigma\rho$ является НОУ W.

2. Пусть σ_1 и σ_2 являются НОУ множества выражений W. Тогда существует такая перестановка переменных ρ, что $\sigma_1 = \sigma_2\rho$. ◁

4.3.2. Алгоритм унификации

В этом разделе мы опишем алгоритм нахождения НОУ выражений. Сначала рассмотрим пример, в котором мы ставим цель найти какой-либо унификатор выражений. Однако этот пример позволит нам подметить существенные черты алгоритма нахождения НОУ.

Пример 4.3.13. Найдём унификатор выражений

$$P(g(x), x, f(g(y)))\ \text{ и }\ P(z, a, f(z)).$$

Устанавливаем по маркеру на начало каждого из данных выражений. Затем сдвигаем маркеры одновременно на одну лексему[6] вперёд до тех пор,

[6] См. определение 2.3.6 на с. 72.

пока маркеры не будут указывать на различные лексемы. Получаем следующее положение маркеров, указанное с помощью подчёркивания лексем:

$$P(\underline{g}(x), x, f(g(y))) \quad \text{и} \quad P(\underline{z}, a, f(z)).$$

Ясно, что искомый унификатор (если он существует) переменной x должен сопоставлять некоторый терм t, а переменной z — терм $g(t)$. Однако незачем излишне конкретизировать терм, сопоставляемый переменной x, и можно положить, что переменной z сопоставляется терм $g(x)$. Заменяем z на $g(x)$ в данных выражениях и получаем выражения

$$P(g(x), x, f(g(y))) \quad \text{и} \quad P(g(x), a, f(g(x))). \qquad (*)$$

При этом имеем подстановку $\sigma_1 \leftrightharpoons \{g(x)/z\}$, являющуюся «частью» искомого унификатора.

Повторив вышеописанные действия с маркерами и выражениями $(*)$, имеем

$$P(g(x), \underline{x}, f(g(y))) \quad \text{и} \quad P(g(x), \underline{a}, f(g(x))).$$

Ясно, что искомый унификатор переменной x должен сопоставлять терм a. Заменим x на a в рассматриваемых сейчас выражениях и в уже построенной «части» искомого унификатора, а также добавим $\{a/x\}$ к уже построенной «части» искомого унификатора. В результате имеем выражения

$$P(g(a), a, f(g(y))) \quad \text{и} \quad P(g(a), a, f(g(a))) \qquad (**)$$

и «часть» искомого унификатора $\sigma_2 \leftrightharpoons \sigma_1 \circ \{a/x\} = \{g(a)/z,\ a/x\}$.

Действуя с выражениями $(**)$ так же, как и выше, остановим маркеры на следующих позициях:

$$P(g(a), a, f(g(\underline{y}))) \quad \text{и} \quad P(g(a), a, f(g(\underline{a}))).$$

Ясно, что искомый унификатор переменной y должен сопоставлять терм a. Продолжим «наращивать» искомый унификатор:

$$\sigma_3 \leftrightharpoons \sigma_2 \circ \{a/y\} = \{g(a)/z,\ a/x,\ a/y\}.$$

Применим подстановку $\{a/y\}$ к текущим выражениям и получим совпадающие выражения $P(g(a), a, f(g(a)))$ и $P(g(a), a, f(g(a)))$. Итак, σ_3 является унификатором исходных выражений. ◁

Определение 4.3.14. Пусть W — множество выражений, d — первая слева позиция, с которой не во всех выражениях-элементах множества W начинается одна и та же лексема. Тогда *множество рассогласований множества* W есть множество всех выражений, которые являются подвыражениями (т. е. подформулами или подтермами) выражений-элементов множества W и начинаются с позиции d. ◁

Пример 4.3.15. Пусть

$$W \leftrightharpoons \{P(g(x), x, f(g(y))),\ P(z, a, f(z))\}.$$

(W есть множество исходных выражений из примера 4.3.13.) Множеством рассогласований множества W является $\{g(x),\ z\}$.

Множеством рассогласований множества

$$\{P(x, f(x)),\ P(x, a),\ P(x, f(g(z, x)))\}$$

является $\{f(x),\ a,\ f(g(z, x))\}$. ◁

Опишем алгоритм, называемый *алгоритмом унификации Робинсона* (короче — *алгоритмом унификации*), который, как мы докажем, находит НОУ множества выражений. Этот алгоритм получает на вход произвольное множество выражений W (непустое и конечное, как мы условились считать) и действует следующим образом:

(1) Присвоить $k = 0$, $W_0 = W$, $\sigma_0 = \{\}$.

(2) Если множество W_k одноэлементное, то завершить работу, выдав в качестве ответа подстановку σ_k. Иначе найти D_k — множество рассогласований множества W_k.

(3) Если существуют такие элементы v_k и t_k в D_k, что v_k — переменная, не входящая в t_k, то присвоить $\sigma_{k+1} = \sigma_k \circ \{t_k/v_k\}$ и $W_{k+1} = W_k\{t_k/v_k\}$. Иначе завершить работу, выдав ответ «неунифицируемо».

(4) Присвоить $k = k + 1$ и перейти к исполнению (2).

Отметим, что в (3) мы не зафиксировали конкретный способ выбора v_k и t_k (обладающих указанными свойствами), хотя это требуется сделать для детерминированности алгоритма унификации. Такой способ выбора можно зафиксировать как угодно, что будет видно из доказательств лемм, приведённых в ниже текущем разделе 4.3.2.

Упражнение 4.3.16. Проверьте, что если алгоритм унификации построил W_k и σ_k, то верно $W_k = W\sigma_k$. ◁

Если алгоритм унификации выдал в качестве ответа подстановку σ_k, то в силу равенства $W_k = W\sigma_k$ эта подстановка является унификатором исходного множества W.

Пример 4.3.13 поиска унификатора по сути соответствует работе алгоритма унификации. Дадим ещё один пример работы этого алгоритма.

Пример 4.3.17. Пусть множество W есть $\{Q(f(x), x),\ Q(y, g(y))\}$.

(a1) Полагаем $W_0 = W$, $\sigma_0 = \{\}$.

(a2) В W_0 более 1 элемента, поэтому находим множество D_0 рассогласований множества W_0: $D_0 = \{f(x),\ y\}$.

(a3) Из D_0 выбираем терм $f(x)$ и переменную y, не входящую в этот терм. Строим подстановку $\sigma_1 = \sigma_0 \circ \{f(x)/y\} = \{f(x)/y\}$ и множество

$$W_1 = W_0\{f(x)/y\} = \{Q(f(x), x),\ Q(f(x), g(f(x)))\}.$$

(a4) Переходим к исполнению (2) при $k = 1$.

(b2) В W_1 более 1 элемента, поэтому находим множество D_1 рассогласований множества W_1: $D_1 = \{x,\ g(f(x))\}$.

(b3) В D_1 только один элемент x является переменной, но x входит во все остальные термы-элементы D_1 (x входит в $g(f(x))$), поэтому выдаём ответ, что W неунифицируемо. ◁

Лемма 4.3.18. *Пусть W — множество выражений. Тогда алгоритм унификации на входе W завершается.*

Доказательство. Предположим, что алгоритм унификации не завершается для (непустого конечного) множества выражений W. Тогда он строит бесконечную последовательность $W_0, W_1, \ldots, W_k, W_{k+1}, \ldots$ (непустых конечных) множеств выражений, причём для каждого $k \in \mathbb{N}$ в множество W_{k+1} входит[7] на одну переменную меньше, чем в множество W_k (переменная v_k входит в W_k, но не входит в $W_{k+1} = W_k\{t_k/v_k\}$). Это невозможно, поскольку в $W_0 = W$ входит лишь конечное число переменных. ◁

Лемма 4.3.19. *Если множество выражений W неунифицируемо, то алгоритм унификации на входе W выдаёт ответ «неунифицируемо».*

Доказательство. В силу леммы 4.3.18 алгоритм унификации завершается. Этот алгоритм может завершиться только при исполнении (2) или (3). Если бы алгоритм завершился при исполнении (2), то, очевидно, алгоритм выдал бы в качестве ответа унификатор множества W. Поэтому алгоритм не завершается при исполнении (2). Значит, алгоритм завершается при исполнении (3) и, следовательно, выдаёт ответ «неунифицируемо». ◁

Лемма 4.3.20. *Если множество выражений W унифицируемо, то алгоритм унификации на входе W выдаёт НОУ σ множества W, причём для любого унификатора θ множества W выполняется $\theta = \sigma\theta$.*

Доказательство. Пусть θ — какой угодно унификатор множества W. Достаточно доказать, что для любого $k \in \mathbb{N}$ верно следующее утверждение $S(k)$: *алгоритм унификации не завершается при исполнении* (3) *до построения σ_k, и если этот алгоритм построил σ_k, то $\theta = \sigma_k\theta$*. Тогда по лемме 4.3.18 алгоритм завершается для некоторого $k = m$ при исполнении (2) и выдаёт унификатор σ_m множества W, который является НОУ в силу того, что $\theta = \sigma_m\theta$.

Для доказательства того, что $S(k)$ справедливо при любом $k \in \mathbb{N}$, воспользуемся индукцией по k.

База индукции $S(0)$ верна, поскольку при исполнении (1) алгоритм построил $\sigma_0 = \{\}$.

Индукционный переход. Предположим, что верно $S(k)$. Докажем, что верно $S(k+1)$.

По индукционному предположению алгоритм не завершается при исполнении (3) до построения σ_k. Если алгоритм завершается при исполнении (2) до построения σ_k, то $S(k+1)$ верно. Иначе алгоритм не завершается до построения σ_k и строит множество W_k. Если W_k одноэлементное, то алгоритм завершается при исполнении (2) и не строит σ_{k+1}; таким образом, в этом случае $S(k+1)$ верно.

Далее разбирается случай, когда в множестве W_k более одного элемента. Тогда алгоритм строит множество рассогласований D_k множества W_k.

[7] Для краткости говоря в текущем доказательстве о том, что переменная входит в множество выражений, мы понимает под этим, что переменная входит хотя бы в одно из выражений, являющихся элементами этого множества.

Так как $\theta = \sigma_k\theta$ (по индукционному предположению) и множество $W\theta$ одноэлементное (вспомним, что θ — унификатор W), то и множество $W_k\theta = W\sigma_k\theta = W\theta$ одноэлементное. Значит, θ — унификатор W_k, и, следовательно, θ — унификатор D_k.

Множество рассогласований D_k унифицируемо, и потому среди элементов D_k имеется переменная (иначе в этом множестве рассогласований все элементы начинались бы с одной и той же лексемы, что невозможно). Пусть v_k — произвольная переменная, являющаяся элементом множества D_k.

Пусть t_k — любой элемент множества рассогласований D_k, отличный от v_k. Так как θ — унификатор D_k, то $v_k\theta = t_k\theta$. Отсюда, поскольку v_k и t_k не совпадают, следует, что v_k не входит в t_k. Поэтому среди элементов D_k алгоритм может как угодно выбрать t_k и переменную v_k, не входящую в t_k. Таким образом, при исполнении (3) алгоритм не завершается и строит $\sigma_{k+1} = \sigma_k\{t_k/v_k\}$.

Для установления $S(k+1)$ осталось доказать, что $\sigma_{k+1}\theta = \theta$. Имеем

$$\sigma_{k+1}\theta = (\sigma_k\{t_k/v_k\})\theta = \sigma_k(\{t_k/v_k\}\theta).$$

Найдём композицию $\{t_k/v_k\}\theta$. Подстановка θ имеет вид $\{t/v_k,\ L\}$, где L — список из элементов вида s/z, причём переменная z отлична от v_k. Как указано выше, выполняется $v_k\theta = t_k\theta$, поэтому

$$\{t_k/v_k\}\theta = \{t_k\theta/v_k,\ L\} = \{v_k\theta/v_k,\ L\} = \theta.$$

Теперь имеем

$$\sigma_{k+1}\theta = \sigma_k(\{t_k/v_k\}\theta) = \sigma_k\theta = \theta,$$

причём последнее равенство в этой цепочке равенств верно по индукционному предположению. Итак, $S(k+1)$ установлено. ◁

Определение 4.3.21. Подстановка η называется *идемпотентной*, если $\eta = \eta\eta$. ◁

Упражнение 4.3.22. Для подстановки θ, представленной в виде множества $\{t_1/x_1, \ldots, t_n/x_n\}$, где для каждого $i = 1, \ldots, n$ t_i не совпадает с x_i, через $Range(\theta)$ обозначим множество всех переменных, входящих в $t_1, \ldots, t_n$.

Докажите, что подстановка θ идемпотентна тогда и только тогда, когда $Dom(\theta) \cap Range(\theta) = \varnothing$. ◁

Упражнение 4.3.23. Пусть σ — НОУ множества выражений W. Докажите, что σ идемпотентен, если и только если для любого унификатора θ множества W выполняется $\theta = \sigma\theta$. ◁

Следующая теорема подытоживает свойства алгоритма унификации, установленные в предыдущих леммах и упражнении.

Теорема 4.3.24. *Если множество выражений W унифицируемо, то алгоритм унификации на входе W выдаёт идемпотентный НОУ множества W, иначе выдаёт ответ «неунифицируемо».*

Упражнение 4.3.25. Укажите множество выражений и его НОУ, который не является идемпотентным. ◁

4.4. Метод резолюций

В этом разделе мы опишем метод резолюций (иначе говоря, определим резолютивное исчисление) для логики предикатов, установим его корректность и полноту и приведём алгоритм доказательства этим методом.

4.4.1. Определение резолютивного вывода и корректность метода резолюций

Мы посвятили раздел 4.3 изучению подстановок и унификаторов, чтобы определить понятие резольвенты дизъюнктов, являющихся предикатными формулами. Теперь мы сможем выполнить намеченное.

Напомним, что мы нередко рассматриваем дизъюнкт как множество литер, причём пустое множество литер считаем пустым дизъюнктом и обозначаем через $\square$.

Определение 4.4.1. Если два или более элемента дизъюнкта C (рассматриваемого как множество литер) имеют НОУ σ, то множество литер $C\sigma$ (рассматриваемое и как дизъюнкт) называется *склейкой дизъюнкта* C. $\triangleleft$

Пример 4.4.2. Пусть дан дизъюнкт

$$C \leftrightharpoons \neg P(x) \vee \neg P(f(y)) \vee Q(x).$$

Литеры $\neg P(x)$ и $\neg P(f(y))$ имеют НОУ $\sigma \leftrightharpoons \{f(y)/x\}$. Дизъюнкт

$$C\sigma = \neg P(f(y)) \vee Q(f(y))$$

является склейкой дизъюнкта C. $\triangleleft$

В дальнейшем нам потребуется переименовывать (свободные) переменные в бескванторных формулах (сравните с понятием переименования связанных переменных в разделе 2.3.2).

Пусть A, B — бескванторные формулы, $x_1, \ldots, x_n$ — попарно различные переменные, $y_1, \ldots, y_n$ — попарно различные переменные. Если B является результатом замены всех свободных вхождений переменных $x_1, \ldots, x_n$ в A на $y_1, \ldots, y_n$ соответственно, то будем говорить, что для получения B мы *переименовали свободные переменные* в A.

Ниже в главе 4 нам потребуется переименовывать связанные переменные только в замкнутых формулах, а свободные — только в бескванторных; поэтому мы будем говорить просто о *переименовании переменных*.

Определение 4.4.3. Пусть C_1 и C_2 — дизъюнкты. Будем считать, что никакая предметная переменная не входит в C_1 и C_2 одновременно (иначе переименуем переменные в C_1 или в C_2 так, чтобы данное условие выполнялось). Пусть элементом дизъюнкта C_1 (рассматриваемого как множество литер) является литера L_1, а элементом дизъюнкта C_2 — литера L_2. Если L_1 и $\neg L_2$ (или $\neg L_1$ и L_2) имеют НОУ σ, то дизъюнкт $(C_1\sigma \backslash \{L_1\sigma\}) \cup (C_2\sigma \backslash \{L_2\sigma\})$ называется *бинарной резольвентой дизъюнктов* C_1 и C_2, а L_1 и L_2 называются *отрезаемыми* литерами. $\triangleleft$

Замечание 4.4.4. Вспомним, что каждому дизъюнкту C из множества дизъюнктов, представляющего ССФ, соответствует формула $\forall C$ (см. конец раздела 4.1). Поэтому переименование переменных в дизъюнкте из множества

дизъюнктов не изменяет семантику этого множества. С другой стороны, в силу этого же соответствия нет никаких оснований полагаться на то, что два дизъюнкта из одного множества дизъюнктов имеют общие переменные. Упражнение 4.4.13 в конце текущего раздела покажет необходимость такого переименования переменных. ◁

Пример 4.4.5. Пусть даны дизъюнкты

$$C_1 \leftrightharpoons \neg P(f(y)) \vee Q(f(y)) \quad \text{и} \quad C_2 \leftrightharpoons P(y) \vee R(y).$$

Переименуем переменную y в дизъюнкте C_2 на новую переменную z и получим дизъюнкт

$$C_2' \leftrightharpoons P(z) \vee R(z).$$

Литера $L_1 \leftrightharpoons \neg P(f(y))$ дизъюнкта C_1 и отрицание литеры $L_2 \leftrightharpoons P(z)$ дизъюнкта C_2' имеют НОУ $\sigma \leftrightharpoons \{f(y)/z\}$. Поэтому

$$\begin{gathered}(C_1\sigma \setminus \{L_1\sigma\}) \cup (C_2'\sigma \setminus \{L_2\sigma\}) = \\ (\{\neg P(f(y)),\ Q(f(y))\} \setminus \{\neg P(f(y))\}) \cup (\{P(f(y)),\ R(f(y))\} \setminus \{P(f(y))\}) = \\ \{Q(f(y))\} \cup \{R(f(y))\} = \{Q(f(y)),\ R(f(y))\} = Q(f(y)) \vee R(f(y))\end{gathered}$$

является бинарной резольвентой дизъюнктов C_1 и C_2. ◁

Определение 4.4.6. *Резольвентой дизъюнктов* C_1 и C_2 называется любая из следующих бинарных резольвент: бинарная резольвента C_1 и C_2, бинарная резольвента C_1 и склейки C_2, бинарная резольвента C_2 и склейки C_1, бинарная резольвента склейки C_1 и склейки C_2. ◁

Пример 4.4.7. Пусть даны дизъюнкты

$$C_1 \leftrightharpoons \neg P(x) \vee \neg P(f(y)) \vee Q(x) \quad \text{и} \quad C_2 \leftrightharpoons P(y) \vee R(y).$$

Из двух предыдущих примеров мы знаем, что дизъюнкт

$$C_1' \leftrightharpoons \neg P(f(y)) \vee Q(f(y))$$

является склейкой дизъюнкта C_1, а дизъюнкт

$$C \leftrightharpoons Q(f(y)) \vee R(f(y))$$

является бинарной резольвентой дизъюнктов C_1' и C_2. Поэтому C является резольвентой дизъюнктов C_1 и C_2.

Если не рассматривать склейку дизъюнкта C_1, то можно получить другую резольвенту дизъюнктов C_1 и C_2. Переименуем переменную y в дизъюнкте C_2 на новую переменную z и получим дизъюнкт

$$C_2' \leftrightharpoons P(z) \vee R(z).$$

Литера $\neg P(x)$ из C_1 и отрицание литеры $P(z)$ из C_2' имеют НОУ $\{z/x\}$, тогда

$$\neg P(f(y)) \vee Q(z) \vee R(z)$$

является бинарной резольвентой (и, следовательно, просто резольвентой) дизъюнктов C_1 и C_2. ◁

Теорема 4.4.8. *Пусть C_1 и C_2 — дизъюнкты, C — их резольвента. Тогда* $\forall C_1, \forall C_2 \models \forall C$.

Упражнение 4.4.9. Докажите теорему 4.4.8. *Указание.* Достаточно доказать, что 1) если C' — склейка дизъюнкта C, то $\forall C \models \forall C'$, и 2) если C — бинарная резольвента двух дизъюнктов C_1 и C_2, то $\forall C_1, \forall C_2 \models \forall C$ (см. также теорему 1.5.4). ◁

Определения *резолютивного вывода* и *резолютивного исчисления* аналогичны определениям, данным в разделе 1.5.

Следующая теорема доказывается аналогично теореме 1.5.6 о корректности метода резолюций для логики высказываний, но с обращением к теореме 4.4.8.

Теорема 4.4.10 (корректность метода резолюций)**.** *Пусть S — множество дизъюнктов. Если существует резолютивный вывод пустого дизъюнкта из S, то S невыполнимо.*

Определение 4.4.11. Пусть C — дизъюнкт, S — множество дизъюнктов. Дерево с конечным числом узлов, каждый из которых является (или помечен) дизъюнктом, называется *деревом* (*резолютивного*) *вывода* дизъюнкта C из множества S, если корнем этого дерева является C, каждый лист этого дерева является некоторым дизъюнктом из S, а каждый внутренний узел $\widetilde{C}$ есть резольвента дизъюнктов, являющихся сыновними узлами узла $\widetilde{C}$. Дерево вывода традиционно изображается так, что его корень располагается ниже всех других узлов этого дерева. ◁

Пример 4.4.12. Пусть дано множество S, состоящее из следующих дизъюнктов:

1) $P(x)$,

2) $\neg P(f(y)) \vee Q(a)$,

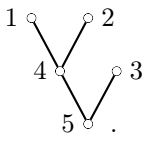
3) $\neg Q(x)$.

Попробуем вывести пустой дизъюнкт из S:

4) $Q(a)$ — резольвента дизъюнктов 1 и 2 (отрицание литеры $P(x)$ дизъюнкта 1 и литера $\neg P(f(y))$ дизъюнкта 2 имеют НОУ $\{f(y)/x\}$),

5) $\square$ — резольвента дизъюнктов 3 и 4 (литера $\neg Q(x)$ дизъюнкта 3 и отрицание литеры $Q(a)$ дизъюнкта 4 имеют НОУ $\{a/x\}$).

Таким образом, S невыполнимо. Дерево вывода дизъюнкта $\square$ из S имеет следующий вид:



В этом дереве вместо дизъюнктов указаны их номера. ◁

Упражнение 4.4.13. Выполнимо ли множество дизъюнктов

$$\{P(a,x),\ \neg P(x,b)\}?$$

Можно ли вывести пустой дизъюнкт из этого множества с помощью правила образования резольвенты, если изменить определение бинарной резольвенты так, чтобы переименование переменных не допускалось? ◁

Упражнение 4.4.14. Выполнимо ли множество дизъюнктов

$$\{P(u) \vee P(v),\ \neg P(x) \vee \neg P(y)\}?$$

Можно ли вывести пустой дизъюнкт из этого множества только с помощью правила образования бинарной резольвенты (без использования склеек)? ◁

Упражнение 4.4.15. Методом резолюций докажите общезначимость всех формул, указанных в упражнении 2.6.14 на с. 89, взяв в качестве A и B формулы $\exists y P(x, f(y))$ и $\forall x Q(x, z)$ соответственно. ◁

4.4.2. Полнота метода резолюций

В этом разделе мы докажем теорему о полноте метода резолюций: из любого невыполнимого множества дизъюнктов с помощью некоторого резолютивного вывода можно получить пустой дизъюнкт. Следующая лемма будет использоваться в доказательстве теоремы о полноте метода резолюций.

Лемма 4.4.16 (лемма подъёма)**.** *Пусть C_1 и C_2 — дизъюнкты, C_1' и C_2' — основные примеры дизъюнктов C_1 и C_2 соответственно, C' — резольвента C_1' и C_2'. Тогда найдётся такая резольвента C дизъюнктов C_1 и C_2, что C' есть пример дизъюнкта C.*

Доказательство. Можно считать (см. определения резольвенты и бинарной резольвенты), что никакая предметная переменная не входит в C_1 и C_2 одновременно.

Поскольку C' — резольвента C_1' и C_2', имеем

$$C' = (C_1' \setminus \{L_1'\}) \cup (C_2' \setminus \{L_2'\}),$$

где L_1' и L_2' — отрезаемые контрарные литеры, причём для определённости будем считать, что $L_1' = \neg L_2'$. Так как для каждого $k = 1, 2$ дизъюнкт C_k' является примером C_k, то $C_k' = C_k\theta_k$, где θ_k — некоторая подстановка. C_1 и C_2 не имеют общих переменных, поэтому можно считать, что $Dom(\theta_1) \cap Dom(\theta_2) = \varnothing$. Положим $\theta \leftrightharpoons \theta_1 \cup \theta_2$. Тогда $C_k' = C_k\theta$.

Для каждого $k = 1, 2$ обозначим через $L_k^1, \ldots, L_k^{n_k}$ $(n_k \in \mathbb{N}_+)$ все литеры-элементы C_k такие, что $L_k' = L_k^1\theta = \ldots = L_k^{n_k}\theta$. (Иначе говоря, литеры $L_k^1, \ldots, L_k^{n_k}$ из C_k склеиваются в литеру L_k' из C_k' в результате применения подстановки θ к C_k.) Таким образом, для каждого $k = 1, 2$ θ — унификатор множества $W_k \leftrightharpoons \{L_k^1, \ldots, L_k^{n_k}\}$. Поэтому в силу леммы 4.3.20 существует НОУ τ_k множества W_k такой, что $\theta = \tau_k\theta$; причём если $n_k = 1$, то в качестве τ_k берём тождественную подстановку. Тогда $C_k\tau_k$ есть склейка C_k при $n_k > 1$, и $C_k\tau_k = C_k$ при $n_k = 1$.

Для каждого $k = 1, 2$ имеем $L'_k = L^1_k\theta = L^1_k\tau_k\theta$. Поскольку $L'_1 = \neg L'_2$, то θ — унификатор литер $L^1_1\tau_1$ и $\neg L^1_2\tau_2$. Тогда по лемме 4.3.20 существует НОУ σ литер $L^1_1\tau_1$ и $\neg L^1_2\tau_2$ такой, что $\theta = \sigma\theta$.

В качестве искомой резольвенты C дизъюнктов C_1 и C_2 возьмём

$$C \leftrightharpoons (C_1\tau_1\sigma \setminus \{L^1_1\tau_1\sigma\}) \cup (C_2\tau_2\sigma \setminus \{L^1_2\tau_2\sigma\}).$$

Осталось показать, что C' — пример C. Действительно,

$$\begin{aligned} C' = (C'_1 \setminus \{L'_1\}) \cup (C'_2 \setminus \{L'_2\}) = (C_1\theta \setminus \{L^1_1\theta\}) \cup (C_2\theta \setminus \{L^1_2\theta\}) = \\ (C_1\tau_1\theta \setminus \{L^1_1\tau_1\theta\}) \cup (C_2\tau_2\theta \setminus \{L^1_2\tau_2\theta\}) = \\ (C_1\tau_1\sigma\theta \setminus \{L^1_1\tau_1\sigma\theta\}) \cup (C_2\tau_2\sigma\theta \setminus \{L^1_2\tau_2\sigma\theta\}) = C\theta. \end{aligned}$$

◁

Теорема 4.4.17 (полнота метода резолюций)**.** *Если множество дизъюнктов S невыполнимо, то существует резолютивный вывод пустого дизъюнкта из S.*

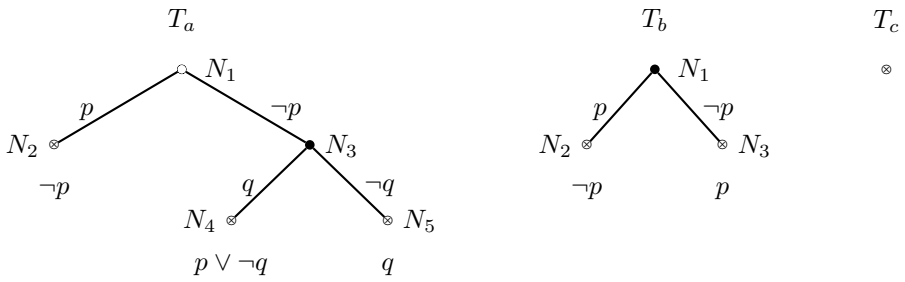


Рис. 4.6. Семантические деревья T_a, T_b, T_c.

Сначала продемонстрируем идею доказательства этой теоремы. Пусть дано невыполнимое множество S, состоящее из следующих дизъюнктов:

1) $\neg p$,

2) $p \vee \neg q$,

3) q.

Рассмотрим закрытое семантическое дерево T_a для S, изображённое на рис. 4.6. Опровергающие узлы помечены крестиками, а выводящие узлы — закрашенными кружками. Каждый лист L дерева T_a помечен дизъюнктом (совпадающим с любым своим основным примером) из S, который опровергается частичной эрбрановской интерпретацией $I(L)$. Сыновья выводящего узла N_3 — узлы N_4 и N_5 — помечены дизъюнктами, которые имеют резольвенту $Res_a \leftrightharpoons p$. Заметим, что Res_a опровергается частичной эрбрановской интерпретацией $I(N_3)$. Тогда дерево T_b, получающееся путём обрезания ветвей дерева T_a непосредственно ниже узла N_3, является закрытым семантическим деревом для множества $S \cup \{Res_a\}$.

Аналогично рассмотрим выводящий узел N_1 дерева T_b и его сыновей — узлы N_2 и N_3. Узлы N_2 и N_3 помечены опровергаемыми ими дизъюнктами, которые имеют резольвенту $Res_b \leftrightharpoons \square$, причём Res_b опровергается пустой частичной эрбрановской интерпретацией $I(N_1)$. Тогда мы получаем закрытое семантическое дерево T_c для множества $S \cup \{Res_a\} \cup \{Res_b\}$, элементом которого является пустой дизъюнкт Res_b.

Таким образом, «стягивая» закрытое семантическое дерево для исходного невыполнимого множества дизъюнктов S, мы получили дерево, состоящее лишь из одного узла. Процесс «стягивания» осуществлялся с помощью порождения резольвент, и этому процессу соответствует резолютивный вывод $\square$ из S (мы продолжаем нумерацию дизъюнктов, начатую при перечислении элементов множества S):

4) p — резольвента дизъюнктов 2 и 3,

5) $\square$ — резольвента дизъюнктов 1 и 4.

Такое «стягивание» закрытого семантического дерева с одновременным построением резолютивного вывода $\square$ мы осуществим и в общем случае.
Доказательство теоремы 4.4.17. По теореме Эрбрана (в терминах семантических деревьев) существует закрытое семантическое дерево T для S.

Для завершения доказательства этой теоремы достаточно доказать, что для любого $k \in \mathbb{N}_+$ верно утверждение: *для любого невыполнимого множества дизъюнктов S и любого закрытого семантического дерева T для S с числом узлов k существует резолютивный вывод пустого дизъюнкта из S*. Воспользуемся индукцией по k.

База индукции. В дереве T имеется лишь один узел N_0. Тогда пустой дизъюнкт принадлежит S, поскольку никакой другой дизъюнкт не опровергается частичной эрбрановской интерпретацией $I(N_0)$, являющейся пустым множеством. Искомый резолютивный вывод состоит из одного пустого дизъюнкта.

Индукционный переход. Рассмотрим дерево T с $m > 1$ узлами, предположив, что доказываемое утверждение верно для любых $k < m$.

Если бы в дереве T не существовало выводящего узла, то каждый узел имел бы сына, который бы не являлся опровергающим. Тогда можно было бы найти бесконечную ветвь, на которой бы не было опровергающих узлов, что противоречило бы закрытости дерева T. Значит, в T существует выводящий узел N. Обозначим сыновей узла N через N_1 и N_2 соответственно, а пометки рёбер, соединяющих N с N_1 и N_2, — через P_n и $\neg P_n$ соответственно (см. также рис. 4.7). Очевидно, имеют место следующие соотношения между частичными эрбрановскими интерпретациями: $I(N_1) = I(N) \cup \{P_n\}$ и $I(N_2) = I(N) \cup \{\neg P_n\}$.

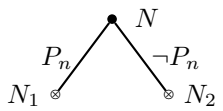


Рис. 4.7. Узлы N, N_1 и N_2 семантического дерева T.

Так как узел N_1 является опровергающим, то для некоторого основного примера C_1' некоторого дизъюнкта C_1 из S верно $I(N) \cup \{P_n\} \models \neg C_1'$ и $I(N) \not\models \neg C_1'$. Следовательно, $\neg P_n$ является элементом множества литер C_1'.

Так как узел N_2 является опровергающим, то для некоторого основного примера C_2' некоторого дизъюнкта C_2 из S верно $I(N) \cup \{\neg P_n\} \models \neg C_2'$ и $I(N) \not\models \neg C_2'$. Значит, P_n является элементом множества литер C_2'.

Тогда существует резольвента $C' \leftrightharpoons (C_1' \setminus \{\neg P_n\}) \cup (C_2' \setminus \{P_n\})$ дизъюнктов C_1' и C_2', причём нетрудно видеть, что $I(N) \models \neg C'$. По лемме подъёма существует такая резольвента C дизъюнктов C_1 и C_2, что C' является примером C. Множество $S \cup \{C\}$ невыполнимо, поскольку S невыполнимо. Так как $I(N) \models \neg C'$, то $I(N)$ опровергает C, следовательно, закрытое семантическое дерево T' для $S \cup \{C\}$ получается из дерева T путём удаления по крайней мере узлов N_1 и N_2.

Применив индукционное предположение к закрытому семантическому дереву T' для невыполнимого множества $S \cup \{C\}$, заключаем, что существует резолютивный вывод пустого дизъюнкта из $S \cup \{C\}$. Дописав в начало этого вывода C_1, C_2 и C, получим резолютивный вывод пустого дизъюнкта из S. ◁

Упражнение 4.4.18. Вместо приведённого на с. 161 определения 4.4.6 резольвенты двух дизъюнктов можно дать следующее определение.

Пусть C_1 и C_2 — дизъюнкты. Будем считать, что никакая предметная переменная не входит в C_1 и C_2 одновременно (иначе переименуем переменные в C_1 или в C_2 так, чтобы данное условие выполнялось). Пусть для каждого $k = 1, 2$ элементами дизъюнкта C_k (рассматриваемого как множество литер) являются литеры $L_k^1, \ldots, L_k^{n_k}$ при некотором $n_k \in \mathbb{N}_+$. Если множество литер $\{L_1^1, \ldots, L_1^{n_1}, \neg L_2^1, \ldots, \neg L_2^{n_2}\}$ (или $\{\neg L_1^1, \ldots, \neg L_1^{n_1}, L_2^1, \ldots, L_2^{n_2}\}$) имеет НОУ σ, то дизъюнкт $(C_1\sigma \setminus \{L_1^1\sigma\}) \cup (C_2\sigma \setminus \{L_2^1\sigma\})$ называется *резольвентой дизъюнктов* C_1 и C_2.

Докажите корректность и полноту метода резолюций при таком определении резольвенты. ◁

4.4.3. Алгоритм доказательства методом резолюций

Для доказательства общезначимости заданной предикатной формулы A методом резолюций строят ССФ формулы $\neg\forall A$ и проводят поиск резолютивного вывода пустого дизъюнкта из множества дизъюнктов S, представляющего эту ССФ. Общезначимость формулы A равносильна невыполнимости множества S. Теоремы о корректности и полноте метода резолюций гарантируют: S невыполнимо, если и только если существует резолютивный вывод пустого дизъюнкта из S. Как найти такой вывод? Можно предложить следующий алгоритм, который при данном множестве дизъюнктов S действует как алгоритм Британского музея для резолютивного исчисления.

Этот алгоритм последовательно порождает множества $S_0, S_1, S_2, \ldots$, где $S_0 = S$, а S_{i+1} есть множество всевозможных (см. следующее замечание) резольвент дизъюнктов C и $\widetilde{C}$, где $C \in S_0 \cup \ldots \cup S_i$, $\widetilde{C} \in S_i$. Как только порождён пустой дизъюнкт, алгоритм заканчивает работу, сообщая, что S невыполнимо.

Замечание 4.4.19. Число всевозможных резольвент двух дизъюнктов не обязательно конечно, поэтому приведённое описание алгоритма требует

уточнения. Из доказательств леммы подъёма и теоремы о полноте метода резолюций следует, что достаточно строить лишь конечное число резольвент двух дизъюнктов: при фиксированном выборе как склеиваемых, так и отрезаемых литер НОУ выбранного множества литер достаточно находить лишь один раз с помощью алгоритма унификации из раздела 4.3.2. Более того, в силу упражнения 4.3.12 подойдёт любой НОУ выбранного множества литер. ◁

В силу теорем о корректности и полноте метода резолюций этот алгоритм заканчивает свою работу с сообщением о невыполнимости S, если и только если S невыполнимо. Если S выполнимо, то этот алгоритм работает бесконечно.

Описанный алгоритм зачастую порождает много лишних дизъюнктов, без которых можно обойтись в дальнейшем поиске вывода пустого дизъюнкта. Для того, чтобы сократить число порождаемых дизъюнктов, на резолютивный вывод накладывают то или иное ограничение, называемое стратегией резолюции. При этом полноту метода резолюций в сочетании с какой-либо стратегией необходимо исследовать дополнительно.

Например, одной из стратегий резолюции является так называемая входная резолюция. При следовании этой стратегии порождаются резольвенты только таких пар дизъюнктов, что хотя бы один дизъюнкт из пары принадлежит исходному (входному) множеству дизъюнктов. С ещё одной стратегией, называемой SLD-резолюцией, мы познакомимся в разделе 4.5.

Упражнение 4.4.20. Опишите алгоритм, который основан на методе резолюций и который по любому (непустому конечному) множеству пропозициональных дизъюнктов определяет, является ли это множество выполнимым или нет. *Указание.* Подчеркнём, что этот алгоритм должен всегда завершаться. При каком условии он должен завершаться? Может ли в резолютивном выводе (из конечного множества пропозициональных дизъюнктов) получиться бесконечное число различных дизъюнктов? ◁

4.5. Основы логического программирования

Языки программирования принято подразделять на императивные и декларативные. В программе на императивном языке (например, Pascal, C++, Java, C#) программист описывает, *как* должно выполняться вычисление. А в программе на декларативном языке программист скорее описывает математическое определение того, *что* должно быть вычислено. К декларативным языкам программирования относятся, по крайней мере, функциональные (например, Lisp, Haskell) и логические (например, Prolog, Datalog).

Программа на языке логического программирования[8] (или логическая программа) представляет собой множество предикатных формул, описывающих предметную область решаемой задачи. Когда к такой программе ставится запрос, представляющий собой формулу, то с помощью специального механизма логического вывода проверяется, является ли этот запрос логическим следствием множества формул, представляющих программу.

[8] Термины «логический язык программирования» и «язык логического программирования» используются как синонимы.

При этом программист освобождается от управления механизмом логического вывода, т. е. освобождается от программирования того, как производится вычисление. (Конечно, особенности реализации языка логического программирования могут потребовать от программиста некоторого учёта механизма вывода, но принципиально ситуация соответствует описанной.) В данном разделе мы определим понятие логической программы, близкой к программе на чистом языке Prolog, и покажем, как может производиться логический вывод при ответе на запрос к логической программе.

4.5.1. Логическая программа и её декларативная семантика

Клаузой называется слово вида

$$B_1, \ldots, B_n \leftarrow A_1, \ldots, A_m,$$

где $B_1, \ldots, B_n, A_1, \ldots, A_m$ — атомарные формулы, ни одна из которых не является истинностной константой, $m, n \in \mathbb{N}$. Каждая из формул A_i $(i = 1, \ldots, m)$ называется *посылкой* этой клаузы, а каждая из формул B_j $(j = 1, \ldots, n)$ — *заключением*.

Клауза $B_1, \ldots, B_n \leftarrow A_1, \ldots, A_m$ при $m > 0$ или $n > 0$ есть иная запись формулы (являющейся замыканием дизъюнкта)

$$\mathbb{V}(B_1 \vee \ldots \vee B_n \vee \neg A_1 \vee \ldots \vee \neg A_m),$$

а при $m = n = 0$ — пустого дизъюнкта $\square$. Для большей наглядности заметим, что данная клауза равносильна формуле

$$\mathbb{V}(A_1 \wedge \ldots \wedge A_m \supset B_1 \vee \ldots \vee B_n),$$

причём мы считаем, что конъюнкция (соответственно, дизъюнкция) нуля формул есть $\mathbf{T}$ (соответственно, $\mathbf{F}$).

Поскольку для любой замкнутой формулы A может быть построена её ССФ $\mathbb{V}(C_1 \wedge \ldots \wedge C_l)$, где для каждого $i = 1, \ldots, l$ C_i — дизъюнкт, то равносильная этой ССФ формула $C \leftrightharpoons \mathbb{V}C_1 \wedge \ldots \wedge \mathbb{V}C_l$ является конъюнкцией клауз и невыполнима одновременно с формулой A (см. теорему 4.1.2). Формулу C называют *клаузальной формой* формулы A.

Клауза называется *клаузой Хорна*, если у неё не более одного заключения.

Замечание 4.5.1. «Клауза» является одним из вариантов перевода англоязычного термина «clause». Встречается и другой вариант перевода — «дизъюнкт». Первый вариант перевода в контексте логического программирования предпочтительнее второго хотя бы потому, что дизъюнкт не обязательно замкнут, а клауза — своеобразная запись замкнутой формулы. Если выбирают второй вариант перевода, то вместо клауз Хорна говорят о *хорновских дизъюнктах*. ◁

Клауза Хорна называется *правилом*, если у неё ровно одно заключение. Правило имеет вид $B \leftarrow A_1, \ldots, A_m$ $(m \geqslant 0)$; формула B называется *головой*, а список формул $A_1, \ldots, A_m$ — *телом* этого правила.

Правило называется *фактом*, если у него нет посылок. Факт имеет вид $B \leftarrow$.

Логической программой называется конечное множество правил.

Упражнение 4.5.2. Докажите, что любая логическая программа (рассматриваемая как множество замкнутых формул) имеет модель. ◁

Запросом G к логической программе (*целью* G для логической программы), называется список $G_1, \ldots, G_n$, где $n \in \mathbb{N}$ и для каждого $i = 1, \ldots, n$ G_i есть атомарная формула, не являющаяся истинностной константой. Каждую из формул G_i $(i = 1, \ldots, n)$ называют *подцелью* цели G. При $n > 0$ запрос G есть иная запись формулы $G_1 \wedge \ldots \wedge G_n$. При $n = 0$ запрос G называется *пустым* и обозначается $\blacksquare$. Там, где пустой запрос будет использоваться как формула, мы будем считать, что эта формула есть истинностная константа $\mathbf{T}$.

Сейчас нам потребуется результат применения подстановки к (бескванторной) формуле, представляющей запрос. Мы получаем определение результата $E\theta$ применения подстановки θ к бескванторной формуле E, заменив в определении 4.3.2 слово «выражение» на слова «бескванторная формула».

Ответить на запрос G к логической программе $\mathcal{P}$ означает выяснить, верно ли, что $\mathcal{P} \models \exists G$, и если это так, то найти подстановку θ такую, что $\mathcal{P} \models G\theta$ и $Dom(\theta)$ есть подмножество множества всех переменных, входящих в G. Такую подстановку θ называют *ответом на запрос* G; если $\mathcal{P} \not\models \exists G$, то *ответом на запрос* G будет слово «неудача». *Декларативная семантика логической программы* заключается в данном определении ответа на запрос к логической программе.

Можно доказать, что если $\mathcal{P} \models \exists G$, то существует такая подстановка θ, что $\mathcal{P} \models G\theta$. Однако это было бы, вообще говоря, не так, если бы в качестве элементов множества $\mathcal{P}$ допускались бы произвольные клаузы, а не только правила. Например, $P(a) \vee P(b) \models \exists x P(x)$, но не существует терма t такого, что $P(a) \vee P(b) \models P(t)$. Всё же попробуем методом резолюций доказать $P(a) \vee P(b) \models \exists x P(x)$, что равносильно $\models P(a) \vee P(b) \supset \exists x P(x)$, и, в свою очередь, равносильно невыполнимости множества $\{P(a) \vee P(b),\ \neg P(x)\}$. Никакой резолютивный вывод $\square$ из него не даёт искомой подстановки, поскольку в любом таком выводе используются два НОУ, один из которых переменной x сопоставляет a, а другой — b.

4.5.2. SLD-резолюция

Пусть даны логическая программа $\mathcal{P} \leftrightharpoons \{C_1, \ldots, C_k\}$, $k \geqslant 0$ (при $k = 0$ множество $\mathcal{P}$ пусто) и запрос $G \leftrightharpoons (G_1, \ldots, G_n)$, $n \geqslant 0$. При $n = 0$ ответом на этот запрос будет тождественная подстановка.

Рассмотрим, как можно устанавливать, что $\mathcal{P} \models \exists G$, при $n > 0$. По теореме 2.4.3 о логическом следствии имеем: $\mathcal{P} \models \exists G$, если и только если $\models C_1 \wedge \ldots \wedge C_k \supset \exists G$. Общезначимость формулы $C_1 \wedge \ldots \wedge C_k \supset \exists G$ равносильна невыполнимости отрицания этой формулы, которое семантически эквивалентно формуле

$$C_1 \wedge \ldots \wedge C_k \wedge \forall\!\!\!\forall(\neg G_1 \vee \ldots \vee \neg G_n).$$

Последняя формула невыполнима тогда и только тогда, когда невыполнимо множество дизъюнктов

$$S \leftrightharpoons \{C'_1,\ \ldots,\ C'_k,\ (\neg G_1 \vee \ldots \vee \neg G_n)\}, \qquad (*)$$

где для каждого $i = 1, \ldots, k$ C'_i — матрица правила C_i (т. е. дизъюнкт C'_i получается из правила C_i удалением всех кванторных приставок).

Невыполнимость множества S может быть установлена с помощью метода резолюций. Для поиска резолютивного вывода из такого множества дизъюнктов может быть применена специальная стратегия, называемая SLD-резолюцией. Перейдём к определению вывода в рамках этой стратегии.

Определение 4.5.3. Пусть даны дизъюнкты

$$\Gamma \leftrightharpoons \neg G_1 \vee \ldots \vee \neg G_{j-1} \vee \neg G_j \vee \neg G_{j+1} \vee \ldots \vee \neg G_n \quad (n > 0) \quad \text{и}$$

$$C \leftrightharpoons B \vee \neg A_1 \vee \ldots \vee \neg A_m \quad (m \geqslant 0).$$

Считаем, что никакая предметная переменная не входит в Γ и C одновременно (иначе переименуем переменные в дизъюнкте C так, чтобы данное условие выполнялось). Пусть также для некоторого $j \in \{1, 2, \ldots, n\}$ подстановка σ является НОУ атомарных формул G_j и B. Тогда *SLD-резольвентой* дизъюнктов Γ и C называется дизъюнкт

$$(\neg G_1 \vee \ldots \vee \neg G_{j-1} \vee \neg A_1 \vee \ldots \vee \neg A_m \vee \neg G_{j+1} \vee \ldots \vee \neg G_n)\sigma,$$

который при $n = 1$ и $m = 0$ является пустым дизъюнктом. ◁

Определение 4.5.4. Пусть Γ_l — дизъюнкт, S — множество дизъюнктов, имеющее вид $(*)$. *SLD-резолютивным выводом* дизъюнкта Γ_l из множества S называется резолютивный вывод, дерево которого изображено на рис. 4.8, где $i_0, \ldots, i_{l-1} \in \{1, 2, \ldots, k\}$, $\Gamma_0 \leftrightharpoons (\neg G_1 \vee \ldots \vee \neg G_n)$ и для каждого $j = 0, 1, \ldots, l-1$ дизъюнкт Γ_{j+1} есть SLD-резольвента дизъюнктов Γ_j и C'_{i_j}.

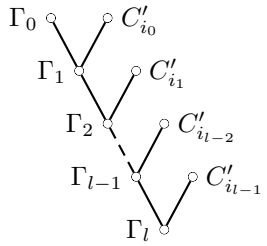


Рис. 4.8. Дерево SLD-резолютивного вывода дизъюнкта Γ_l из множества S.

Далее, пусть в данном SLD-резолютивном выводе при построении SLD-резольвенты дизъюнктов Γ_j и C'_{i_j} использовался НОУ σ_{j+1} для каждого $j = 0, 1, \ldots, l-1$. Тогда подстановка $\sigma_1\sigma_2 \ldots \sigma_l$ при $l > 0$ или тождественная подстановка $\{\}$ при $l = 0$ называется *вычисленной подстановкой* этого вывода.

Наконец, пусть $\{t_1/x_1, \ldots, t_j/x_j,\ t_{j+1}/x_{j+1}, \ldots, t_{l'}/x_{l'}\}$ является вычисленной подстановкой этого вывода, каждая из переменных $x_1, \ldots, x_j$ входит в Γ_0, но ни одна из переменных $x_{j+1}, \ldots, x_{l'}$ не входит в Γ_0. Тогда *вычисленной ответной подстановкой* этого вывода называется $\{t_1/x_1, \ldots, t_j/x_j\}$. ◁

Логическая программа, определяющая предков человека

Приведём пример того, как может осуществляться поиск ответа на запрос к логической программе с помощью построения SLD-резолютивного вывода.

Представим некоторые генеалогические знания в виде логической программы. Чтобы задать в виде фактов то, что предикат «человек x является родителем человека y» истинен для нескольких пар людей, введём двухместный предикатный символ Par, и для каждого рассматриваемого человека введём соответствующую ему предметную константу. Пусть даны следующие факты:

1) $Par(cAdam, cCain) \leftarrow$,

2) $Par(cEve, cCain) \leftarrow$,

3) $Par(cAdam, cAbel) \leftarrow$,

4) $Par(cEve, cAbel) \leftarrow$,

5) $Par(cCain, cEnoch) \leftarrow$.

Теперь зададим в виде правил предикат «человек x является предком человека z». Для этого введём двухместный предикатный символ $Pred$ и положим, что формула $Pred(x, z)$ выражает этот предикат (в подразумеваемой интерпретации). С другой стороны, для любых людей x и z верно, что человек x является предком человека z, если x — родитель человека z, или существует человек y такой, что x — родитель человека z, а y — предок человека z. Запишем утверждение предыдущего предложения в виде формулы

$$\forall(Par(x, z) \vee \exists y(Par(x, y) \wedge Pred(y, z)) \supset Pred(x, z)),$$

которая в силу $p \vee q \supset r \sim (p \supset r) \wedge (q \supset r)$ равносильна формуле

$$\forall((Par(x, z) \supset Pred(x, z)) \wedge (\exists y(Par(x, y) \wedge Pred(y, z)) \supset Pred(x, z))).$$

Последняя формула равносильна следующей формуле:

$$\forall(Par(x, z) \supset Pred(x, z)) \wedge \forall(Par(x, y) \wedge Pred(y, z) \supset Pred(x, z)). \quad (**)$$

Согласно декларативной семантике в логической программе факты и правила по сути соединены конъюнкцией; действительно, $\{C_1, \ldots, C_k\} \vDash E$, если и только если $C_1 \wedge \ldots \wedge C_k \vDash E$. Таким образом, формула $(**)$ записывается в виде двух правил[9]:

6) $Pred(x, z) \leftarrow Par(x, z)$,

7) $Pred(x, z) \leftarrow Par(x, y),\ Pred(y, z)$.

Итак, множество правил 1–7 представляет собой логическую программу.

Зададим запрос

[9]Мы могли бы мыслить в терминах правил и записать эти два правила сразу, без предварительных рассуждений с предикатными формулами более общего вида.

$$Pred(x, cEnoch),\ \ Par(x, cAbel)$$

к этой программе, который требует выяснить, кто является предком человека, обозначенного $cEnoch$, и при этом родителем человека, обозначенного $cAbel$. Попробуем выяснить, невыполнимо ли множество, которое состоит из дизъюнктов, являющихся матрицами правил 1–7, и следующего дизъюнкта, соответствующего запросу:

8) $\neg Pred(x, cEnoch) \vee \neg Par(x, cAbel)$.

Попытаемся найти SLD-резолютивный вывод пустого дизъюнкта $\square$ из этого множества.

(a) Последовательно просматриваем правила 1–7, начиная с первого. В каждом просматриваемом правиле переименовываем переменные так, чтобы после переименования правило и запрос не имели общих переменных; затем проверяем, унифицируема ли голова полученного правила с первой подцелью $Pred(x, cEnoch)$ запроса. Таким образом найдём правило

6′) $Pred(x_1, z_1) \leftarrow Par(x_1, z_1)$,

которое конгруэнтно правилу 6. Построим SLD-резольвенту дизъюнкта[10] 6′ и дизъюнкта 8. Подстановка $\sigma_1^1 \rightleftharpoons \{x/x_1,\ cEnoch/z_1\}$ является НОУ головы правила 6′ и подцели $Pred(x, cEnoch)$; тогда искомая резольвента такова:

1.9) $\neg Par(x, cEnoch) \vee \neg Par(x, cAbel)$.

Полученную резольвенту 1.9 рассматриваем как текущий запрос

$$Par(x, cEnoch),\ \ Par(x, cAbel).$$

(b) Снова просматриваем правила 1–7, начиная с первого, и ищем правило, голова которого (после переименования переменных в этом правиле) унифицируема с первой подцелью $Par(x, cEnoch)$ текущего запроса 1.9. Таково правило 5, а найденный НОУ — $\sigma_2^1 \rightleftharpoons \{cCain/x\}$. SLD-резольвентой дизъюнктов 5 и 1.9 является

1.10) $\neg Par(cCain, cAbel)$,

которая рассматривается как текущий запрос

$$Par(cCain, cAbel).$$

(c) Опять просматриваем правила 1–7 с первого, пытаясь найти правило, голова которого унифицируема с текущим запросом $Par(cCain, cAbel)$. Таких правил нет, поэтому на данном пути поиска вывода не получить $\square$.

Вернёмся к начатому на шаге (b) нахождению правила (с переименованными переменными), голова которого унифицируема с первой подцелью $Par(x, cEnoch)$ запроса 1.9: ведь на шаге (b) мы выбрали правило 5, не досмотрев до конца все правила программы. Просматривая правила программы, идущие за правилом 5, мы не находим правило, голова которого (после переименования переменных в этом правиле) унифицируема с $Par(x, cEnoch)$.

[10] Для краткости здесь и ниже мы обозначаем дизъюнкт, являющийся матрицей правила, номером этого правила.

Тогда вернёмся к начатому на шаге (a) нахождению правила (с переименованными переменными), голова которого унифицируема с первой подцелью $Pred(x, cEnoch)$ запроса 8: ведь на шаге (a) мы выбрали правило 6, не досмотрев до конца все правила программы. Просматривая правила программы, идущие за правилом 6, найдём правило, голова которого (после переименования переменных в этом правиле) унифицируема с $Pred(x, cEnoch)$. Таково правило

7′) $Pred(x_2, z_2) \leftarrow Par(x_2, y_2),\ Pred(y_2, z_2),$

конгруэнтное правилу 7. Построим SLD-резольвенту дизъюнктов 7′ и 8. Подстановка $\sigma_1^2 \leftrightharpoons \{x/x_2,\ cEnoch/z_2\}$ является НОУ головы правила 7′ и подцели $Pred(x, cEnoch)$; тогда искомая резольвента такова:

2.9) $\neg Par(x, y_2) \vee \neg Pred(y_2, cEnoch) \vee \neg Par(x, cAbel).$

Полученную резольвенту 2.9 рассматриваем как текущий запрос

$$Par(x, y_2),\ Pred(y_2, cEnoch),\ Par(x, cAbel).$$

(d) Просматриваем правила 1–7 с первого, пытаясь найти правило (с переименованными переменными), голова которого унифицируема с первой подцелью $Par(x, y_1)$ текущего запроса 2.9. Таково правио 1, а найденный НОУ — $\sigma_2^2 \leftrightharpoons \{cAdam/x,\ cCain/y_2\}$. SLD-резольвентой дизъюнктов 1 и 2.9 является

2.10) $\neg Pred(cCain, cEnoch) \vee \neg Par(cAdam, cAbel).$

(e) Повторяем просмотр всех правил программы, начиная с первого, и находим правило, конгруэнтное правилу 6 (в данном случае необязательно переименовывать переменные в правиле 6, но мы делаем это для единообразия):

6″) $Pred(x_3, z_3) \leftarrow Par(x_3, z_3),$

голова которого унифицируема с первой подцелью $Pred(cCain, cEnoch)$ текущего запроса. При этом находим НОУ $\sigma_3^2 \leftrightharpoons \{cCain/x_3,\ cEnoch/z_3\}$. Строим SLD-резольвенту дизъюнктов 6″ и 2.10:

2.11) $\neg Par(cCain, cEnoch) \vee \neg Par(cAdam, cAbel).$

(f) Опять просматриваем все правила программы, начиная с первого, и строим SLD-резольвенту дизъюнктов 5 и 2.11, при этом используется НОУ $\sigma_4^2 \leftrightharpoons \{\}$:

2.12) $\neg Par(cAdam, cAbel).$

(g) Ещё раз просматриваем все правила программы, начиная с первого, и строим SLD-резольвенту дизъюнктов 3 и 2.12, при этом используется НОУ $\sigma_5^2 \leftrightharpoons \{\}$:

2.13) $\square$.

Итак, мы получили SLD-резолютивный вывод (1, ..., 8, 2.9, ..., 2.13) пустого дизъюнкта из множества дизъюнктов 1–8. Вычисленная подстановка этого вывода есть $\sigma_1^2\sigma_2^2\sigma_3^2\sigma_4^2\sigma_5^2$, из неё мы выделяем вычисленную ответную подстановку $\{cAdam/x\}$, которая, очевидно, является ответом на исходный запрос. Вопрос о том, будет ли вычисленная ответная подстановка любого SLD-резолютивного вывода пустого дизъюнкта ответом на запрос к логической программе, мы скоро рассмотрим (см. теорему 4.5.6 и замечание 4.5.7).

Вычисления, произведённые при поиске ответа на запрос, можно наглядно представить в виде дерева (называемого *SLD-деревом*), каждый узел которого является (или помечен) некоторым запросом, причём

а) корнем этого дерева является исходный запрос, и

б) для каждого узла G этого дерева, если из G и некоторого правила C был образован запрос G' посредством построения резольвенты с помощью НОУ σ, то G' является сыном узла G, и ребро, соединяющее G и G', помечено C и σ.

Такое дерево, соответствующее проведённому нами поиску ответа на запрос, изображено на рис. 4.9.

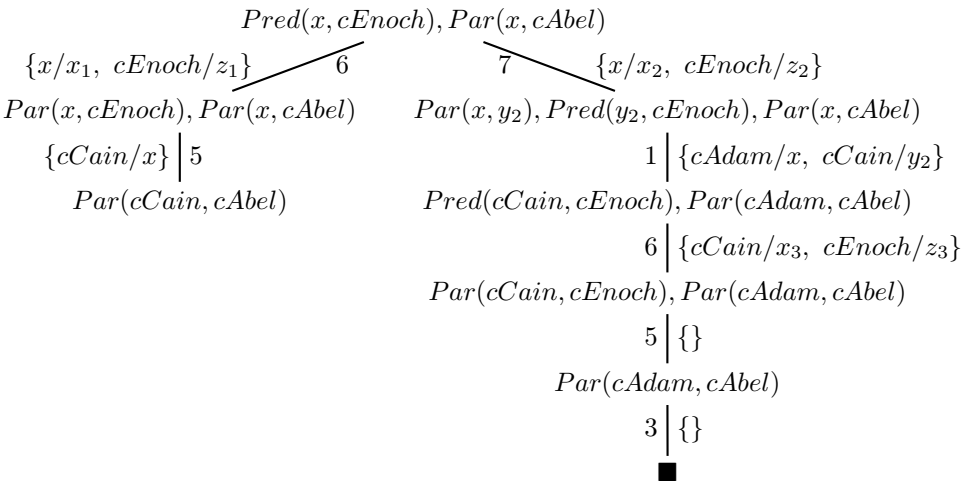


Рис. 4.9. SLD-дерево.

Заметим, что, совершая возвраты (аналогично тому, как мы это делали на шаге (c)) и в случае получения $\square$, мы можем по-новому продолжить поиск вывода $\square$ и получить ещё одну вычисленную ответную подстановку $\{cEve/x\}$.

Упражнение 4.5.5. Осуществите пошаговый поиск второй вычисленной ответной подстановки. $\triangleleft$

4.5.3. Операционная семантика логической программы

Вышеприведённый пример поиска ответа на запрос к логической программе показывает существенные черты некоторого алгоритма, который

может осуществлять такой поиск. Опишем такой алгоритм *search*, тем самым задав *операционную семантику логической программы*. Вообще, под операционной семантикой какой-либо программы понимают точное описание того, как эта программа исполняется некоторой виртуальной машиной. В качестве виртуальной машины может выступать как компьютер (иногда гипотетический), так и программа, исполняемая в свою очередь некоторой виртуальной машиной и называемая интерпретатором. Стандартные реализации Prolog используют усовершенствованные варианты алгоритма *search*.

Пусть дана произвольная логическая программа

$$\mathcal{P} \leftrightharpoons \{C_1, \ldots, C_k\},$$

все правила которой занумерованы числами $1, 2, \ldots, k$. Представим алгоритм *search* в виде следующей рекурсивной функции $search(G)$, аргументом которой является цель $G \leftrightharpoons (G_1, \ldots, G_n)$:

(1) Если $n = 0$, то выдать в качестве ответа тождественную подстановку.

(2) Для каждого $i = 1, \ldots, k$ выполнить следующие действия.

  (2.1) Переименовать в правиле C_i переменные на новые. Пусть в результате этого переименования переменных в C_i получилось правило $B \leftarrow A_1, \ldots, A_m$ $(m \geqslant 0)$, которое не имеет общих переменных с текущей целью G.

  (2.2) Если литеры B и G_1 унифицируемы, то выполнить следующие действия.

    (2.2.1) Найти НОУ σ литер B и G_1.

    (2.2.2) Если вызов $search(A_1\sigma, \ldots, A_m\sigma, G_2\sigma, \ldots, G_n\sigma)$ вернул подстановку τ, то выдать в качестве ответа $\sigma\tau$.

(3) Выдать в качестве ответа «неудача».

Алгоритм *search* ищет SLD-резолютивный вывод пустого дизъюнкта из множества S (см. $(*)$ в начале раздела 4.5.2). Такой вывод найден, когда цель становится пуста. Для образования SLD-резольвенты всегда выбирается первая подцель и голова правила, причём это правило ищется путём последовательного просмотра программы.

В логическом программировании наряду с термином «операционная семантика» применяется синонимичный термин «процедурная семантика», что объясняется следующей наглядной трактовкой алгоритма *search*. Правило $B \leftarrow A_1, \ldots, A_m$ $(m \geqslant 0)$ трактуется как процедура с именем B и телом, состоящим из вызовов процедур $A_1, \ldots, A_m$. Для достижения цели $G_1, \ldots, G_n$ требуется исполнить каждую из процедур с именами $G_1, \ldots, G_n$. Для исполнения процедуры G_1 находится такое правило (с переименованными переменными) $B \leftarrow A_1, \ldots, A_m$, что существует НОУ σ B и G_1, и строится новая текущая цель $A_1\sigma, \ldots, A_m\sigma, G_2\sigma, \ldots, G_n\sigma$. Таким образом, исполнение процедуры моделирует построение SLD-резольвенты.

Теорема 4.5.6 (корректность вычисленной подстановки)**.** *Пусть даны программа $\mathcal{P}$ и запрос $G \leftrightharpoons (G_1, \ldots, G_n)$. Тогда если вызов $search(G)$ выдал подстановку θ, то $\mathcal{P} \vDash G\theta$.*

Д о к а з а т е л ь с т в о. Воспользуемся индукцией по длине l вывода пустого дизъюнкта при исполнении вызова $search(G)$. База индукции ($l = 1$) очевидна.

Индукционный переход. Рассмотрим вызов $search(G)$, который построил вывод длины $l > 1$ и выдал подстановку θ. Первое применение правила резолюции в этом выводе выглядит следующим образом. Из дизъюнкта $B \vee \neg A_1 \vee \ldots \vee \neg A_m$, являющегося матрицей правила (с переименованными переменными) $B \leftarrow A_1, \ldots, A_m$ программы $\mathcal{P}$, и дизъюнкта $\neg G_1 \vee \ldots \vee \neg G_n$, соответствующего цели $G_1, \ldots, G_n$, образована SLD-резольвента

$$(\neg A_1 \vee \ldots \vee \neg A_m \vee \neg G_2 \vee \ldots \vee \neg G_n)\sigma,$$

где σ — НОУ B и G_1. Эта резольвента соответствует цели

$$\widetilde{G} \leftrightharpoons (A_1\sigma, \ldots, A_m\sigma, G_2\sigma, \ldots G_n\sigma).$$

Далее при исполнении вызова $search(G)$ был произведён вызов $search(\widetilde{G})$, который построил вывод длины, меньшей l, и выдал подстановку τ. Наконец, вызов $search(G)$ в качестве результата выдал подстановку $\theta = \sigma\tau$. По индукционному предположению $\mathcal{P} \models \widetilde{G}\tau$, т. е.

$$\mathcal{P} \models (A_1 \wedge \ldots \wedge A_m \wedge G_2 \wedge \ldots \wedge G_n)\sigma\tau. \qquad (\star)$$

Рассмотрим случай, когда $m > 0$ (в случае, когда $m = 0$, проводимые далее рассуждения упрощаются). Правило $B \leftarrow A_1, \ldots, A_m$ принадлежит программе $\mathcal{P}$, следовательно,

$$\mathcal{P} \models (A_1 \wedge \ldots \wedge A_m)\sigma \supset B\sigma.$$

Так как $B\sigma$ совпадает с $G_1\sigma$, то

$$\mathcal{P} \models (A_1 \wedge \ldots \wedge A_m)\sigma \supset G_1\sigma,$$

поэтому из $(\star)$ получаем

$$\mathcal{P} \models (G_1 \wedge G_2 \wedge \ldots \wedge G_n)\sigma\tau,$$

т. е. $\mathcal{P} \models G\theta$. ◁

Замечание 4.5.7. Доказательство предыдущей теоремы по сути не изменится, если рассматривать произвольный SLD-резолютивный вывод пустого дизъюнкта из множества S (см. $(*)$ в начале раздела 4.5.2) вместо вывода, который строит алгоритм *search*. Поэтому вычисленная ответная подстановка любого SLD-резолютивного вывода пустого дизъюнкта из множества S является ответом на запрос к логической программе $\mathcal{P}$.

Можно доказать, что и ответ «неудача» алгоритма *search* корректен, точнее, если вызов $search(G)$ выдал «неудача», то $\mathcal{P} \not\models \exists G$. Это утверждение и теорема 4.5.6 говорят о том, что операционная семантика логической программы корректна относительно декларативной.

При заданной операционной семантике большую роль играет порядок правил в программе. Поиск ответа на один и тот же запрос при одном порядке правил может завершиться, а при другом — нет. Аналогичное утверждение справедливо и про порядок атомарных формул в теле правила. Поэтому программисту приходится учитывать этот аспект операционной семантики.

Упражнение 4.5.8. Приведите пример такой логической программы $\mathcal{P}$ и такого запроса G к ней, что $\mathcal{P} \models \exists G$, но поиск ответа на этот запрос (в соответствии с заданной операционной семантикой) не завершается. Можно ли переупорядочить формулы в правилах или правила этой программы так, чтобы поиск ответа на этот запрос завершился? ◁

Мы бы сказали, что операционная семантика, заданная некоторым алгоритмом $search'$, полна относительно декларативной семантики логической программы, если бы для любой логической программы $\mathcal{P}$ и любого запроса G выполнялось следующее: если $\mathcal{P} \models \exists G$, то $search'(G)$ выдаёт такую подстановку θ, что $\mathcal{P} \models G\theta$; иначе $search'(G)$ выдаёт «неудача». Как показывает предыдущее упражнение, операционная семантика, заданная алгоритмом $search$, неполна относительно декларативной семантики. Однако описанная операционная семантика позволяет эффективно реализовать Prolog, тогда как реализация полной операционной семантики была бы практически непригодна для использования из-за неэффективности.

Возврат из рекурсивного вызова функции $search$ при неудаче (т. е. невозможности унифицировать первую подцель с головой какого бы то ни было правила после переименования переменных в этом правиле) и последующая попытка найти вывод пустого дизъюнкта на другом пути поиска называется *откатом* (или *бектрекингом*). В фактических реализациях Prolog такие же возврат и попытка (называемые так же — откатом) производятся и при нахождении вычисленной подстановки, чтобы найти другие вычисленные подстановки.

Упражнение 4.5.9. Модифицируйте алгоритм $search$ так, чтобы после каждого нахождения вычисленной подстановки этот алгоритм продолжал поиск других вычисленных подстановок. ◁

В заключение приведём ещё один пример логической программы.

Логическая программа, определяющая сумму и произведение натуральных чисел

Пусть $Padd$ и $Pmult$ — трёхместные предикатные символы, формула $Padd(x, y, z)$ выражает предикат «сумма натуральных чисел x и y равна z», а $Pmult(x, y, z)$ — предикат «произведение натуральных чисел x и y равно z». Далее, пусть 0 — предметная константа, S — одноместный функциональный символ, предметной константе 0 сопоставляется натуральное число 0, функциональному символу S — функция прибавления единицы к натуральному числу.

Зададим то, как вычисляется сумма и произведение натуральных чисел, следующей логической программой:

1) $Padd(x, 0, x)$,

2) $Padd(x, S(y), S(z)) \leftarrow Padd(x, y, z)$,

3) $Pmult(x, 0, 0)$,

4) $Pmult(x, S(y), z) \leftarrow Pmult(x, y, u),\ Padd(u, x, z)$.

Правила 1 и 2 задают то, как вычисляется сумма натуральных чисел согласно аксиомам элементарной арифметики: факт 1 говорит, что $x + 0 = x$,

а правило 2 — что $x + S(y) = S(x + y)$. Правила 3 и 4 задают то, как вычисляется произведение натуральных чисел: факт 3 говорит, что $x \cdot 0 = 0$, а правило 4 — что $x \cdot S(y) = x \cdot y + x$.

Поиск ответа на запрос $Pmult(S(S(0)), S(S(0)), x)$, требующий вычислить $2 \cdot 2$, даст вычисленную ответную подстановку $\{S(S(S(S(0))))/x\}$, а на запрос $Pmult(x, S(0), S(S(0)))$, требующий решить уравнение $x \cdot 1 = 2$, — $\{S(S(0))/x\}$.

Упражнение 4.5.10. Осуществите пошаговый поиск ответов на эти запросы. ◁

Упражнение 4.5.11. Напишите логические программы, которые могут служить для вычисления различных функций на натуральных числах, например, возведения в степень, факториала, минимума из двух чисел, частного и остатка от (целочисленного) деления. ◁

Глава 5.

Теория вычислимости

Теория вычислимости (также называемая *общей теорией алгоритмов*) изучает вопросы о том, существует ли алгоритм, представляющий общий метод решения какой-либо задачи, или нет. Когда утверждают, что для задачи существует алгоритм, то в качестве обоснования, как правило, достаточно предъявить неформальное описание последовательности действий, ведущей к требуемому результату. Но для доказательства того, что для решения некоторой задачи не существует алгоритма, неформального понятия алгоритма недостаточно и требуется точное определение алгоритма.[1] В данной главе мы изучим несколько подходов к определению алгоритма, докажем, что для решения некоторых задач не существует никакого алгоритма, а также рассмотрим знаменитые теоремы Гёделя о неполноте, которые тесно связаны с теорией вычислимости и показывают принципиальные ограничения формализации рассуждений.

5.1. Определения понятия вычислимости

До сих пор мы употребляли понятие «алгоритм» неформально (интуитивно), рассчитывая на компьютерную грамотность читателя. Мы в общих чертах описывали последовательность действий, согласно которой входные данные преобразуются в результат, и называли такое описание алгоритмом. Но мы рассчитывали, что при необходимости такое описание может быть детализировано, т. е. представлено как точное описание некоторых элементарных действий. Описание каждого действия элементарно в том смысле, что может быть исполнено некоторой машиной, которая «понимает» такие описания. Иначе говоря, мы предполагали, что любое вышеприведённое неформальное описание алгоритма может быть записано на некотором языке программирования, являющимся формальным языком.

Однако ключевое отличие программы, предназначенной для исполнения на компьютере, от алгоритма в теории вычислимости заключается в следующем. В теории вычислимости отвлекаются от временны́х и материальных возможностей, предполагая, что исполнение алгоритма может быть сколь угодно долгим и может потребовать сколь угодно много материалов

[1] Сравните с причиной развития формально аксиоматического метода, указанной в конце раздела 3.5.5.

для осуществления этого исполнения. В этом предположении заключается так называемая *абстракция потенциальной осуществимости.*

Ещё до появления компьютеров было дано несколько точных определений

а) функции, которая (по замыслу автора каждого определения) может быть вычислена некоторым интуитивно (неформально) понимаемым алгоритмом, и

б) алгоритма и, соответственно, функции, которая вычисляется некоторым таким точно определённым алгоритмом.

Поэтому лучше говорить не о подходах к определению понятия алгоритма, а о подходах к определению понятия вычислимой (некоторым алгоритмом) функции — понятия вычислимости.

В данном разделе мы рассмотрим следующие подходы к определению понятия вычислимости:

1) машины Тьюринга,

2) нормальные алгоритмы (алгорифмы[2]) Маркова,

3) лямбда-определимые функции Чёрча (основанные на так называемом лямбда-исчислении),

4) частично рекурсивные функции Клини.

Перечисленные подходы (и многие другие подходы к определению понятия вычислимости) оказались эквивалентными, о чём пойдёт речь в разделе 5.1.5.

Знакомство с несколькими фундаментальными определениями понятия вычислимости полезно не только для дальнейшего изучения теории вычислимости, но и для развития программистского опыта: ведь эти определения по существу представляют собой языки программирования, более того, некоторые из этих определений легли в основу практически используемых языков программирования. Так называемая универсальная машина Тьюринга явилась прообразом компьютера с хранимой в памяти программой (компьютера фон-неймановской архитектуры); машины Тьюринга и компьютеры фон-неймановской архитектуры послужили развитию императивного программирования. Лямдба-исчисление привело к созданию функционального программирования. Нормальные алгорифмы легли в основу ряда декларативных языков программирования (например, языка РЕФАЛ), ориентированных на обработку строк.

5.1.1. Машины Тьюринга

Машину Тьюринга можно интуитивно представлять как ленту с управляющей головкой. Лента поделена на ячейки, в каждой ячейке находится символ из некоторого заранее заданного алфавита. Число ячеек на ленте конечно, но лента считается потенциально бесконечной в обе стороны. Это

[2] «Алгорифм» — использовавшийся А. А. Марковым вариант написания слова «алгоритм».

означает, что при необходимости новая ячейка пристраивается к одному или другому концу ленты. Управляющая головка находится в некотором (текущем) состоянии из конечного числа состояний и наблюдает символ в некоторой ячейке. Машина Тьюринга имеет программу, которая в зависимости от текущего состояния и наблюдаемого символа предписывает заменить этот символ на некоторый символ, перейти в некоторое состояние и сдвигуть головку на одну ячейку вправо или влево или оставить головку на том же месте. Теперь перейдём к определениям.

Пусть X, Y, X' — произвольные множества такие, что $X' \subseteq X$. Тогда функцию f из X' в Y называют *частичной функцией из X в Y*. Если $X' = \varnothing$, то функцию f называют *нигде не определённой*. То, что f является частичной функцией из X в Y, мы будем обозначать посредством $f : X \xrightarrow{p} Y$.

Определение 5.1.1. *Машиной Тьюринга* называется упорядоченная семёрка

$$\langle \Gamma,\ \Sigma,\ \Delta,\ S,\ q_0,\ F,\ P \rangle,$$

где

1) Γ — алфавит, называемый *ленточным* (*рабочим*) *алфавитом*, причём символ _, называемый *пробелом* (*пустым символом*), принадлежит Γ;

2) Σ — алфавит, называемый *входным* (*внешним*) *алфавитом*, причём $\Sigma \subseteq \Gamma \setminus \{\_\}$;

3) Δ — алфавит, называемый *выходным алфавитом*, $\Delta \subseteq \Gamma \setminus \{\_\}$;

4) S — алфавит, называемый *алфавитом состояний* (*внутренним алфавитом*), каждый символ из S называется *состоянием*, $\Gamma \cap S = \varnothing$;

5) $q_0 \in S$ — символ, называемый *начальным состоянием*;

6) $F \subseteq S$ — непустое множество *заключительных состояний*;

7) P — частичная функция из $(S \setminus F) \times \Gamma$ в $S \times \Gamma \times \{L, C, R\}$, называемая *программой*. ◁

Зафиксируем произвольную машину Тьюринга

$$M \leftrightharpoons \langle \Gamma,\ \Sigma,\ \Delta,\ S,\ q_0,\ F,\ P \rangle$$

и будем рассматривать её до конца текущего раздела 5.1.1.

Пусть $q, r \in S,\ a, b \in \Gamma$. Соотношение $P(q, a) = \langle r, b, L \rangle$ мы будем записывать в виде слова $qa \to rbL$, $P(q, a) = \langle r, b, C \rangle$ — в виде слова $qa \to rb$, а $P(q, a) = \langle r, b, R \rangle$ — в виде слова $qa \to rbR$; будем называть слова этих трёх видов *командами*. Программу P будем записывать в виде списка команд, перечисленных в произвольном порядке.

Конфигурацией (*машинным словом*) называется любое слово вида $UqaV$, где $q \in S,\ a \in \Gamma,\ U, V \in \Gamma^*$. При этом говорят, что символ a *наблюдается* (или является *наблюдаемым*) в состоянии q. Конфигурация называется *начальной*, если q есть начальное состояние и слово U пусто. Конфигурация называется *заключительной*, если $q \in F$.

Определение 5.1.2. Пусть дана конфигурация $X \leftrightharpoons UqaV$. *Следующая за X конфигурация*, обозначаемая в этом определении через Y, задаётся таким образом:

1) если в программе P нет команды, начинающейся с qa, то конфигурация Y считается не определённой;

2) если в P имеется команда $qa \to rb$, то $Y = UrbV$;

3) если в P имеется команда $qa \to rbR$, то

   а) если V не пусто, то $Y = UbrV$;

   б) если V пусто, то $Y = Ubr\_$;[3]

4) если в P имеется команда $qa \to rbL$, то

   а) если U не пусто (пусть $U = U'c$ для некоторых $U' \in \Gamma^*$ и $c \in \Gamma$), то $Y = U'rcbV$;

   б) если U пусто, то $Y = r\_bV$.[3]

Будем говорить, что указанная в каждом из пунктов 2–4 *команда применима к конфигурации X*, и что при построении конфигурации Y из конфигурации X машина M совершает *шаг*. То, что Y является следующей за X конфигурацией, будем обозначать через $X \Vdash Y$. $\triangleleft$

Определение 5.1.3. Пусть даны слова $\alpha \in \Sigma^*$, $\beta \in \Delta^*$. Будем говорить, что *машина Тьюринга M на входе α выдаёт β* (или что *результатом работы машины M на входе α является β*) и писать $M(\alpha) = \beta$, если существует такая конечная последовательность конфигураций $Z_0, \ldots, Z_n$ $(n \geqslant 0)$, что выполняются следующие условия:

1) если α не пусто, то $Z_0 = q_0\alpha$, иначе $Z_0 = q_0\_$;

2) для каждого $i = 1, \ldots, n$ $\;Z_{i-1} \Vdash Z_i$;

3) $Z_n = UqW$ для некоторого заключительного состояния $q \in F$ и некоторых слов $U, W \in \Gamma^*$;

4) β есть наибольшее по длине слово, которое является началом слова W и состоит только из символов выходного алфавита Δ.

Говорят, что *машина Тьюринга M останавливается на входе α* (*M завершает работу на входе α*, *M завершается на входе α*, или *M применима к слову α*) и пишут $!M(\alpha)$, если существует результат работы машины M на входе α. $\triangleleft$

Словарной функцией назовём произвольную частичную функцию из $\Sigma_1^* \times \ldots \times \Sigma_n^*$ в Δ_0^*, где $\Sigma_1, \ldots, \Sigma_n, \Delta_0$ — какие угодно алфавиты, $n \in \mathbb{N}_+$.

Определение 5.1.4. Пусть символ # принадлежит входному алфавиту Σ, даны алфавиты $\Sigma_1, \ldots, \Sigma_n \subseteq \Sigma \setminus \{\#\}$, алфавит $\Delta_0 \subseteq \Delta$ и словарная функция $f : \Sigma_1^* \times \ldots \times \Sigma_n^* \xrightarrow{p} \Delta_0^*$. Будем говорить, что *машина Тьюринга M вычисляет функцию f*, если выполняются два условия:

[3] Этот подпункт уточняет интуитивное представление о том, что при необходимости новая ячейка добавляется к концу ленты машины Тьюринга.

1) для всех $\sigma_1 \in \Sigma_1^*, \ldots, \sigma_n \in \Sigma_n^*$ верно: $f(\sigma_1, \ldots, \sigma_n)$ определена тогда и только тогда, когда $!M(\sigma_1 \# \sigma_2 \# \ldots \# \sigma_n)$;

2) для всех $\sigma_1 \in \Sigma_1^*, \ldots, \sigma_n \in \Sigma_n^*$ верно: если $f(\sigma_1, \ldots, \sigma_n)$ определена, то $f(\sigma_1, \ldots, \sigma_n) = M(\sigma_1 \# \sigma_2 \# \ldots \# \sigma_n)$.

Словарная функция называется *вычислимой по Тьюрингу*, если существует машина Тьюринга, которая вычисляет эту функцию. ◁

Замечание 5.1.5. Для любого $n \in \mathbb{N}_+$ любая машина Тьюринга M естественным образом задаёт словарную функцию, которая упорядоченной n-ке слов $\langle \sigma_1, \ldots, \sigma_n \rangle$ сопоставляет слово $M(\sigma_1 \# \ldots \# \sigma_n)$, если $!M(\sigma_1 \# \ldots \# \sigma_n)$, и не определена на этой n-ке в противном случае. ◁

Итак, мы определили понятие вычислимой по Тьюрингу словарной функции. Теперь распространим это понятие вычислимости на функции, аргументами и значениями которых являются натуральные числа.

Для какого угодно $n \in \mathbb{N}_+$ произвольную частичную функцию из $\mathbb{N}^n$ в $\mathbb{N}$ назовём *числовой функцией*. n-местную числовую функцию f будем называть *всюду определённой*, если $dom(f) = \mathbb{N}^n$.

Для каждого натурального числа x назовём *нумералом* числа x и обозначим через $\overline{x}$ слово $||\ldots|$ (в алфавите $\{|\}$), в котором имеется ровно x вхождений символа $|$; при этом $\overline{0}$ есть пустое слово.

Для каждой числовой функции $f : \mathbb{N}^n \xrightarrow{p} \mathbb{N}$ определим словарную функцию $\overline{f} : (\{|\}^*)^n \xrightarrow{p} \{|\}^*$ так, что для любых $x_1, \ldots, x_n, x \in \mathbb{N}$

$$f(x_1, \ldots, x_n) = x \quad \Leftrightarrow \quad \overline{f}(\overline{x}_1, \ldots, \overline{x}_n) = \overline{x}$$

(в частности, $f(x_1, \ldots, x_n)$ и $\overline{f}(\overline{x}_1, \ldots, \overline{x}_n)$ определены или не определены одновременно).

Будем говорить, что *машина Тьюринга M вычисляет числовую функцию f*, если M вычисляет словарную функцию $\overline{f}$. Числовая функция называется *вычислимой по Тьюрингу*, если существует машина Тьюринга, которая вычисляет эту функцию.

Замечание 5.1.6. Ясно, что целые и рациональные числа можно представить словами в подходящих алфавитах и можно аналогично определить вычислимые по Тьюрингу функции, перерабатывающие такие числа. ◁

Машину Тьюринга M будем называть *числовой*, если её входной алфавит есть $\{|, \#\}$, а её выходной алфавит — $\{|\}$.

Замечание 5.1.7. (Частный случай замечания 5.1.5.) Для любого $n \in \mathbb{N}_+$ любая числовая машина Тьюринга M естественным образом задаёт числовую функцию, которая упорядоченной n-ке натуральных чисел $\langle x_1, \ldots, x_n \rangle$ сопоставляет длину слова $M(\overline{x}_1 \# \ldots \# \overline{x}_n)$ (совпадающую с числом вхождений символа $|$ в это слово), если $!M(\overline{x}_1 \# \ldots \# \overline{x}_n)$, и не определена на этой n-ке в противном случае. ◁

Для натуральных чисел $x_1, \ldots, x_n, y$ условимся писать $M(x_1, \ldots, x_n)$ наряду с $M(\overline{x}_1 \# \ldots \# \overline{x}_n)$ и $M(x_1, \ldots, x_n) = y$ наряду с $M(\overline{x}_1 \# \ldots \# \overline{x}_n) = \overline{y}$.

Пример 5.1.8. Построим числовую машину Тьюринга M_+, вычисляющую числовую всюду определённую функцию $f(x, y) = x + y$.

Входной и выходной алфавиты числовой машины Тьюринга определены как $\{|, \#\}$ и $\{|\}$ соответственно. Положим, что $\{\_, \#, |\}$ есть ленточный

алфавит машины M_+, $\{q_0, q_1, q_2, q_3, q_f\}$ — алфавит состояний, q_0 — начальное состояние, $\{q_f\}$ — множество заключительных состояний. Программа машины M_+ такова:

$$
\begin{aligned}
&1)\ q_0| \to q_0|R,\\
&2)\ q_0\# \to q_1|R,\\
&3)\ q_1| \to q_1|R,\\
&4)\ q_1\_ \to q_2\_L,\\
&5)\ q_2| \to q_3\_L,\\
&6)\ q_3| \to q_3|L,\\
&7)\ q_3\_ \to q_f\_R.
\end{aligned}
$$

Неформально поясним работу этого алгоритма. На его вход подаётся слово вида $||\ldots|\#||\ldots|$. Коротко говоря, алгоритм заменяет символ $\#$ на $|$, а затем заменяет последний символ $|$ на $\_$. Более подробное объяснение приведено ниже.

Изначально в состоянии q_0 мы наблюдаем первый символ входного слова вида $||\ldots|\#||\ldots|$. В состоянии q_0 (согласно команде 1) мы пропускаем все символы $|$ до $\#$. Далее, исполняя команду 2, мы заменяем $\#$ на $|$ и переходим в состояние q_1. В этом состоянии мы доходим (команда 3) до $\_$, следующего за последним $|$, и отступаем (команда 4) на один символ назад, переходя в состояние q_2. Заменяем (команда 5) последний $|$ на $\_$ (мы помним, что ранее мы поставили $|$ вместо $\#$) и переходим в состояние q_3. Далее двигаемся влево (команда 6) до $\_$, находящегося перед первым $|$. Наконец, делаем шаг вправо (команда 7), переходя в заключительное состояние q_f. Теперь слово из $|$, начинающееся сразу за q_f, является результатом работы машины Тьюринга M.

Проиллюстрируем то, как производится вычисление по шагам, записав так называемый протокол вычислений машины Тьюринга M_+:

$$
\begin{aligned}
&q_0||\#| \ \Vdash_1\ |q_0|\#| \ \Vdash_1\ ||q_0\#| \ \Vdash_2\ |||q_1| \ \Vdash_3\ ||||q_1\_ \ \Vdash_4\ |||q_2|\_ \ \Vdash_5\\
&||q_3|\_\_ \ \Vdash_6\ |q_3||\_\_ \ \Vdash_6\ q_3|||\_\_ \ \Vdash_6\ q_3\_|||\_\_ \ \Vdash_7\ \_q_f|||\_\_\ .
\end{aligned}
$$

Рядом со знаком $\Vdash$ мы указали номер команды, которая была применена к конфигурации, стоящей непосредственно слева от этого знака.

Доказательство того, что машина M_+ действительно вычисляет функцию f, можно провести с помощью индукции по длинам нумералов аргументов f; такое доказательство остаётся в качестве упражнения.

Легко видеть, что машина M_+ задаёт числовую нигде не определённую трёхместную функцию. Однако, если в программу машины M_+ добавить команду $q_1\# \to q_2\#L$, то получившаяся машина будет задавать числовую всюду определённую функцию $g(x, y, z) = x + y$. ◁

Упражнение 5.1.9. Как было сказано в предпоследнем абзаце предыдущего примера 5.1.8, докажите, что машина M_+ вычисляет функцию f. ◁

Упражнение 5.1.10. Докажите, что нигде не определённая функция (какой угодно местности) вычислима по Тьюрингу. ◁

Упражнение 5.1.11. Постройте машины Тьюринга, вычисляющие сумму любого количества (натуральных) чисел, модуль разности двух чисел, произведение двух чисел, остаток от деления одного числа на другое; машину

Тьюринга, выдающую 1, если входное слово в алфавите $\{a, b\}$ является палиндромом (определение палиндрома см. в примере 1.2.2 на с. 32), и 0 в противном случае. ◁

Упражнение 5.1.12. Докажите, что композиция двух одноместных вычислимых по Тьюрингу функций вычислима по Тьюрингу. ◁

5.1.2. Нормальные алгоритмы

Кратко опишем ещё одно точное понятие алгоритма — понятие нормального алгоритма. Нормальный алгоритм осуществляет определённые замены подслов в перерабатываемом слове. Мы упомянем предпосылку возникновения понятия нормального алгоритма в конце раздела 5.5.1, а сейчас дадим определения.

Пусть Γ — алфавит, в который не входят символы $\cdot$ и $\to$.

Продукцией (формулой подстановки) в алфавите Γ называется слово вида $\alpha \to \beta$ и слово вида $\alpha \to \cdot\, \beta$, где $\alpha, \beta \in \Gamma^*$. Слово первого вида называется *простой продукцией*, а второго вида — *заключительной продукцией*.

Конечная последовательность продукций в алфавите Γ называется *схемой в алфавите* Γ.

Нормальным алгоритмом в алфавите Γ называется упорядоченная пара, состоящая из Γ и схемы в алфавите Γ.

Пусть схема нормального алгоритма $\mathfrak{A}$ в алфавите Γ имеет вид

$$\begin{aligned} &1)\ \alpha_1 \to \pi_1\ \beta_1\ , \\ &\dots \\ &n)\ \alpha_n \to \pi_n\ \beta_n\ , \end{aligned}$$

где для каждого $i = 1, \dots, n$ $\alpha_i, \beta_i \in \Gamma^*$, π_i есть символ $\cdot$ или пустое слово.

Опишем, как работает алгоритм $\mathfrak{A}$ на входе $\alpha \in \Gamma^*$, попутно определив $\mathfrak{A}(\alpha)$ — *результат работы алгоритма* $\mathfrak{A}$ *на входе* α. Если не существует такого i, что α_i входит в α, то результатом $\mathfrak{A}(\alpha)$ является α; иначе обозначим через i наименьшее число, для которого α_i входит в α. Тогда первое вхождение α_i в α заменяется на β_i; пусть результатом этой замены является слово α'. Если $\pi_i = \cdot$ (т. е. если i-я продукция заключительная), то результатом $\mathfrak{A}(\alpha)$ является α'; иначе (т. е. если i-я продукция простая) $\mathfrak{A}(\alpha)$ есть $\mathfrak{A}(\alpha')$, т. е. со словом α' делается то же, что и с α. Описанный процесс может никогда не закончиться, в таком случае результат работы алгоритма $\mathfrak{A}$ на входе α не определён.

Говорят, что *нормальный алгоритм* $\mathfrak{A}$ *останавливается на входе* α ($\mathfrak{A}$ *применим к слову* α) и пишут $!\mathfrak{A}(\alpha)$, если существует результат работы алгоритма $\mathfrak{A}$ на входе α.

Нормальным алгоритмом над алфавитом Σ называется нормальный алгоритм в некотором алфавите Γ, содержащем Σ. (Это определение отражает то, что символы из $\Gamma \setminus \Sigma$ могут играть лишь вспомогательную роль в реализации алгоритма и не присутствовать ни во входе, ни в результате работы алгоритма.)

Пусть $\mathfrak{A}$ — нормальный алгоритм над алфавитом Σ, $\# \in \Sigma$, даны алфавиты $\Sigma_1, \dots, \Sigma_n \subseteq \Sigma \setminus \{\#\}$, алфавит $\Delta \subseteq \Sigma$ и словарная функция $f : \Sigma_1^* \times \ldots \times \Sigma_n^* \xrightarrow{p} \Delta^*$. Будем говорить, что *алгоритм* $\mathfrak{A}$ *вычисляет функцию* f, если выполнены два условия:

1) для всех $\sigma_1 \in \Sigma_1^*, \ldots, \sigma_n \in \Sigma_n^*$ верно: $f(\sigma_1, \ldots, \sigma_n)$ определена тогда и только тогда, когда $!\mathfrak{A}(\sigma_1 \# \sigma_2 \# \ldots \# \sigma_n)$;

2) для всех $\sigma_1 \in \Sigma_1^*, \ldots, \sigma_n \in \Sigma_n^*$ верно: если $f(\sigma_1, \ldots, \sigma_n)$ определена, то

$$f(\sigma_1, \ldots, \sigma_n) = \mathfrak{A}(\sigma_1 \# \sigma_2 \# \ldots \# \sigma_n).$$

Словарная функция называется *нормально вычислимой*, если существует нормальный алгоритм, который вычисляет эту функцию.

Определение нормально вычислимой числовой функции получается из определения вычислимой по Тьюрингу числовой функции (см. раздел 5.1.1) заменой слов «вычислима по Тьюрингу» на слова «нормально вычислима» (с соответствующими изменениями окончаний этих слов).

Пример 5.1.13. 1. Пусть $\mathfrak{A}$ — нормальный алгоритм в алфавите $\{a, b, c, d\}$ с одной продукцией $ab \to \cdot\, bac$, и пусть на вход подаётся слово (обозначенное через) α. Тогда $\mathfrak{A}$ заменяет первое вхождение ab в α на bac, если такое вхождение есть, иначе оставляет α без изменений.

2. Нормальный алгоритм в каком угодно алфавите с одной продукцией $\to$ не применим ни к какому слову (он бесконечное число раз заменяет вхождение пустого слова на пустое слово).

3. Нормальный алгоритм в каком угодно алфавите с одной продукцией $\to \cdot$ оставляет входное слово без изменений (он единожды заменяет вхождение пустого слова на пустое слово и останавливается).

4. Нормальный алгоритм в любом алфавите, содержащем символ a, с одной продукцией $\to \cdot\, a$ приписывает символ a в начало входного слова.

5. Нормальный алгоритм в алфавите $\{|, \#\}$ с одной продукцией $\# \to$ выполняет сложение любого количества натуральных чисел, в частности, вычисляет числовые всюду определённые функции $f(x, y) = x + y$ и $g(x, y, z) = x + y + z$. ◁

Пример 5.1.14. Определим нормальный алгоритм $\mathfrak{A}$, приписывающий слово ba в конец входного слова в алфавите $\{a, b\}$. Используем вспомогательный символ q для того, чтобы поместить q в конец входного слова, а затем заменить q на ba. $\mathfrak{A}$ есть нормальный алгоритм в алфавите $\{a, b, q\}$ (и над алфавитом $\{a, b\}$) со следующей схемой:

$$\begin{aligned} &1)\ qa \to aq\ , \\ &2)\ qb \to bq\ , \\ &3)\ q \to \cdot\, ba\ , \\ &4)\ \to q\ . \end{aligned}$$

Продемонстрируем работу алгоритма $\mathfrak{A}$:

$$abb \ \Vdash_4 \ qabb \ \Vdash_1 \ aqbb \ \Vdash_2 \ abqb \ \Vdash_2 \ abbq \ \Vdash_3 \ abbba\ .$$

Рядом со знаком $\Vdash$ мы указали номер продукции, в соответствии с которой произошла замена подслова. ◁

Имеет место следующая теорема, доказательство которой мы не приводим из-за его громоздкости (см. также упражнение 5.5.7 на с. 231).

Теорема 5.1.15. *Пусть f — словарная функция. Тогда f нормально вычислима, если и только если f вычислима по Тьюрингу.*

Упражнение 5.1.16. Постройте нормальные алгоритмы, которые делают то же, что и машины Тьюринга из упражнения 5.1.11. ◁

Упражнение 5.1.17. Постройте нормальный алгоритм, выдающий 1, если входное слово в алфавите языка логики высказываний является пропозициональной формулой (см. определение 1.1.3 на с. 13), и 0 в противном случае. ◁

5.1.3. Лямбда-исчисление и лямбда-определимые функции

Введение

Программы на чистом функциональном языке программирования строятся из единственного элементарного действия — вызова функции. Функциональная программа по сути является композицией функций, и в такой программе разработчик скорее описывает математическое (точнее, функциональное) отношение между исходными данными и результатом работы программы, чем конкретные шаги для получения результата (см. также начало раздела 4.5). Теоретической основой функционального программирования явилось лямбда-исчисление.

Лямбда-исчисление есть исчисление в определённом нами смысле (см. раздел 1.2). Оно ориентировано на изучение функций как правил вычисления значений функций по значениям аргументов. Здесь функция рассматривается с иной — вычислительной — точки зрения, нежели в рамках теории множеств, где функция — это множество пар, каждая из которых состоит из значения аргумента и значения функции.

В рамках лямбда-исчисления функция представляет собой определённым образом построенное слово, а аргументы функции — слова, которые являются функциями или переменными. Именно применение функции к аргументу (аппликация) является основным предметом исследования в теории, изучающей лямбда-исчисление. Аксиомы и правила вывода лямбда-исчисления задают то, как может вычисляться значение функции.

Прежде чем сформулировать лямбда-исчисление, приведём ещё два наводящих соображения, чтобы формулировка этого исчисления показалась более естественной.

Рассмотрим, например, одноместную функцию f, определённую на множестве всех натуральных чисел так, что $f(x) = x + 2$. Выбор обозначения f для этой функции произволен. Мы можем избежать такого выбора, используя вместо f выражение $\lambda x.(x + 2)$, которое естественным образом понимается как функция, сопоставляющая своему единственному аргументу x значение $x + 2$. При этом неважно, как назван аргумент функции: в том же качестве можно использовать выражение $\lambda y.(y + 2)$. Так же можно поступать с любой одноместной функцией f, значения которой задаются выражением $f(x)$: можно использовать безымянную функцию $\lambda x.f(x)$, полагая при этом, что значение этой функции при значении аргумента a есть $(\lambda x.f(x))(a) = f(a)$.

Далее, пусть имеется двухместная функция f, значения которой задаются выражением $f(x, y)$, зависящим от переменных x и y. Для каждого x можно ввести одноместную функцию f_x такую, что для любого y

$f_x(y) = f(x, y)$. В соответствии с принятой договорённостью, вместо f_x используем выражение $\lambda y.f(x, y)$. Тогда вместо f можно использовать одноместную функцию $\lambda x.f_x(y) = \lambda x.(\lambda y.f(x, y))$, которую удобно записать в сокращённом виде как $\lambda xy.f(x, y)$. Аналогично можно поступать и с функциями большей местности. Поэтому достаточно ограничиться рассмотрением одноместных функций. На этом мы закончим обсуждение наводящих соображений и перейдём к определениям.

Данные в текущем разделе 5.1.3 определения понятий, носящих названия или обозначения, которые встречаются в остальных разделах данной книги, имеют силу лишь в разделе 5.1.3.

Лямбда-термы и формулы

Функции в лямбда-исчислении представляются так называемыми лямбда-термами.

Переменной назовём любое слово языка $Vars$, который задан в определении 1.1.3. Будем обозначать произвольную переменную одной из букв f, v, w, x, y, z, возможно, с индексом.

Лямбда-терм (*ламбда-терм*, *λ-терм* или просто *терм*) задаётся следующим индуктивным определением:

1) любая переменная является термом;
2) если M — терм, x — переменная, то (λxM) является термом, про который говорят, что он получен из терма M с помощью *абстракции* (*лямбда-абстракции*);
3) если M и N — термы, то (MN) является термом, который называется *аппликацией* терма M к терму N.

При записи термов мы будем использовать следующие соглашения. Во-первых, будем иногда опускать внешние скобки в записи отдельно стоящего терма. Во-вторых, терм вида $(\lambda x_1(\lambda x_2 \dots (\lambda x_n M) \dots))$ будем иногда записывать как $\lambda x_1 x_2 \dots x_n.M$, а также, когда M имеет вид (M'), — как $\lambda x_1 x_2 \dots x_n.M'$. В-третьих, вместо терма вида $(((M_1 M_2)M_3) \dots M_n)$ будем иногда писать $M_1 M_2 \dots M_n$. Чтобы избежать неоднозначности, второе соглашение будем считать более приоритетным, чем третье, и потому при восстановлении опущенных в записи терма скобок после точки будем заключать в скобки максимальный по длине терм.

Пример 5.1.18. Ниже приведены записи термов и обозначаемые этими записями термы:

$$x, \quad xx \leftrightharpoons (xx), \quad yx \leftrightharpoons (yx), \quad \lambda x.yx \leftrightharpoons (\lambda x(yx)), \quad \lambda xy.yx \leftrightharpoons (\lambda x(\lambda y(yx))),$$
$$(\lambda xy.yx)\lambda z.z \leftrightharpoons ((\lambda x(\lambda y(yx)))(\lambda zz)), \quad \lambda xy.yx\lambda z.z \leftrightharpoons (\lambda x(\lambda y((yx)(\lambda zz)))).$$

$\triangleleft$

Понятия *свободного* и *связанного вхождения переменной* в лямбда-терм, *замкнутого* лямбда-терма определяются так же, как и для предикатной формулы (см. раздел 2.1.2), причём λ играет роль квантора. Например, если переменная x не совпадает ни с y, ни с z, то в лямбда-терме $(\lambda x.x\underline{y})(\lambda z.(\lambda y.\underline{x})z\underline{x})$, неподчёркнутые вхождения переменных являются связанными, а подчёркнутые — свободными. Любой замкнутый лямбда-терм называется *комбинатором*.

Аналогично тому, как это было сделано для логики предикатов (см. раздел 2.3.1), определяется $[M]^x_N$ — результат подстановки лямбда-терма N вместо всех свободных вхождений переменной x в лямбда-терм M, а также условие, при котором *лямбда-терм N свободен для переменной x в лямбда-терме M* (*подстановка терма N вместо x в M свободна*).

Определение подтерма лямбда-терма не в полной мере аналогично определению подтерма терма языка первого порядка. Мы не хотим считать x подтермом лямбда-терма $\lambda x.y$, где переменные x и y не совпадают. Дадим следующее индуктивное определение множества $Sub(M)$ всех подтермов лямбда-терма M:

1) если M есть переменная x, то $Sub(M) = \{x\}$,

2) если M имеет вид $\lambda x.N$, то $Sub(M) = Sub(N) \cup \{\lambda x.N\}$,

3) если M имеет вид N_1N_2, то $Sub(M) = Sub(N_1) \cup Sub(N_2) \cup \{N_1N_2\}$.

Лямбда-терм N назовём *подтермом* лямбда-терма M, если $N \in Sub(M)$.

Формулой называется любое слово вида $M = N$, где M и N — лямбда-термы.

Аксиомы и правила вывода лямбда-исчисления

Зададим исчисление, называемое *лямбда-исчислением* (*ламбда-исчислением* или *λ-исчислением*). Язык лямбда-исчисления составляют всевозможные формулы. Приведём аксиомы и правила вывода этого исчисления; в них L, M и N обозначают произвольные лямбда-термы, x и y — произвольные переменные.

Аксиомами лямбда-исчисления являются формулы, имеющие один из следующих видов:

1) $\lambda x.M = \lambda y.[M]^x_y$, где y не входит свободно в M (любая аксиома этого вида называется *α-конверсией*);

2) $(\lambda x.M)N = [M]^x_N$, где N свободен для x в M (любая аксиома этого вида называется *β-конверсией*);

3) $M = M$.

Правила вывода лямбда-исчисления таковы:

$$\frac{M = N}{N = M}, \qquad \frac{M = N}{LM = LN}, \qquad \frac{M = N}{ML = NL},$$

$$\frac{L = M; \quad M = N}{L = N}, \qquad \frac{M = N}{\lambda x.M = \lambda x.N}.$$

Выводимость формулы $M = N$ в лямбда-исчислении мы обозначаем через $\vdash M = N$. Если $\vdash L = M$ и $\vdash M = N$, то эти факты мы будем записывать в виде цепочки $L = M = N$. Аналогично будем понимать (в текущем разделе 5.1.3) и цепочки с бóльшим числом вхождений $=$.

Говорят, что лямбда-термы M и N *взаимно конвертируемы* (или что M *конвертируем* в N), если $\vdash M = N$. Очевидно, что взаимная конвертируемость есть отношение эквивалентности на множестве всех лямбда-термов.

Легко показать, что лямбда-исчисление позволяет заменить в терме некоторый подтерм на взаимно конвертируемый и получить в результате терм, конвертируемый в исходный. Представим этот факт следующей теоремой, которая является аналогом теоремы 2.6.37 о синтаксически эквивалентной замене.

Теорема 5.1.19. *Пусть M, N и N' — лямбда-термы, M' есть результат замены некоторого вхождения N в M на N', причём непосредственно перед этим вхождением нет вхождения символа λ. Тогда если $\vdash N = N'$, то $\vdash M = M'$.*

Упражнение 5.1.20. Докажите теорему 5.1.19. ◁

Среди аксиом и правил вывода лямбда-исчисления наиболее существенной является аксиома, называемая β-конверсией. Остальные аксиомы и правила вывода по сути служат для обеспечения применимости β-конверсии при построении цепочки взаимно конвертируемых термов (такая цепочка строится согласно теореме 5.1.19 аналогично тому, как выполняются равносильные преобразования предикатных формул). β-конверсия формализует то, как вычисляется значение безымянной функции. Эта аксиома гарантирует, что коллизии переменных не возникнет, поскольку подстановка должна быть свободной. Поэтому, чтобы воспользоваться β-конверсией иногда сначала требуется применить α-конверсию, которая позволяет переименовать в терме связанную переменную на новую переменную (сравните с понятием правильной подстановки, определённым в разделе 2.3.2).

Далее в этом разделе 5.1.3 мы будем считать различными переменные, которым даны различные обозначения.

Пример 5.1.21. Имеем следующую цепочку взаимно конвертируемых термов:

$$(\lambda x.(\lambda y.xy))(\lambda z.yz) = (\lambda x.(\lambda v.xv))(\lambda z.yz) = \lambda v.(\lambda z.yz)v = \lambda v.yv.$$

Таким образом, установлено, что в лямбда-исчислении выводима формула

$$(\lambda x.(\lambda y.xy))(\lambda z.yz) = \lambda v.yv. \qquad (*)$$

Построим следующие две цепочки взаимно конвертируемых термов:

$$(\lambda x.xy)(\lambda z.z) = (\lambda z.z)y = y,$$
$$(\lambda x.x)(\lambda x.x)y = (\lambda x.x)y = y.$$

Эти цепочки заканчиваются одним и тем же термом, так что одну из них можно перевернуть и соединить с другой с помощью знака $=$ (всё это можно сделать в силу аксиом и правил вывода лямбда-исчисления: вспомним, что взаимная конвертируемость термов есть отношение эквивалентности). Поэтому в лямбда-исчислении выводима формула

$$(\lambda x.xy)(\lambda z.z) = (\lambda x.x)(\lambda x.x)y. \qquad (**)$$

◁

Упражнение 5.1.22. Постройте вывод (в лямбда-исчислении) формул $(*)$ и $(**)$ из примера 5.1.21. ◁

Приняв за определение непротиворечивости исчисления то, что не все слова языка этого исчисления выводимы (сравните с леммой 3.1.6), можно доказать, что лямбда-исчисление непротиворечиво, т. е. существует невыводимая в этом исчислении формула.

Нормальная форма лямбда-терма

Любой терм вида $(\lambda x.M)N$ называется *редексом*[4]. Говорят, что терм *находится в нормальной форме*, если никакой подтерм этого терма не является редексом.

Терм N называется *нормальной формой терма* M, если N находится в нормальной форме и конвертируем в M. Если для терма M существует его нормальная форма, то говорят, что терм M *имеет нормальную форму*.

С интуитивной точки зрения вычисление терма, находящегося в нормальной форме, не может быть продолжено.

Пример 5.1.23. Терм $\lambda v.yv$ находится в нормальной форме.

Терм $(\lambda x.(\lambda y.xy))(\lambda z.yz)$ не находится в нормальной форме, но имеет нормальную форму $\lambda v.yv$ (см. пример 5.1.21), а также нормальные формы $\lambda w.yw$ и $\lambda x.yx$, получающиеся из первой нормальной формы путём переименования связанной переменной.

Как терм $(\lambda x.xy)(\lambda z.z)$, так и терм $(\lambda x.x)(\lambda x.x)y$ имеет нормальную форму y (см. пример 5.1.21).

Применив β-конверсию к терму $(\lambda x.xx)(\lambda x.xx)$, получим тот же самый терм. Неочевидно, как получить нормальную форму этого терма. Можно доказать, что он не имеет нормальной формы. ◁

Также можно установить, что нормальная форма терма единственна с точностью до α-конверсий, т. е. с точностью до переименования связанных переменных.

Лямбда-исчисление как язык программирования

Покажем, как средствами лямбда-исчисления можно представить натуральные числа и некоторые конструкции функциональных языков программирования.

Положим

$$\mathbf{T} \leftrightharpoons \lambda xy.x,$$
$$\mathbf{F} \leftrightharpoons \lambda xy.y.$$

Любой лямбда-терм, нормальная форма которого есть $\mathbf{T}$ или $\mathbf{F}$, назовём *булевским*. В частности, $\mathbf{T}$ и $\mathbf{F}$ являются булевскими.

Считая терм B булевским, а термы M и N — произвольными, условное выражение «если B, то M, иначе N» можно представить термом BMN. Очевидно, имеем

$$\vdash \mathbf{T}MN = M,$$
$$\vdash \mathbf{F}MN = N.$$

Считая термы X и Y булевскими, представим лямбда-термами дизъюнкцию X и Y и отрицание X. Желаемое действие дизъюнкции на X и Y задаётся условным выражением «если X, то $\mathbf{T}$, иначе Y», поэтому, положив

$$\mathbf{Or} \leftrightharpoons \lambda xy.x\mathbf{T}y,$$

[4]Слово «редекс» происходит от англоязычного словосочетания «*red*ucible *ex*pression».

будем иметь

$$\begin{gathered}\vdash \mathbf{Or}\,\mathbf{F}\,\mathbf{F} = \mathbf{F},\\ \vdash \mathbf{Or}\,\mathbf{F}\,\mathbf{T} = \mathbf{T},\\ \vdash \mathbf{Or}\,\mathbf{T}\,\mathbf{F} = \mathbf{T},\\ \vdash \mathbf{Or}\,\mathbf{T}\,\mathbf{T} = \mathbf{T}.\end{gathered}$$

Подобным образом пользуясь условным выражением, для отрицания введём терм

$$\mathbf{Not} \leftrightharpoons \lambda x.x\mathbf{F}\mathbf{T}$$

и будем иметь

$$\begin{gathered}\vdash \mathbf{Not}\,\mathbf{F} = \mathbf{T},\\ \vdash \mathbf{Not}\,\mathbf{T} = \mathbf{F}.\end{gathered}$$

Упражнение 5.1.24. Представьте конъюнкцию лямбда-термом. ◁

Для любых термов M и N положим

$$[M, N] \leftrightharpoons \lambda x.xMN,$$

где переменная x не входит свободно ни в M, ни в N. Тогда

$$\begin{gathered}\vdash [M, N]\mathbf{T} = M,\\ \vdash [M, N]\mathbf{F} = N,\end{gathered}$$

так что $[M, N]$ можно рассматривать как упорядоченную пару M и N.

Для каждого натурального числа n определим *нумерал* $\overline{n}$ таким образом:

$$\begin{aligned}\overline{0} &\leftrightharpoons \lambda x.x,\\ \overline{n+1} &\leftrightharpoons [\mathbf{F}, \overline{n}].\end{aligned}$$

Нетрудно видеть, что каждый нумерал находится в нормальной форме.

Введя обозначения для комбинаторов

$$\begin{aligned}\mathbf{Succ} &\leftrightharpoons \lambda x.[\mathbf{F}, x],\\ \mathbf{Pred} &\leftrightharpoons \lambda x.x\mathbf{F},\\ \mathbf{IsZero} &\leftrightharpoons \lambda x.x\mathbf{T},\end{aligned}$$

легко убеждаемся, что для любого натурального числа n

$$\begin{gathered}\vdash \mathbf{Succ}\,\overline{n} = \overline{n+1},\\ \vdash \mathbf{Pred}\,\overline{n+1} = \overline{n},\\ \vdash \mathbf{IsZero}\,\overline{0} = \mathbf{T},\\ \vdash \mathbf{IsZero}\,\overline{n+1} = \mathbf{F}.\end{gathered}$$

Таким образом, мы представили функции прибавления единицы к натуральному числу, вычитания единицы из натурального числа и предикат, проверяющий натуральное число на равенство нулю.

Представление рекурсивных функций. В языках программирования высокого уровня функции можно дать имя и вызвать её рекурсивно, использовав это имя. В лямбда-исчислении функции не имеют имён, поэтому следует предложить иной способ представления рекурсивных функций. Идея этого способа заключается в том, что рекурсивная функция имеет себя в качестве своего аргумента. Следующая теорема поможет нам справиться с задачей представления рекурсивных функций в лямбда-исчислении.

Теорема 5.1.25 (о неподвижной точке)**.** (a) *Для любого терма F существует терм X такой, что $\vdash X = FX$.*
(b) *Пусть*

$$\mathbf{Y} \leftrightharpoons \lambda f.(\lambda x.f(xx))(\lambda x.f(xx)). \qquad (\star)$$

Тогда для любого терма F

$$\vdash \mathbf{Y}F = F(\mathbf{Y}F).$$

Доказательство. (a) Положим $W \leftrightharpoons \lambda y.F(yy)$, где переменная y не входит свободно в F, и $X \leftrightharpoons WW$. Тогда

$$X = WW = (\lambda y.F(yy))W = F(WW) = FX.$$

(b) В (a) терм X определён так, что $\vdash X = \mathbf{Y}F$, и установлено $\vdash X = FX$; поэтому по теореме 5.1.19 $\vdash \mathbf{Y}F = F(\mathbf{Y}F)$. ◁

Пусть F — произвольный терм; тогда терм X такой, что $\vdash X = FX$, называется *неподвижной точкой терма F*. *Комбинатором неподвижной точки* называется какой угодно замкнутый терм Y, удовлетворяющий условию: $\vdash YF = F(YF)$ для любого терма F. Таким образом, терм $\mathbf{Y}$ (см. $(\star)$ в теореме 5.1.25) является комбинатором неподвижной точки.

Обратимся непосредственно к поставленной задаче представления рекурсивных функций в лямбда-исчислении. Пусть имеется рекурсивная функция с телом, представленным термом M, и в M входит переменная f, что означает рекурсивный вызов этой функции. Введём терм $F \leftrightharpoons \lambda f.M$, тем самым сделав f аргументом функции, представленной термом F. Теперь мы можем представить исходную функцию термом $\mathbf{Y}F$. Действительно, по теореме о неподвижной точке вычисление этого терма сводится к вычислению терма $F(\mathbf{Y}F)$, который подробнее записывается как $(\lambda f.M)(\mathbf{Y}F)$. При вычислении последнего терма в тело M вместо f подставляется терм $\mathbf{Y}F$, представляющий исходную функцию, что и обеспечивает рекурсивный вызов этой функции.

Рассмотрим применение теоремы о неподвижной точке для представления конкретной рекурсивной функции, при этом мы проведём рассуждение, аналогичное интуитивному рассуждению из предыдущего абзаца.

Пример 5.1.26. Найдём терм $\mathbf{Add}$ такой, что для любых натуральных чисел m и n выполняется

$$\vdash \mathbf{Add}\,\overline{m}\,\overline{n} = \overline{m+n}.$$

Ясно, что вычисление значения функции сложения $Add(x,\, y)$ натуральных чисел x и y может производиться согласно следующему (неформально описанному) алгоритму:

если $IsZero(y)$, то x, иначе $Succ(Add(x,\, Pred(y)))$,

где $IsZero$ — предикат, проверяющий свой аргумент на равенство нулю, $Succ$ и $Pred$ — функции прибавления и вычитания единицы соответственно. Поэтому искомый терм **Add** таков, что

$$\vdash \mathbf{Add}\, x\, y = (\mathbf{IsZero}\, y)\, x\, \mathbf{Succ}(\mathbf{Add}\, x\, (\mathbf{Pred}\, y)).$$

Очевидно, достаточно найти терм **Add** такой, что

$$\vdash \mathbf{Add} = \lambda xy.(\mathbf{IsZero}\, y)\, x\, \mathbf{Succ}(\mathbf{Add}\, x\, (\mathbf{Pred}\, y)),$$

и потому теперь достаточно найти терм **Add** такой, что

$$\vdash \mathbf{Add} = (\lambda fxy.(\mathbf{IsZero}\, y)\, x\, \mathbf{Succ}(f\, x\, (\mathbf{Pred}\, y)))\mathbf{Add}.$$

В силу теоремы о неподвижной точке в качестве **Add** подойдёт терм

$$\mathbf{Y}(\lambda fxy.(\mathbf{IsZero}\, y)\, x\, \mathbf{Succ}(f\, x\, (\mathbf{Pred}\, y))).$$

◁

Упражнение 5.1.27. Продемонстрируйте вычисление $\mathbf{Add}\,\overline{2}\,\overline{1}$, тем самым показав основные этапы вывода формулы $\mathbf{Add}\,\overline{2}\,\overline{1} = \overline{3}$, где **Add** — терм, определённый в предыдущем примере. ◁

Упражнение 5.1.28. Постройте термы **Mult**, **Exp** и **Tsub** такие, что для любых натуральных чисел m и n выполняется

$$\vdash \mathbf{Mult}\,\overline{m}\,\overline{n} = \overline{m \cdot n},$$
$$\vdash \mathbf{Exp}\,\overline{m}\,\overline{n} = \overline{m^n} \quad \text{(полагаем } 0^0 = 1\text{, как и в [7])},$$
$$\vdash \mathbf{Tsub}\,\overline{m}\,\overline{n} = \overline{m \dot{-} n}, \quad \text{где}$$
$$m \dot{-} n = \begin{cases} m - n, & \text{если } m \geqslant n, \\ 0, & \text{если } m < n. \end{cases}$$

◁

Упражнение 5.1.29. Постройте терм **Sub** такой, что для любых натуральных чисел m и n

$$\vdash \mathbf{Sub}\,\overline{m}\,\overline{n} = \overline{m - n},$$

если $m \geqslant n$, иначе $\mathbf{Sub}\,\overline{m}\,\overline{n}$ не имеет нормальной формы. ◁

Лямбда-определимые функции

Функция $f : \mathbb{N}^k \xrightarrow{p} \mathbb{N}$ называется *лямбда-определимой* (λ-*определимой*), если существует терм F такой, что для любых $n_1, \ldots, n_k \in \mathbb{N}$

$$\vdash F\overline{n}_1 \ldots \overline{n}_k = \overline{f(n_1, \ldots, n_k)}, \text{ если } f(n_1, \ldots, n_k) \text{ определена, и}$$

$F\overline{n}_1 \ldots \overline{n}_k$ не имеет нормальной формы, если $f(n_1, \ldots, n_k)$ не определена.

Теорема 5.1.30. *Пусть f — числовая функция. Тогда f лямбда-определима, если и только если f вычислима по Тьюрингу.*

Мы оставляем эту теорему без доказательства, поскольку оно громоздко. Однако представленные выше конструкции призваны убедить нас в том, что лямбда-исчисление является языком программирования, на котором, обладая достаточным запасом времени, можно запрограммировать любую интуитивно вычислимую числовую функцию.

5.1.4. Частично рекурсивные функции

В данном разделе мы опишем один класс[5] числовых функций, который совпадает с классом всех числовых вычислимых по Тьюрингу функций. Вводимые функции будут некоторыми очевидно вычислимыми по Тьюрингу функциями, а также будут получаться из последних с помощью некоторых операций над функциями, причём эти операции не выведут нас за пределы класса всех числовых вычислимых по Тьюрингу функций.

Следующие функции называются *простейшими*:

1) нуль-функция o, определённая как $o(x) = 0$ для любого $x \in \mathbb{N}$;

2) функция прибавления единицы s, определённая как $s(x) = x + 1$ для любого $x \in \mathbb{N}$;

3) для каждого $n \in \mathbb{N}_+$ и каждого $m = 1, \ldots, n$ функция проекции I_m^n, определённая как $I_m^n(x_1, \ldots, x_n) = x_m$ для любых $x_1, \ldots, x_n \in \mathbb{N}$.

Ниже в текущем разделе 5.1.4 переменные m, n будут использоваться для обозначения каких угодно чисел из $\mathbb{N}_+$, если иное явно не оговорено. Для любых (частичных) числовых n-местных функций f и g и любых $x_1, \ldots, x_n \in \mathbb{N}$ условимся записывать $f(x_1, \ldots, x_n) = g(x_1, \ldots, x_n)$, если обе части этого равенства не определены, или обе части этого равенства определены и совпадают.

Определим операции, с помощью которых будут получаться новые функции из уже имеющихся.

Пусть даны числовые функции: m-местная функция f и n-местные функции $f_1, \ldots, f_m$. Зададим n-местную функцию g так, что для любых $x_1, \ldots, x_n \in \mathbb{N}$

$$g(x_1, \ldots, x_n) = f(f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)),$$

если все $f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)$ определены, иначе $g(x_1, \ldots, x_n)$ не определено. Будем говорить, что g *получена суперпозицией* (*подстановкой*) *из* $f, f_1, \ldots, f_m$.

Пусть даны числовые функции: n-местная функция f и $(n+2)$-местная функция g. Зададим $(n+1)$-местную функцию h так, что для любых $x_1, \ldots, x_n \in \mathbb{N}$

$$h(x_1, \ldots, x_n, 0) = f(x_1, \ldots, x_n),$$

и для любых $x_1, \ldots, x_n, y \in \mathbb{N}$

$$h(x_1, \ldots, x_n, y + 1) = g(x_1, \ldots, x_n, y, h(x_1, \ldots, x_n, y)),$$

если $h(x_1, \ldots, x_n, y)$ определено, иначе $h(x_1, \ldots, x_n, y + 1)$ не определено. Будем говорить, что g *получена примитивной рекурсией из* f *и* g.

Распространим предыдущее определение на случай $n = 0$. Пусть a — натуральное число, g — числовая двухместная функция. Будем говорить, что одноместная функция h получена примитивной рекурсией из одноместной всюду определённой функции, тождественно равной a, и функции g, если

$$h(0) = a,$$

[5] Здесь и далее под классом понимается множество.

и для любого $y \in \mathbb{N}$

$$h(y+1) = g(y, h(y)),$$

если $h(y)$ определено, иначе $h(y+1)$ не определено.

Функция называется *примитивно рекурсивной*, если её можно получить конечным числом операций суперпозиции и примитивной рекурсии из простейших функций.

Упражнение 5.1.31. Задайте исчисление (с непустым множеством правил вывода), в котором будут выводимы все примитивно рекурсивные функции и только они. ◁

Легко видеть, что применение операций суперпозиции и примитивной рекурсии ко всюду определённым функциям даёт всюду определённые функции. Поэтому примитивно рекурсивные функции всюду определены.

Пример 5.1.32. Функция $o^n : \mathbb{N}^n \to \mathbb{N}$, тождественно равная 0, примитивно рекурсивна. Действительно, для любых $x_1, \ldots, x_n \in \mathbb{N}$ имеем

$$o^n(x_1, \ldots, x_n) = o(I^n_1(x_1, \ldots, x_n)).$$

Таким образом, o^n получена суперпозицией из o, I^n_1. ◁

Пример 5.1.33. Пусть a — произвольное натуральное число. Тогда функция $g^n : \mathbb{N}^n \to \mathbb{N}$, тождественно равная a, примитивно рекурсивна. Действительно, для любых $x_1, \ldots, x_n \in \mathbb{N}$ имеем

$$g^n(x_1, \ldots, x_n) = s(s(\ldots s(o^n(x_1, \ldots, x_n))\ldots)),$$

где в правой части равенства ровно a вхождений s. Таким образом, g^n получена (многократной) суперпозицией из простейшей функции s и примитивно рекурсивной функции o^n. ◁

Пример 5.1.34. Всюду определённая числовая функция $add(x, y) = x + y$ примитивно рекурсивна. Действительно, для любых $x, y \in \mathbb{N}$ имеем

$$add(x, 0) = x = I^1_1(x),$$
$$add(x, y+1) = add(x, y) + 1 = s(add(x, y)) = s(I^3_3(x, y, add(x, y))).$$

Таким образом, add получена примитивной рекурсией из I^1_1 и g, где g получена суперпозицией из s, I^3_3: $g(x, y, z) = s(I^3_3(x, y, z))$. ◁

Пример 5.1.35. Докажем, что функция $x \dot{-} y$ (см. её определение в упражнении 5.1.28 на с. 194) примитивно рекурсивна.

Сначала покажем, что $x \dot{-} 1$ примитивно рекурсивна. Для любого $x \in \mathbb{N}$ имеем

$$0 \dot{-} 1 = 0 = o(x),$$
$$(x+1) \dot{-} 1 = x = I^2_1(x, x \dot{-} 1).$$

Таким образом, $x \dot{-} 1$ получена примитивной рекурсией из o и I^2_1.

Далее, для любых $x, y \in \mathbb{N}$ имеем

$$x \dot{-} 0 = x = I^1_1(x),$$
$$x \dot{-} (y+1) = (x \dot{-} y) \dot{-} 1.$$

Поэтому $x \dot{-} y$ получается примитивной рекурсией из I^1_1 и $g(x, y, z) = z \dot{-} 1$, причём функция g, очевидно, примитивно рекурсивна. ◁

Пример 5.1.36. Функция $|x - y|$ (модуль разности x и y) примитивно рекурсивна, поскольку для любых $x, y \in \mathbb{N}$ имеем $|x - y| = (x \dot{-} y) + (y \dot{-} x)$, а выше мы установили, что функции $u + v$, $u \dot{-} v$ примитивно рекурсивны. $\triangleleft$

Пусть дана числовая $(n+1)$-местная функция f. Зададим n-местную функцию g так, что для любых $x_1, \ldots, x_n \in \mathbb{N}$ выполняется следующее: если существует $y \in \mathbb{N}$ такое, что $f(x_1, \ldots, x_n, y)$ определено и равно 0, и для любого $y' < y$ $f(x_1, \ldots, x_n, y')$ определено и не равно 0, то $g(x_1, \ldots, x_n) = y$; если такого y не существует, то $g(x_1, \ldots, x_n)$ не определено. Будем говорить, что g *получена минимизацией из* f и писать

$$g(x_1, \ldots, x_n) = \mu_y(f(x_1, \ldots, x_n, y) = 0).$$

Функция называется *частично рекурсивной*, если её можно получить конечным числом операций суперпозиции, примитивной рекурсии и минимизации из простейших функций.

Если частично рекурсивная функция всюду определена, то её называют *общерекурсивной* (или просто *рекурсивной*). Можно доказать, что существует общерекурсивная функция, не являющаяся примитивно рекурсивной.

Как показывает следующий пример, с помощью операции минимизации из всюду определённой функции может получиться не всюду определённая функция.

Пример 5.1.37. Рассмотрим (всюду определённую) примитивно рекурсивную функцию $f(x, y) = s(x) + y$. Легко видеть, что частично рекурсивная функция $g(x) = \mu_y(f(x, y) = 0)$ нигде не определена. $\triangleleft$

Пример 5.1.38. Докажем, что функция

$$sub(x, y) = \begin{cases} x - y, & \text{если } x \geqslant y, \\ \text{не определено}, & \text{если } x < y, \end{cases}$$

частично рекурсивна. Введём функцию $g(x, y, z) = |x - (y + z)|$, которая примитивно рекурсивна. Тогда $sub(x, y) = \mu_z(g(x, y, z) = 0)$. $\triangleleft$

Воспользовавшись каким-либо языком программирования, скажем, языком Pascal (и считая, что целочисленные переменные могут хранить любые целые числа), несложно запрограммировать вычисление каждой простейшей функции и каждой из трёх введённых операций над функциями. Так что можно запрограммировать вычисление любой частично рекурсивной функции. Машина Тьюринга для вычисления какой-либо частично рекурсивной функции, вообще говоря, будет громоздкой, но нет сомнений в том, что и такую машину можно описать. Можно доказать и обратное утверждение: любая вычислимая по Тьюрингу числовая функция является частично рекурсивной. Итак, можно доказать следующую теорему.

Теорема 5.1.39. *Пусть f — числовая функция. Тогда f частично рекурсивна, если и только если f вычислима по Тьюрингу.*

На определение частично рекурсивных функций можно смотреть как на декларативный язык программирования. С помощью этого языка можно устанавливать вычислимость (по Тьюрингу) функций, не прибегая к императивному программированию, каковым является построение машин Тьюринга.

Упражнение 5.1.40. Докажите, что следующие функции примитивно рекурсивны:

$$sg(x) = \begin{cases} 0, & \text{если } x = 0, \\ 1, & \text{если } x \neq 0; \end{cases}$$

$$mult(x, y) = x \cdot y;$$

$$h(x) = \begin{cases} f(x), & \text{если } b(x) = 0, \\ g(x), & \text{если } b(x) \neq 0, \end{cases}$$

где b, f, g — одноместные примитивно рекурсивные функции;

$$mod(x, y) = \begin{cases} \text{остаток от деления } x \text{ на } y, & \text{если } y \neq 0, \\ x, & \text{если } y = 0; \end{cases}$$

$$div(x, y) = \begin{cases} \text{частное от деления } x \text{ на } y, & \text{если } y \neq 0, \\ x, & \text{если } y = 0; \end{cases}$$

$$p(x) = \ x\text{-е простое число } (p(0) = 2,\ p(1) = 3,\ p(2) = 5,\ \ldots).$$

◁

Упражнение 5.1.41. Докажите, что следующая функция частично рекурсивна:

$$f(x, y) = \begin{cases} x/y, \text{ если } y \neq 0 \text{ и } y \text{ — делитель } x, \\ \text{не определено в противном случае.} \end{cases}$$

◁

Упражнение 5.1.42. Докажите, что функция, обратная к инъективной общерекурсивной функции, частично рекурсивна. ◁

5.1.5. Тезис Чёрча

Рассмотренные нами определения понятия вычислимости равносильны (см. теоремы 5.1.15, 5.1.30, 5.1.39). Каждое из этих понятий призвано уточнить интуитивное понятие вычислимости. В 1936 г. А. Чёрч впервые высказал следующее предложение, которое назвали *тезисом Чёрча*: *любая интуитивно вычислимая всюду определённая числовая функция является как лямбда-определимой, так и рекурсивной*. Затем тезис Чёрча был принят и для не всюду определённых (т. е. частичных) числовых функций. В силу теоремы 5.1.30 тезис Чёрча можно переформулировать так: *любая интуитивно вычислимая числовая функция является вычислимой по Тьюрингу*. Тезис Чёрча также принимают в следующей расширенной формулировке (не ограничиваясь числовыми функциями): *любое неформальное описание алгоритма может быть уточнено предъявлением некоторой машины Тьюринга*.

Тезис Чёрча — не теорема, поскольку в нём фигурирует математически неопределённое понятие интуитивно вычислимой функции (или, соответственно, понятие неформально описанного алгоритма). Однако то, что рассмотренные здесь и многие другие подходы к определению понятия вычислимости оказались эквивалентными, подкрепляет этот тезис. Кроме того, принципиально машина Тьюринга действует как человек-вычислитель, выполняющий арифметические операции и обладающий потенциально неограниченным запасом бумаги, чернил и времени. Лента соответствует бумаге,

состояние моделирует память вычислителя, а программа — правила выполнения арифметических операций, например, правило сложения чисел в столбик. Поэтому есть основания полагать, что всё, что может быть вычислено человеком по данному ему точному предписанию (алгоритму), может быть вычислено и подходящей машиной Тьюринга.

В дальнейшем изложении мы будем говорить о вычислимых функциях, считая их вычислимыми по Тьюрингу (если явно не сказано иное). Также мы будем давать неформальные описания алгоритмов, полагая, что при наличии времени можно построить соответствующие машины Тьюринга. В математике без тезиса Чёрча можно вообще обойтись, если использовать машины Тьюринга и вычислимые по Тьюрингу функции вместо неформальных описаний алгоритмов и интуитивно вычислимых функций. Принятие тезиса Чёрча позволяет не искать никакой алгоритм решения задачи, если доказано, что не существует машины Тьюринга, решающей эту задачу.

Замечание 5.1.43. То, что некоторые точные понятия вычислимости ограничиваются числовыми функциями (и не включают в себя словарных функций), по существу не умаляет общности этих понятий. Действительно, слова можно закодировать натуральными числами наподобие того, как любая информация в компьютере закодирована с помощью двоичных цифр. Мы также приведём один из способов кодировки слов натуральными числами в разделе 5.2.4. ◁

Доказательство вычислимости функции

Ответим на принципиальный вопрос: что является доказательством вычислимости функции? В классическом направлении в математике, которого мы придерживаемся в данной книге, такое доказательство при некоторых уточнениях является доказательством в теории множеств Цермело-Френкеля с аксиомой выбора. Такое (вообще говоря, неконструктивное) доказательство может и не предъявить алгоритма вычисления рассматриваемой функции.

Рассмотрим одну функцию и докажем её вычислимость. Так называемая гипотеза Гольдбаха утверждает, что любое натуральное чётное число, большее 2, есть сумма некоторых простых натуральных чисел. Определим функцию $f : \mathbb{N} \to \mathbb{N}$ так, что для любого $x \in \mathbb{N}$

$$f(x) = \begin{cases} 1, & \text{если гипотеза Гольдбаха верна,} \\ 0 & \text{в противном случае.} \end{cases}$$

В настоящее время неизвестно, верна ли гипотеза Гольдбаха. Однако f тождественно равна 0 или 1, функции f_0 и f_1 такие, что для любого $x \in \mathbb{N}$ $f_0(x) = 0$ и $f_1(x) = 1$, вычислимы, следовательно, f вычислима. Доказав вычислимость функции f, мы не предъявили алгоритма, вычисляющего эту функцию; мы можем лишь утверждать, что f вычисляется одним из двух алгоритмов: либо алгоритмом, на любом входе выдающим 1, либо алгоритмом, на любом входе выдающим 0. Заметим, что это доказательство вычислимости функции f опирается на закон исключённого третьего, который, вообще говоря, неприемлем в конструктивном доказательстве (если не указан явный способ выяснения верного члена дизъюнкции; см. замечание 2.6.23).

Доказательство существования алгоритма, вычисляющего функцию, без предъявления такого алгоритма по меньшей мере затрудняет практическое применение этой функции, не давая возможности запрограммировать вычисление значений этой функции. С этой точки зрения следует предпочитать конструктивные доказательства. Напомним (см. замечание 2.5.7), что в конструктивном доказательстве существования объекта даётся явный способ (точнее, алгоритм) построения такого объекта.

5.2. Разрешимые и перечислимые множества

В данном разделе понятие вычислимости, ранее определённое лишь для некоторых функций, будет распространено на некоторые множества.

5.2.1. Разрешимые и перечислимые множества натуральных чисел

Определение 5.2.1. *Характеристической функцией множества* $A \subseteq \mathbb{N}$ называется такая функция $\chi_A : \mathbb{N} \to \mathbb{N}$, что

$$\chi_A(x) = \begin{cases} 1, & \text{если } x \in A, \\ 0, & \text{если } x \notin A. \end{cases}$$

◁

Определение 5.2.2. Множество $A \subseteq \mathbb{N}$ называется *разрешимым* (*рекурсивным* или *вычислимым*), если функция χ_A вычислима. ◁

Другими словами, множество $A \subseteq \mathbb{N}$ разрешимо, если существует алгоритм, который для любого натурального числа x выясняет, $x \in A$ или нет.

Пример 5.2.3. Разрешимыми множествами являются $\varnothing$, $\mathbb{N}$, произвольное конечное множество, множество всех чётных чисел, множество всех простых чисел. В самом деле, для каждого из этих множеств легко описать алгоритм, проверяющий, принадлежит ли произвольное заданное число этому множеству или нет. ◁

Легко показать, что объединение, пересечение и разность двух разрешимых множеств являются разрешимыми множествами. Действительно, пусть множества A и B разрешимы. Это означает, что существуют машины Тьюринга M_A и M_B, вычисляющие функции χ_A и χ_B соответственно. Тогда следующая неформально описанная машина Тьюринга вычисляет характеристическую функцию $\chi_{A \setminus B}$ множества $A \setminus B$: на входе x исполнить вычисление $M_A(x)$, а затем вычисление $M_B(x)$; если результат вычисления $M_A(x)$ есть 1, и результат вычисления $M_B(x)$ есть 0, то выдать 1, иначе выдать 0. Похожее построение легко провести для объединения и пересечения разрешимых множеств.

Определение 5.2.4. *Частичной характеристической функцией множества* $A \subseteq \mathbb{N}$ называется такая функция $\chi_A^* : \mathbb{N} \xrightarrow{p} \mathbb{N}$, что

$$\chi_A^*(x) = \begin{cases} 1, & \text{если } x \in A, \\ \text{не определено}, & \text{если } x \notin A. \end{cases}$$

◁

Определение 5.2.5. Множество $A \subseteq \mathbb{N}$ называется *перечислимым* (*рекурсивно перечислимым*, *полуразрешимым* или *полувычислимым*), если функция χ_A^* вычислима. ◁

Другими словами, множество $A \subseteq \mathbb{N}$ перечислимо, если существует алгоритм, который для любого натурального числа x выдаёт 1, если $x \in A$, и не заканчивает работу в противном случае.

Очевидно, что любое разрешимое множество A перечислимо. В самом деле, если имеется машина Тьюринга M, вычисляющая функцию χ_A, то машина Тьюринга M', вычисляющая функцию χ_A^*, может быть устроена таким образом: на входе x исполнить вычисление $M(x)$, если результат работы $M(x)$ есть 1, то выдать 1, иначе продолжать работу бесконечно.

Пример перечислимого, но неразрешимого множества нам даст теорема 5.4.5.

Теорема 5.2.6. *Объединение и пересечение перечислимых множеств являются перечислимыми множествами.*

Доказательство. Пусть множества A и B перечислимы. Тогда существуют машины Тьюринга M_A и M_B, вычисляющие функции χ_A^* и χ_B^* соответственно.

Функция $\chi_{A\cup B}^*$ вычисляется следующей машиной Тьюринга: на входе x одновременно вычислять $M_A(x)$ и $M_B(x)$, исполняя команды этих двух машин поочерёдно, и как только $M_A(x)$ или $M_B(x)$ остановится, выдать 1.

Функция $\chi_{A\cap B}^*$ вычисляется следующей машиной Тьюринга: на входе x одновременно вычислять $M_A(x)$ и $M_B(x)$, исполняя команды этих двух машин поочерёдно, как только $M_A(x)$ или $M_B(x)$ остановится, продолжить исполнение команд ещё не остановившейся машины; когда и эта машина остановится, выдать 1. ◁

Как мы убедимся (см. теорему 5.4.5 и её следствие 5.4.6), существует перечислимое множество A такое, что $\mathbb{N} \setminus A$ неперечислимо.

Теорема 5.2.7 (теорема Пóста)**.** *Пусть $A \subseteq \mathbb{N}$. Тогда A разрешимо, если и только если A и $\mathbb{N} \setminus A$ перечислимы.*

Доказательство. Если A разрешимо, то разрешимо и $\mathbb{N} \setminus A$, следовательно, оба эти множества перечислимы.

Обратно, пусть A и $\mathbb{N} \setminus A$ перечислимы. Тогда существуют машины Тьюринга M' и M'', вычисляющие функции χ_A^* и $\chi_{\mathbb{N}\setminus A}^*$ соответственно. Построим машину Тьюринга M, которая на входе $x \in \mathbb{N}$ одновременно вычисляет $M'(x)$ и $M''(x)$, исполняя команды этих двух машин поочерёдно, причём как только M' остановится, M выдаёт результат 1, а как только M'' остановится, M выдаёт результат 0. Поскольку для любого $x \in \mathbb{N}$ ровно одна из машин M' или M'' на входе x должна остановиться, то машина M вычисляет функцию χ_A. Следовательно, A разрешимо. ◁

Теорема 5.2.8. *Пусть $A \subseteq \mathbb{N}$. Следующие утверждения равносильны:*

(a) *A перечислимо;*

(b) *$A = dom(\varphi)$ для некоторой вычислимой функции $\varphi : \mathbb{N} \xrightarrow{p} \mathbb{N}$;*

(c) $A = \varnothing$ *или* $A = rng(f)$ *для некоторой вычислимой функции* $f : \mathbb{N} \to \mathbb{N}$*;*

(d) $A = rng(\varphi)$ *для некоторой вычислимой функции* $\varphi : \mathbb{N} \xrightarrow{p} \mathbb{N}$.

Д о к а з а т е л ь с т в о. Установим равносильность (a) и (b). Если A перечислимо, то функция χ_A^* вычислима и $dom(\chi_A^*) = A$, так что функцию χ_A^* можно взять в качестве функции φ из (b).

Обратно, пусть $A = dom(\varphi)$ для некоторой вычислимой функции $\varphi : \mathbb{N} \xrightarrow{p} \mathbb{N}$. Тогда имеем машину Тьюринга M, вычисляющую функцию φ. Построим машину Тьюринга M', которая действует в соответствии со следующим неформальным описанием: на входе x вычисляет $M(x)$ и (если это вычисление завершается) выдаёт 1. Очевидно, машина M' вычисляет функцию χ_A^*, поэтому A перечислимо.

Установим равносильность (a) и (c). Пусть A перечислимо и непусто. Тогда выберем какой угодно элемент $j_0 \in A$. Определим функцию f как функцию, задаваемую следующим алгоритмом $\mathfrak{A}$ со входом $x \in \mathbb{N}$:

1. Алгоритм $\mathfrak{A}$ исполняет x шагов такого алгоритма $\mathfrak{B}$ (вход алгоритма $\mathfrak{B}$ не используется):

   для каждого $i = 0, 1, 2, \ldots$ исполнить $i + 1$ шаг в каждом из вычислений $\chi_A^*(0)$, $\chi_A^*(1)$, $\ldots$, $\chi_A^*(i)$.

2. Если на x-м шаге алгоритма $\mathfrak{B}$ завершилось вычисление $\chi_A^*(j)$ для некоторого j (такого j не может быть более одного), то алгоритм $\mathfrak{A}$ выдаёт j, иначе выдаёт j_0.

По построению алгоритма $\mathfrak{A}$, задающего функцию f, для любого $x \in \mathbb{N}$ имеем $f(x) \in dom(\chi_A^*) = A$, и для любого $j \in A$ найдётся $x \in \mathbb{N}$, при котором $f(x) = j$. Поэтому $rng(f) = A$ и функция f удовлетворяет (c).

Обратно, пусть имеет место (c). Если A пусто, то A перечислимо. Иначе $A = rng(f)$ для некоторой вычислимой функции $f : \mathbb{N} \to \mathbb{N}$. Тогда $\chi_A^*(x)$ вычисляется следующим неформально описанным алгоритмом: на входе x для каждого $i = 0, 1, 2, \ldots$ вычислить значение $f(i)$ и, если это значение равно числу x, выдать 1. Таким образом, A перечислимо.

Установим равносильность (c) и (d). Пусть имеет место (c). Если $A = \varnothing$, то A есть область значений одноместной числовой нигде не определённой функции, которая является вычислимой. Если $A \neq \varnothing$, то $A = rng(f)$ для некоторой вычислимой функции $f : \mathbb{N} \to \mathbb{N}$, которая годится на роль функции φ из (d). Итак, в обоих случаях имеет место (d).

Доказательство того, что (d) влечёт (c), аналогично вышеприведённому доказательству того, что (a) влечёт (c), лишь алгоритм, задающий искомую функцию f, выдаёт значения функции φ вместо аргументов функции χ_A^*. ◁

Пункт (c) теоремы 5.2.8 объясняет происхождение термина «(рекурсивно) перечислимое множество». Действительно, все элементы непустого перечислимого множества можно перечислить (возможно, с повторениями) как $f(0), f(1), f(2), \ldots$, где f — некоторая всюду определённая вычислимая функция (также называемая рекурсивной функцией).

Упражнение 5.2.9. Пусть A — разрешимое, B — перечислимое множества натуральных чисел, функция $f : \mathbb{N} \to \mathbb{N}$ вычислима. Что можно сказать о разрешимости и перечислимости образов и прообразов множеств A и B при отображении f? ◁

5.2.2. Разрешимые и перечислимые числовые отношения

Числовым отношением мы называем произвольное отношение на множестве $\mathbb{N}$. Перенесём понятия разрешимого и перечислимого множеств на числовые отношения. С этой целью мы сначала закодируем упорядоченные n-ки натуральных чисел натуральными числами.

Прежде всего закодируем упорядоченные пары натуральных чисел. Расположим все элементы множества $\mathbb{N}^2$ в виде бесконечной таблицы

$$\begin{array}{l}\langle 0,0\rangle,\ \langle 0,1\rangle,\ \langle 0,2\rangle,\ \langle 0,3\rangle,\ \ldots\\ \langle 1,0\rangle,\ \langle 1,1\rangle,\ \langle 1,2\rangle,\ \ldots\\ \langle 2,0\rangle,\ \langle 2,1\rangle,\ \ldots\\ \langle 3,0\rangle,\ \ldots\\ \ldots\end{array}$$

и занумеруем их по диагоналям этой таблицы, выстроив в такую последовательность:

$$\langle 0,0\rangle,\ \langle 0,1\rangle,\ \langle 1,0\rangle,\ \langle 0,2\rangle,\ \langle 1,1\rangle,\ \langle 2,0\rangle,\ \langle 0,3\rangle,\ \langle 1,2\rangle,\ \ldots\ . \qquad (*)$$

Номер пары $\langle x,y\rangle$ в этой последовательности (мы начинаем нумерацию с 0) обозначим через $\boldsymbol{c}(x,y)$ и назовём *канторовским номером* (или просто *номером*) этой пары. Функцию $\boldsymbol{c}:\mathbb{N}^2\to\mathbb{N}$, сопоставляющую каждой паре её канторовский номер, назовём *канторовской нумерующей функцией*. Легко видеть, что она биективна и вычислима. В дальнейшем изложении в качестве $\boldsymbol{c}$ может выступать любая вычислимая биекция из $\mathbb{N}^2$ в $\mathbb{N}$. Для определённости мы зафиксируем данную функцию $\boldsymbol{c}$ до конца текущей главы.

Получим алгебраическую формулу, задающую функцию $\boldsymbol{c}$. Пара $\langle x,y\rangle$ находится на $(x+y)$-й диагонали таблицы и является x-й на своей диагонали (мы начинаем отсчёт с 0). На предыдущих диагоналях находится $1+2+\ldots+(x+y)$ пар. Таким образом,

$$\boldsymbol{c}(x,y)=1+2+\ldots+(x+y)+x=(x+y)(x+y+1)/2+x.$$

Существуют вычислимые функции $\boldsymbol{l}$ и $\boldsymbol{r}$ из $\mathbb{N}$ в $\mathbb{N}$, которые любому натуральному числу z сопоставляют, соответственно, первый и второй член пары с номером z. Действительно, функция $\boldsymbol{l}$ вычисляется следующим алгоритмом: на входе z для каждого $i=0,1,2,\ldots$ породить i-й член $\langle x,y\rangle$ последовательности $(*)$ и, если $\boldsymbol{c}(x,y)=z$, выдать x. Функция $\boldsymbol{r}$ может быть вычислена аналогично.

Из определения функций $\boldsymbol{l}$ и $\boldsymbol{r}$ получаем, что для любых $x,y,z\in\mathbb{N}$

$$\boldsymbol{l}(\boldsymbol{c}(x,y))=x,\quad \boldsymbol{r}(\boldsymbol{c}(x,y))=y\quad\text{и}\quad \boldsymbol{c}(\boldsymbol{l}(z),\boldsymbol{r}(z))=z.$$

Функции $\boldsymbol{c}$, $\boldsymbol{l}$, $\boldsymbol{r}$ будем называть *канторовскими функциями*.

Упражнение 5.2.10. Найдите формулы для вычисления функций $\boldsymbol{l}$ и $\boldsymbol{r}$. ◁

Чтобы закодировать упорядоченные n-ки натуральных чисел, для каждого $n\in\mathbb{N}_+$ определим функцию $\boldsymbol{c}^{(n)}:\mathbb{N}^n\to\mathbb{N}$ так, что

$$\begin{aligned}\boldsymbol{c}^{(1)}(x_1)&=\ x_1,\\ \boldsymbol{c}^{(n+1)}(x_1,\ldots,x_n,x_{n+1})&=\ \boldsymbol{c}(\boldsymbol{c}^{(n)}(x_1,\ldots,x_n),\,x_{n+1}).\end{aligned}$$

Очевидно, функция $\boldsymbol{c}^{(n)}$ является вычислимой биекцией. Будем называть $\boldsymbol{c}^{(n)}(x_1, \ldots, x_n)$ (*канторовским*) *номером* n-ки $\langle x_1, \ldots, x_n \rangle$. Заметим, что $\boldsymbol{c}^{(2)} = \boldsymbol{c}$.

Для каждого $n \in \mathbb{N}_+$ и каждого $m = 1, \ldots, n$ существует вычислимая функция $\boldsymbol{c}_m^{(n)} : \mathbb{N} \to \mathbb{N}$, которая любому натуральному числу z сопоставляет m-й член n-ки с номером z. В частности, $\boldsymbol{c}_1^{(2)} = \boldsymbol{l}$ и $\boldsymbol{c}_2^{(2)} = \boldsymbol{r}$.

Упражнение 5.2.11. Выразите функцию $\boldsymbol{c}_m^{(n)}$ через $\boldsymbol{l}$ и $\boldsymbol{r}$. ◁

Числовое отношение $R \subseteq \mathbb{N}^n$ назовём *разрешимым* (соответственно, *перечислимым*), если $\boldsymbol{c}^{(n)}(R)$ (образ множества R при отображении $\boldsymbol{c}^{(n)}$) является разрешимым (соответственно, перечислимым) множеством.

Теперь мы можем легко перенести (объясните, как) некоторые результаты, установленные ранее для множеств натуральных чисел, на числовые отношения. Объединение и пересечение разрешимых (соответственно, перечислимых) числовых отношений разрешимы (соответственно, перечислимы). Разность разрешимых числовых отношений разрешима. Также имеет место аналог теоремы Поста: числовое отношение $R \subseteq \mathbb{N}^n$ разрешимо, если и только если R и $\mathbb{N}^n \setminus R$ перечислимы.

Для числового отношения $R \subseteq \mathbb{N}^n$ его *характеристическая функция* χ_R и *частичная характеристическая функция* χ_R^* определяются аналогично тому, как это сделано в определениях 5.2.1 и 5.2.4, только теперь аргументами этих функций являются упорядоченные n-ки натуральных чисел.

Упражнение 5.2.12. Пусть $R \subseteq \mathbb{N}^n$. Найдите соотношения, связывающие функции χ_R и $\chi_{\boldsymbol{c}^{(n)}(R)}$. ◁

Упражнение 5.2.13. Докажите, что множество $R \subseteq \mathbb{N}^n$ разрешимо (соответственно, перечислимо) тогда и только тогда, когда χ_R (соответственно, χ_R^*) вычислима. ◁

Упражнение 5.2.14. Верно ли, что n-местная числовая всюду определённая функция вычислима тогда и только тогда, когда она, рассматриваемая как $(n+1)$-местное отношение, разрешима? ◁

Упражнение 5.2.15. Докажите, что n-местная числовая функция вычислима тогда и только тогда, когда она, рассматриваемая как $(n+1)$-местное отношение, перечислима. ◁

Любой предикат Π с областью определения $\mathbb{N}^n$ назовём *числовым предикатом*. Такой предикат Π назовём *разрешимым* (соответственно, *перечислимым*), если разрешимо (соответственно, перечислимо) отношение на $\mathbb{N}$, задаваемое этим предикатом (см. замечание 2.2.2).

Характеристической функцией χ_Π (соответственно, *частичной характеристической функцией* χ_Π^*) числового предиката Π будем называть характеристическую функцию (соответственно, частичную характеристическую функцию) отношения, задаваемого этим предикатом.

5.2.3. Теоремы о проекции

Изложим ещё один (в дополнение к теореме 5.2.8) вариант описания перечислимых множеств: такие множества можно охарактеризовать как проекции разрешимых отношений.

Определение 5.2.16. Пусть X — произвольное множество, R — n-местное отношение на X, $n \geqslant 2$. Тогда множество

$$\{\langle x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n\rangle \in X^{n-1} \mid \exists x_j\, (\langle x_1, \ldots, x_{j-1}, x_j, x_{j+1}, \ldots, x_n\rangle \in R)\}$$

называется *проекцией отношения R вдоль j-й координаты.* ◁

Теорема 5.2.17. *Если отношение $R \subseteq \mathbb{N}^n$ перечислимо, то существует разрешимое отношение $\widetilde{R} \subseteq \mathbb{N}^{n+1}$ такое, что R является проекцией отношения $\widetilde{R}$ вдоль $(n+1)$-й координаты.*

Доказательство. Так как R перечислимо, то перечислимо и множество $R' \leftrightharpoons \boldsymbol{c}^{(n)}(R)$. Следовательно, функция $\chi^*_{R'}$ вычисляется некоторой машиной Тьюринга M. Для любого $\langle x_1, \ldots, x_n\rangle \in \mathbb{N}^n$ имеем: $\langle x_1, \ldots, x_n\rangle \in R$, если и только если $\boldsymbol{c}^{(n)}(x_1, \ldots, x_n) \in R'$, что в свою очередь равносильно $!M(\boldsymbol{c}^{(n)}(x_1, \ldots, x_n))$.

Определим отношение $\widetilde{R} \subseteq \mathbb{N}^{n+1}$ так, что $\langle x_1, \ldots, x_n, x_{n+1}\rangle \in \widetilde{R}$, если и только если M на входе $\boldsymbol{c}^{(n)}(x_1, \ldots, x_n)$ останавливается менее чем за x_{n+1} шагов. Ясно, что отношение $\widetilde{R}$ разрешимо, и для $!M(\boldsymbol{c}^{(n)}(x_1, \ldots, x_n))$ необходимо и достаточно существование такого натурального числа x_{n+1}, что $\langle x_1, \ldots, x_n, x_{n+1}\rangle \in \widetilde{R}$. Поэтому R является проекцией отношения $\widetilde{R}$ вдоль $(n+1)$-й координаты. ◁

Теорема 5.2.18. *Если отношение $R \subseteq \mathbb{N}^n$ перечислимо, $n \geqslant 2$, то любая проекция отношения R перечислима.*

Доказательство. Ограничимся рассмотрением проекции Q перечислимого отношения R вдоль n-й координаты, оставив рассмотрение проекции вдоль произвольной координаты в качестве упражнения.

Используем определение перечислимого множества из пункта (d) теоремы 5.2.8. Имеем $\boldsymbol{c}^{(n)}(R) = rng(\varphi)$ для некоторой вычислимой функции $\varphi : \mathbb{N} \xrightarrow{p} \mathbb{N}$. Тогда $\boldsymbol{c}^{(n-1)}(Q) = \boldsymbol{l}(\boldsymbol{c}^{(n)}(R)) = rng(\boldsymbol{l} \circ \varphi)$, композиция $\boldsymbol{l} \circ \varphi$ вычислимых функций φ и $\boldsymbol{l}$ вычислима, следовательно, $\boldsymbol{c}^{(n-1)}(Q)$ и Q перечислимы. ◁

Упражнение 5.2.19. Дополните доказательство предыдущей теоремы рассмотрением проекции вдоль произвольной координаты. ◁

Из двух предыдущих теорем получаем

Следствие 5.2.20. *Числовое отношение перечислимо, если и только если оно является проекцией некоторого разрешимого числового отношения.*

Эти две теоремы и их следствие называют *теоремами о проекции.*

5.2.4. Разрешимые и перечислимые языки и словарные отношения

Пусть дан произвольный алфавит Σ. Распространим понятия разрешимого и перечислимого множеств на языки в алфавите Σ.

Сначала закодируем все слова языка Σ^* натуральными числами. Номер пустого слова положим равным 0. Далее, пусть в алфавите Σ ровно k символов ($k \geqslant 1$). Занумеруем все символы алфавита Σ числами $1, 2, \ldots, k$ и обозначим символ этого алфавита с номером i через a_i. Номер непустого слова $\alpha \leftrightharpoons a_{i_j} \ldots a_{i_1} a_{i_0}$ положим равным числу $\tau(\alpha) = i_0 + i_1 \cdot k + \ldots + i_j \cdot k^j$.

При фиксированном $k \in \mathbb{N}_+$ любое число $n \in \mathbb{N}_+$ представляется ровно одним способом в виде $n = i_0 + i_1 \cdot k + \ldots + i_j \cdot k^j$, где $i_l \in \{1, 2, \ldots, k\}$ для каждого $l = 0, 1, \ldots, j$. Таким образом, функция $\tau : \Sigma^* \to \mathbb{N}$, сопоставляющая каждому слову его номер, является биекцией. Также функция τ, очевидно, вычислима. Как и раньше, для нас существенно лишь то, что кодирующая функция τ является вычислимой биекцией.

Язык $L \subseteq \Sigma^*$ будем называть *разрешимым* (соответственно, *перечислимым*), если $\tau(L)$ является разрешимым (соответственно, перечислимым) множеством.

Словарным отношением назовём любое отношение $R \subseteq \Sigma_1^* \times \ldots \times \Sigma_n^*$, где $\Sigma_1, \ldots, \Sigma_n$ — какие угодно алфавиты, $n \geqslant 1$.

Пусть даны алфавиты $\Sigma_1, \ldots, \Sigma_n \subseteq \Sigma$, $n \geqslant 1$. Словарное отношение $R \subseteq \Sigma_1^* \times \ldots \times \Sigma_n^*$ будем называть *разрешимым* (соответственно, *перечислимым*), если $\{\langle \tau(\alpha_1), \ldots, \tau(\alpha_n) \rangle \in \mathbb{N}^n \mid \langle \alpha_1, \ldots, \alpha_n \rangle \in R\}$ является разрешимым (соответственно, перечислимым) отношением на $\mathbb{N}$.

Как и в разделе 5.2.2, мы можем перенести некоторые ранее установленные результаты о разрешимости и перечислимости на словарные отношения. Кроме того, для словарных отношений имеют место аналоги теорем о проекции.

Упражнение 5.2.21. Осуществите перенос ранее установленных результатов о разрешимости и перечислимости, в том числе и теорем о проекции, на словарные отношения. ◁

Для словарного отношения R его *характеристическая функция* χ_R и *частичная характеристическая функция* χ_R^* определяются аналогично тому, как это сделано в определениях 5.2.1 и 5.2.4, только теперь аргументами этих функций являются упорядоченные n-ки слов.

Упражнение 5.2.22. Докажите, что словарное отношение R разрешимо (соответственно, перечислимо) тогда и только тогда, когда χ_R (соответственно, χ_R^*) вычислима. ◁

Любой предикат Π с областью определения $\Sigma_1^* \times \ldots \times \Sigma_n^*$ назовём *словарным предикатом*. Такой предикат Π назовём *разрешимым* (соответственно, *перечислимым*), если разрешимо (соответственно, перечислимо) отношение, задаваемое этим предикатом (см. замечание 2.2.2).

Характеристической функцией χ_Π (соответственно, *частичной характеристической функцией* χ_Π^*) словарного предиката Π будем называть характеристическую функцию (соответственно, частичную характеристическую функцию) отношения, задаваемого этим предикатом.

5.3. Нумерация вычислимых функций

Для программиста привычно то, что исходный текст программы на языке программирования в компьютере представляется конечной последовательностью двоичных цифр, так же как и результат компиляции про-

граммы. Подобным образом каждую программу можно рассматривать как некоторое (возможно, длинное) натуральное число. Такое число может подаваться на вход программы-интерпретатора, которая исполняет программу, представленную этим числом.

В данном разделе мы закодируем числовые машины Тьюринга натуральными числами, иначе говоря, занумеруем числовые машины Тьюринга. С помощью этой нумерации мы занумеруем и числовые вычислимые функции. Затем мы введём в рассмотрение так называемую универсальную машину Тьюринга, которая является интерпретатором числовых машин Тьюринга. Универсальная машина Тьюринга послужит нам для дальнейшего развития теории вычислимости.

Далее в текущей главе под *функцией* и под *машиной Тьюринга* мы будем понимать числовую функцию и числовую машину Тьюринга соответственно, если не указано иначе.

5.3.1. Нумерация машин Тьюринга и вычислимых функций

Нумерацией множества Ξ называется любая сюръективная функция $f : \mathbb{N} \to \Xi$. Если для некоторых $\xi \in \Xi$ и $x \in \mathbb{N}$ имеет место $f(x) = \xi$, то x называют *номером* элемента ξ (в нумерации f). Заметим, что некоторые элементы множества Ξ могут иметь более одного номера.

Для каждого $n \in \mathbb{N}_+$ определим нумерацию всех n-местных (числовых) вычислимых функций. Сначала закодируем натуральными числами (числовые) машины Тьюринга. Здесь возникает трудность, связанная с тем, что машина Тьюринга может иметь любой алфавит в качестве внутреннего алфавита и любой алфавит, содержащий символы _, # и |, в качестве ленточного алфавита. Однако ясно, что для машины Тьюринга все символы, кроме трёх указанных, носят чисто технический характер и без изменения поведения машины Тьюринга могут быть заменены на любые другие попарно различные символы. Поэтому, не умаляя общности, мы будем считать, что все рассматриваемые нами числовые машины Тьюринга имеют ленточный и внутренний алфавиты, содержащиеся в некотором заранее выбранном счётном множестве символов $\{b_0, b_1, \ldots\}$.

Тогда любую машину Тьюринга можно однозначно представить словом в некотором алфавите Σ_0, используя вместо символа b_i десятичную запись числа i, при этом потребуется лишь несколько дополнительных символов-разделителей. Легко обеспечить, чтобы это представление было таким, что существовал бы алгоритм, который бы по любому слову в алфавите Σ_0 проверял, представляет ли оно некоторую машину Тьюринга или нет. Детализация такого представления остаётся в качестве упражнения. (На самом деле машину Тьюринга можно было определить не в теоретико-множественных терминах, как мы это сделали в определении 5.1.1, а как специальным образом построенное слово, хотя такое определение было бы более громоздким.) Обозначим через $\mathcal{M}$ множество всех слов, представляющих машины Тьюринга ($\mathcal{M} \subseteq \Sigma_0^*$), и будем отождествлять каждый элемент этого множества с машиной Тьюринга, представленной этим элементом.

Неформально опишем алгоритм, который на произвольном входном слове $M \in \Sigma_0^*$ не завершает свою работу, если M не представляет ника-

кую машину Тьюринга, а в противном случае выдаёт натуральное число[6]. Установим счётчик x в 0. Будем последовательно порождать все слова в алфавите Σ_0. Как только очередное слово α порождено, проверим, совпадает ли α с M, и если совпадает, то завершаем работу, выдавая в качестве ответа текущее значение x; иначе проверим, представляет ли α машину Тьюринга, и если представляет, то увеличим x на 1; затем переходим к порождению следующего слова. (Этот алгоритм легко модифицировать так, чтобы он останавливался на любом входе, но здесь это необязательно.)

Нет сомнений в том, что можно описать словарную машину Тьюринга, реализующую этот алгоритм. Он задаёт некоторую частичную функцию $\gamma_0 : \Sigma_0^* \xrightarrow{p} \mathbb{N}$ с областью определения $\mathcal{M}$. Тогда положим, что функция $\gamma : \mathcal{M} \to \mathbb{N}$ совпадает с γ_0 на множестве $\mathcal{M}$. По построению этого алгоритма функция γ является биекцией и сопоставляет каждой машине Тьюринга M натуральное число $\gamma(M)$.

Для вычисления функции $\gamma^{-1} : \mathbb{N} \to \mathcal{M}$, обратной к γ, имеется следующий алгоритм. Этот алгоритм по любому входному натуральному числу x выдаёт такую машину Тьюринга M, что $\gamma(M) = x$. Точно так же, как и в предыдущем алгоритме, будем порождать машины Тьюринга до тех пор, пока не будет порождена x-ая по порядку машина Тьюринга, её и выдадим в качестве ответа.

Итак, последний описанный алгоритм вычисляет функцию γ^{-1}, которая является нумерацией множества $\mathcal{M}$. Отметим, что эта нумерация есть вычислимая биекция из $\mathbb{N}$ в $\mathcal{M}$. Для дальнейшего изложения подойдёт любая нумерация множества $\mathcal{M}$, являющаяся вычислимой биекцией (это условие можно ослабить, но мы не будем на этом останавливаться). Зафиксируем данную нумерацию γ^{-1} до конца текущей главы.

Для каждой машины Тьюринга M $\gamma(M)$ мы назовём *гёделевым номером* (*номером* или *индексом*) этой машины Тьюринга. Для каждого $x \in \mathbb{N}$ будем обозначать через M_x машину Тьюринга, гёделев номер которой есть x.

Через $\varphi_x^{(n)}$ будем обозначать n-местную (числовую) функцию, задаваемую машиной Тьюринга M_x. Число x назовём *гёделевым номером* (*номером* или *индексом*) функции $\varphi_x^{(n)}$. Таким образом, для каждого $n \in \mathbb{N}_+$ мы имеем нумерацию множества всех числовых вычислимых n-местных функций: $x \mapsto M_x \mapsto \varphi_x^{(n)}$. Поскольку одну и ту же функцию вычисляют различные машины Тьюринга, то функция имеет не единственный гёделев номер. Вместо $\varphi_x^{(1)}$ мы будем часто писать φ_x.

Упражнение 5.3.1. Докажите, что любая вычислимая функция имеет бесконечное число гёделевых номеров. ◁

Поскольку множество всех n-местных числовых вычислимых функций счётно, а множество всех n-местных числовых функций несчётно, то для любого $n \in \mathbb{N}_+$ существуют n-местные невычислимые функции. Важные примеры невычислимых функций мы рассмотрим позже. Здесь же мы приведём доказательство утверждения о существовании одноместной невычислимой функции, продемонстрировав метод доказательства, называемый диагональным и нередко используемый в теории вычислимости. Этот метод

[6] Оно будет номером машины Тьюринга M в некоторой нумерации, которую мы скоро определим.

был применён Кантором для доказательства несчётности множества всех вещественных чисел.

Докажем, что существует одноместная числовая невычислимая функция, определив такую функцию f. Положим, что для любого $x \in \mathbb{N}$

$$f(x) = \begin{cases} \varphi_x(x) + 1, & \text{если } \varphi_x(x) \text{ определено,} \\ 0, & \text{если } \varphi_x(x) \text{ не определено.} \end{cases}$$

$f(x)$ отлично от $\varphi_x(x)$ при каждом x, следовательно, функция f отличается от каждой одноместной числовой вычислимой функции, и потому f не является вычислимой.

Поясним происхождение названия «диагональный метод». Поместим значения функций φ_x в строки бесконечной таблицы (если $\varphi_x(y)$ не определено, то в этой таблице вместо значения $\varphi_x(y)$ стоит «не определено»):

$$\begin{array}{l} \underline{\varphi_0(0)},\ \varphi_0(1),\ \varphi_0(2),\ \ldots \\ \varphi_1(0),\ \underline{\varphi_1(1)},\ \varphi_1(2),\ \ldots \\ \varphi_2(0),\ \varphi_2(1),\ \underline{\varphi_2(2)},\ \ldots \\ \ldots \end{array}$$

Для каждого x мы определили $f(x)$ так, чтобы $f(x)$ отличалось от $\varphi_x(x)$, находящегося на *диагонали* этой таблицы.

5.3.2. Универсальные машины Тьюринга и универсальные функции

Неформально опишем машину Тьюринга U, интерпретирующую любую (числовую) машину Тьюринга, т. е. исполняющую те же действия, что и данная машина Тьюринга.

Машине U на вход подаётся номер x произвольной машины Тьюринга и произвольное число любых входных данных $y_1, \ldots, y_n$ для этой машины ($n \geqslant 1$). По номеру x машина U находит машину Тьюринга M_x; затем вычисляет $M_x(y_1, \ldots, y_n)$, моделируя действия машины M_x на входе $y_1, \ldots, y_n$; когда моделирование действий машины M_x завершится (это может и не произойти), выдаёт результат работы $M_x(y_1, \ldots, y_n)$.

Итак, можно построить машину Тьюринга U, обладающую следующим свойством:

$$\text{для любого } n \in \mathbb{N}_+ \text{ и для любых } x, y_1, \ldots, y_n \in \mathbb{N}$$
$$U(x, y_1, \ldots, y_n) = M_x(y_1, \ldots, y_n).$$

(Здесь и в дальнейшем знак $=$ между двумя результатами работы машин Тьюринга означает, что эти результаты определены или не определены одновременно, и в случае определённости эти результаты равны.) Такую машину Тьюринга U назовём *универсальной* машиной Тьюринга (или *интерпретатором*).

Примем соглашение о так называемых лямбда-обозначениях функций. Лямбда-обозначения, описанные в начале раздела 5.1.3, удобно использовать не только в связи с лямбда-исчислением. Часто мы будем фиксировать значения нескольких аргументов функции и рассматривать её как функцию

от оставшихся аргументов. Например, при заданных значениях u_0, v_0 первых двух аргументов четырёхместной функции g мы получаем двухместную функцию $\lambda xy.g(u_0, v_0, x, y)$. Саму функцию g можно записать также как $\lambda uvxy.g(u, v, x, y)$. Аналогичным образом мы иногда будем явно указывать аргументы какой угодно функции.

При фиксированном $n \in \mathbb{N}_+$ универсальная машина Тьюринга U задаёт $(n+1)$-местную функцию $\psi_U^{(n)}$, обладающую свойством:

$$\text{для любого } x \in \mathbb{N} \quad \lambda y_1 \dots y_n.\psi_U^{(n)}(x, y_1, \dots, y_n) = \varphi_x^{(n)}. \qquad (*)$$

Дадим определение, характеризующее такие функции, как $\psi_U^{(n)}$.

Определение 5.3.2. Пусть $\mathcal{C}$ — произвольный класс числовых n-местных функций. $(n+1)$-местная функция ψ называется *универсальной для класса* $\mathcal{C}$, если выполняются два условия:

1) для любого $x \in \mathbb{N}$ функция $\lambda y_1 \dots y_n.\psi(x, y_1, \dots, y_n)$ принадлежит $\mathcal{C}$;

2) для всякой функции φ из $\mathcal{C}$ найдётся $x \in \mathbb{N}$ такое, что $\lambda y_1 \dots y_n.\psi(x, y_1, \dots, y_n) = \varphi$. ◁

Таким образом, из $(*)$ следует, что функция $\psi_U^{(n)}$ универсальна для класса всех (числовых) вычислимых n-местных функций. Оформим полученные результаты в виде следующей теоремы.

Теорема 5.3.3. *Существует универсальная машина Тьюринга. Для каждого $n \in \mathbb{N}_+$ существует функция, которая вычислима и универсальна для класса всех вычислимых n-местных функций.*

В дальнейшем мы будем использовать сокращение $\psi_U \rightleftharpoons \psi_U^{(1)}$.

Функцию ψ' называют *продолжением* функции ψ, если $\psi \subseteq \psi'$, где обе функции рассматриваются как бинарные отношения.

Мы нашли (частичную) вычислимую функцию, универсальную для класса всех вычислимых n-местных функций. Ясно, что любую функцию можно продолжить до всюду определённой. Естественно поставить вопрос: можно ли продолжить функцию, универсальную для класса всех вычислимых n-местных функций, до всюду определённой *вычислимой* функции? Отрицательный ответ на этот вопрос даёт следующая теорема.

Теорема 5.3.4. *Никакая функция ψ, универсальная для класса всех вычислимых n-местных функций, не имеет всюду определённого вычислимого продолжения.*

Доказательство. Предположим, что нашлись функция ψ, универсальная для класса всех вычислимых n-местных функций, и её вычислимое всюду определённое продолжение ψ'. Тогда функция $\lambda y_1 \dots y_n.(\psi'(y_1, y_1, y_2, \dots, y_n)+1)$ всюду определена и вычислима. Найдётся такое $x_0 \in \mathbb{N}$, что

$$\lambda y_1 \dots y_n.\psi(x_0, y_1, \dots, y_n) = \lambda y_1 \dots y_n.(\psi'(y_1, y_1, y_2, \dots, y_n) + 1).$$

Отсюда следует, что $\lambda y_1 \dots y_n.\psi(x_0, y_1, \dots, y_n)$ всюду определена. Положив $y_1, \dots, y_n$ равными x_0, получим равенство $\psi(x_0, \dots, x_0) = \psi'(x_0, \dots, x_0) + 1$,

которое в силу того, что ψ' является продолжением ψ, перепишем как $\psi'(x_0,\ldots,x_0) = \psi'(x_0,\ldots,x_0)+1$. Получили противоречие, поэтому сделанное предположение неверно. ◁

Упражнение 5.3.5. Докажите, что ни для какого $n \in \mathbb{N}_+$ не существует функции, которая вычислима и универсальна для класса всех (числовых) вычислимых всюду определённых n-местных функций. ◁

5.3.3. Теорема о параметризации

Пусть имеется двухместная вычислимая функция f, иначе записываемая как $\lambda xy.f(x,y)$. Тогда для каждого натурального числа a одноместная функция $\lambda y.f(a,y)$ вычислима. Действительно, чтобы для какого угодно значения b аргумента y вычислить значение функции $\lambda y.f(a,y)$, достаточно вычислить $f(a,b)$. Более того, можно написать алгоритм, который по алгоритму вычисления любой двухместной функции f и любому натуральному числу a, выдаёт алгоритм вычисления функции $\lambda y.f(a,y)$. В терминах вычислимых функций это означает, что существует вычислимая всюду определённая функция, которая по номеру двухместной функции f и натуральному числу a даёт номер функции $\lambda y.f(a,y)$. Обобщим этот факт следующей теоремой.

Теорема 5.3.6 (теорема о параметризации, или s-m-n-теорема)**.** *Для любых $m,n \in \mathbb{N}_+$ существует вычислимая функция $s_n^m : \mathbb{N}^{m+1} \to \mathbb{N}$ такая, что для любых $x,y_1,\ldots,y_m \in \mathbb{N}$ имеет место*

$$\lambda z_1\ldots z_n.\varphi_x^{(m+n)}(y_1,\ldots,y_m,z_1,\ldots,z_n) = \varphi^{(n)}_{s_n^m(x,y_1,\ldots,y_m)}.$$

Доказательство. Неформально опишем алгоритм, который вычисляет искомую функцию s_n^m. На входе $x,y_1,\ldots,y_n$ этот алгоритм выполняет следующие действия:

1) построить машину Тьюринга M', которая на входе $z_1,\ldots,z_n$ вычисляет $\varphi_x^{(m+n)}(y_1,\ldots,y_m,z_1,\ldots,z_n)$, моделируя действия машины M_x на входе $y_1,\ldots,y_m,z_1,\ldots,z_n$;
2) найти и выдать номер машины M'.

Действительно, результатом работы этого алгоритма на входе $x,y_1,\ldots,y_n$ является номер функции $\lambda z_1\ldots z_n.\varphi_x^{(m+n)}(y_1,\ldots,y_m,z_1,\ldots,z_n)$. ◁

Следствие 5.3.7. *Для любых $m,n \in \mathbb{N}_+$ и любой вычислимой функции $g : \mathbb{N}^{m+n} \xrightarrow{p} \mathbb{N}$ существует вычислимая функция $s : \mathbb{N}^m \to \mathbb{N}$ такая, что для любых $y_1,\ldots,y_m \in \mathbb{N}$ имеет место*

$$\lambda z_1\ldots z_n.g(y_1,\ldots,y_m,z_1,\ldots,z_n) = \varphi^{(n)}_{s(y_1,\ldots,y_m)}.$$

Доказательство. Для некоторого $x_0 \in \mathbb{N}$ $g = \varphi_{x_0}^{(m+n)}$. По предыдущей теореме в качестве s можно взять $\lambda y_1\ldots y_m.s_n^m(x_0,y_1,\ldots,y_m)$. ◁

5.4. Неразрешимые проблемы. Сводимость

В данном разделе мы определим, что понимается под проблемой (задачей)[7] в теории вычислимости и докажем, что для решения некоторых проблем не существует никакого алгоритма.

5.4.1. Понятие массовой проблемы

Поясним, что мы понимаем под проблемой в контексте теории вычислимости. Под массовой проблемой (короче — проблемой) мы понимаем общий вопрос, содержащий параметры и требующий ответа либо «да», либо «нет». Например, вопрос «x и y взаимно просты?» с натуральными числами x и y в качестве параметров является массовой проблемой. При задании значений всем параметрам массовой проблемы получается индивидуальная проблема. Например, индивидуальными проблемами являются «2 и 3 взаимно просты?» и «2 и 4 взаимно просты?». Ставя массовую проблему $\Pi(x_1, \ldots, x_n)$ с параметрами $x_1, \ldots, x_n$, мы желаем выяснить, существует ли алгоритм, который по любым заданным значениям параметров $x_1 = k_1$, $\ldots$, $x_n = k_n$ выдаёт ответ 1, если ответом на поставленную индивидуальную проблему $\Pi(k_1, \ldots, k_n)$ является «да», и ответ 0, если ответом на эту индивидуальную проблему является «нет». Если такой алгоритм существует, то говорят, что эта проблема (алгоритмически) разрешима; иначе говорят, что эта проблема (алгоритмически) неразрешима.

Дадим точное определение проблемы. Любой числовой и любой словарный предикат называют *массовой проблемой разрешения* (или *массовой алгоритмической проблемой*), а также короче — *массовой проблемой* или просто *проблемой*. Имея проблему, представленную предикатом, мы будем выяснять, является ли этот предикат разрешимым.

Как мы знаем (см. окончания разделов 5.2.2 и 5.2.4, а также замечание 2.2.2), любой числовой или словарный n-местный предикат Π с областью определения $X_1 \times \ldots \times X_n$ задаёт множество

$$S_\Pi \rightleftharpoons \{\langle x_1, \ldots, x_n\rangle \in X_1 \times \ldots \times X_n \mid \Pi(x_1, \ldots, x_n) \text{ истинно}\},$$

характеристическая функция которого совпадает с характеристической функцией предиката Π. Поэтому проблему, представленную предикатом Π, можно также представить как множество S_Π и выяснять, разрешимо ли это множество.

Будем говорить, что а) *функция χ решает проблему* Π, если χ является вычислимой характеристической функцией проблемы Π, и б) *алгоритм M решает проблему* Π, если M вычисляет характеристическую функцию проблемы Π.

Пример 5.4.1. Следующие проблемы являются разрешимыми:

1) «x чётно» с параметром $x \in \mathbb{N}$
(множество $\{x \in \mathbb{N} \mid x \text{ чётно}\}$ разрешимо);

2) «x и y взаимно просты» с параметрами $x, y \in \mathbb{N}$
(множество $\{\langle x, y\rangle \in \mathbb{N}^2 \mid x \text{ и } y \text{ взаимно просты}\}$ разрешимо);

[7] В теории вычислимости, излагаемой на русском языке, из двух синонимов «проблема» и «задача» чаще употребляют первый. В предисловии и в аннотации к текущей главе мы употребляли слово «задача», предполагая, что его смысл лучше угадывается.

3) «x — палиндром» с параметром $x \in \Sigma^*$, где $\Sigma \leftrightharpoons \{a, b\}$ (множество $\{x \in \Sigma^* \mid x$ — палиндром$\}$ разрешимо);

4) «машина Тьюринга с номером x имеет ровно 5 команд» с параметром $x \in \mathbb{N}$;

5) «машина Тьюринга с номером x на входе y останавливается, совершив ровно t шагов» с параметрами $x, y, t \in \mathbb{N}$. ◁

Упражнение 5.4.2. Постройте машины Тьюринга и нормальные алгоритмы для решения проблем 1–3, указанных в предыдущем примере. Неформально опишите алгоритмы, решающие проблемы 4 и 5. ◁

5.4.2. Проблемы самоприменимости, остановки и всюду определённости

Важной проблемой и для теории вычислимости, и для программирования является так называемая *проблема остановки* (или *проблема применимости*). Эта проблема заключается в выяснении по произвольным натуральным числам x и y, остановится ли машина Тьюринга M_x на входе y или нет. В терминах функций проблема остановки имеет вид: «$\varphi_x(y)$ определено» с параметрами $x, y \in \mathbb{N}$. Две следующие теоремы покажут, что проблема остановки неразрешима; говоря подробнее, не существует алгоритма, который по произвольным натуральным числам x и y определяет, остановится ли машина Тьюринга M_x на входе y или нет.

В первой теореме мы рассмотрим частный случай проблемы остановки — так называемую *проблему самоприменимости*. В этом случае требуется по произвольному натуральному числу x выяснить, применима ли машина Тьюринга M_x к собственному номеру x или нет. В терминах функций проблема самоприменимости ставится таким образом: «$\varphi_x(x)$ определено» с параметром $x \in \mathbb{N}$.

Теорема 5.4.3 (неразрешимость проблемы самоприменимости)**.** *Проблема «$\varphi_x(x)$ определено» с параметром $x \in \mathbb{N}$ неразрешима.*

Доказательство. Предположим, что эта проблема разрешима. Значит, вычислима характеристическая функция χ этой проблемы. Функция χ такова, что для любого $x \in \mathbb{N}$

$$\chi(x) = \begin{cases} 1, & \text{если } \varphi_x(x) \text{ определено,} \\ 0, & \text{если } \varphi_x(x) \text{ не определено.} \end{cases}$$

Используем вариант диагонального метода доказательства, сходный с парадоксом Рассела. Определим функцию f так, что для любого $x \in \mathbb{N}$

$$f(x) = \begin{cases} \text{не определено}, & \text{если } \chi(x) = 1, \\ 0, & \text{если } \chi(x) = 0. \end{cases}$$

Функция f вычислима, поскольку, имея машину Тьюринга, вычисляющую функцию χ, можно построить машину Тьюринга, вычисляющую функции f, таким образом: на входе x, если $\chi(x) = 0$, то выдать 0, иначе продолжать работу бесконечно.

Для вычислимой функции f найдётся её номер x_0: $f = \varphi_{x_0}$. Полагая значение аргумента x функции f равным x_0 и вспоминая определение функции χ, получаем

$$\varphi_{x_0}(x_0) = f(x_0) = \begin{cases} \text{не определено}, & \text{если } \varphi_{x_0}(x_0) \text{ определено}, \\ 0, & \text{если } \varphi_{x_0}(x_0) \text{ не определено}. \end{cases}$$

Итак, мы имеем противоречие: $\varphi_{x_0}(x_0)$ определено тогда и только тогда, когда $\varphi_{x_0}(x_0)$ не определено. Следовательно, проблема самоприменимости неразрешима. ◁

Теорема 5.4.4 (неразрешимость проблемы остановки)**.** *Проблема «$\varphi_x(y)$ определено» с параметрами $x, y \in \mathbb{N}$ неразрешима.*

Доказательство. Предположим, что эта проблема разрешима. Тогда характеристическая функция χ этой проблемы вычислима. χ такова, что для любых $x, y \in \mathbb{N}$

$$\chi(x, y) = \begin{cases} 1, & \text{если } \varphi_x(y) \text{ определено}, \\ 0, & \text{если } \varphi_x(y) \text{ не определено}. \end{cases}$$

Следовательно, функция $\lambda x.\chi(x, x)$, определённая на $\mathbb{N}$, вычислима. Но эта функция является характеристической функцией проблемы самоприменимости, так что её вычислимость противоречит теореме 5.4.3. Поэтому проблема остановки неразрешима. ◁

Мы доказали, что не существует алгоритма, решающего проблему остановки, но это не означает, что мы не сможем в некоторых случаях по конкретной машине Тьюринга M и конкретному входу x для неё выяснить, остановится ли M на x или нет. Например, машина Тьюринга из примера 5.1.8 применима ко входу $||\#|$ и вообще к любому входу вида $||\ldots|\#||\ldots|$.

Пусть

$$K \leftrightharpoons \{x \in \mathbb{N} \mid \varphi_x(x) \text{ определено}\}.$$

Как мы отмечали в разделе 5.4.1, проблему можно представить в виде множества, характеристическая функция которого совпадает с характеристической функцией этой проблемы, и вместо вопроса о разрешимости этой проблемы ставить вопрос о разрешимости этого множества. Значит, теорему 5.4.3 можно переформулировать так: *множество K неразрешимо.* Отсюда и из легко устанавливаемой перечислимости множества K вытекает следующая

Теорема 5.4.5. *Множество K перечислимо и неразрешимо.*

Доказательство. Неразрешимость множества K следует из теоремы 5.4.3. По теореме 5.2.8 множество K перечислимо, поскольку оно есть область определения вычислимой функции $\lambda x.\psi_U(x, x)$, где ψ_U — вычислимая функция (универсальная для класса всех вычислимых одноместных функций, см. раздел 5.3.2) такая, что $\varphi_x(x) = \psi_U(x, x)$ для любого $x \in \mathbb{N}$. ◁

До сих пор мы не предъявили пример неперечислимого множества, теперь мы можем дать такой пример.

Следствие 5.4.6. *Множество*

$$\{x \in \mathbb{N} \mid \varphi_x(x) \text{ не определено}\} = \mathbb{N} \setminus K$$

неперечислимо.

Доказательство. Если бы $\mathbb{N} \setminus K$ было перечислимо, то, учитывая перечислимость K, по теореме Поста (см. теорему 5.2.7) мы получили бы, что K разрешимо, — противоречие. ◁

Следующая теорема показывает, что не существует алгоритма, который по всякому натуральному числу x определяет, на любом ли входе останавливается машина Тьюринга с номером x или нет.

Теорема 5.4.7 (неразрешимость проблемы всюду определённости)**.** *Проблема «φ_x всюду определена» с параметром $x \in \mathbb{N}$ неразрешима.*

Доказательство. Предположим, что эта проблема разрешима, т. е. существует вычислимая функция χ такая, что для любого $x \in \mathbb{N}$

$$\chi(x) = \begin{cases} 1, & \text{если } \varphi_x \text{ всюду определена,} \\ 0, & \text{если } \varphi_x \text{ не всюду определена.} \end{cases}$$

Воспользуемся диагональным методом доказательства. Определим функцию f, которая является вычислимой, если вычислима χ, но отличается от каждой вычислимой функции.

Пусть функция f такова, что для любого $x \in \mathbb{N}$

$$f(x) = \begin{cases} \varphi_x(x) + 1, & \text{если } \chi(x) = 1, \\ 0, & \text{если } \chi(x) = 0. \end{cases}$$

f вычисляется следующим неформально описанным алгоритмом: на входе x, если $\chi(x) = 1$, то вычислить значение $\psi_U(x, x) + 1$ и выдать его, иначе выдать 0.

С другой стороны, f всюду определена и потому отличается от каждой вычислимой функции, не являющейся всюду определённой. Если же вычислимая функция φ_x всюду определена, то f отличается и от такой φ_x, поскольку $f(x) \neq \varphi_x(x)$.

Итак, мы получили противоречие: функция f вычислима, но отлична от каждой вычислимой функции. ◁

5.4.3. Доказательство неразрешимости проблем методом сведения

В доказательстве теоремы 5.4.4 о неразрешимости проблемы остановки мы показали, что разрешимость этой проблемы повлекла бы разрешимость проблемы самоприменимости. Однако проблема самоприменимости неразрешима в силу теоремы 5.4.3. Опишем использованный метод доказательства в общем виде.

Пусть даны проблемы Π и Π'. Если удаётся доказать, что разрешимость проблемы Π' повлечёт разрешимость проблемы Π, то говорят, что

проблема Π *сводится* к проблеме Π'. Если к тому же известно, что проблема Π неразрешима, то тем самым доказана неразрешимость проблемы Π'. Такой метод доказательства называют *сведе́нием* известной неразрешимой проблемы к исследуемой проблеме. Метод сведения нередко используется в теории вычислимости.

Часто доказательство методом сведения проводится с помощью описания того, как (в предположении, что имеется алгоритм решения проблемы Π') по алгоритму решения проблемы Π' построить алгоритм решения проблемы Π. Поэтому изобретение такого доказательства по сути является программистской задачей: грубо говоря, как воспользоваться подпрограммой, решающей проблему Π', для решения проблемы Π?

Докажем методом сведения несколько теорем о неразрешимости проблем.

Теорема 5.4.8. *Проблема «$\varphi_x = 0$» (т. е. «φ_x есть всюду определённая функция, тождественно равная 0») с параметром $x \in \mathbb{N}$ неразрешима.*

Доказательство. Сведём проблему самоприменимости к поставленной проблеме.

Предположим, что поставленная проблема разрешима, т. е. нашлась вычислимая функция χ такая, что для любого $x \in \mathbb{N}$

$$\chi(x) = \begin{cases} 1, & \text{если } \varphi_x = 0, \\ 0, & \text{если } \varphi_x \neq 0. \end{cases}$$

Определим функцию f так, что для любых $x, y \in \mathbb{N}$

$$f(x,y) = \begin{cases} 0, & \text{если } \varphi_x(x) \text{ определено}, \\ \text{не определено}, & \text{если } \varphi_x(x) \text{ не определено}. \end{cases}$$

Функция f вычисляется следующим неформально описанным алгоритмом: по входным данным x и y вычислить значение универсальной функции $\psi_U(x,x)$ и затем выдать 0.

По следствию 5.3.7 теоремы о параметризации существует вычислимая функция $s : \mathbb{N} \to \mathbb{N}$ такая, что для любого $x \in \mathbb{N}$ имеем $\lambda y.f(x,y) = \varphi_{s(x)}$. Тогда для любых $x, y \in \mathbb{N}$

$$\varphi_{s(x)}(y) = \begin{cases} 0, & \text{если } \varphi_x(x) \text{ определено}, \\ \text{не определено}, & \text{если } \varphi_x(x) \text{ не определено}. \end{cases}$$

Отсюда получаем, что для любого $x \in \mathbb{N}$ верно следующее: $\varphi_{s(x)} = 0$ тогда и только тогда, когда $\varphi_x(x)$ определено.

Поскольку функции s и χ всюду определены и вычислимы, то такова же и их композиция $\lambda x.\chi(s(x))$, причём для любого $x \in \mathbb{N}$

$$\chi(s(x)) = \begin{cases} 1, & \text{если } \varphi_x(x) \text{ определено}, \\ 0, & \text{если } \varphi_x(x) \text{ не определено}. \end{cases}$$

Мы получили, что вычислимая функция $\lambda x.\chi(s(x))$ является характеристической функцией проблемы самоприменимости, что противоречит теореме 5.4.3 о неразрешимости проблемы самоприменимости. Значит, предположение о существовании вычислимой функции χ неверно, и теорема доказана.

Изложим и другое доказательство этой теоремы, которое проведём не в терминах функций, а в терминах машин Тьюринга. Предположим, что имеется машина Тьюринга χ, решающая проблему «$\varphi_x = 0$» с параметром x.

Пусть машина Тьюринга h выполняет следующие действия на входе x:

1) построить машину Тьюринга $s^{(x)}$, которая выполняет такие действия на входе y: «запустить универсальную машину Тьюринга $U(x, x)$ и затем выдать 0»;

2) выдать результат работы машины χ на номере машины $s^{(x)}$.

Для каждого x имеем: машина Тьюринга $s^{(x)}$ на любом входе y выдаёт 0, если и только если $U(x, x) = \varphi_x(x)$ определено. Следовательно, машина Тьюринга h решает проблему самоприменимости, что противоречит неразрешимости этой проблемы. ◁

Следствие 5.4.9. *Проблема «$\varphi_x = \varphi_y$» с параметрами $x, y \in \mathbb{N}$ неразрешима.*

Доказательство. Покажем, что проблема «$\varphi_x = 0$», которая неразрешима по теореме 5.4.8, сводится к поставленной проблеме.

Предположим, что поставленная проблема разрешима. Значит, существует вычислимая функция χ такая, что для любых $x, y \in \mathbb{N}$

$$\chi(x, y) = \begin{cases} 1, & \text{если } \varphi_x = \varphi_y, \\ 0, & \text{если } \varphi_x \neq \varphi_y. \end{cases}$$

Для вычислимой функции $\lambda y.0$ найдётся её номер y_0. Тогда функция $\lambda x.\chi(x, y_0)$ вычислима и является характеристической функцией проблемы «$\varphi_x = 0$», что противоречит теореме 5.4.8. ◁

Упражнение 5.4.10. Докажите, что для любого фиксированного $a \in \mathbb{N}$ проблема «$\varphi_x(a)$ определено» с параметром $x \in \mathbb{N}$ неразрешима. ◁

Упражнение 5.4.11. Докажите, что для любого фиксированного $a \in \mathbb{N}$ проблема «$a \in rng(\varphi_x)$» с параметром $x \in \mathbb{N}$ неразрешима. ◁

Упражнение 5.4.12. Докажите, что проблема «φ_x нигде не определена» с параметром $x \in \mathbb{N}$ неразрешима. ◁

Упражнение 5.4.13. Докажите, что проблема «$dom(\varphi_x) \neq \varnothing$» с параметром $x \in \mathbb{N}$ неразрешима. ◁

Сейчас мы докажем теорему, частными случаями которой являются многие утверждения о неразрешимости проблем.

Обозначим через $\mathcal{C}^{(1)}$ множество всех (числовых) одноместных вычислимых функций.

Теорема 5.4.14 (теорема Райса, или теорема Успенского-Райса)**.** *Пусть $\mathcal{C}$ — произвольное подмножество множества $\mathcal{C}^{(1)}$. Тогда если $\mathcal{C} \neq \varnothing$ и $\mathcal{C} \neq \mathcal{C}^{(1)}$, то проблема «$\varphi_x \in \mathcal{C}$» с параметром $x \in \mathbb{N}$ неразрешима.*

Д о к а з а т е л ь с т в о. Поскольку проблемы «$\varphi_x \in \mathcal{C}$» и «$\varphi_x \in \mathcal{C}^{(1)} \setminus \mathcal{C}$» разрешимы или неразрешимы одновременно, то, не умаляя общности, можно считать, что нигде не определённая функция не принадлежит $\mathcal{C}$ (иначе вместо поставленной проблемы «$\varphi_x \in \mathcal{C}$» рассмотрим проблему «$\varphi_x \in \mathcal{C}^{(1)} \setminus \mathcal{C}$»). Так как $\mathcal{C} \neq \varnothing$, то существует функция $\zeta \in \mathcal{C}$.

Далее это доказательство будет походить на доказательство теоремы 5.4.8. Предположим, что поставленная проблема разрешима, т. е. нашлась вычислимая функция χ такая, что для любого $x \in \mathbb{N}$

$$\chi(x) = \begin{cases} 1, & \text{если } \varphi_x \in \mathcal{C}, \\ 0, & \text{если } \varphi_x \notin \mathcal{C}. \end{cases}$$

Определим функцию f так, что для любых $x, y \in \mathbb{N}$

$$f(x,y) = \begin{cases} \zeta(y), & \text{если } \varphi_x(x) \text{ определено}, \\ \text{не определено}, & \text{если } \varphi_x(x) \text{ не определено}. \end{cases}$$

Функция f вычисляется следующим неформально описанным алгоритмом: по входным данным x и y вычислить значение универсальной функции $\psi_U(x,x)$, затем вычислить и выдать $\zeta(y)$.

По следствию 5.3.7 теоремы о параметризации существует вычислимая функция $s : \mathbb{N} \to \mathbb{N}$ такая, что для любого $x \in \mathbb{N}$ $\lambda y.f(x,y) = \varphi_{s(x)}$. Тогда для любого $x \in \mathbb{N}$ имеем:

1) если $\varphi_x(x)$ определено, то $\varphi_{s(x)} = \zeta$ и, следовательно, $\varphi_{s(x)} \in \mathcal{C}$;

2) если $\varphi_x(x)$ не определено, то функция $\varphi_{s(x)}$ нигде не определена и, следовательно, $\varphi_{s(x)} \notin \mathcal{C}$.

Таким образом, вычислимая функция $\lambda x.\chi(s(x))$ является характеристической функцией проблемы самоприменимости, что противоречит теореме 5.4.3 о неразрешимости проблемы самоприменимости. Теорема доказана.

Приведём и другое доказательство этой теоремы — в терминах машин Тьюринга. Аналогично началу первого доказательства возьмём машину Тьюринга ζ, вычисляющую некоторую функцию из множества $\mathcal{C}$, которому не принадлежит нигде не определённая функция. Предположим, что имеется машина Тьюринга χ, решающая проблему «$\varphi_x \in \mathcal{C}$» с параметром x.

Пусть машина Тьюринга h выполняет следующие действия на входе x:

1) построить машину Тьюринга $s^{(x)}$, которая выполняет такие действия на входе y: «запустить универсальную машину Тьюринга $U(x,x)$, затем вычислить и выдать $\zeta(y)$»;

2) выдать результат работы машины χ на номере машины $s^{(x)}$.

Для каждого x функция, вычисляемая машиной Тьюринга $s^{(x)}$, принадлежит $\mathcal{C}$, если и только если $U(x,x) = \varphi_x(x)$ определено. Следовательно, машина Тьюринга h решает проблему самоприменимости, что противоречит неразрешимости этой проблемы. ◁

Иногда вместо множества $\mathcal{C}$ одноместных вычислимых функций говорят о соответствующем свойстве таких функций, понимая под этим свойством одноместный предикат на множестве $\mathcal{C}^{(1)}$, который задаёт множество

$\mathcal{C}$ (см. замечание 2.2.2). Такое свойство называют *нетривиальным*, если $\mathcal{C} \neq \varnothing$ и $\mathcal{C} \neq \mathcal{C}^{(1)}$, иначе говоря, существует как функция, обладающая этим свойством, так и функция, не обладающая им. Тогда теорему Райса можно переформулировать так: *всякое нетривиальное свойство одноместных вычислимых функций неразрешимо.*

Покажем, что следствием теоремы Райса является теорема 5.4.8 о неразрешимости проблемы «$\varphi_x = 0$». Действительно, свойство одноместной вычислимой функции совпадать с функцией $\lambda y.0$, нетривиально: вычислимая функция $\lambda y.0$ обладает этим свойством, а вычислимая функция $\lambda y.1$ не обладает им.

Теперь переформулируем теорему Райса в терминах машин Тьюринга. Вычислимые функции являются абстракциями машин Тьюринга: некоторые (быть может, важные с какой-нибудь точки зрения) свойства машин Тьюринга не учитываются при рассмотрении вычислимых функций.

Определение нетривиального свойства переносится на машины Тьюринга почти дословно: свойство машин Тьюринга называется *нетривиальным*, если существуют как машина Тьюринга, обладающая этим свойством, так и машина Тьюринга, не обладающая им. Далее, одна и та же (одноместная числовая вычислимая) функция представляет класс всех вычисляющих её машин Тьюринга. Поэтому введём следующее определение: свойство машин Тьюринга назовём *инвариантным*, если любые машины Тьюринга, вычисляющие одну и ту же функцию, обладают или не обладают этим свойством одновременно. Тогда теорему Райса в терминах машин Тьюринга можно сформулировать так: *всякое нетривиальное инвариантное свойство машин Тьюринга неразрешимо.*

Например, свойство машины Тьюринга выдавать 0 на любом входе является нетривиальным инвариантным и потому неразрешимым. Однако свойство машины Тьюринга иметь ровно 5 команд не инвариантно, поэтому последний вариант теоремы Райса не применим (это свойство разрешимо, как мы знаем из примера 5.4.1 и упражнения 5.4.2).

Упражнение 5.4.15. Докажите, что свойство машины Тьюринга иметь ровно 5 команд не инвариантно, построив две машины Тьюринга, вычисляющие одну и ту же функцию: первую машину — ровно с 5 командами, а вторую — с числом команд, отличным от 5. ◁

Упражнение 5.4.16. Докажите, что проблема «$rng(\varphi_x)$ состоит ровно из y элементов» с параметрами $x, y \in \mathbb{N}$ неразрешима. *Указание.* К данной проблеме теорема Райса не применима, но если зафиксировать значение параметра y, то к полученному частному случаю этой проблемы можно применить теорему Райса. ◁

5.4.4. m-сводимость и m-полнота

Для доказательства неразрешимости некоторых проблем мы неоднократно пользовались методом сведения одной проблемы к другой (см. раздел 5.4.3). Мы говорили, что проблема Π сводится к проблеме Π', если разрешимость проблемы Π' повлечёт разрешимость проблемы Π. Каждый раз при использовании метода сведения мы понимали, что означает сведение проблемы Π к проблеме Π'. Однако конкретные способы сведения

одной проблемы к другой до сих пор не уточнены. В данном разделе мы рассмотрим один из подходов к уточнению понятия сводимости проблем.

Напомним, что проблема Π с параметром $x \in \mathbb{N}$ представима в виде множества $\{x \in \mathbb{N} \mid \Pi(x) \text{ истинно}\}$. Уточним понятие сводимости таких проблем как следующее отношение между множествами натуральных чисел.

Определение 5.4.17. Говорят, что множество $A \subseteq \mathbb{N}$ *m-сводится* (*многосводимо* или *много-односводимо*) к множеству $B \subseteq \mathbb{N}$, и обозначают это через $A \leqslant_m B$, если существует вычислимая функция $f : \mathbb{N} \to \mathbb{N}$ такая, что для любого $x \in \mathbb{N}$ выполняется: $x \in A$ тогда и только тогда, когда $f(x) \in B$. Функцию f называют функцией, *m-сводящей* A к B. ◁

Замечание 5.4.18. Термин «m-сводимость» (и производные от него) происходит от словосочетания «many-one reducibility», которое указывает на то, что функция f не обязательно инъективна. Также известно понятие односводимости, которое отличается от m-сводимости лишь дополнительным требованием инъективности функции f. ◁

Напомним ранее введённое обозначение:

$$K \leftrightharpoons \{x \in \mathbb{N} \mid \varphi_x(x) \text{ определено}\}.$$

Пример 5.4.19. 1. В доказательстве теоремы 5.4.8 мы построили функцию s, m-сводящую множество K к множеству $\{x \in \mathbb{N} \mid \varphi_x = 0\}$.

2. В доказательстве теоремы Райса (см. теорему 5.4.14) мы построили функцию s, m-сводящую множество K к множеству $\{x \in \mathbb{N} \mid \varphi_x \in \mathcal{C}\}$, где $\mathcal{C}$ — любое непустое множество одноместных вычислимых функций, которому не принадлежит нигде не определённая функция.

3. Функция $\lambda x.\boldsymbol{c}(x, x)$ m-сводит $\mathbb{N}$ к $\{y \in \mathbb{N} \mid \exists x(y = \boldsymbol{c}(x, x))\}$, где $\boldsymbol{c}$ — канторовская нумерующая функция, определённая в разделе 5.2.2.

4. Множество

$$K^{(2)} \leftrightharpoons \{\langle x, y\rangle \in \mathbb{N}^2 \mid \varphi_x(y) \text{ определено}\}$$

представляет проблему остановки. В доказательстве теоремы 5.4.4 мы свели проблему самоприменимости к проблеме остановки. Проблема самоприменимости представляется множеством K, но, согласно данному определению m-сводимости, множество $K^{(2)}$ не может m-сводиться к множеству K, поскольку $K^{(2)}$ есть множество упорядоченных пар чисел. Закодировав все пары множества $K^{(2)}$ с помощью канторовской нумерующей функции $\boldsymbol{c}$, получаем множество

$$K_h \leftrightharpoons \{\boldsymbol{c}(x, y) \mid \varphi_x(y) \text{ определено}\},$$

которое более формально можно записать как

$$\{z \in \mathbb{N} \mid \exists x \exists y(z = \boldsymbol{c}(x, y) \wedge \varphi_x(y) \text{ определено})\}.$$

Теперь функция $\lambda x.\boldsymbol{c}(x, x)$ m-сводит множество K к множеству K_h.

5. В доказательстве теоремы 5.4.9 мы свели проблему «$\varphi_x = 0$» к проблеме «$\varphi_x = \varphi_y$». Пусть y_0 — номер функции $\lambda y.0$. Функция $\lambda x.\boldsymbol{c}(x, y_0)$ m-сводит множество $\{x \in \mathbb{N} \mid \varphi_x = 0\}$ к множеству $\{\boldsymbol{c}(x, y) \mid \varphi_x = \varphi_y\}$, которое более формально можно записать как

$$\{z \in \mathbb{N} \mid \exists x \exists y(z = \boldsymbol{c}(x, y) \wedge \varphi_x = \varphi_y)\}.$$

◁

Установим несколько свойств m-сводимости.

Теорема 5.4.20. *Пусть A, B, C — произвольные подмножества $\mathbb{N}$. Тогда*

(a) $A \leqslant_m A$ *(т. е. отношение $\leqslant_m$ рефлексивно); если $A \leqslant_m B$ и $B \leqslant_m C$, то $A \leqslant_m C$ (т. е. отношение $\leqslant_m$ транзитивно);*

(b) *если $A \leqslant_m B$, то $\mathbb{N} \setminus A \leqslant_m \mathbb{N} \setminus B$;*

(c) *если $A \leqslant_m B$ и B разрешимо, то A разрешимо;*

(d) *если $A \leqslant_m B$ и B перечислимо, то A перечислимо;*

(e) *если A разрешимо, $B \neq \varnothing$ и $B \neq \mathbb{N}$, то $A \leqslant_m B$;*

(f) $A \leqslant_m \mathbb{N}$ *тогда и только тогда, когда* $A = \mathbb{N}$;

(g) $A \leqslant_m \varnothing$ *тогда и только тогда, когда* $A = \varnothing$;

(h) $\mathbb{N} \leqslant_m A$ *тогда и только тогда, когда* $A \neq \varnothing$;

(i) $\varnothing \leqslant_m A$ *тогда и только тогда, когда* $A \neq \mathbb{N}$.

Доказательство. (a) Функция $\lambda x.x$ m-сводит A к A. Если функция f m-сводит A к B и функция g m-сводит B к C, то функция $\lambda x.g(f(x))$ m-сводит A к C.

(b) Функция, m-сводящая A к B, m-сводит и $\mathbb{N} \setminus A$ к $\mathbb{N} \setminus B$.

(c) Пусть функция f m-сводит A к B и χ_B — вычислимая характеристическая функция множества B. Тогда $\lambda x.\chi_B(f(x))$ является вычислимой характеристической функцией множества A.

(d) Пусть функция f m-сводит A к B и χ_B^* — вычислимая частичная характеристическая функция множества B. Тогда $\lambda x.\chi_B^*(f(x))$ является вычислимой частичной характеристической функцией множества A.

(e) Выберем $b \in B$ и $c \in \mathbb{N} \setminus B$ и определим функцию f так, что для любого $x \in \mathbb{N}$

$$f(x) = \begin{cases} b, & \text{если } x \in A, \\ c, & \text{если } x \notin A. \end{cases}$$

Поскольку множество A разрешимо, то функция f вычислима. Кроме того, для любого $x \in \mathbb{N}$ выполняется: $x \in A$, если и только если $f(x) \in B$. Таким образом, f m-сводит A к B.

Доказательство пунктов (f)–(i) остаётся в качестве лёгкого упражнения. ◁

Упражнение 5.4.21. Докажите пунктов (f)–(i) из теоремы 5.4.20. ◁

Свойства m-сводимости, установленные теоремой 5.4.20, позволяют нам предъявить в следующей теореме два неразрешимых множества, ни одно из которых не m-сводится к другому.

Теорема 5.4.22. *Неверно, что $\mathbb{N} \setminus K \leqslant_m K$, и неверно, что $K \leqslant_m \mathbb{N} \setminus K$.*

Доказательство. По теореме 5.4.5 множество K перечислимо, а по теореме 5.4.6 множество $\mathbb{N} \setminus K$ неперечислимо. Следовательно, в силу пункта (d) теоремы 5.4.20 неверно, что $\mathbb{N} \setminus K \leqslant_m K$. Тогда в силу пункта (b) теоремы 5.4.20 неверно, что $K \leqslant_m \mathbb{N} \setminus K$. ◁

Упражнение 5.4.23. Докажите, что множество K m-сводится к каждому из множеств, представляющих проблемы из упражнений 5.4.10–5.4.13. ◁

Оказывается, что в классе всех перечислимых подмножеств $\mathbb{N}$ существуют множества, к которым m-сводится любое перечислимое подмножество $\mathbb{N}$. Интуитивно говоря, эти множества обладают «максимальной степенью неразрешимости» среди всех перечислимых множеств: согласно пункту (c) теоремы 5.4.20, если одно из этих множеств было бы разрешимо, то и все перечислимые подмножества $\mathbb{N}$ оказались бы разрешимы. Как утверждает следующая теорема, таково множество K_h, определённое в примере 5.4.19.

Теорема 5.4.24. *Множество K_h перечислимо и $A \leqslant_m K_h$ для любого перечислимого множества $A \subseteq \mathbb{N}$.*

Доказательство. Множество $K^{(2)}$ (определённое в примере 5.4.19), очевидно, перечислимо, поскольку $K^{(2)}$ есть область определения вычислимой функции ψ_U (универсальной для класса всех вычислимых одноместных функций, см. раздел 5.3.2). Поэтому $K_h = \boldsymbol{c}(K^{(2)})$ перечислимо.

По теореме 5.2.8 для любого перечислимого множества $A \subseteq \mathbb{N}$ найдётся вычислимая функция φ_a (с номером a), областью определения которой является A. Для любого y имеем: $y \in A$, если и только если $\varphi_a(y)$ определено, что в свою очередь равносильно $\boldsymbol{c}(a, y) \in K_h$. Таким образом, функция $\lambda y.\boldsymbol{c}(a, y)$ m-сводит A к K_h. ◁

Определение 5.4.25. Множество $B \subseteq \mathbb{N}$ называется *m-полным* (подробнее — *m-полным в классе всех перечислимых множеств натуральных чисел относительно $\leqslant_m$*), если B перечислимо и $A \leqslant_m B$ для любого перечислимого множества $A \subseteq \mathbb{N}$. ◁

Согласно теореме 5.4.24 множество K_h является m-полным. Следующая теорема показывает, как можно устанавливать m-полноту исследуемого множества, опираясь на известную m-полноту некоторого множества.

Теорема 5.4.26. *Если A — m-полное множество, B — перечислимое подмножество $\mathbb{N}$ и $A \leqslant_m B$, то B является m-полным.*

Доказательство. Пусть дано какое угодно перечислимое множество $A' \subseteq \mathbb{N}$. m-полнота множества A влечёт $A' \leqslant_m A$. В силу транзитивности отношения $\leqslant_m$ (см. пункт (a) теоремы 5.4.20) из $A' \leqslant_m A$ и $A \leqslant_m B$ следует $A' \leqslant_m B$. ◁

Предыдущая теорема показывает, что все m-полные множества m-сводятся друг к другу. Каждое m-полное множество неразрешимо: иначе в силу пункта (c) теоремы 5.4.20 было бы разрешимо любое перечислимое множество, но среди перечислимых множеств имеются неразрешимые.

Также теорема 5.4.26 обеспечивает такой метод доказательства m-полноты множества B: достаточно установить, что а) B перечислимо, и б) некоторое m-полное множество m-сводится к B.

Докажем этим методом, что множество K m-полно.

Теорема 5.4.27. *Множество K m-полно.*

Доказательство. Перечислимость множества K установлена теоремой 5.4.5. По теореме 5.4.24 множество K_h m-полно. Построим функцию, m-сводящую множество K_h к множеству K.

С помощью канторовских функций $\boldsymbol{l}$ и $\boldsymbol{r}$ (см. раздел 5.2.2) множество K_h можно описать таким образом:

$$K_h = \{x \in \mathbb{N} \mid \varphi_{\boldsymbol{l}(x)}(\boldsymbol{r}(x)) \text{ определено}\}.$$

Поэтому для любого $x \in \mathbb{N}$ имеем: $x \in K_h$ равносильно тому, что $\varphi_{\boldsymbol{l}(x)}(\boldsymbol{r}(x))$ определено.

Для любого $x \in \mathbb{N}$ сконструируем машину Тьюринга $M^{(x)}$, которая на входе y вычисляет $\varphi_{\boldsymbol{l}(x)}(\boldsymbol{r}(x))$ и выдаёт 0. Теперь для любого $x \in \mathbb{N}$ имеем: $x \in K_h$ тогда и только тогда, когла $M^{(x)}$ останавливается хотя бы на одном входе y, причём всё равно на каком, например, на номере машины Тьюринга $M^{(x)}$.

Определим функцию $s : \mathbb{N} \to \mathbb{N}$, указав неформальный алгоритм её вычисления: каждому x функция s сопоставляет номер машины Тьюринга $M^{(x)}$. Для каждого $x \in \mathbb{N}$ машина $M^{(x)}$ задаёт функцию $\varphi_{s(x)}$, и потому $M^{(x)}$ останавливается на входе $s(x)$, если и только если $\varphi_{s(x)}(s(x))$ определено. Следовательно, для любого $x \in \mathbb{N}$ верно: $x \in K_h$ тогда и только тогда, когда $\varphi_{s(x)}(s(x))$ определено, т. е. когда $s(x) \in K$. Итак, функция s m-сводит K_h к K.

Приведём и другое доказательство того, что $K_h \leqslant_m K$, — доказательство в терминах функций, использующее перечислимость множества K_h вместо деталей внутреннего устройства этого множества. Определим функцию g так, что для любых $x, y \in \mathbb{N}$

$$g(x,y) = \begin{cases} 0, & \text{если } x \in K_h, \\ \text{не определено}, & \text{если } x \notin K_h. \end{cases}$$

Функция g вычислима, поскольку K_h перечислимо. По следствию 5.3.7 теоремы о параметризации существует вычислимая функция $s : \mathbb{N} \to \mathbb{N}$ такая, что для любого $x \in \mathbb{N}$ $\lambda y.g(x,y) = \varphi_{s(x)}$. Следовательно, для любого $x \in \mathbb{N}$ имеем: $x \in K_h$ тогда и только тогда, когда $g(x, s(x)) = \varphi_{s(x)}(s(x))$ определено, т. е. когда $s(x) \in K$. Значит, функция s m-сводит K_h к K. ◁

Замечание 5.4.28. Во втором доказательстве предыдущей теоремы из свойств множества K_h мы использовали лишь его перечислимость. Поэтому, заменив в этом доказательстве K_h на произвольное перечислимое множество $A \subseteq \mathbb{N}$, получим доказательство того, что $A \leqslant_m K$. ◁

Пример 5.4.29. 1. Если перечислимо множество $\{x \in \mathbb{N} \mid \varphi_x \in \mathcal{C}\}$, где $\mathcal{C}$ — любое непустое множество одноместных вычислимых функций, которому не принадлежит нигде не определённая функция, то это множество m-полно, поскольку оно m-сводится к K (см. пример 5.4.19).

2. Множество $\{x \in \mathbb{N} \mid x \text{ чётно}\}$ не является m-полным, так как оно разрешимо.

3. Множество $\{x \in \mathbb{N} \mid \varphi_x(x) \text{ не определено}\}$ не m-полно, так как по теореме 5.4.6 оно неперечислимо. ◁

Упражнение 5.4.30. Конкретизируйте пункт 1 примера 5.4.29, указав несколько примеров m-полных множеств. ◁

Упражнение 5.4.31. Выясните, являются ли следующие множества m-полными:

1) $\{x \in \mathbb{N} \mid \varphi_x$ нигде не определена$\}$,

2) $\{x \in \mathbb{N} \mid \varphi_x$ всюду определена$\}$,

3) $\{x \in \mathbb{N} \mid \varphi_x = 0\}$,

4) $\{z \in \mathbb{N} \mid \exists x \exists y (z = \boldsymbol{c}(x, y) \wedge \varphi_x = \varphi_y)\}$,

5) $\{x \in \mathbb{N} \mid \varphi_x(a)$ определено$\}$ для какого угодно $a \in \mathbb{N}$. ◁

5.4.5. Теорема о рекурсии

Теорема о рекурсии, которой посвящён данный раздел, является удобным инструментом для развития теории вычислимости. В программистских терминах эта теорема по сути утверждает, что всякий алгоритм, преобразующий любые программы, хотя бы для одной программы выдаёт эквивалентную ей.

Теорема 5.4.32 (теорема о рекурсии, или о неподвижной точке)**.** (a) *Для любой вычислимой функции $f : \mathbb{N} \to \mathbb{N}$ существует такое $n \in \mathbb{N}$, что $\varphi_n = \varphi_{f(n)}$ (т. е. n и $f(n)$ — номера одной и той же функции).*
(b) *Для любой вычислимой функции $g : \mathbb{N}^2 \xrightarrow{p} \mathbb{N}$ сущетвует такое $n \in \mathbb{N}$, что $\varphi_n = \lambda y.g(n, y)$.*

Доказательство. (a) Рассмотрим функцию $\lambda xy.\varphi_{f(\varphi_x(x))}(y)$. Эта функция вычисляется следующим неформально описанным алгоритмом, на вход которого подаются произвольные $x, y \in \mathbb{N}$: исполнить $M_x(x)$; результат (если он будет получен) подать на вход машины Тьюринга, вычисляющей функцию f; используя вычисленное значение w как номер машины Тьюринга M_w, вычислить $M_w(y)$ и, если это вычисление завершится, выдать полученный результат $M_w(y)$.

По следствию 5.3.7 теоремы о параметризации существует вычислимая функция $s : \mathbb{N} \to \mathbb{N}$ такая, что для любого $x \in \mathbb{N}$ $\lambda y.\varphi_{f(\varphi_x(x))}(y) = \varphi_{s(x)}$. Для вычислимой функции s найдётся её номер m: $s = \varphi_m$. Тогда для любого $x \in \mathbb{N}$ имеем $\varphi_{f(\varphi_x(x))} = \varphi_{\varphi_m(x)}$. Беря m в качестве x, получаем $\varphi_{f(\varphi_m(m))} = \varphi_{\varphi_m(m)}$, причём $\varphi_m(m)$ определено, поскольку $s = \varphi_m$ и s всюду определена. Положив $n = \varphi_m(m)$, получаем $\varphi_{f(n)} = \varphi_n$, что и требовалось.

(b) Воспользовавшись следствием 5.3.7 теоремы о параметризации, получим вычислимую функцию $s : \mathbb{N} \to \mathbb{N}$ такую, что для любого $x \in \mathbb{N}$ $\lambda y.g(x, y) = \varphi_{s(x)}$. По пункту (a) настоящей теоремы существует $n \in \mathbb{N}$, для которого $\varphi_n = \varphi_{s(n)}$. Из двух последних равенств при $x = n$ получаем $\varphi_n = \lambda y.g(n, y)$. ◁

Пусть дана функция f, удовлетворяющая условию теоремы о рекурсии. Тогда любое $n \in \mathbb{N}$, для которого $\varphi_n = \varphi_{f(n)}$, называют *неподвижной точкой для функции f*. (Следует отличать этот термин от более традиционного математического термина «неподвижная точка функции f», которым называют любое x такое, что $x = f(x)$.)

Упражнение 5.4.33. Пусть функция $f : \mathbb{N} \to \mathbb{N}$ вычислима. Докажите, что множество неподвижных точек для f бесконечно. ◁

Упражнение 5.4.34. Обобщите теорему о рекурсии на случай, когда рассматриваются функции $\varphi_x^{(k)}$ и m-местная функция f. ◁

Продемонстрируем несколько применений теоремы о рекурсии.

Следующая теорема говорит о том, что существует самовоспроизводящаяся машина Тьюринга, которая на любом входе выдаёт свой номер.

Теорема 5.4.35. *Существует $p \in \mathbb{N}$ такое, что $\varphi_p = p$.*

Доказательство. Построим машину Тьюринга M, которая на произвольном входе x выдаёт номер машины Тьюринга $M^{(x)}$, на любом входе выдающей x. Машина M задаёт некоторую функцию $f : \mathbb{N} \to \mathbb{N}$ такую, что $\varphi_{f(x)} = x$ для любого $x \in \mathbb{N}$. По теореме о рекурсии существует такое $p \in \mathbb{N}$, что $\varphi_p = \varphi_{f(p)}$. Из двух последних равенств при $x = p$ получаем $\varphi_p = p$. ◁

Упражнение 5.4.36. Напишите на каком-либо языке программирования программу, которая печатает свой текст. *Указание.* Имея возможность вызывать интерпретатор используемого языка программирования как подпрограмму, такую программу можно построить, следуя доказательству теоремы о рекурсии. Другой подход — записать в виде программы следующее неформальное описание: «напечатать два раза, второй раз в апострофах, такой текст: 'напечатать два раза, второй раз в апострофах, такой текст:'». ◁

Докажем теорему Райса (см. теорему 5.4.14) с помощью теоремы о рекурсии.

Теорема 5.4.37 (теорема Райса, или теорема Успенского-Райса)**.** *Пусть $\mathcal{C}$ — произвольное подмножество множества $\mathcal{C}^{(1)}$ всех числовых одноместных вычислимых функций. Тогда если $\mathcal{C} \neq \varnothing$ и $\mathcal{C} \neq \mathcal{C}^{(1)}$, то проблема «$\varphi_x \in \mathcal{C}$» с параметром $x \in \mathbb{N}$ неразрешима.*

Доказательство. Предположим, что данная проблема разрешима, т. е. разрешимо множество $A \leftrightharpoons \{x \in \mathbb{N} \mid \varphi_x \in \mathcal{C}\}$. Поскольку $\mathcal{C} \neq \varnothing$ (т. е. $A \neq \varnothing$) и $\mathcal{C} \neq \mathcal{C}^{(1)}$ (т. е. $A \neq \mathbb{N}$), то найдутся $a \in A$ и $b \in \mathbb{N} \setminus A$. Тогда функция f такая, что для любого $x \in \mathbb{N}$

$$f(x) = \begin{cases} a, & \text{если } x \notin A, \\ b, & \text{если } x \in A, \end{cases}$$

вычислима.

По теореме о рекурсии существует такое $n \in \mathbb{N}$, что $\varphi_n = \varphi_{f(n)}$. Отсюда

$$n \in A, \text{ если и только если } f(n) \in A. \qquad (*)$$

По определению функции f для любого $x \in \mathbb{N}$ имеет место следующее: $x \in A$, если и только если $f(x) \notin A$. Беря n в качестве x, получаем:

$$n \in A, \text{ если и только если } f(n) \notin A,$$

что вместе с $(*)$ даёт противоречие. ◁

5.5. Применения теории вычислимости

В данном разделе мы определим проблему равенства для полугруппы, заданной образующими и определяющими соотношениями, и установим неразрешимость этой проблемы для некоторой полугруппы, опираясь на неразрешимость проблемы самоприменимости. Далее, используя полученный результат о полугруппе, мы докажем, что неразрешима и проблема выяснения по формуле языка элементарной теории полугрупп, является ли эта формула общезначимой или нет; тем самым мы раскроем принципиальное ограничение поиска доказательств с помощью компьютера. Наконец, мы применим развитую теорию вычислимости для доказательства теорем Гёделя о неполноте арифметики, которые показывают принципиальные ограничения формализации содержательных математических рассуждений.

5.5.1. Проблема равенства для полугрупп

Пусть дан алфавит $\Sigma \leftrightharpoons \{a_1, \ldots, a_m\}$ и конечный список пар слов в этом алфавите $\langle A_i, B_i \rangle$, $i = 1, \ldots, n$. *Ассоциативным исчислением в алфавите* Σ называется исчисление в этом алфавите без аксиом и со следующими $2n$ правилами вывода:

$$\frac{UA_iV}{UB_iV}, \qquad \frac{UB_iV}{UA_iV},$$

где U, V — произвольные слова в алфавите Σ, $i = 1, \ldots, n$. Обозначим исчисление с этими правилами через $\mathcal{C}$.

Для произвольных заданных слов D и E в алфавите Σ правило

$$\frac{UDV}{UEV},$$

где U, V — произвольные слова в алфавите Σ, мы будем кратко записывать в виде $D \to E$, предполагая, что $\to$ не является символом алфавита Σ (в дальнейшем аналогичные естественные предположения мы не упоминаем). Таким образом, правила исчисления $\mathcal{C}$ записываются в виде

$$A_i \to B_i, \qquad B_i \to A_i \qquad (i = 1, \ldots, n).$$

Будем говорить, что правило $D \to E$ является *обратным* к правилу $E \to D$.

Назовём слова X и Y *эквивалентными в* $\mathcal{C}$, если $X \vdash_{\mathcal{C}} Y$ (см. раздел 1.2.2 о выводе из гипотез).

Проблема эквивалентности (*проблема Туэ*) *для ассоциативного исчисления* $\mathcal{C}$ в алфавите Σ ставится таким образом: по произвольным словам X и Y в алфавите Σ выяснить, эквивалентны в $\mathcal{C}$ X и Y или нет.

Теперь сформулируем эту проблему в алгебраических терминах. Для каждого $i = 1, \ldots, n$ объединим записи правила $A_i \to B_i$ и правила $B_i \to A_i$, получив следующее представление правил вывода исчисления $\mathcal{C}$:

$$A_i \leftrightarrow B_i \qquad (i = 1, \ldots, n). \qquad (*)$$

Введём на множестве слов Σ^* отношение $\leftrightarrow$, положив по определению $X \leftrightarrow Y$, если $X \vdash_{\mathcal{C}} Y$. В равносильной форме $X \leftrightarrow Y$ можно определить так:

существует такая последовательность слов $Z_1, \ldots, Z_k$ $(k \geqslant 1)$, что $X = Z_1$, $Z_k = Y$ и для каждого $l = 2, \ldots, k$ Z_l является результатом замены некоторого вхождения A_i в Z_{l-1} на B_i или результатом замены некоторого вхождения B_i в Z_{l-1} на A_i при некотором $i = 1, \ldots, n$.

Легко видеть, что отношение $\leftrightarrow$ является отношением эквивалентности на Σ^*. Обозначим множество всех классов эквивалентности по отношению $\leftrightarrow$ через $\mathcal{S}$, а класс эквивалентности с представителем Z — через $[Z]$. Определим на множестве $\mathcal{S}$ операцию $\cdot$, положив $[U] \cdot [V] = [UV]$. Множество $\mathcal{S}$ с операцией $\cdot$ является полугруппой, которая называется *полугруппой с образующими* $a_1, \ldots, a_m$ *и определяющими соотношениями* $(*)$.

Итак, проблему эквивалентности для ассоциативного исчисления $\mathcal{C}$ можно переформулировать как *проблему равенства для полугруппы* с образующими $a_1, \ldots, a_m$ и определяющими соотношениями $(*)$: по произвольным словам X и Y в алфавите Σ выяснить, $[X] = [Y]$ или нет.

Упражнение 5.5.1. Докажите, что разрешима проблема равенства для полугруппы с образующими $a_1, \ldots, a_m$ и определяющими соотношениями $a_i a_j \leftrightarrow a_j a_i$ (для всех $i < j$). $\triangleleft$

Упражнение 5.5.2. Докажите, что для любого ассоциативного исчисления множество всех пар эквивалентных слов перечислимо. $\triangleleft$

Теорема 5.5.3. *Существует ассоциативное исчисление с неразрешимой проблемой эквивалентности.*

Доказательство. Сначала рассмотрим исчисления без аксиом с правилами вывода вида $A \to B$. Такие исчисления мы назовём *полуассоциативными*; они необязательно являются ассоциативными, т. е. необязательно имеют для каждого правила вывода обратное. В текущем доказательстве будем называть *проблемой выводимости* для какого угодно полуассоциативного исчисления $\mathcal{C}_0$ проблему выяснения по любым словам X и Y (в алфавите исчисления $\mathcal{C}_0$), $X \vdash_{\mathcal{C}_0} Y$ или нет. Докажем, что существует полуассоциативное исчисление, для которого неразрешима проблема выводимости. С целью сведения проблемы самоприменимости к проблеме выводимости для некоторого полуассоциативного исчисления, опишем построение по произвольной машине Тьюринга полуассоциативного исчисления, в котором применение правил соответствует исполнению команд данной машины Тьюринга.

Пусть дана произвольная машина Тьюринга

$$M \leftrightharpoons \langle \Gamma,\ \Sigma,\ \Delta,\ S,\ q_0,\ F,\ P \rangle.$$

Будем представлять конфигурацию Z_M машины M словом вида $\langle Z_0 \rangle$, где ни символ $\langle$, ни символ $\rangle$ не принадлежит $\Gamma \cup S$, и слово Z_0 получается из Z_M удалением с обоих концов слова Z_M всех пробелов за исключением наблюдаемого пробела. Например, пусть $q_2 \in S$, $a_3, a_5 \in \Gamma \setminus \{\_\}$, тогда конфигурация $\_\_a_3\_\_q_2a_5\_$ представляется словом $\langle a_3\_\_q_2a_5 \rangle$, конфигурация $\_a_3\_\_q_2\_\_$ представляется словом $\langle a_3\_\_q_2\_ \rangle$, и конфигурация $\_\_q_2\_\_$ представляется словом $\langle q_2\_ \rangle$.

Определим полуассоциативное исчисление $\mathcal{C}$ в алфавите $\Gamma \cup S \cup \{\langle, \rangle\}$. В этом исчислении нет аксиом.

Правила вывода исчисления $\mathcal{C}$, соответствующие исполнению команд, будем формулировать в виде $A_i \to B_i$ так, что для любого слова Z, представляющего конфигурацию Z_M, к которой применима некоторая команда, выполняется: а) к Z применимо ровно одно правило, причём единственным способом (т. е. в Z имеется единственное вхождение A_i), и б) при применении правила к Z получается слово, представляющее следующую за Z_M конфигурацию.

Для каждой команды (машины M) вида $qa \to rb$ исчисление $\mathcal{C}$ имеет правило вывода $qa \to rb$.

Для каждой команды вида $qa \to rbL$ исчисление $\mathcal{C}$ имеет следующие правила вывода:

1) если $b \neq \_$, то для каждого символа c алфавита Γ правило

$$cqa \to rcb$$

и правило

$$\langle qa \to \langle r\_b \ ;$$

2) если $b = \_$, то для каждой пары символов c и d алфавита Γ правило

$$cqad \to rc\_d \ ;$$

для каждого символа d алфавита Γ правило

$$\langle qad \to \langle r\_\_d \ ;$$

для каждого символа c алфавита Γ правило

$$cqa\rangle \to rc\rangle \ ;$$

и, наконец, правило

$$\langle qa\rangle \to \langle r\_\rangle \ .$$

Аналогично формулируются правила вывода исчисления $\mathcal{C}$ для каждой команды вида $qa \to rbR$.

В текущем доказательстве для произвольного слова X мы будем обозначать через $\widetilde{X}$ слово $\_$, если X пусто, и слово X иначе.[8]

Мы бы хотели показать, что

$$\textit{для любых } X \in \Sigma^*,\ Y \in \Delta^* \textit{ верно: } M(X) = Y \ \Leftrightarrow\ \langle q_0\widetilde{X}\rangle \vdash_{\mathcal{C}} \langle Y\rangle \ . \qquad (\star)$$

Однако это не так, но лишь поскольку результат работы машины M не есть всё слово $\langle UqYV\rangle$, представляющее заключительную конфигурацию ($q \in F$), а есть наибольшее по длине подслово Y слова YV, состоящее только из символов выходного алфавита Δ. Чтобы утверждение $(\star)$ стало верным, добавим правила вывода в исчисление $\mathcal{C}$, после применения которых к любому слову, представляющему заключительную конфигурацию, получается $\langle Y\rangle$, где Y — результат работы машины M.

[8] Это обозначение будет удобно использовать для записи начальной конфигурации (см. пункт 1 определения 5.1.3 на с. 182), например, в $(\star)$.

Добавим в алфавит исчисления $\mathcal{C}$ 3 новых символа $\blacktriangleleft, \triangleright, \blacktriangleright$ и добавим следующие правила вывода в исчисление $\mathcal{C}$:

$$\begin{aligned}
q &\to \blacktriangleleft \quad \text{для каждого } q \in F,\\
a \blacktriangleleft &\to \blacktriangleleft \quad \text{для каждого } a \in \Gamma,\\
\langle\blacktriangleleft &\to \langle\triangleright\ ,\\
\triangleright\, d &\to d \triangleright \quad \text{для каждого } d \in \Delta,\\
\triangleright\, b &\to \blacktriangleright b \quad \text{для каждого } b \in (\Gamma \setminus \Delta) \cup \{\rangle\},\\
\blacktriangleright a &\to \blacktriangleright \quad \text{для каждого } a \in \Gamma,\\
\blacktriangleright\rangle &\to \rangle\ .
\end{aligned}$$

Применение этих правил к любому слову, представляющему заключительную конфигурацию, состоит в удалении всех символов слева от заключительного состояния до $\langle$, оставлении всех символов из Δ, представляющих результат, и удалении всех символов справа от результата до $\rangle$. Добавление этих правил в исчисление $\mathcal{C}$ делает утверждение $(\star)$ верным.

Теперь возьмём в качестве M машину Тьюринга, которая на входе $x \in \mathbb{N}$ (представленным словом $\overline{x}$) вычисляет $\varphi_x(x)$ (запуская универсальную машину Тьюринга $U(x,x)$) и затем выдаёт 0 (представленный пустым словом). Ясно, что для любого $x \in \mathbb{N}$ имеем: $M(x) = 0$ тогда и только тогда, когда $\varphi_x(x)$ определено. В качестве $\mathcal{C}$ возьмём полуассоциативное исчисление, построенное по машине M так, как описано выше. Тогда в силу $(\star)$ для любого $x \in \mathbb{N}$ имеем: $\varphi_x(x)$ определено, если и только если $\langle q_0\widetilde{\overline{x}}\rangle \vdash_{\mathcal{C}} \langle\rangle$. Проблема выводимости для этого исчисления $\mathcal{C}$ неразрешима, поскольку иначе была бы разрешима проблема самоприменимости.

Наконец, докажем, что для некоторого *ассоциативного* исчисления неразрешима проблема эквивалентности. Выше было построено полуассоциативное исчисление $\mathcal{C}$ с неразрешимой проблемой выводимости. Рассмотрим исчисление $\mathcal{C}'$, которое получается из $\mathcal{C}$ добавлением нового символа $\bullet$ и заменой правила $\blacktriangleright\rangle \to \rangle$ на правило $\blacktriangleright\rangle \to \rangle\bullet$. Очевидно, для исчисления $\mathcal{C}'$ имеет место утверждение, аналогичное $(\star)$:

$$\textit{для любых } X \in \Sigma^*,\ Y \in \Delta^*:\ M(X) = Y \ \Leftrightarrow\ \langle q_0\widetilde{X}\rangle \vdash_{\mathcal{C}'} \langle Y\rangle \bullet\ .$$

В качестве искомого ассоциативного исчисления $\mathcal{C}''$ возьмём исчисление, которое получается из $\mathcal{C}'$ добавлением для каждого правила вывода обратного. Каждое правило вывода исчисления $\mathcal{C}''$, являющееся правилом вывода исчисления $\mathcal{C}'$, в текущем доказательстве для краткости будем называть *прямым*. Докажем, что проблема эквивалентности для ассоциативного исчисления $\mathcal{C}''$ неразрешима. Для этого достаточно установить следующее утверждение:

$$\textit{для любых } X \in \Sigma^*,\ Y \in \Delta^*:\ \langle q_0\widetilde{X}\rangle \vdash_{\mathcal{C}''} \langle Y\rangle\bullet\ \Rightarrow\ \langle q_0\widetilde{X}\rangle \vdash_{\mathcal{C}'} \langle Y\rangle \bullet\ . \quad (\star\star)$$

Действительно, так как все правила исчисления $\mathcal{C}'$ являются правилами исчисления $\mathcal{C}''$, то имеет место импликация в обратную сторону, а тогда для исчисления $\mathcal{C}''$ справедливо утверждение, аналогичное $(\star)$.

По существу $(\star\star)$ утверждает, что из вывода $\langle q_0\widetilde{X}\rangle \vdash_{\mathcal{C}''} \langle Y\rangle\bullet$ в исчислении $\mathcal{C}''$ можно исключить все применения правил, обратных к прямым.

Доказательство утверждения ($\star\star$) проведём индукцией по длине вывода $\langle q_0\widetilde{X}\rangle \vdash_{\mathcal{C}''} \langle Y\rangle\bullet$. Пусть этот вывод имеет вид $Z_1, \ldots, Z_k$. При $k = 1$ доказываемое утверждение очевидно.

Пусть $k > 1$. При переходе от Z_{k-1} к Z_k применено прямое правило, поскольку $\bullet$ не порождается никаким правилом, обратным к прямому. Последнее применение правила, обратного к некоторому прямому правилу R_1, могло произойти лишь при переходе от Z_{l-1} к Z_l для некоторого $l < k$. Тогда Z_{l-1} получается из Z_l по прямому правилу R_1. Через R_2 обозначим прямое правило, применённое при переходе от Z_l к Z_{l+1}. Ситуацию можно наглядно представить так: $Z_{l-1} \overset{R_1}{\leftarrow\!\!-} Z_l \overset{R_2}{-\!\!\rightarrow} Z_{l+1}$. Прямые правила таковы, что ровно одно прямое правило применимо к Z_l, причём единственным способом (для прямых правил, моделирующих работу машины Тьюринга, мы указывали это требование, но тогда это легко видеть и для всех прямых правил исчисления $\mathcal{C}''$). Поэтому R_1 и R_2 — это одно и то же правило, и Z_{l-1} совпадает с Z_{l+1}. Таким образом, в выводе $Z_1, \ldots, Z_{l-1}, Z_l, Z_{l+1}, \ldots Z_k$ можно опустить Z_l, Z_{l+1} и тем самым избавиться от одного применения правила, обратного к прямому правилу R_1, затем к укороченному выводу применить индукционное предположение и получить искомый вывод в исчислении $\mathcal{C}'$. ◁

Упражнение 5.5.4. Дополните доказательство предыдущей теоремы, выписав правила вывода, соответствующие команде $qa \to rbR$ машины Тьюринга. ◁

В доказательстве теоремы 5.5.3 описано построение ассоциативного исчисления, для которого неразрешима такая проблема: по произвольным *непустым* словам X и Y в алфавите этого исчисления выяснить, эквивалентны X и Y или нет. Назовём эту проблему *проблемой эквивалентности непустых слов для данного ассоциативного исчисления.* В дальнейшем нам понадобится

Следствие 5.5.5. *Существует ассоциативное исчисление с неразрешимой проблемой эквивалентности непустых слов, причём во всех правилах вывода вида $A \leftrightarrow B$ этого исчисления ни A, ни B не пусто.*

В доказательстве теоремы 5.5.3 простота используемой вычислительной модели (машины Тьюринга) сыграла ключевую роль для моделирования работы машины Тьюринга с помощью ассоциативного исчисления. Использование более сложной (и, возможно, более удобной для программирования) вычислительной модели по крайней мере существенно затруднило бы поиск подобного доказательства.

Можно заметить, что определения нормального алгоритма и ассоциативного исчисления схожи. Это сходство не случайно. В 1947 г. А. А. Марков (и одновременно и независимо Э. Пост) впервые опубликовал пример ассоциативного исчисления с неразрешимой проблемой эквивалентности, причём доказательство А. А. Маркова основывалось на лямбда-исчислении (см. [27, 29]). Понятие нормального алгоритма возникло в процессе поиска более естественного изложения этого результата (см. [30]).

Упражнение 5.5.6. Докажите теорему 5.5.3, используя нормальные алгоритмы вместо машин Тьюринга. ◁

Упражнение 5.5.7. Докажите, что любая вычислимая по Тьюрингу функция нормально вычислима. *Указание.* В доказательстве теоремы 5.5.3 описано построение исчисления, моделирующего машину Тьюринга. Воспользуйтесь этим построением. ◁

Ассоциативное исчисление с неразрешимой проблемой эквивалентности, построение которого описано в доказательстве теоремы 5.5.3, имеет много правил вывода, поскольку оно моделирует работу универсальной машины Тьюринга, а такая машина довольно громоздка. Интересно отметить ассоциативное исчисление с малым числом правил вывода, для которого неразрешима проблема эквивалентности. Установлено (см. [47, 48]), что неразрешима проблема эквивалентности для ассоциативного исчисления в алфавите $\{a, b, c, d, e\}$ со следующими правилами:

$$ac \leftrightarrow ca, \quad ad \leftrightarrow da, \quad bc \leftrightarrow cb, \quad bd \leftrightarrow db,$$
$$eca \leftrightarrow ce, \quad edb \leftrightarrow de, \quad cca \leftrightarrow ccae.$$

5.5.2. Проблема общезначимости и проблема выводимости для исчисления предикатов

Определение 5.5.8. Пусть Ω — произвольный язык первого порядка, Σ_Ω — алфавит языка Ω. *Проблема общезначимости для языка первого порядка* Ω заключается в выяснении по любому слову A в алфавите Σ_Ω, является ли A общезначимой формулой языка Ω или нет. ◁

Эта проблема очень важна для представления знаний и проведения рассуждений с помощью компьютера. Как показывает следующая теорема, даже для довольно «бедного» языка первого порядка (лишь с одним двухместным функциональным символом и одним двухместным предикатным символом) не существует алгоритма, проверяющего, является ли произвольная заданная формула этого языка логическим законом или нет.

Обозначим через $\mathfrak{S}$ элементарную теорию полугрупп, а через $\Omega_\mathfrak{S}$ — язык этой теории. (Определения этой теории и полугруппы см. в примере 3.2.4 на с. 115.)

Теорема 5.5.9. *Неразрешима проблема общезначимости для языка* $\Omega_\mathfrak{S}$ *элементарной теории полугрупп.*

Доказательство. По теореме 5.5.5 существует ассоциативное исчисление $\mathcal{C}$ с неразрешимой проблемой эквивалентности непустых слов и правилами вывода

$$A_i \leftrightarrow B_i \quad (i = 1, \ldots, n), \qquad (*)$$

где для каждого $i = 1, \ldots, n$ ни A_i, ни B_i не пусто.

Пусть $\Sigma \leftrightharpoons \{a_1, \ldots, a_m\}$ — алфавит исчисления $\mathcal{C}$, $x_1, \ldots, x_m$ — попарно различные предметные переменные. Каждому символу a_i $(i = 1, \ldots, m)$ сопоставим переменную x_i. Каждому непустому слову $A \leftrightharpoons a_{j_1} a_{j_2} a_{j_3} \ldots a_{j_k}$ в алфавите Σ сопоставим терм

$$t_A \leftrightharpoons (\ldots((x_{j_1} \cdot x_{j_2}) \cdot x_{j_3}) \cdot \ldots \cdot x_{j_k})$$

языка $\Omega_\mathfrak{S}$.

Покажем, что *для любых непустых слов A и B в алфавите Σ верно: A и B эквивалентны (в $\mathcal{C}$) тогда и только тогда, когда формула*

$$\Psi_{A,B} \leftrightharpoons ((t_{A_1} = t_{B_1} \wedge \ldots \wedge t_{A_n} = t_{B_n}) \supset t_A = t_B)$$

истинна в любой модели теории $\mathfrak{S}$.

Пусть слова A и B эквивалентны. Тогда существует такая последовательность слов $Z_1, \ldots, Z_k$ $(k \geqslant 1)$, что $A = Z_1$, $Z_k = B$ и для каждого $l = 2, \ldots, k$ Z_l является результатом замены некоторого вхождения A_i в Z_{l-1} на B_i или результатом замены некоторого вхождения B_i в Z_{l-1} на A_i при некотором $i = 1, \ldots, n$. Возьмём любую модель M теории $\mathfrak{S}$ и любую оценку ν такие, что $M, \nu \vDash (t_{A_1} = t_{B_1} \wedge \ldots \wedge t_{A_n} = t_{B_n})$. Делая по сути те же замены, что и в последовательности $Z_1, \ldots, Z_k$, получаем цепочку равенств $t_A = t_{Z_1} = t_{Z_2} = \ldots = t_{Z_k} = t_B$, истинных в модели M при оценке ν. Поэтому $M, \nu \vDash \Psi_{A,B}$. Таким образом, $\Psi_{A,B}$ истинна в любой модели теории $\mathfrak{S}$ (при любой оценке).

Обратно, если $\Psi_{A,B}$ истинна в любой модели теории $\mathfrak{S}$, то $\Psi_{A,B}$ истинна в модели M, являющейся полугруппой с образующими $a_1, \ldots, a_m$ и определяющими соотношениями $(*)$, при такой оценке ν, что $\nu(x_i) = a_i$ $(i = 1, \ldots, m)$. Тогда $M, \nu \vDash (t_{A_1} = t_{B_1} \wedge \ldots \wedge t_{A_n} = t_{B_n})$ в силу $(*)$. Поэтому $M, \nu \vDash (t_A = t_B)$, следовательно, слова A и B эквивалентны. Таким образом, мы установили утверждение, выделенное курсивом в текущем доказательстве.

Зафиксируем какие угодно непустые слова A и B в алфавите Σ и продолжим использовать введённое выше обозначение формулы $\Psi_{A,B}$. Обозначим через Γ конъюнкцию всех собственных аксиом теории $\mathfrak{S}$ и положим $\widetilde{\Psi}_{A,B} \leftrightharpoons (\Gamma \supset \Psi_{A,B})$. По теореме 2.4.3 о логическом следствии $\Psi_{A,B}$ истинна в любой модели теории $\mathfrak{S}$, если и только если $\vDash \widetilde{\Psi}_{A,B}$. Итак, произвольные непустые слова A и B в алфавите Σ эквивалентны тогда и только тогда, когда $\vDash \widetilde{\Psi}_{A,B}$. Следовательно, проблема общезначимости для языка $\Omega_{\mathfrak{S}}$ неразрешима, поскольку иначе была бы разрешима проблема эквивалентности непустых слов для ассоциативного исчисления $\mathcal{C}$. $\triangleleft$

Определение 5.5.10. Пусть $\mathcal{C}$ — какое угодно исчисление в произвольном алфавите Σ. *Проблема выводимости для исчисления $\mathcal{C}$* заключается в выяснении по любому слову α в алфавите Σ, $\vdash_{\mathcal{C}} \alpha$ или нет. *Исчисление $\mathcal{C}$* называют *разрешимым*, если эта проблема разрешима, иначе — *неразрешимым*. $\triangleleft$

Общезначимость формулы языка первого порядка Ω равносильна выводимости этой формулы в исчислении предикатов гильбертовского типа $\mathcal{H}\forall(\Omega)$ в силу теорем о корректности и полноте этого исчисления. Поэтому из теоремы 5.5.9 получаем такое

Следствие 5.5.11. *Неразрешимо исчисление $\mathcal{H}\forall(\Omega_{\mathfrak{S}})$, где $\Omega_{\mathfrak{S}}$ — язык элементарной теории полугрупп.*

Замечание 5.5.12. Теорему 5.5.11 можно доказать и конструктивно. Для этого достаточно конструктивно установить утверждение, которое получается из утверждения, выделенного курсивом в доказательстве теоремы 5.5.9, если вместо истинности формулы $\Psi_{A,B}$ говорить о выводимости

$\Psi_{A,B}$ в элементарной теории полугрупп $\mathfrak{S}$. Одна импликация, являющаяся частью этого утверждения, доказывается несложно, что и предлагается сделать в следующем упражнении. ◁

Упражнение 5.5.13. Докажите, что если непустые слова A и B в алфавите Σ эквивалентны, то формула $\Psi_{A,B}$ выводима в $\mathfrak{S}$ (см. обозначения в доказательстве теоремы 5.5.9). Не опирайтесь на семантику языка элементарной теории полугрупп, а по любому выводу слова B из гипотезы A в ассоциативном исчислении опишите построение вывода формулы $\Psi_{A,B}$ в $\mathfrak{S}$. ◁

Для некоторых языков Ω исчисление $\mathcal{H}\forall(\Omega)$ разрешимо. Например, если в языке Ω предикатные символы лишь нульместные, то для любой формулы A этого языка путём составления конечной таблицы истинности можно определить, общезначима A или нет, и значит, выводима она или нет.

Упражнение 5.5.14. Пусть в сигнатуре языка первого порядка имеется конечное число n одноместных предикатных символов, нет других предикатных символов и нет функциональных символов. Докажите, что разрешима проблема общезначимости для этого языка. *Указание.* Сначала докажите, что любая выполнимая замкнутая формула этого языка истинна в некоторой интерпретации с носителем, содержащим не более 2^n элементов. ◁

Определение 5.5.15. Формальная аксиоматическая теория первого порядка T называется *разрешимой*, если множество всех замкнутых формул, выводимых в T, разрешимо; иначе T называется *неразрешимой*. ◁

Упражнение 5.5.16. Докажите, что теория разрешима, если и только если множество всех формул, выводимых в T, разрешимо. ◁

Из доказательства теоремы 5.5.9 нетрудно получить такое утверждение: *элементарная теория полугрупп неразрешима*.

Упражнение 5.5.17. Дайте более подробное доказательство этого утверждения. ◁

5.5.3. Перечислимость теорем теории

Пусть $T \leftrightharpoons \langle \Omega, \Gamma \rangle$ — какая угодно формальная аксиоматическая теория первого порядка, Σ_Ω — алфавит языка Ω. Займёмся вопросом о перечислимости множества всех теорем теории T. В частности, при пустом множестве Γ мы имеем вопрос о перечислимости множества всех теорем исчисления $\mathcal{H}\forall(\Omega)$. Как мы установим, множество всех теорем теории T действительно перечислимо, если принять естественные предположения. Естественно желать, чтобы правильность формального доказательства (вывода) в теории T могла быть проверена алгоритмом. Обсудим предположения, обеспечивающие выполнение этого желания.

Первое предположение состоит в том, что имеется алгоритм, который по любому слову в алфавите Σ_Ω проверяет, принадлежит ли это слово языку Ω или нет; т. е. это предположение о разрешимости языка Ω. Языки первого порядка, обычно используемые для задания (формальных аксиоматических) теорий, разрешимы. Разрешимость языка Ω, очевидно, влечёт разрешимость множества всех аксиом исчисления $\mathcal{H}\forall(\Omega)$.

Далее, пусть дана произвольная формула разрешимого языка Ω. Можно ли с помощью некоторого алгоритма определить, является ли она выводом (длины 1) в теории T или нет? Ясно, что для существования такого алгоритма необходимо и достаточно разрешимости множества Γ всех собственных аксиом теории T. Теперь наше второе предположение, включающее первое, можно сформулировать так: множества Ω и Γ разрешимы.

Вывод в теории T является словом, состоящим из формул, разделённых некоторым символом, не входящим в алфавит Σ_Ω. Будем считать, что в качестве такого разделителя выступает символ «;», и вывод является словом в алфавите $\Sigma_\Omega \cup \{;\}$. Если выполнено второе предположение, то силу того, что правила вывода MP и Gen исчисления $\mathcal{H}\forall(\Omega)$ — разрешимые отношения, существует алгоритм, который по любому слову в алфавите $\Sigma_\Omega \cup \{;\}$ проверяет, является это слово выводом в теории T или нет.

Упражнение 5.5.18. Разработайте такой алгоритм. ◁

Следующее определение обобщает только что обсуждавшиеся естественные предположения о теории.

Определение 5.5.19. Теория $T \leftrightharpoons \langle \Omega, \Gamma \rangle$ называется *рекурсивно аксиоматизируемой*, если язык Ω разрешим, и существует теория $T' \leftrightharpoons \langle \Omega, \Gamma' \rangle$ такая, что множества всех теорем теорий T и T' равны, и множество Γ' разрешимо (говоря иначе, рекурсивно). ◁

Пример 5.5.20. Языки элементарной арифметики, теории множеств Цермело-Френкеля, элементарной теории полугрупп разрешимы, множества всех собственных аксиом этих теорий разрешимы, так что эти теории рекурсивно аксиоматизируемы. ◁

Теорема 5.5.21. *Пусть T — рекурсивно аксиоматизируемая теория. Тогда множество всех теорем теории T перечислимо.*

Доказательство. Пусть Σ_Ω — алфавит языка теории T, символ «;» не входит в алфавит Σ_Ω. Определим двухместный предикат $Proof$ на множестве всех слов в алфавите $\Sigma_\Omega \cup \{;\}$, используемом для записи выводов в теории T. Положим, что $Proof(\alpha, \beta)$ есть истина, если и только если β является выводом в теории T формулы α. Из рассуждений, проведённых перед настоящей теоремой, следует, что предикат $Proof$ разрешим.

Тогда (по аналогу теоремы о проекции для словарных отношений, см. упражнение 5.2.21 на с. 206) одноместный предикат $\exists\beta\, Proof(\alpha, \beta)$ является перечислимым. Этот предикат истинен, если и только если α является теоремой теории T. Следовательно, множество всех теорем теории T перечислимо. ◁

Упражнение 5.5.22. Докажите, что множество всех теорем теории $\langle \Omega, \Gamma \rangle$ перечислимо, если язык Ω разрешим и множество Γ перечислимо. ◁

Упражнение 5.5.23. Докажите, что полная рекурсивно аксиоматизируемая теория разрешима. ◁

5.5.4. Теоремы Гёделя о неполноте

Пусть имеются какие угодно язык первого порядка Ω и его интерпретация M. С целью формализовать получение истинных в M формул языка Ω

вводят некоторую формальную аксиоматическую теорию T, для которой M является моделью. Непротиворечивость теории T равносильна существованию модели этой теории и является естественным требованием к теории T (см. раздел 3.1, в частности, лемму 3.1.6). Определяя теорию T, естественно желать, чтобы в точности все истинные в M формулы языка Ω были выводимы в T. Упражнение 3.1.8 на с. 112 показывает,[9] что это желание выполнимо, если и только если теория T полна.

Рассматриваемая в текущем разделе теорема Гёделя о неполноте элементарной арифметики PA утверждает, что теория PA не является одновременно непротиворечивой и полной; поэтому указанное желание невыполнимо для теории PA.

Подготавливаясь к доказательству этой теоремы, мы введём понятие эффективно неотделимых множеств, относящееся непосредственно к теории вычислимости.

Эффективно неотделимые множества

Пусть $W_x \leftrightharpoons dom(\varphi_x)$. Поскольку по теореме 5.2.8 перечислимые множества натуральных чисел есть в точности области определения одноместных числовых вычислимых функций, то $W_0, W_1, \ldots$ — нумерация всех перечислимых множеств натуральных чисел.

Определение 5.5.24. Непересекающиеся множества $A \subseteq \mathbb{N}$ и $B \subseteq \mathbb{N}$ называются *эффективно неотделимыми*, если существует вычислимая функция $f : \mathbb{N}^2 \to \mathbb{N}$ такая, что для любых $u, v \in \mathbb{N}$ из $A \subseteq W_u$, $B \subseteq W_v$ и $W_u \cap W_v = \varnothing$ следует, что $f(u, v)$ определено и $f(u, v) \notin W_u \cup W_v$. ◁

Упражнение 5.5.25. Докажите, что каждое из двух эффективно неотделимых множеств является неразрешимым. ◁

Теорема 5.5.26. *Множества*

$$K_0 \leftrightharpoons \{x \in \mathbb{N} \mid \varphi_x(x) = 0\} \quad u \quad K_1 \leftrightharpoons \{x \in \mathbb{N} \mid \varphi_x(x) = 1\}$$

перечислимы и эффективно неотделимы.

Доказательство. Легко видеть, что K_0 и K_1 перечислимы. Докажем эффективную неотделимость этих множеств. Сначала отметим, что K_0 и K_1 не пересекаются.

Определим трёхместную функцию g, которая вычисляется следующим алгоритмом со входом $u, v, x \in \mathbb{N}$: одновременно перечислять элементы множеств W_u и W_v,[10] причём если порождён элемент множества W_u, совпадающий с x, то выдать 1, иначе, если порождён элемент множества W_v, совпадающий с x, то выдать 0.

По следствию 5.3.7 теоремы о параметризации существует вычислимая функция $f : \mathbb{N}^2 \to \mathbb{N}$ такая, что для любых $u, v \in \mathbb{N}$ $\lambda x.g(u, v, x) = \varphi_{f(u,v)}$. Зафиксируем произвольные u и v, для которых $K_0 \subseteq W_u$, $K_1 \subseteq W_v$ и

[9] См. также упражнение 2.6.32 на с. 94.

[10] Для детализации такого одновременного перечисления можно воспользоваться идеей из доказательства теоремы 5.2.8.

$W_u \cap W_v = \varnothing$. Для завершения доказательства достаточно показать, что $f(u,v) \notin W_u \cup W_v$. Для данных u, v и любого x имеем

$$g(u,v,x) = \varphi_{f(u,v)}(x) = \begin{cases} 1, & \text{если } x \in W_u, \\ 0, & \text{если } x \in W_v, \\ \text{не определено} & \text{в остальных случаях.} \end{cases}$$

Если бы $f(u,v) \in W_u$, то $\varphi_{f(u,v)}(f(u,v)) = 1$, тогда $f(u,v) \in K_1 \subseteq W_v$, и мы получили бы противоречие с $W_u \cap W_v = \varnothing$. Аналогично $f(u,v) \in W_v$ привело бы к противоречию. Поэтому $f(u,v) \notin W_u \cup W_v$. ◁

Теорема Гёделя о неполноте и теорема о неразрешимости арифметики

Для любого $m \in \mathbb{N}$ мы обозначаем через $\widetilde{m}$ такой терм $SS\ldots S0$ языка Ar элементарной арифметики PA, что символ S входит ровно m раз в этот терм.

Пусть $x_1, \ldots, x_n$ — попарно различные предметные переменные, A — формула, $t_1, \ldots, t_n$ — термы. Через $[A]^{x_1,\ldots,x_n}_{t_1,\ldots,t_n}$ обозначим результат одновременной замены всех свободных вхождений переменных $x_1, \ldots, x_n$ в A на $t_1, \ldots, t_n$ соответственно.[11]

Определение 5.5.27. Пусть Π — n-местный числовой предикат; $x_1, \ldots, x_n$ — список попарно различных предметных переменных; A — формула языка Ar элементарной арифметики PA, все параметры A входят в этот список. Говорят, что *формула A выражает в теории PA предикат* $\Pi(x_1, \ldots, x_n)$, если для любых $m_1, \ldots, m_n \in \mathbb{N}$

1) если $\Pi(m_1, \ldots, m_n)$ истинно, то $PA \vdash [A]^{x_1,\ldots,x_n}_{\widetilde{m}_1,\ldots,\widetilde{m}_n}$, и
2) если $\Pi(m_1, \ldots, m_n)$ ложно, то $PA \vdash \neg[A]^{x_1,\ldots,x_n}_{\widetilde{m}_1,\ldots,\widetilde{m}_n}$.

Числовой предикат $\Pi(x_1, \ldots, x_n)$ называется *выразимым в теории PA*, если существует формула, которая выражает в PA предикат $\Pi(x_1, \ldots, x_n)$. ◁

Пример 5.5.28. Покажем, что двухместный предикат $Eq(x,y)$ на $\mathbb{N}$, определённый как «x равно y», выразим в PA формулой $x = y$.

1) Если $Eq(m,n)$ есть истина, то по собственной аксиоме $E1$ $\forall x(x = x)$ теории PA имеем $PA \vdash \widetilde{m} = \widetilde{n}$.
2) Индукцией по n докажем ($\star$): ложность $Eq(m,n)$ влечёт $PA \vdash \widetilde{m} \neq \widetilde{n}$.

   База индукции. n равно[12] 0. В этом случае утверждение ($\star$) верно по собственной аксиоме $P1$ $\forall x(Sx \neq 0)$ теории PA.

   Индукционный переход. Предположим, что ($\star$) верно при n, равном k. Рассмотрим ($\star$) при n, равном $k+1$. При m, равном 0, ($\star$) верно по

[11] Мы вводим такое обозначение результата замены, напоминающее обозначение из определения 2.3.1, но отличное от обозначения из определения 4.3.2, поскольку здесь оно представляется нам прагматичным.

[12] Мы намеренно пишем здесь слово «равно», а не обозначение предиката равенства $=$, чтобы знак $=$ в этом доказательстве употреблялся лишь как предикатный символ языка Ar.

аксиоме $P1$. Иначе m равно $j+1$ для некоторого j. Если $Eq(m, k+1)$ ложно, то $Eq(j,k)$ ложно, откуда по индукционному предположению $PA \vdash \widetilde{j} \neq \widetilde{k}$. Тогда по аксиоме $P2$ $\forall x \forall y(Sx = Sy \supset x = y)$ получаем $PA \vdash S\widetilde{j} \neq S\widetilde{k}$, т. е. $PA \vdash \widetilde{m} \neq \widetilde{k+1}$. $\triangleleft$

Упражнение 5.5.29. Докажите, что числовой предикат «x меньше y» выразúм в PA формулой $\exists z(y = x + Sz)$. $\triangleleft$

Лемма 5.5.30. *Любой разрешимый числовой предикат выразúм в теории PA.*

Мы не будем излагать довольно длинное доказательство леммы 5.5.30. Эта лемма показывает, что элементарная арифметика обладает богатыми выразительными возможностями.

Упражнение 5.5.31. Докажите, что если теория PA непротиворечива, то любой выразимый в PA предикат разрешим. $\triangleleft$

Множества K_0 и K_1, определённые в теореме 5.5.26, перечислимы, следовательно, по теореме 5.2.17 K_0 и K_1 являются проекциями некоторых разрешимых подмножеств $\mathbb{N}^2$. Поэтому для каждого $i = 0, 1$ найдётся двухместный числовой разрешимый предикат Π_i такой, что для любого $m \in \mathbb{N}$

$$m \in K_i \quad \Leftrightarrow \quad \text{существует такое } n, \text{ что истинно } \Pi_i(m,n). \qquad (*)$$

Поскольку $K_0 \cap K_1 = \varnothing$, то

(0) $m \in K_0 \quad \Leftrightarrow \quad$ существует такое n, что истинно $\Pi_0(m,n)$ и для любого $l \leqslant n$ ложно $\Pi_1(m,l)$,

(1) $m \in K_1 \quad \Leftrightarrow \quad$ существует такое n, что истинно $\Pi_1(m,n)$ и для любого $l \leqslant n$ ложно $\Pi_0(m,l)$.

По лемме 5.5.30 для каждого $i = 0, 1$ предикат $\Pi_i(x,y)$ выразúм в PA некоторой формулой A_i. (Следовательно, все параметры формулы A_i входят в список x, y.) Пусть

$$B_0 \leftrightharpoons \exists y(A_0 \wedge \forall w(w \leqslant y \supset \neg[A_1]^y_w)),$$
$$B_1 \leftrightharpoons \exists y(A_1 \wedge \forall w(w \leqslant y \supset \neg[A_0]^y_w)),$$

где $w \leqslant y \leftrightharpoons \exists z(y = w + z)$. Формулы B_0 и B_1 содержательно согласованы с (0) и (1) соответственно, точнее, для $i = 0, 1$, стандартной интерпретации ω языка Ar и любой оценки ν имеем: $\omega, \nu \vDash B_i$, если и только если $\nu(x) \in K_i$.

Лемма 5.5.32. *Для любого $m \in \mathbb{N}$*

(a) *если $m \in K_0$, то $PA \vdash [B_0]^x_{\widetilde{m}}$;*

(b) *если $m \in K_1$, то $PA \vdash [B_1]^x_{\widetilde{m}}$;*

(c) *если $PA \vdash [B_1]^x_{\widetilde{m}}$, то $PA \vdash \neg[B_0]^x_{\widetilde{m}}$.*

Мы не приводим доказательство леммы 5.5.32. Построение такого доказательства является хорошим упражнением на вывод в PA.

Упражнение 5.5.33. Докажите лемму 5.5.32. $\triangleleft$

Замечание 5.5.34. Из $(*)$ и выражения в PA предиката $\Pi_i(x,y)$ формулой A_i $(i=0,1)$ следует, что если $m \in K_i$, то существует такое n, что $PA \vdash [A_i]^{x,y}_{\widetilde{m},\widetilde{n}}$. Отсюда по $\exists$-введению получаем $PA \vdash \exists y [A_i]^x_{\widetilde{m}}$. Поэтому в пункте (a) леммы 5.5.32 вместо B_0 можно взять и $\exists y A_0$; аналогично и в пункте (b). Однако формулы B_i $(i=0,1)$ выбраны так, чтобы обеспечивался пункт (c) этой леммы. $\triangleleft$

Теорема 5.5.35 ((первая) теорема Гёделя о неполноте арифметики (в форме Россера))**.** *Если теория PA непротиворечива, то PA неполна.*

Доказательство. Пусть

$$Pr \leftrightharpoons \{m \in \mathbb{N} \mid PA \vdash [B_0]^x_{\widetilde{m}}\}, \qquad Ref \leftrightharpoons \{m \in \mathbb{N} \mid PA \vdash \neg[B_0]^x_{\widetilde{m}}\}.$$

$Pr \cap Ref = \varnothing$ в силу непротиворечивости PA. По лемме 5.5.32 $K_0 \subseteq Pr$ и $K_1 \subseteq Ref$.

Из теоремы 5.5.21 следует, что множества Pr и Ref перечислимы, поэтому найдутся такие $u, v \in \mathbb{N}$, что $Pr = W_u$ и $Ref = W_v$. В силу эффективной неотделимости множеств K_0 и K_1 существует вычислимая функция f и натуральное число $k = f(u,v) \notin W_u \cup W_v$, при котором $PA \nvdash [B_0]^x_{\widetilde{k}}$ и $PA \nvdash \neg[B_0]^x_{\widetilde{k}}$. Следовательно, PA неполна. $\triangleleft$

В доказательстве теоремы 5.5.35 указана замкнутая формула $\neg[B_0]^x_{\widetilde{k}}$, которая невыводима в теории PA (если PA непротиворечива). Но эта формула истинна в модели ω теории PA. Действительно, $\omega \vDash \neg[B_0]^x_{\widetilde{k}}$, поскольку

а) как видно из доказательства теоремы 5.5.35, $k \notin K_0$, и

б) $\omega, \nu \vDash B_0$, если и только если оценка ν такова, что $\nu(x) \in K_0$.

Теорема 5.5.36. *Если теория PA непротиворечива, то PA неразрешима.*

Доказательство. В доказательстве теоремы 5.5.35 определены множества Pr и Ref такие, что $K_0 \subseteq Pr$, $K_1 \subseteq Ref$ и (в силу непротиворечивости PA) $Pr \cap Ref = \varnothing$. Следовательно, $K_1 \subseteq Ref \subseteq \mathbb{N} \setminus Pr$.

Предположим, что теория PA разрешима. Тогда множество Pr разрешимо. Следовательно, множества Pr и $\mathbb{N} \setminus Pr$ перечислимы, и потому найдутся такие $u, v \in \mathbb{N}$, что $Pr = W_u$ и $\mathbb{N} \setminus Pr = W_v$. В силу эффективной неотделимости K_0 и K_1 существует вычислимая функция f и натуральное число $k = f(u,v) \notin W_u \cup W_v = \mathbb{N}$. Полученное противоречие (натуральное число $k \notin \mathbb{N}$) доказывает неразрешимость теории PA. $\triangleleft$

Вторая теорема Гёделя о неполноте

Формализуем понятие вывода в теории PA в рамках самой теории PA. С этой целью определим биекцию γ из множества $\mathbb{S}$ всех конечных последовательностей формул языка Ar на $\mathbb{N}$ и вычислимую нумерацию γ^{-1} множества $\mathbb{S}$; это можно сделать аналогично тому, как были занумерованы машины Тьюринга в разделе 5.3.1. Для каждой последовательности $\mathcal{S} \in \mathbb{S}$ $\gamma(\mathcal{S})$ мы назовём *гёделевым номером* (или просто *номером*) этой последовательности. Для любого m, являющегося номером некоторой формулы (т. е.

последовательности из одной формулы), через Φ_m будем обозначать эту формулу.

Определим двухместный числовой предикат Π так, что для любых $m, n \in \mathbb{N}$ имеет место следующее: $\Pi(m, n)$ есть истина, если и только если m является номером некоторой формулы Φ_m, и n является номером вывода формулы $[\Phi_m]^x_{\widetilde{m}}$. Легко видеть, что предикат Π разрешим, следовательно, по лемме 5.5.30 найдётся формула A, которая выражает в PA предикат $\Pi(x, y)$. (Следовательно, все параметры формулы A входят в список x, y.)

Обозначим через k номер формулы $\neg\exists y A$. Рассмотрим замкнутую формулу

$$\Psi \leftrightharpoons \neg\exists y [A]^x_{\widetilde{k}},$$

которая есть $[\Phi_k]^x_{\widetilde{k}}$ и содержательно (в модели ω) выражает то, что сама формула Ψ невыводима в PA.

Лемма 5.5.37. *Если теория PA непротиворечива, то $PA \nvdash \Psi$.*

Доказательство. Предположим, что $PA \vdash \Psi$. Тогда существует номер l вывода формулы Ψ. В силу определения предиката $\Pi(x, y)$ истинно $\Pi(k, l)$. Поскольку формула A выражает предикат $\Pi(x, y)$, имеем $PA \vdash [A]^{x,y}_{\widetilde{k},\widetilde{l}}$. По $\exists$-введению получаем $PA \vdash \exists y [A]^x_{\widetilde{k}}$, что вместе со сделанным предположением $PA \vdash \Psi$ противоречит непротиворечивости PA. Поэтому $PA \nvdash \Psi$. ◁

Заметим, что в силу леммы 5.5.37, если теория PA непротиворечива, то формула Ψ невыводима в PA, но истинна в модели ω этой теории, так как Ψ содержательно выражает то, что Ψ невыводима в PA.

По лемме 3.1.6 непротиворечивость теории PA равносильна существованию невыводимой в PA формулы языка Ar. Поскольку $PA \vdash S0 \neq 0$ (см. упражнение 3.3.3 на с. 122), то для непротиворечивости PA необходимо и достаточно $PA \nvdash S0 = 0$. Обозначим через m номер формулы $S0 = 0$. Тогда непротиворечивость PA содержательно выражается формулой

$$Consis \leftrightharpoons \neg\exists y [A]^x_{\widetilde{m}}.$$

Вспомним, что невыводимость в PA формулы Ψ содержательно выражается формулой Ψ. Доказательство леммы 5.5.37 является содержательным доказательством того, что непротиворечивость PA влечёт невыводимость в PA формулы Ψ. Это доказательство можно формализовать в теории PA, тем самым записав вывод формулы $Consis \supset \Psi$ в теории PA (мы не будем проводить эту формализацию). Если бы $PA \vdash Consis$, то отсюда и из $PA \vdash Consis \supset \Psi$ получили бы $PA \vdash \Psi$, что по лемме 5.5.37 невозможно, если PA непротиворечива. Таким образом, имеет место следующая

Теорема 5.5.38 (вторая теорема Гёделя о неполноте арифметики)**.** *Если теория PA непротиворечива, то $PA \nvdash Consis$.*

По сути эта теорема утверждает, что если теория PA непротиворечива, то доказательство непротиворечивости PA не может быть проведено средствами, формализуемыми в PA.

Непротиворечивость PA можно доказать, например, средствами теории множеств ZFC при условии, что ZFC непротиворечива. В самом деле,

PA непротиворечива по теореме 2.6.20, поскольку существует модель ω этой теории. Однако формализация этого доказательства в ZFC сведёт вопрос о непротиворечивости PA к вопросу о непротиворечивости ZFC. А непротиворечивость ZFC до сих пор не установлена (см. раздел 3.5.5). (В том же разделе 3.5.5 намечено другое сведение вопроса непротиворечивости PA к вопросу о непротиворечивости ZFC.)

Теоремы 5.5.35, 5.5.36, 5.5.38 имеют место не только для теории PA, но и для любой рекурсивно аксиоматизируемой теории T, в естественном смысле содержащей теорию PA. Мы не будем точно определять, в каком смысле T содержит PA. Например, в качестве T, очевидно, может выступать любая рекурсивно аксиоматизируемая теория, которая получается добавлением к PA новых предикатных, функциональных символов или собственных аксиом. Также в качестве T могут выступать теории ZF и ZFC.

Глава 6.

Исчисление Хоара для доказательства корректности программ

В этой главе определяется язык программирования и исчисление, с помощью которого можно формально доказывать правильность программ, написанных на данном языке программирования.

6.1. Подходы к верификации программ

Проблема качества компьютерных программ всегда стоит остро. Одним из путей обеспечения качества программы является её верификация, т. е. проверка её корректности (правильности). Под корректностью программы понимается соответствие этой программы установленным для неё требованиям, или её спецификации.

Тестирование программы — один из наиболее распространённых подходов к верификации. При тестировании программа исполняется конечное число раз и результаты её работы проверяются на соответствие спецификации этой программы.

Пусть дана программа, на вход которой подаются три 32-битовых целых числа, и результатом работы которой должно являться наибольшее из этих чисел. Чтобы лишь с помощью тестирования показать корректность этой программы, необходимо провести исчерпывающее тестирование: проверить результаты работы программы на всех допустимых входных данных. Таким образом, нужно исполнить программу $(2^{32})^3 > 10^{28}$ раз. Даже если предположить, что в нашем распоряжении имеется суперкомпьютер, на котором мы можем исполнять программу 10^{15} раз в секунду, то тестирование займёт более 10^{13} секунд, что составляет более 3000 веков. Итак, исчерпывающее тестирование практически неосуществимо.

Если мы отказываемся от исчерпывающего тестирования, то приходим к следующему выводу. Тестирование может обнаружить, что программа не удовлетворяет спецификации, но, вообще говоря, не может гарантировать, что программа удовлетворяет ей. (Вместе с тем мы не отрицаем, что в про-

мышленном программировании во многих случаях тестирование различных видов оказывается целесообразным для верификации программ.)

Если же построить математическую модель системы, исполняющей программу, и изложить спецификацию программы на некотором формальном языке, то можно попытаться доказать, что данная программа удовлетворяет спецификации. В этом и заключается так называемая формальная верификация программы. Основам одного из подходов к формальной верификации посвящена настоящая глава.

Мы зададим простой язык программирования, сходный с подмножеством практически любого современного императивного языка программирования высокого уровня. Для программы, написанной на этом языке, мы определим её операционную семантику — то, как исполняется программа некоторой виртуальной машиной (см. также начало раздела 4.5.3). Спецификация программы будет представлять собой две предикатные формулы — условия на переменные этой программы до и после её исполнения. Наконец, мы приведём исчисление, в котором выводимые программы будут корректными по отношению к своим спецификациям.

6.2. Программа и её операционная семантика

Определим программу с помощью формы Бэкуса-Наура (БНФ) — аппарата, часто используемого для задания синтаксиса языков программирования. *Программа* представляет собой инструкцию, возможно, составную. Инструкция задаётся следующими правилами, конец каждого из которых обозначается точкой.

```
Инструкция ::= ИнструкцияПрисваивания |
               СоставнаяИнструкция |
               СокращённаяУсловнаяИнструкция |
               ПолнаяУсловнаяИнструкция |
               ЦиклическаяИнструкция.
ИнструкцияПрисваивания ::= ПредметнаяПеременная ':=' Терм.
СоставнаяИнструкция ::= Инструкция ';' Инструкция.
СокращённаяУсловнаяИнструкция ::=
                            'if' Условие 'then' Инструкция 'fi'.
ПолнаяУсловнаяИнструкция ::= 'if' Условие 'then' Инструкция
                              'else' Инструкция 'fi'.
ЦиклическаяИнструкция ::= 'while' Условие 'do' Инструкция 'od'.
Условие ::= БесквантornаяФормула.
```

Знак `::=` читается как «есть», знак `|` — как «или». В апострофах записаны слова, которые могут входить в программу непосредственно.

Отметим, что `ПредметнаяПеременная`, `Терм` и `БесквантornаяФормула` пока остались неопределёнными. Зададим язык первого порядка Ω, сигнатура которого дополняет сигнатуру языка элементарной арифметики (см. раздел 2.2.2) двухместным функциональным символом $\hat{}$. Мы будем сокращённо записывать $t_1\hat{}t_2$ как $t_1{}^{t_2}$. Теперь по определению положим, что `ПредметнаяПеременная`, `Терм` и `БесквантornаяФормула` являются предметной переменной, термом и бескванторной формулой языка Ω соответственно.

Циклическую инструкцию короче называют *циклом*, а инструкцию S — *телом цикла* `while` B `do` S `od`.

Наконец, условимся, что в любом месте программы может находиться комментарий, ограниченный с двух сторон фигурными скобками { и }, причём внутри такой пары фигурных скобок не может стоять никакая фигурная скобка. На этом определение синтаксиса рассматриваемого языка программирования завершено.

Определим *операционную семантику* какой угодно *программы* S. Зададим интерпретацию M языка Ω, носителем которой является множество $\mathbb{N}$, и которая расширяет стандартную интерпретацию языка элементарной арифметики, сопоставляя функциональному символу $\hat{}$ функцию возведения в степень, причём (как и в [7]) полагаем $0^0 = 1$.

Опишем, как программа S исполняется некоторой виртуальной машиной, тем самым определив эту виртуальную машину. Эта виртуальная машина последовательно исполняет инструкции программы, пропуская комментарии, и поддерживает текущее *состояние программы* θ, являющееся оценкой языка Ω в интерпретации M, т. е. отображением из множества всех предметных переменных в $\mathbb{N}$. $\theta(x)$ будем называть текущим *значением переменной* x. (Для краткости изложения мы выбрали такую абстракцию, в которой состояние программы есть оценка, хотя, как несложно будет увидеть, в качестве состояния программы достаточно использовать значения конечного числа переменных.)

Предполагается, что до начала исполнения программы некоторым переменным сопоставляются значения путём построения начального состояния этой программы. Мы не указываем, каким образом строится начальное состояние: это выходит за рамки определяемой нами операционной семантики. Но мы не будем безосновательно опираться на то, что переменные изначально имеют определённые значения.

Исполнение каждой инструкции может изменить состояние программы. Далее для каждой инструкции S мы определим, как в результате исполнения инструкции S, начатого в состоянии θ (для краткости — исполнения инструкции S из состояния θ), происходит переход в состояние $S(\theta)$, если это исполнение завершается.

1. Исполнение инструкции присваивания $x := t$ из состояния θ заключается в вычислении значения $|t| \leftharpoons |t|_{M,\theta}$ терма t и переходе в состояние $\theta^x_{|t|}$ (см. определения 2.2.7 и 2.2.8 на с. 67).

2. Исполнение составной инструкции $S_1; S_2$ из состояния θ производится так. Начинает исполняться инструкция S_1. Если исполнение S_1 не завершается, то и исполнение этой составной инструкции не завершается, иначе исполнение S_1 завершается в состоянии $S_1(\theta)$. Затем в состоянии $S_1(\theta)$ начинает исполняться S_2, и результат исполнения S_2 является результатом исполнения этой составной инструкции. (Здесь и далее в этом разделе 6.2 мы говорим, что результат исполнения инструкции S' является результатом исполнения инструкции S, естественным образом понимая под этим, что если исполнение S' не завершается, то исполнение S не завершается, а если исполнение S' завершается в некотором состоянии θ', то исполнение S завершается в том же состоянии θ'.)

3. При исполнении сокращённой условной инструкции `if` B `then` S `fi` из состояния θ сначала вычисляется истинностное значение $|B| \leftharpoons |B|_{M,\theta}$

формулы B (см. определение 2.2.9 на с. 67). Затем, если $|B|$ есть истина, то результат исполнения S является результатом исполнения этой условной инструкции, иначе исполнение этой условной инструкции завершается в том же состоянии θ.

4. При исполнении из состояния θ полной условной инструкции `if` B `then` S_1 `else` S_2 `fi` сначала вычисляется истинностное значения $|B| \leftrightharpoons |B|_{M,\theta}$ формулы B. Затем, если $|B|$ есть истина, то результат исполнения S_1 является результатом исполнения этой условной инструкции, иначе результат исполнения S_2 является результатом исполнения этой условной инструкции.

5. При исполнении цикла `while` B `do` S `od` из состояния θ сначала вычисляется истинностное значение $|B| \leftrightharpoons |B|_{M,\theta}$ формулы B. Затем, если $|B|$ есть ложь, то исполнение этого цикла завершается в том же состоянии θ, иначе начинает исполняться S. Если исполнение S не завершается, то и исполнение этого цикла не завершается, иначе исполнение S завершается в состоянии $S(\theta)$. Тогда результат исполнения этого цикла из состояния $S(\theta)$ является результатом исполнения этого цикла из состояния θ (иначе говоря, происходит повторение исполнения S, пока условие B истинно).

Определение операционной семантики программы закончено. Из этого определения следует, что исполнение инструкции может не завершиться, только если она является циклом или включает в себя некоторый цикл.

В этой главе для термов $S0$, $SS0$, $SSS0$, ... языка Ω мы будем иногда использовать сокращения 1, 2, 3, ... соответственно (из контекста будет понятно, имеются ли в виду именно такие термы или же натуральные числа). Например, при этом соглашении инструкция присваивания $x := S0$ записывается в виде $x := 1$, более привычном для знакомых с программированием.

Пример 6.2.1. Пусть дана следующая программа Exp с переменными x, n, k, exp. Слева от каждой инструкции мы поставили (в виде комментария) её номер.

$\{1\}$ $k := 0$;
$\{2\}$ $exp := 1$;
$\{3\}$ `while` $k \neq n$ `do`
$\{4\}$ $\quad exp := exp \cdot x$;
$\{5\}$ $\quad k := k + 1$
`od`

Если в программу входят только попарно различные переменные $x_1, \ldots, x_n$, то состояние θ этой программы такое, что $\theta(x_1) = t_1$, ..., $\theta(x_n) = t_n$, мы будем записывать в виде $\{t_1/x_1, \ldots, t_n/x_n\}$. (Мы знаем, что значения прочих переменных не влияют на исполнение программы.)

Продемонстрируем, как изменяется состояние этой программы, если её исполнение начинается в состоянии $\{3/x,\ 2/n,\ 594/k,\ 47/exp\}$. Исполнение инструкции 1 вызывает переход в состояние $\{3/x,\ 2/n,\ 0/k,\ 47/exp\}$, инструкции 2 — в состояние $\{3/x,\ 2/n,\ 0/k,\ 1/exp\}$. При исполнении циклической инструкции 3 сначала исполняется инструкция 4 и происходит переход в состояние $\{3/x,\ 2/n,\ 0/k,\ 3/exp\}$, а затем — инструкция 5 и происходит переход в состояние $\{3/x,\ 2/n,\ 1/k,\ 3/exp\}$. Далее циклическая инструкция 3 продолжает исполняться: исполняется инструкция 4 с переходом в состояние $\{3/x,\ 2/n,\ 1/k,\ 9/exp\}$, а затем — инструкция 5 с переходом в состояние $\theta_f \leftrightharpoons \{3/x,\ 2/n,\ 2/k,\ 9/exp\}$. Наконец, поскольку истинностное

значение формулы $k \neq n$ в текущем состоянии θ_f становится ложным, исполнение цикла 3 завершается, и исполнение всей программы завершается в состоянии θ_f. ◁

Упражнение 6.2.2. В каком состоянии завершится начатое в состоянии $\{0/x,\ 3/n,\ 0/k,\ 0/exp\}$ исполнение программы, приведённой в предыдущем примере? ◁

6.3. Аксиоматическая семантика программы

Сначала обрисуем подход к формальной верификации, который будем изучать в данном разделе. Пусть S — инструкция в программе $Prog$. Напомним, что программа является инструкцией, поэтому S может совпадать с $Prog$. Спецификация инструкции S будет представлять собой две предикатные формулы Pre и $Post$, которые выражают некоторые условия относительно значений переменных программы $Prog$ до и после исполнения инструкции S соответственно. Мы (пока грубо) скажем, что инструкция S частично корректна относительно своей спецификации, если для любых значений переменных, удовлетворяющих условию Pre, после исполнения S (если оно завершается) значения переменных удовлетворяют условию $Post$. Мы сформулируем так называемое исчисления Хóара, в котором можно выводить спецификацию программы из спецификаций составляющих её инструкций и тем самым доказывать частичную корректность программы относительно её спецификации. Говорят, что в рамках этого подхода к формальной верификации посредством исчисления задаётся *аксиоматическая* (или *деривационная*) *семантика программы*.

Теперь перейдём к определению понятия частичной корректности инструкции и формулировке исчисления Хоара. Напомним, что язык первого порядка Ω и его интерпретация M заданы в начале раздела 6.2.

Определение 6.3.1. *Триадой Хоара* называется слово вида

$$\{Pre\}\ S\ \{Post\},$$

где S — инструкция, Pre и $Post$ — формулы языка Ω, называемые *предусловием* и *постусловием* инструкции S соответственно. ◁

Определение 6.3.2. Пусть Pre и $Post$ — формулы языка Ω. Говорят, что инструкция (программа) S *частично корректна* относительно предусловия Pre и постусловия $Post$, если для любых состояний ν и ξ верно следующее: если $M, \nu \vDash Pre$ и исполнение инструкции S из состояния ν завершается в состоянии ξ, то $M, \xi \vDash Post$. В этом случае для краткости будем также говорить, что триада Хоара $\{Pre\}\ S\ \{Post\}$ *частично корректна*. ◁

Замечание 6.3.3. В определении частичной корректности программы не требуется, чтобы исполнение этой программы завершалось. Например, программа `while` $x = x$ `do` $x := x$ `od` частично корректна относительно любых предусловий и постусловий, поскольку её исполнение (из какого угодно начального состояния) не завершается. ◁

Язык *исчисления Хоара* содержит всевозможные триады Хоара и формулы языка Ω.

Аксиомами этого исчисления являются все формулы языка Ω, которые истинны в интерпретации M (при любой оценке), а также все триады вида

$$\{[[A]]_t^x\}\ x := t\ \{A\},$$

где x — произвольная переменная, t — произвольный терм, A — произвольная формула языка Ω, $[[A]]_t^x$ — результат правильной подстановки t вместо x в A (см. определение 2.3.15 на с. 74). Любая триада такого вида называется *аксиомой присваивания*. Аксиома присваивания формализует операционную семантику инструкции присваивания; как установит теорема 6.3.7, аксиома присваивания частично корректна.

Правила вывода исчисления Хоара таковы (здесь A, $\widetilde{B}$, C, I — любые формулы языка Ω; B — любая бескванторная формула языка Ω; S, S_1, S_2 — любые инструкции):

$$\frac{A \supset \widetilde{B};\quad \{\widetilde{B}\}\ S\ \{C\}}{\{A\}\ S\ \{C\}}\ (SPRE),\qquad \frac{\{A\}\ S\ \{\widetilde{B}\};\quad \widetilde{B} \supset C}{\{A\}\ S\ \{C\}}\ (WPOST),$$

$$\frac{\{A\}\ S_1\ \{\widetilde{B}\};\quad \{\widetilde{B}\}\ S_2\ \{C\}}{\{A\}\ S_1; S_2\ \{C\}}\ (COMP),$$

$$\frac{\{A \wedge B\}\ S\ \{C\};\quad A \wedge \neg B \supset C}{\{A\}\ \texttt{if}\ B\ \texttt{then}\ S\ \texttt{fi}\ \{C\}}\ (IF),$$

$$\frac{\{A \wedge B\}\ S_1\ \{C\};\quad \{A \wedge \neg B\}\ S_2\ \{C\}}{\{A\}\ \texttt{if}\ B\ \texttt{then}\ S_1\ \texttt{else}\ S_2\ \texttt{fi}\ \{C\}}\ (IFELSE),$$

$$\frac{\{I \wedge B\}\ S\ \{I\}}{\{I\}\ \texttt{while}\ B\ \texttt{do}\ S\ \texttt{od}\ \{I \wedge \neg B\}}\ (WHILE).$$

Программистская интуиция подсказывает нам, что эти правила вывода формализуют операционную семантику инструкций; как мы увидим в доказательстве теоремы 6.3.7, из частично корректных триад и истинных в M формул по этим правилам получаются только частично корректные триады-заключения. Справа от каждого правила вывода в скобках указано его обозначение. Правила $SPRE$ и $WPOST$ называются *правилами усиления предусловия* и *ослабления постусловия* соответственно. В правиле $WHILE$ формула I называется *инвариантом цикла* (это название объясняется тем, что содержательно формула I является условием, которое неизменно истинно в текущем состоянии до и после каждого исполнения тела S цикла `while` B `do` S `od`).

Пример 6.3.4. Рассмотрим программу Exp из примера 6.2.1 и докажем, что триада $\{x \neq 0\}\ Exp\ \{exp = x^n\}$ выводима в исчислении Хоара.

Будем обозначать инструкции 1, 2, 3, 4 и 5 программы Exp через S_1, S_2, S_3, S_4 и S_5 соответственно. При этом $S_1; S_2; S_3$ есть Exp, а $S_4; S_5$ — тело цикла S_3.

Поиск доказательства выводимости (и поиск вывода) в исчислени Хоара удобно осуществлять, пытаясь найти по постусловию инструкции «наименее ограничительное» (иначе говоря, слабейшее) предусловие и начав с последней инструкции программы, не вложенной в другую инструкцию. Удобство такого способа объясняется в первую очередь тем, что в соответствии с аксиомой присваивания по постусловию инструкции присваивания легко найти предусловие, но не наоборот.

Последней инструкцией S_3 программы Exp является цикл, и его постусловием должна быть формула $exp = x^n$. Для применения правила $WHILE$ достаточно найти инвариант I этого цикла. Попробуем в качестве I взять $exp = x^k$. Если мы докажем выводимость триады

$$\{exp = x^k \wedge k \neq n\}\ S_4; S_5\ \{exp = x^k\}, \qquad (*)$$

то тогда по правилу $WHILE$ получим $\{exp = x^k\}\ S_3\ \{exp = x^k \wedge \neg(k \neq n)\}$. Отсюда, поскольку $M \vDash (exp = x^k \wedge \neg(k \neq n)) \supset exp = x^n$, по правилу $WPOST$ получим $\{exp = x^k\}\ S_3\ \{exp = x^n\}$.

Займёмся доказательством выводимости триады $(*)$, которое можно представлять себе как доказательство леммы. Согласно аксиоме присваивания предусловием инструкции S_5 с постусловием $exp = x^k$ является $exp = x^{k+1}$. Далее, согласно той же аксиоме предусловием инструкции S_4 с постусловием $exp = x^{k+1}$ является $exp \cdot x = x^{k+1}$. Так как $M \vDash (exp = x^k \wedge k \neq n) \supset exp \cdot x = x^{k+1}$, то по правилу $SPRE$ в качестве предусловия инструкции S_4 можно взять и $exp = x^k \wedge k \neq n$. Таким образом, доказана выводимость триады $(*)$. Тогда, как уже было показано, выводима триада $\{exp = x^k\}\ S_3\ \{exp = x^n\}$.

Теперь переходим к инструкции S_2. Её постусловие есть предусловие цикла S_3, т. е. формула $exp = x^k$. Согласно аксиоме присваивания предусловием инструкции S_2 с постусловием $exp = x^k$ является $1 = x^k$. Далее, согласно той же аксиоме предусловием инструкции S_1 с постусловием $1 = x^k$ является $1 = x^0$. Поскольку $M \vDash x \neq 0 \supset 1 = x^0$, то по правилу $SPRE$ можно усилить предусловие $1 = x^0$, взяв $x \neq 0$ в качестве предусловия инструкции S_1. Таким образом, доказательство выводимости триады $\{x \neq 0\}\ Exp\ \{exp = x^n\}$ завершено.

Заметим, что $M \vDash \mathbf{T} \supset 1 = x^0$, и потому предусловие инструкции S_1 можно усилить по правилу $SPRE$ до $\mathbf{T}$, тем самым доказав выводимость триады $\{\mathbf{T}\}\ Exp\ \{exp = x^n\}$. (Предусловие $\mathbf{T}$ не накладывает никаких условий на начальное состояние.) Также поясним, что для сокращения вышеприведенного доказательства применение правила $COMP$ не оговаривалось: предусловие инструкции непосредственно рассматривалось как постусловие предыдущей инструкции.

Проведённое доказательство можно компактно представить с помощью комментариев в программе:

$\{x \neq 0\}\{\supset\}\{1 = x^0\}$
$\{1\}\ k := 0;\ \{1 = x^k\}$
$\{2\}\ exp := 1;\ \{exp = x^k\}$
$\{3\}$ `while` $k \neq n$ `do` $\{exp = x^k \wedge k \neq n\}\{\supset\}\{exp \cdot x = x^{k+1}\}$
$\{4\}$ $\quad exp := exp \cdot x;\ \{exp = x^{k+1}\}$
$\{5\}$ $\quad k := k + 1\ \{exp = x^k\}$
`od` $\{exp = x^k \wedge \neg(k \neq n)\}\{\supset\}\{exp = x^n\}$

Заменив в вышеприведённой программе первый комментарий $\{x \neq 0\}$ на $\{\mathbf{T}\}$, получим компактную запись доказательства частичной корректности этой программы относительно предусловия $\mathbf{T}$ и постусловия $exp = x^n$.

Приведём и формальный вывод триады $\{x \neq 0\}\ Exp\ \{exp = x^n\}$:

1) $\{exp = x^{k+1}\}\ S_5\ \{exp = x^k\}$ — аксиома присваивания;

2) $\{exp \cdot x = x^{k+1}\}\ S_4\ \{exp = x^{k+1}\}$ — аксиома присваивания;

3) $(exp = x^k \wedge k \neq n) \supset exp \cdot x = x^{k+1}$ — аксиома (истинная в M формула);

4) $\{exp = x^k \wedge k \neq n\}\ S_4\ \{exp = x^{k+1}\}$ из 3, 2 по $SPRE$;

5) $\{exp = x^k \wedge k \neq n\}\ S_4; S_5\ \{exp = x^k\}$ из 4, 1 по $COMP$;

6) $\{exp = x^k\}\ S_3\ \{exp = x^k \wedge \neg(k \neq n)\}$ из 5 по $WHILE$;

7) $(exp = x^k \wedge \neg(k \neq n)) \supset exp = x^n$ — аксиома (истинная в M формула);

8) $\{exp = x^k\}\ S_3\ \{exp = x^n\}$ из 6, 7 по $WPOST$;

9) $\{1 = x^k\}\ S_2\ \{exp = x^k\}$ — аксиома присваивания;

10) $\{1 = x^k\}\ S_2; S_3\ \{exp = x^n\}$ из 9, 8 по $COMP$;

11) $\{1 = x^0\}\ S_1\ \{1 = x^k\}$ — аксиома присваивания;

12) $x \neq 0 \supset 1 = x^0$ — аксиома (истинная в M формула);

13) $\{x \neq 0\}\ S_1\ \{1 = x^k\}$ из 12, 11 по $SPRE$;

14) $\{x \neq 0\}\ S_1; S_2; S_3\ \{exp = x^n\}$ из 13, 10 по $COMP$.

Вывод триады $\{\mathbf{T}\}\ Exp\ \{exp = x^n\}$ можно получить из предыдущего вывода, заменив 12, 13 и 14 на $12'$, $13'$ и $14'$ соответственно:

$12'$) $\mathbf{T} \supset 1 = x^0$ — аксиома (истинная в M формула);

$13'$) $\{\mathbf{T}\}\ S_1\ \{1 = x^k\}$ из $12'$, 11 по $SPRE$;

$14'$) $\{\mathbf{T}\}\ S_1; S_2; S_3\ \{exp = x^n\}$ из $13'$, 10 по $COMP$. ◁

Замечание 6.3.5. Вместо того, чтобы включать в число аксиом исчисления Хоара все формулы языка Ω, истинные в интерпретации M, можно добавить в это исчисление все аксиомы и правила вывода исчисления $\mathcal{H}\forall(\Omega)$, все собственные аксиомы элементарной арифметики и следующие аксиомы для функционального символа $\hat{}$: $\forall x(x^0 = S0)$, $\forall x \forall y(x^{Sy} = x^y \cdot x)$. При этом получается исчисление, не опирающееся на семантику языка Ω, в отличие от сформулированного выше варианта исчисления Хоара. ◁

Упражнение 6.3.6. Выводимы ли в этих двух исчислениях одни и те же триады Хоара? ◁

Следующая теорема показывает, как связана выводимость в исчислении Хоара с частичной корректностью, определённой в терминах операционной семантики.

Теорема 6.3.7 (корректность исчисления Хоара относительно операционной семантики программы)**.** *Любая триада Хоара, выводимая в исчислении Хоара, частично корректна.*

Доказательство проведём индукцией по длине вывода триады.

Для установления базы индукции достаточно показать, что частично корректна любая аксиома присваивания вида $\{[[A]]^x_t\}\ S\ \{A\}$, где S есть $x := t$. Действительно, пусть даны произвольные состояния ν и ξ, верно $M, \nu \vDash [[A]]^x_t$ и исполнение инструкции S из состояния ν завершается в состоянии ξ. Покажем, что $M, \xi \vDash A$.

Согласно операционной семантике инструкции присваивания имеем $\xi = \nu^x_{|t|}$, где $|t| \leftrightharpoons |t|_{M,\nu}$. По лемме 2.4.17 $M, \nu \vDash [[A]]^x_t$ влечёт $M, \nu^x_{|t|} \vDash A$. Таким образом, $M, \nu^x_{|t|} \vDash A$, что и требовалось, поскольку $\xi = \nu^x_{|t|}$.

Для установления индукционного перехода достаточно показать, что для всякого правила вывода исчисления Хоара, если каждая посылка частично корректна или является истинной в M формулой, то заключение частично корректно. Рассмотрим, например, правило IF, а рассмотрение остальных правил оставим в качестве упражнения.

Пусть $\{A \wedge B\}\ S\ \{C\}$ частично корректна и $M \vDash A \wedge \neg B \supset C$. Покажем, что частично корректна $\{A\}\ ifS\ \{C\}$, где ifS есть `if` B `then` S `fi`. Для этого зададим какие угодно состояния ν и ξ и предположим, что $M, \nu \vDash A$ и исполнение инструкции ifS из состояния ν завершается в состоянии ξ. Требуется доказать, что $M, \xi \vDash C$.

Если $M, \nu \vDash B$, то (согласно операционной семантике инструкции ifS) исполнение инструкции S из состояния ν завершается в состоянии ξ. Тогда из $M, \nu \vDash A$, $M, \nu \vDash B$ и из частичной корректности $\{A \wedge B\}\ S\ \{C\}$ следует $M, \xi \vDash C$.

Если же $M, \nu \nvDash B$, то исполнение инструкции ifS из состояния ν завершается в том же состоянии, поэтому $\nu = \xi$. Тогда из $M, \nu \vDash A$, $M, \nu \nvDash B$ и $M \vDash A \wedge \neg B \supset C$ следует $M, \xi \vDash C$. ◁

Упражнение 6.3.8. Закончите индукционный переход в доказательстве предыдущей теоремы, разобрав оставшиеся правила вывода. ◁

Упражнение 6.3.9. Задайте операционную семантику нескольких инструкций, встречающихся во многих языках программирования, например, `case`, `for`, `repeat until` в языке Pascal или `switch`, `for`, `do while` в языках C-семейства. Сформулируйте правила вывода, задающие аксиоматическую семантику этих инструкций. Для каждого из этих правил докажите, что при переходе от посылок к заключению частичная корректность триад Хоара сохраняется. ◁

Упражнение 6.3.10. Дополним сигнатуру языка Ω двумя одноместными функциональными символами $div2$ и $mod2$. Расширим интерпретацию M языка Ω, сопоставив функциональным символам $div2$ и $mod2$ функции, вычисляющие, соответственно, частное и остаток от (целочисленного) деления на 2. Формально докажите частичную корректность следующей программы (с переменными x, n, p, q, exp) относительно предусловия $\mathbf{T}$ и постусловия $exp = x^n$.

$exp := 1$;
$p := x$;
$q := n$;
`while` $q \neq 0$ `do`
  `if` $mod2(q) = 1$ `then` $exp := exp \cdot p$ `fi`;
  $p := p \cdot p$;

$q := div2(q)$
`od` ◁

Упражнение 6.3.11. Специфицируйте программу нахождения наибольшего общего делителя двух натуральных чисел, задав предусловие и постусловие. Затем напишите такую программу и формально докажите её частичную корректность относительно составленной ранее спецификации. При необходимости расширьте язык Ω. ◁

В заключение данного раздела упомянем о некоторых проблемах и расширениях рассмотренного подхода к формальной верификации.

I. Нахождение инварианта цикла является, вообще говоря, нелёгкой задачей. Формула $\mathbf{T}$ — инвариант любого цикла, но этот тривиальный инвариант обычно не отражает существенных свойств цикла и не подходит для получения вывода триады, специфицирующей программу. Автоматическое нахождение инвариантов циклов является предметом исследований.

II. Нетрудно проследить, что если мы выберем любой другой язык первого порядка в качестве Ω или любую другую интерпретацию этого языка в качестве M, то теорема о корректности исчисления Хоара будет иметь место. Мы не вели изложение в столь общем виде лишь с целью большей наглядности, поскольку рассматриваемый язык программирования привычен для многих читателей, знакомящихся с основами программирования ещё в школе.

Упражнение 6.3.12. Пересмотрите доказательства частичной корректности программ из примера и предыдущих упражнений этого раздела, выбрав в качестве интерпретации языка Ω множество всех целых чисел с обычным образом определёнными арифметическими операциями над ними. (Если вместо указанной интерпретации рассматривать конечное множество компьютерных целых чисел с компьютерными арифметическими операциями, то такие задачи становятся гораздо менее тривиальными.) ◁

III. К настоящему времени создано много исчислений для доказательства корректности программ, включающих в себя массивы, записи, указатели, функции, подтипы и другие конструкции языков программирования.

Скажем чуть подробнее об исчислении, позволяющем доказывать не только частичную корректность, но и завершаемость исполнения программ, определённых нами в разделе 6.2.

Определение 6.3.13. Пусть Pre и $Post$ — формулы языка Ω. Инструкция (программа) S называется *тотально* (или *полностью*) *корректной* относительно предусловия Pre и постусловия $Post$, если для любого состояния ν существует состояние ξ такое, что верно следующее: если $M, \nu \vDash Pre$, то исполнение S из состояния ν завершается в состоянии ξ и $M, \xi \vDash Post$. В этом случае для краткости будем также говорить, что триада Хоара $\{Pre\}\ S\ \{Post\}$ *тотально* (или *полностью*) *корректна*. ◁

Упражнение 6.3.14. Покажите, что тотальная корректность триады Хоара влечёт её частичную корректность. ◁

Как отмечалось выше, исполнение рассматриваемых нами программ может не завершиться только по причине незавершаемости исполнения некоторого цикла. Ясно, что для содержательного доказательства завершаемости исполнения цикла достаточно найти такое выражение, которое принимает только натуральные значения, и значение которого уменьшается с каждым исполнением тела цикла. На основе таких соображений можно сформулировать новое правило вывода $FINWHILE$ для цикла:

$$\frac{[I]^n_{n+1} \supset B; \quad \{[I]^n_{n+1}\}\ S\ \{I\}; \quad [I]^n_0 \supset \neg B}{\{\exists n I\}\ \texttt{while}\ B\ \texttt{do}\ S\ \texttt{od}\ \{[I]^n_0\}},$$

где n — какая угодно переменная, не входящая в инструкцию S.

Заменив в сформулированном выше исчислении Хоара правило $WHILE$ на правило $FINWHILE$, мы получим исчисление для доказательства тотальной корректности программ.

Упражнение 6.3.15. Докажите, что любая триада, выводимая в полученном исчислении, тотально корректна. ◁

Упражнение 6.3.16. Докажите (как содержательно, так и формально) тотальную корректность программ из примера и предыдущих упражнений этого раздела. При необходимости расширьте язык Ω. ◁

Приложение А.

Свойства секвенциального исчисления предикатов и формальные аксиоматические теории на его основе

Исчисление предикатов гильбертовского типа является исторически первым исчислением для формализации математики, и формальные аксиоматические теории первого порядка традиционно строятся на основе этого исчисления. Однако представляет интерес и систематическое построение таких теорий на основе секвенциального исчисления предикатов генценовского типа, более удобного для поиска вывода, чем исчисление предикатов гильбертовского типа. В настоящем приложении мы изложим начала построения формальных аксиоматических теорий на основе секвенциального исчисления предикатов, предварительно изучив достаточный для такого построения ряд свойств этого исчисления.

А.1. Некоторые свойства секвенциального исчисления предикатов

В данном разделе мы установим некоторые свойства секвенциального исчисления предикатов с целью подготовки к построению формальных аксиоматических теорий на основе этого исчисления. Эти свойства относятся к теории доказательств (разделу математической логики, в котором исследуются формальные доказательства) и имеют собственную ценность. Здесь мы ограничимся изучением небольшого набора свойств, достаточного для доказательства базовых теорем о формальных аксиоматических теориях, построенных на основе секвенциального исчисления предикатов.

Зафиксируем произвольный язык первого порядка Ω, который будем рассматривать в текущем разделе А.1.

Определения правила и правила, допустимого в секвенциальном исчислении предикатов $\mathcal{G}\forall(\Omega)$, аналогичны соответствующим определениям для исчисления $\mathcal{G}$, данным в разделе 1.4.4.

А.1.1. Допустимость правил добавления

Правила добавления в антецедент и *сукцедент* соответственно имеют следующий вид:

$$\frac{\Gamma \to \Delta}{\Gamma, A \to \Delta}, \qquad \frac{\Gamma \to \Delta}{\Gamma \to \Delta, A},$$

где Γ, Δ — любые конечные списки формул языка Ω, A — любая формула языка Ω.

Теорема А.1.1. *Правила добавления допустимы в исчислении* $\mathcal{G}\forall(\Omega)$.

Д о к а з а т е л ь с т в о. Допустимость правила добавления в антецедент мы докажем индукцией по длине вывода посылки. (Допустимость правила добавления в сукцедент доказывается аналогично.)

Пусть выводима секвенция $\Gamma \to \Delta$. Установим, что выводима секвенция $\Gamma, A \to \Delta$.

База индукции. Секвенция $\Gamma \to \Delta$ является аксиомой. Тогда и секвенция $\Gamma, A \to \Delta$ является аксиомой.

Индукционный переход. Предположим, что из выводимости секвенции вида $\Gamma_0 \to \Delta_0$, имеющей вывод длины, меньшей n, следует выводимость секвенции $\Gamma_0, A_0 \to \Delta_0$. Рассмотрим секвенцию $\Gamma \to \Delta$, которая является последней в выводе длины n и получается по некоторому правилу вывода $\mathcal{R}$.

Пусть $\mathcal{R}$ является правилом $(\wedge \to)$, и секвенция $\Gamma \to \Delta$, имеющая вид $\Gamma_1, B \wedge C \to \Delta$, получена по правилу $\mathcal{R}$ из секвенции $\Gamma_1, B, C \to \Delta$. По индукционному предположению выводима секвенция $\Gamma_1, B, C, A \to \Delta$, из которой по правилу $\mathcal{R}$ получается секвенция $\Gamma, A \to \Delta$.

Пусть $\mathcal{R}$ является правилом $(\to \forall)$, и секвенция $\Gamma \to \Delta$, имеющая вид

$$\Gamma \to \Delta_1, \forall x B, \tag{$*$}$$

получена по правилу $\mathcal{R}$ из секвенции

$$\Gamma \to \Delta_1, [B]^x_y, \tag{$**$}$$

где переменная y не входит свободно в $(*)$ и свободна для x в B. Выберем переменную z, которая не входит ни в рассматриваемый вывод секвенции $(**)$, ни в формулу A. Заменим в этом выводе все свободные вхождения переменной y на z и в результате получим вывод секвенции $\Gamma \to \Delta_1, [B]^x_z$ (более подробное рассуждение остаётся в качестве упражнения). По индукционному предположению выводима секвенция $\Gamma, A \to \Delta_1, [B]^x_z$, из которой по правилу $\mathcal{R}$ получается секвенция $\Gamma, A \to \Delta$.

Каждое из оставшихся правил вывода, взятое в качестве $\mathcal{R}$, рассматривается аналогично одному из рассмотренных выше. ◁

Упражнение А.1.2. Проведите рассуждение, оставленное в доказательстве предыдущей теоремы в качестве упражнения. ◁

Упражнение А.1.3. Дополните доказательство предыдущей теоремы рассмотрением оставшихся правил вывода. ◁

А.1.2. Обратимость правил

Правилом, *обратным* к n-посылочному правилу вывода

$$\frac{S_1;\ldots;S_n}{S},$$

называется каждое из правил

$$\frac{S}{S_1},\quad \ldots,\quad \frac{S}{S_n}.$$

Правило вывода называется *обратимым*, если выводимость его заключения влечет выводимость каждой из посылок (т. е. каждое из правил, обратных к данному, допустимо).

Если правило вывода исчисления $\mathcal{G}\forall(\Omega)$ является правилом введения логического символа $\odot$ в антецедент (соответственно, в сукцедент), то каждое из правил, обратных к данному, называется правилом удаления логического символа $\odot$ из антецедента (соответственно, из сукцедента).

Перечислим правила, обратные к пропозициональным правилам вывода секвенциального исчисления:

$$\frac{\Gamma,\neg A\to\Delta}{\Gamma\to\Delta,A},\qquad \frac{\Gamma\to\Delta,\neg A}{\Gamma,A\to\Delta},$$

$$\frac{\Gamma,A\wedge B\to\Delta}{\Gamma,A,B\to\Delta},\qquad \frac{\Gamma\to\Delta,A\wedge B}{\Gamma\to\Delta,A},\ \frac{\Gamma\to\Delta,A\wedge B}{\Gamma\to\Delta,B},$$

$$\frac{\Gamma,A\vee B\to\Delta}{\Gamma,A\to\Delta},\ \frac{\Gamma,A\vee B\to\Delta}{\Gamma,B\to\Delta},\qquad \frac{\Gamma\to\Delta,A\vee B}{\Gamma\to\Delta,A,B},$$

$$\frac{\Gamma,A\supset B\to\Delta}{\Gamma\to\Delta,A},\ \frac{\Gamma,A\supset B\to\Delta}{\Gamma,B\to\Delta},\qquad \frac{\Gamma\to\Delta,A\supset B}{\Gamma,A\to\Delta,B},$$

где Γ, Δ — любые конечные списки формул языка Ω, A, B — любые формулы языка Ω.

Следующая теорема устанавливает допустимость каждого из этих правил.

Теорема А.1.4. *Каждое пропозициональное правило вывода исчисления* $\mathcal{G}\forall(\Omega)$ *обратимо.*

Доказательство. Обратимость каждого пропозиционального правила вывода мы докажем индукцией по длине вывода заключения этого правила.

Рассмотрим, например, правило $(\to\supset)$, а аналогичное рассмотрение остальных пропозициональных правил вывода останется в качестве упражнения. Пусть выводима секвенция $\Gamma\to\Delta,A\supset B$. Докажем, что выводима секвенция $\Gamma,A\to\Delta,B$.

База индукции. Секвенция $\Gamma\to\Delta,A\supset B$ является аксиомой.

Если в Γ и в Δ есть одна и та же формула, отличная от $A\supset B$, то секвенция $\Gamma,A\to\Delta,B$ является аксиомой.

Иначе Γ имеет вид $\Gamma_1,A\supset B$. Тогда секвенция $\Gamma,A\to\Delta,B$ (также записываемая как $\Gamma_1,A\supset B,A\to\Delta,B$) выводима, поскольку получается по правилу $(\supset\to)$ из аксиом $\Gamma_1,B,A\to\Delta,B$ и $\Gamma_1,A\to\Delta,B,A$.

Индукционный переход. Предположим, что из выводимости секвенции вида $\Gamma_0 \to \Delta_0, A_0 \supset B_0$, имеющей вывод длины, меньшей n, следует выводимость секвенции $\Gamma_0, A_0 \to \Delta_0, B_0$. Рассмотрим секвенцию $\Gamma \to \Delta, A \supset B$, которая является последней в выводе длины n и получается по некоторому правилу вывода $\mathcal{R}$.

Если подчёркнутый в секвенции член $\Gamma \to \Delta, \underline{A \supset B}$ является главным для рассматриваемого применения правила $\mathcal{R}$, то секвенция $\Gamma, A \to \Delta, B$ является посылкой этого применения и потому выводима.

В противном случае рассуждения аналогичны для каждого правила вывода $\mathcal{R}$. Пусть, например, $\mathcal{R}$ является правилом $(\exists \to)$, и секвенция $\Gamma \to \Delta, A \supset B$, имеющая вид $\Gamma_1, \exists x C \to \Delta, A \supset B$, получена по правилу $\mathcal{R}$ из секвенции $\Gamma_1, [C]^x_y \to \Delta, A \supset B$. По индукционному предположению выводима секвенция $\Gamma_1, [C]^x_y, A \to \Delta, B$, из которой по правилу $\mathcal{R}$ получается секвенция $\Gamma, A \to \Delta, B$. ◁

Упражнение А.1.5. Дополните доказательство предыдущей теоремы рассмотрением оставшихся пропозициональных правил вывода. ◁

Упражнение А.1.6. Докажите, что все кванторные правила вывода секвенциального исчисления предикатов обратимы. ◁

А.1.3. Допустимость правил, формализующих некоторые обычные способы математических рассуждений

Выводы в секвенциальном исчислении мало похожи на обычные математические рассуждения. Однако обычные способы математических рассуждений можно формализовать как допустимые в этом исчислении правила. С помощью таких правил мы сможем доказывать выводимость некоторых секвенций в этом исчислении, не строя их выводы полностью. (В разделах 1.3.3 и 2.6.2 мы поступали аналогично по отношению к формулам и исчислению гильбертовского типа.)

Приведём правила, которые (как мы скоро поясним) формализуют некоторые обычные способы математических рассуждений:

Связка $\odot$	$\odot$-введение	$\odot$-удаление
$\supset$	$\dfrac{\Gamma, A \to B}{\Gamma \to A \supset B}$,	$\dfrac{\Gamma \to A; \quad \Gamma \to A \supset B}{\Gamma \to B}$,
$\neg$	$\dfrac{\Gamma, A \to B; \quad \Gamma, A \to \neg B}{\Gamma \to \neg A}$,	$\dfrac{\Gamma \to \neg\neg A}{\Gamma \to A}$,
$\wedge$	$\dfrac{\Gamma \to A; \quad \Gamma \to B}{\Gamma \to A \wedge B}$,	$\dfrac{\Gamma \to A \wedge B}{\Gamma \to A}$, $\quad \dfrac{\Gamma \to A \wedge B}{\Gamma \to B}$,
$\vee$	$\dfrac{\Gamma \to A}{\Gamma \to A \vee B}$, $\quad \dfrac{\Gamma \to B}{\Gamma \to A \vee B}$,	$\dfrac{\Gamma, A \to C; \quad \Gamma, B \to C}{\Gamma, A \vee B \to C}$,

где Γ, Δ — любые конечные списки формул языка Ω, A, B, C — любые формулы языка Ω.

Докажем, что каждое из этих правил допустимо, т. е. из выводимости

всех посылок правила следует выводимость заключения этого правила.
Д о к а з а т е л ь с т в о. Допустимость $\supset$-введения очевидна, поскольку в исчислении $\mathcal{G}\forall(\Omega)$ имеется правило вывода $(\to\supset)$.

Установим допустимость $\supset$-удаления. Пусть выводимы секвенции

$$(*)\ \Gamma \to A \qquad \text{и} \qquad (**)\ \Gamma \to A \supset B.$$

В силу обратимости правила вывода $(\to\supset)$ выводимость секвенции $(**)$ влечёт выводимость секвенции $\Gamma, A \to B$. Применив допустимое правило сечения к $(*)$ и к $\Gamma, A \to B$, получим, что выводима требуемая секвенция $\Gamma \to B$.

Для доказательства допустимости $\neg$-введения из $\Gamma, A \to B$ и $\Gamma, A \to \neg B$ по правилу $(\to\supset)$ получим

$$(\star)\ \Gamma \to A \supset B \qquad \text{и} \qquad (\star\star)\ \Gamma \to A \supset \neg B.$$

Затем по $\supset$-удалению из выводимой (как легко проверить в качестве упражнения) секвенции

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

и $(\star)$ получим $\Gamma \to (A \supset \neg B) \supset \neg A$. Отсюда и из $(\star\star)$ по $\supset$-удалению получим $\Gamma \to \neg A$, что и требуется.

Далее, $\neg$-удаление допустимо: из $\Gamma \to \neg\neg A$ и выводимой секвенции $\Gamma \to \neg\neg A \supset A$ по $\supset$-удалению получим $\Gamma \to A$.

$\wedge$-введение допустимо: из $\Gamma \to A$, $\Gamma \to B$ и выводимой секвенции

$$\Gamma \to A \supset (B \supset (A \wedge B)),$$

дважды применив $\supset$-удаление, получим $\Gamma \to A \wedge B$.

Каждое правило, называемое $\wedge$-удалением, допустимо: из $\Gamma \to A \wedge B$ и выводимой секвенции $\Gamma \to (A \wedge B) \supset A$ по $\supset$-удалению получим $\Gamma \to A$; аналогично, из $\Gamma \to A \wedge B$ и выводимой секвенции $\Gamma \to (A \wedge B) \supset B$ по $\supset$-удалению получим $\Gamma \to B$.

Каждое правило, называемое $\vee$-введением, допустимо: из $\Gamma \to A$ и выводимой секвенции $\Gamma \to A \supset A \vee B$ по $\supset$-удалению получим $\Gamma \to A \vee B$; аналогично, из $\Gamma \to B$ и выводимой секвенции $\Gamma \to B \supset A \vee B$ по $\supset$-удалению получим $\Gamma \to A \vee B$.

Наконец, $\vee$-удаление допустимо, поскольку в исчислении $\mathcal{G}\forall(\Omega)$ имеется правило вывода $(\vee \to)$. $\triangleleft$

Упражнение А.1.7. Постройте выводы секвенций, выводимость которых необоснованно утверждалась в предыдущем доказательстве. $\triangleleft$

Вспомним (см. раздел 1.4.2), что секвенцию вида $A_1, \ldots, A_m \to B$ можно содержательно трактовать как формулу $A_1 \wedge \ldots \wedge A_m \supset B$. Тогда легко видеть, что допустимые правила, приведённые выше в этом разделе А.1.3, соответствуют некоторым обычным способам математических рассуждений.

Скажем, $\supset$-введение формализует такой способ доказательства того, что утверждение A влечёт утверждение B: предполагают, что верно A, и в этом предположении доказывают B. Утверждение о допустимости $\supset$-введения также называют *теоремой о дедукции* для рассматриваемого исчисления.

$\supset$-удаление формализует следующее общеупотребительное умозаключение: если имеет место A и известно, что A влечёт B, то имеет место B. $\supset$-удаление также назвают *модусом поненс* или *правилом отделения*.

$\neg$-введение соответствует способу рассуждения, называемому *приведением к абсурду* (*нелепости*, *противоречию*): чтобы доказать $\neg A$, достаточно допустить A и получить из этого допущения противоречие, т. е. для некоторого B получить одновременно B и $\neg B$.

$\vee$-удаление формализует способ доказательства *разбором случаев*: чтобы доказать, что из A или B следует C, достаточно доказать, что и из A следует C, и из B следует C.

Упражнение А.1.8. Докажите, что следующее правило, являющееся одним из вариантов $\vee$-удаления, допустимо:

$$\frac{\Gamma \to A \vee B; \quad \Gamma, A \to C; \quad \Gamma, B \to C}{\Gamma \to C}.$$

$\triangleleft$

Наконец, перечислим допустимые в исчислении $\mathcal{G}\forall(\Omega)$ правила, которые соответствуют некоторым обычным способам математических рассуждений со словами «для любого» и «существует»:

$$\frac{\Gamma \to [A]^x_y}{\Gamma \to \forall x A}\ (\forall\text{-введение}), \qquad \frac{\Gamma \to \forall x A}{\Gamma \to [A]^x_t}\ (\forall\text{-удаление}),$$

$$\frac{\Gamma \to [A]^x_t}{\Gamma \to \exists x A}\ (\exists\text{-введение}), \qquad \frac{\Gamma, [A]^x_y \to B}{\Gamma, \exists x A \to B}\ (\exists\text{-удаление}),$$

причём 1) в $\forall$-введении и $\exists$-удалении переменная y (являющаяся термом) свободна для переменной x в формуле A и не входит свободно в заключение ни одного из этих двух правил, 2) в $\forall$-удалении и $\exists$-введении терм t свободен для переменной x в формуле A.

Скажем, $\forall$-введение формализует следующее рассуждение: для доказательства утверждения вида $\forall x A$ достаточно выбрать новую переменную y и доказать, что имеет место $[A]^x_y$. $\exists$-удаление соответствует так называемому *правилу единичного выбора* (или *правилу* C — от слова «Choice»): чтобы доказать, что из $\exists x A$ следует B, обозначают через новую переменную y такой x, для которого A верно в силу $\exists x A$, и доказывают, что $[A]^x_y$ влечёт B.

$\forall$-введение действительно допустимо: из $\Gamma \to [A]^x_y$ по правилу $(\to \forall)$ получаем $\Gamma \to \forall x A$.

$\forall$-удаление также допустимо: из $\Gamma \to \forall x A$ и выводимой секвенции $\Gamma \to \forall x A \supset [A]^x_t$ по $\supset$-удалению получаем $\Gamma \to [A]^x_t$.

Упражнение А.1.9. Докажите допустимость $\exists$-введения и $\exists$-удаления. $\triangleleft$

Упражнение А.1.10. Докажите, что следующее правило, являющееся одним из вариантов $\exists$-удаления, допустимо:

$$\frac{\Gamma \to \exists x A; \quad \Gamma, [A]^x_y \to B}{\Gamma \to B},$$

где переменная y свободна для переменной x в формуле A и не входит свободно в заключение этого правила. $\triangleleft$

А.2. Формальные аксиоматические теории на основе секвенциального исчисления предикатов

В данном разделе мы заново определим некоторые понятия, связанные с формальными аксиоматическими теориями первого порядка, так, что для вывода в таких теориях вместо исчисления предикатов гильбертовского типа будет использоваться секвенциальное исчисление предикатов генценовского типа. При этом мы не будем повторять все определения из главы 3, которые сохраняют свою силу в настоящем разделе. Затем мы докажем несколько базовых теорем о свойствах теорий на основе секвенциального исчисления, не предполагая, что предварительно изучались разделы 1.3 и 2.6 об исчислении гильбертовского типа.

Мотивировка к введению в рассмотрение формальных аксиоматических теорий по сути аналогична приведённой в начале главы 3. С помощью вывода в секвенциальном исчислении предикатов $\mathcal{G}\forall(Ar)$ для языка элементарной арифметики Ar мы можем получать формулы, истинные в любой интерпретации этого языка. Как выводить формулы, выражающие свойства натуральных чисел (т. е. истинные в стандартной интерпретации ω языка Ar)? Например, формула $S(0)+0=S(0)$ языка Ar в интерпретации ω говорит о том, что сумма натуральных чисел 1 и 0 равна 1. Но эта формула невыводима в исчислении $\mathcal{G}\forall(Ar)$, поскольку она не является общезначимой. Опишем одно из свойств натуральных чисел с операцией сложения формулой $\forall x(x+0=x)$, которая истинна в интерпретации ω. Теперь формула $\forall x(x+0=x)\supset S(0)+0=S(0)$ выводима в исчислении $\mathcal{G}\forall(Ar)$ и, следовательно, общезначима. Тогда, так как $\omega\vDash\forall x(x+0=x)$, то и $\omega\vDash S(0)+0=S(0)$. Таким образом, с помощью вывода в исчислении $\mathcal{G}\forall(Ar)$ мы установили, что формула $S(0)+0=S(0)$ истинна в интерпретации ω, т. е. описывает некоторое свойство натуральных чисел.

Аналогичным образом мы будем поступать и для формального доказательства свойств других математических объектов. Сначала запишем базовые свойства рассматриваемых объектов на некотором языке первого порядка Ω. Этот язык и множество формул, выражающих эти базовые свойства, будут составлять формальную аксиоматическую теорию первого порядка. Теперь для формального доказательства формулы A языка Ω, выражающей свойство рассматриваемых объектов, будем устанавливать выводимость в исчислении $\mathcal{G}\forall(\Omega)$ формулы $G_1\wedge\ldots\wedge G_n\supset A$, где $G_1,\ldots,G_n$ — формулы, описывающие некоторые базовые свойства этих объектов.

А.2.1. Новые определения

Пусть дан произвольный язык первого порядка Ω.

Добавим в секвенциальное исчисление предикатов $\mathcal{G}\forall(\Omega)$ правило вывода (обозначаемое нами через $(\approx)$)

$$\frac{S_1}{S_2}\ (\approx),$$

где S_1 и S_2 — произвольные конгруэнтные секвенции (определение конгру-

энтных секвенций см. на с. 103). Таким образом, в полученном «новом» исчислении из вывода какой угодно заданной секвенции S одним применением этого правила можно получить вывод любой секвенции, конгруэнтной S.

Легко проследить, что все вышеприведённые в данной книге утверждения о «старом» исчислении $\mathcal{G}\forall(\Omega)$ (кроме перечня правил вывода этого исчисления), остаются в силе и для «нового» исчисления.

Ниже в разделе А.2 под секвенциальным исчислением предикатов (генценовского типа) $\mathcal{G}\forall(\Omega)$ мы будем понимать «новое» исчисление с правилом ($\approx$).

Сейчас теорема о полноте секвенциального исчисления предикатов может быть сформулирована так: *любая общезначимая секвенция выводима в этом исчислении.* Действительно, чистота секвенции, требовавшаяся в теореме 2.7.10 о полноте «старого» секвенциального исчисления предикатов, теперь необязательна, поскольку для любой общезначимой секвенции S мы можем найти чистую конгруэнтную ей секвенцию S_0, затем, воспользовавшись теоремой 2.7.10, получить вывод секвенции S_0 и наконец по правилу ($\approx$) получить вывод исходной секвенции S.

Кроме того, теперь для любой формулы её выводимость в секвенциальном исчислении предикатов равносильна её общезначимости и потому равносильна её выводимости в исчислении предикатов гильбертовского типа (см. также раздел 2.7.4). Теперь мы сможем заменить гильбертовское исчисление секвенциальным при выводе формул в формальной аксиоматической теории так, чтобы сохранить свойства таких теорий (напомним, традиционно основанных на исчислении предикатов гильбертовского типа).

Определение 3.1.1 формальной аксиоматической теории первого порядка (короче — теории) остаётся неизменным, так же как и другие определения из главы 3, замену которым мы не предлагаем в текущем разделе.

Определение А.2.1. Пусть $T \leftrightharpoons \langle \Omega, \Gamma \rangle$ — теория, A — формула языка Ω.

Формула A называется *теоремой теории* T (или *выводимой в теории* T), если существует такое множество $\{G_1, \ldots, G_n\} \subseteq \Gamma$, что в исчислении $\mathcal{G}\forall(\Omega)$ выводима формула $G_1 \wedge \ldots \wedge G_n \supset A$ (причём мы считаем, что конъюнкция нуля формул есть $\mathbf{T}$).

То, что A — теорема теории T, будем обозначать через $T \vdash A$. В противном случае будем писать $T \nvdash A$.

Вывод формулы $G_1 \wedge \ldots \wedge G_n \supset A$, где $\{G_1, \ldots, G_n\} \subseteq \Gamma$ будем называть *выводом в теории* T, а также, более конкретно, *выводом формулы* A *в теории* T. $\triangleleft$

Упражнение А.2.2. Докажите, что выводимость в секвенциальном исчислении предикатов формулы $G_1 \wedge \ldots \wedge G_n \supset A$ равносильна выводимости секвенции $G_1, \ldots, G_n \to A$. $\triangleleft$

Определение А.2.3. Теория T в языке Ω называется *непротиворечивой*, если не существует формулы A языка Ω такой, что $T \vdash A$ и $T \vdash \neg A$. Иначе T называется *противоречивой*. $\triangleleft$

Определение А.2.4. Теория T в языке Ω называется *полной*, если для любой замкнутой формулы A языка Ω имеем $T \vdash A$ или $T \vdash \neg A$. Иначе T называется *неполной*. $\triangleleft$

Определение А.2.5. Пусть T — теория в языке Ω. *Интерпретацией теории* T называется любая интерпретация языка Ω. *Моделью теории* T называется любая интерпретация языка Ω, в которой истинны все собственные аксиомы этой теории. $\triangleleft$

А.2.2. Базовые теоремы

В настоящем разделе мы докажем несколько базовых теорем о формальных аксиоматических теориях, основанных на секвенциальном исчислении предикатов. В частности, теорема А.2.15 покажет, что вывод в теории является адекватным способом получения формул, истинных в любой модели этой теории.

Теорема А.2.6 (теорема о корректности исчисления $\mathcal{G}\forall(\Omega)$ для теорий, или обобщённая теорема о корректности исчисления $\mathcal{G}\forall(\Omega)$)**.** *Пусть* $T \leftrightharpoons \langle \Omega, \Gamma \rangle$ *— теория,* A *— формула языка* Ω*. Тогда если* $T \vdash A$*, то* A *истинна в любой модели теории* T*.*

Доказательство. Согласно определению А.2.1, $T \vdash A$ означает, что существует такое множество $\{G_1, \ldots, G_n\} \subseteq \Gamma$, что выводима формула $G_1 \wedge \ldots \wedge G_n \supset A$. По теореме 2.7.7 о корректности секвенциального исчисления предикатов $\vDash G_1 \wedge \ldots \wedge G_n \supset A$.

Итак, существует такое множество $\{G_1, \ldots, G_n\} \subseteq \Gamma$, что в любой интерпретации языка Ω (при любой оценке) выполняется: если все формулы $G_1, \ldots, G_n$ истинны, то формула A истинна. Поскольку в любой модели теории T собственные аксиомы $G_1, \ldots, G_n$ этой теории истинны, то в любой модели теории T формула A истинна. $\triangleleft$

Теорема А.2.7. *Всякая теория, имеющая модель, непротиворечива.*

Доказательство. Пусть теория T имеет модель M. Если бы T была противоречива, то для некоторой формулы A выполнялось бы $T \vdash A$ и $T \vdash \neg A$. Тогда по теореме А.2.6 $M \vDash A$ и $M \vDash \neg A$, что невозможно. $\triangleleft$

Лемма А.2.8. *Пусть* T *— теория в языке* Ω*. Тогда* T *непротиворечива, если и только если существует невыводимая в* T *формула языка* Ω*.*

Доказательство. Очевидно, что если T непротиворечива, то существует невыводимая в T формула языка Ω. Чтобы доказать обратное утверждение, докажем утверждение, равносильное ему: если T противоречива, то любая формула языка Ω выводима в T.

Пусть теория T противоречива, т. е. существует формула A языка Ω такая, что $T \vdash A$ и $T \vdash \neg A$. Тогда (см. определение А.2.1 и упражнение А.2.2) выводимы секвенции $\Gamma_A \to A$ и $\Gamma_{\neg A} \to \neg A$ для некоторых конечных списков Γ_A и $\Gamma_{\neg A}$ собственных аксиом теории T. По допустимому правилу добавления в антецедент выводимы секвенции

$$\Gamma_A, \Gamma_{\neg A} \to A \qquad \text{и} \qquad \Gamma_A, \Gamma_{\neg A} \to \neg A.$$

Для любой формулы B языка Ω выводима секвенция

$$\Gamma_A, \Gamma_{\neg A} \to \neg A \supset (A \supset B).$$

Из трёх последних секвенций с помощью $\supset$-удаления получаем секвенцию $\Gamma_A, \Gamma_{\neg A} \to B$. Следовательно, $T \vdash B$. Таким образом, в противоречивой теории T выводима любая формула языка Ω. $\triangleleft$

Теорема А.2.9 (о существовании модели)**.** *Всякая непротиворечивая теория имеет счётную модель.*

Мы не приводим довольно длинное доказательство теоремы А.2.9. Эту теорему можно доказать аналогично теореме 2.6.21 (см. также теорему 2.6.28), используя допустимые в секвенциальном исчислении правила.

Упражнение А.2.10. Докажите теорему А.2.9. $\triangleleft$

Теорема А.2.11 (теорема Лёвенгейма-Скулема)**.** *Если теория имеет какую-нибудь модель, то эта теория имеет счётную модель.*

Доказательство. По теореме А.2.7 теория, имеющая модель, непротиворечива. Тогда по предыдущей теореме А.2.9 эта теория имеет счётную модель. $\triangleleft$

Теорема А.2.12 (теорема о компактности (для логики предикатов))**.** *Пусть $T \leftrightharpoons \langle \Omega, \Gamma \rangle$ — теория. Тогда если всякая теория $\langle \Omega, \Gamma' \rangle$, где Γ' — конечное подмножество множества Γ, имеет модель, то T имеет модель.*

Доказательство. Предположим, что теория T не имеет модели. Тогда по теореме А.2.9 о существовании модели T противоречива, т. е. существует формула A такая, что $T \vdash A$ и $T \vdash \neg A$.

Найдётся (объясните, почему) такое конечное множество $\Gamma_0 \subseteq \Gamma$ и теория $T_0 \leftrightharpoons \langle \Omega, \Gamma_0 \rangle$, что $T_0 \vdash A$ и $T_0 \vdash \neg A$. Таким образом, T_0 противоречива. По теореме А.2.7 теория T_0 не имеет модели, что противоречит условию доказываемой теоремы. Следовательно, T имеет модель. $\triangleleft$

Для доказательства очередной теоремы нам понадобится утверждение, которое мы сформулируем в следующем упражнении (являющемся аналогом упражнения 2.6.32).

Упражнение А.2.13. Пусть T — теория в языке Ω, A — формула языка Ω. Докажите, что тогда а) A истинна в любой модели теории T, если и только если $\mathbb{W}A$ истинна в любой модели теории T; и б) $T \vdash A$ равносильно $T \vdash \mathbb{W}A$. $\triangleleft$

Теорема А.2.14 (теорема о полноте исчисления $\mathcal{G}\forall(\Omega)$ для теорий, или обобщённая теорема о полноте исчисления $\mathcal{G}\forall(\Omega)$)**.** *Пусть $T \leftrightharpoons \langle \Omega, \Gamma \rangle$ — теория, A — формула языка Ω. Тогда если в любой модели теории T формула A истинна, то $T \vdash A$.*

Доказательство. Формулу A можно считать замкнутой в силу предыдущего упражнения.

Пусть в любой модели теории T замкнутая формула A истинна. Тогда теория $T' \leftrightharpoons \langle \Omega,\ \Gamma \cup \{\neg A\} \rangle$ не имеет модели. По теореме А.2.9 о существовании модели T' противоречива, т. е. существует формула B такая, что $T' \vdash B$

и $T' \vdash \neg B$. Отсюда с помощью $\neg$-введения получаем (детально объясните, как) $T \vdash \neg\neg A$, откуда по $\neg$-удалению $T \vdash A$. $\triangleleft$

Непосредственным следствием теорем А.2.6 и А.2.14 является следующий аналог теоремы 3.1.5.

Теорема А.2.15. *Пусть $T \leftrightharpoons \langle \Omega, \Gamma \rangle$ — теория, A — формула языка Ω. Тогда $T \vdash A$, если и только если A истинна в любой модели теории T.*

Дальнейшее построение формальных аксиоматических теорий, основанных на секвенциальном исчислении, может следовать главе 3, начиная сразу за леммой 3.1.6. В заключительном разделе А.2.3 мы укажем изменения, которые необходимо для этого внести.

А.2.3. Примеры содержательного и формального доказательств в теории

I. Одно из содержательных доказательств того, что в элементарной арифметике PA имеет место $SS0 + S0 = SSS0$, приведено в начале раздела 3.3.3. Это доказательство можно формализовать, записав вывод формулы $SS0 + S0 = SSS0$ в теории PA (теперь основанной на секвенциальном исчислении $\mathcal{G}\forall(Ar)$). Однако такая формализация несколько утомительна, и здесь мы её не проводим. Чтобы всё же получить более надёжное обоснование выводимости этой формулы в теории PA, мы дадим более подробное содержательное доказательство этой выводимости.

С помощью допустимых в секвенциальном исчислении $\mathcal{G}\forall(Ar)$ правил мы докажем, что выводима секвенция

$$\Gamma \to SS0 + S0 = SSS0, \qquad (*)$$

где Γ — список собственных аксиом $E3$, $E4$, $A1$, $A2$ теории PA. Тогда, в силу упражнения А.2.2, формула $E3 \wedge E4 \wedge A1 \wedge A2 \supset SS0 + S0 = SSS0$ выводима в исчислении $\mathcal{G}\forall(Ar)$, что означает выводимость формулы $SS0 + S0 = SSS0$ в теории PA.

Доказательство выводимости секвенции $(*)$ приведено ниже в пунктах 1–7.

1. Из аксиомы секвенциального исчисления $\mathcal{G}\forall(Ar)$ $\Gamma \to A2$, где $A2$ есть собственная аксиома теории PA $\forall x \forall y(x + Sy = S(x + y))$, дважды применяя $\forall$-удаление, получаем, что выводима секвенция
$$\Gamma \to SS0 + S0 = S(SS0 + 0). \qquad (1)$$

2. Из аксиомы исчисления $\Gamma \to A1$, где $A1$ есть собственная аксиома $\forall x(x + 0 = x)$, по $\forall$-удалению получаем секвенцию
$$\Gamma \to SS0 + 0 = SS0. \qquad (2)$$

3. Из аксиомы исчисления $\Gamma \to E4$, где $E4$ есть собственная аксиома $\forall x \forall y(x = y \supset Sx = Sy)$, дважды применяя $\forall$-удаление, получаем
$$\Gamma \to SS0 + 0 = SS0 \supset S(SS0 + 0) = SSS0. \qquad (3)$$

4. Из секвенций (2) и (3) по $\supset$-удалению получаем

$$\Gamma \to S(SS0+0) = SSS0. \tag{4}$$

5. Из секвенций (1) и (4) по $\wedge$-введению получаем

$$\Gamma \to SS0 + S0 = S(SS0+0) \wedge S(SS0+0) = SSS0. \tag{5}$$

6. Из аксиомы исчисления $\Gamma \to E3$, где $E3$ есть собственная аксиома $\forall x \forall y \forall z(x = y \wedge y = z \supset x = z)$, трижды применяя $\forall$-удаление, получаем

$$\Gamma \to SS0 + S0 = S(SS0+0) \wedge S(SS0+0) = SSS0 \supset SS0 + S0 = SSS0. \tag{6}$$

7. Из секвенций (5) и (6) по $\supset$-удалению получаем требуемую секвенцию $\Gamma \to SS0 + S0 = SSS0$.

Как видно, список Γ было необязательно подбирать заранее, а было можно постепенно конкретизировать его при поиске доказательства выводимости, считая, что в него входит потребовавшаяся собственная аксиома.

II. Теперь приведём пример формального доказательства в теории. Построим вывод формулы $S0 \neq 0$ в теории PA. Для этого найдём вывод формулы $P1 \supset S0 \neq 0$ в исчислении $\mathcal{G}\forall(Ar)$, где $P1$ есть собственная аксиома $\forall x(Sx \neq 0)$ теории PA:

1) $\to \forall x(Sx \neq 0) \supset S0 \neq 0$ из 2 по $(\to\supset)$;

2) $\forall x(Sx \neq 0) \to S0 \neq 0$ из 3 по $(\forall \to)$;

3) $\forall x(Sx \neq 0),\ S0 \neq 0 \to S0 \neq 0$ — аксиома исчисления $\mathcal{G}\forall(Ar)$.

III. Наконец, несложная адаптация некоторых рассуждений из разделов 3.3.2 и 5.5.2–5.5.4 к секвенциальному исчислению предикатов и теориям на его основе остаётся в качестве упражнений.

Литература

[1] Барендрегт Х. *Ламбда-исчисление. Его синтаксис и семантика.* М.: Мир, 1985.

[2] Братко И. *Алгоритмы искусственного интеллекта на языке PROLOG.* М.: Издательский дом «Вильямс», 2004.

[3] Верещагин Н. К., Шень А. *Лекции по математической логике и теории алгоритмов.* Часть 2. *Языки и исчисления.* М.: МЦНМО, 2002.

[4] Верещагин Н. К., Шень А. *Лекции по математической логике и теории алгоритмов.* Часть 3. *Вычислимые функции.* М.: МЦНМО, 2002.

[5] Гильберт Д., Бернайс П. *Основания математики. Логические исчисления и формализация арифметики.* М.: Наука, 1979.

[6] Гильберт Д., Бернайс П. *Основания математики. Теория доказательств.* М.: Наука, 1982.

[7] Грэхем Р., Кнут Д., Паташник О. *Конкретная математика. Основание информатики.* М.: Мир, 1998.

[8] Гэри М., Джонсон Д. *Вычислительные машины и труднорешаемые задачи.* М.: Мир, 1982.

[9] Ершов Ю. Л., Палютин Е. А. *Математическая логика: Учебное пособие.* СПб.: Издательство «Лань», 2004.

[10] Йех Т. *Теория множеств и метод форсинга.* М.: Мир, 1973.

[11] Катленд Н. *Вычислимость. Введение в теорию рекурсивных функций.* М.: Мир, 1983.

[12] Кейслер Г., Чэн Ч. Ч. *Теория моделей.* М.: Мир, 1977.

[13] Клини С. К. *Введение в метаматематику.* М.: Издательство иностранной литературы, 1957.

[14] Клини С. К. *Математическая логика.* М.: Мир, 1973.

[15] Ковальски Р. *Логика в решении проблем.* М.: Наука, 1990.

[16] Ковальски Р., Хайс П. Дж. *Семантические деревья в автоматическом поиске доказательств.* // Кибернетический сборник, новая серия, вып. 9. М.: Мир, 1972, с. 66–83.

[17] Колмогоров А. Н., Драгалин А. Г. *Математическая логика.* М.: Едиториал УРСС, 2004.

[18] Косовский Н. К. *Элементы математической логики и её приложения к теории субрекурсивных алгоритмов: Учеб. пособие.* Л.: Изд-во Ленингр. ун-та, 1981.

[19] Коэн П. Дж. *Теория множеств и континуум-гипотеза.* М.: Мир, 1969.

[20] Кузнецов О. П., Адельсон-Вельский Г. М. *Дискретная математика для инженера.* М.: Энергоатомиздат, 1988.

[21] Куратовский К., Мостовский А. *Теория множеств.* М.: Мир, 1970.

[22] Лавров И. А., Максимова Л. Л. *Задачи по теории множеств, математической логике и теории алгоритмов.* М.: ФИЗМАТЛИТ, 2003.

[23] Лавров С. С. *Программирование. Математические основы, средства, теория.* СПб.: БХВ-Петербург, 2001.

[24] Ловягин Ю. Н. *Элементарная математическая логика: Учебное пособие.* СПб.: Издательство РГПУ им. А. И. Герцена, 2007.

[25] *Логическое программирование.* М.: Мир, 1988.

[26] Мальцев А. И. *Алгоритмы и рекурсивные функции.* М.: Наука, 1965.

[27] Марков А. А. *Невозможность алгорифмов тождества и делимости в теории ассоциативных систем.* // ДАН СССР, Т. 55, N 7, 1947, с. 587–590.

[28] Марков А. А. *Избранные труды.* Т. I. *Математика, механика, физика.* М.: Изд-во МЦНМО, 2002.

[29] Марков А. А. *Избранные труды.* Т. II. *Теория алгорифмов и конструктивная математика, математическая логика, информатика и смежные вопросы.* М.: Изд-во МЦНМО, 2003.

[30] Марков А. А., Нагорный Н. М. *Теория алгорифмов.* М.: ФАЗИС, 1996.

[31] Мартыненко Б. К. *Языки и трансляции: Учеб. пособие.* СПб.: Издательство С.-Петербургского университета, 2002.

[32] Маслов С. Ю. *Теория дедуктивных систем и её применения.* М.: Радио и связь, 1986.

[33] *Математическая теория логического вывода.* Сборник переводов под редакцией А. В. Идельсона, Г. Е. Минца. М.: Наука, 1967.

[34] Мендельсон Э. *Введение в математическую логику.* М.: Наука, 1984.

[35] Непейвода Н. Н. *Прикладная логика.* Новосибирск: Изд-во НГУ, 2000.

[36] Новиков П. С. *Элементы математической логики.* М.: Наука, 1973.

[37] Ньюэлл А., Шоу Дж., Саймон Г. *Эмпирические исследования машины «Логик-теоретик»; пример изучения эвристики.* В кн.: *Вычислительные машины и мышление.* М.: Мир, 1967, с. 113–144.

[38] Робинсон Дж. А. *Машинно-ориентированная логика, основанная на принципе резолюции.* // Кибернетический сборник, новая серия, вып. 7. М.: Мир, 1970, с. 194–218.

[39] Роджерс Х. *Теория рекурсивных функций и эффективная вычислимость.* М.: Мир, 1972.

[40] Смальян Р. *Теория формальных систем.* М.: Наука, 1981.

[41] *Справочная книга по математической логике.* Ч. I–IV. М.: Наука, 1982, 1983.

[42] Такеути Г. *Теория доказательств.* М.: Мир, 1978.

[43] Успенский В. А., Верещагин Н. К., Плиско В. Е. *Вводный курс математической логики.* М.: ФИЗМАТЛИТ, 2004.

[44] Филд А., Харрисон П. *Функциональное программирование.* М.: Мир, 1993.

[45] Фихтенгольц Г. М. *Курс дифференциального и интегрального исчисления.* В трёх томах. Т. I. СПб.: Издательство «Лань», 1997.

[46] Френкель А., Бар-Хиллел И. *Основания теории множеств.* М.: Мир, 1966.

[47] Цейтин Г. С. *Ассоциативное исчисление с неразрешимой проблемой эквивалентности.* // ДАН СССР, Т. 107, N 3, 1956, с. 370–371.

[48] Цейтин Г. С. *Ассоциативное исчисление с неразрешимой проблемой эквивалентности.* // Тр. Матем. ин-та им. В. А. Стеклова АН СССР, Т. 52, 1958, с. 172–189.

[49] Чень Ч., Ли Р. *Математическая логика и автоматическое доказательство теорем.* М.: Наука, 1983.

[50] Чёрч А. *Введение в математическую логику.* М.: Издательство иностранной литературы, 1960.

[51] Шенфилд Дж. *Математическая логика.* М.: Наука, 1975.

[52] Яблонский С. В. *Введение в дискретную математику.* М.: Наука, 1979.

[53] Apt K. R. *Ten Years of Hoare's Logic: A Survey Part I.* // ACM Transactions on Programming Languages and Systems, Vol. 3, No. 4, October 1981, Pp. 431–483.

[54] Barendregt H., Barendsen E. *Introduction to Lambda Calculus.* 1994.

[55] Church A. *An Unsolvable Problem of Elementary Number Theory.* // American Journal of Mathematics, Vol. 58, No. 2, April 1936, Pp. 345–363.

[56] Deransart P., Ed-Dbali A., Cervoni L., Biro C., Scowen R. S. *Prolog: The Standard: Reference Manual.* Springer, 1996.

[57] Fitting M. *First-Order Logic and Automated Theorem Proving.* Springer, 1996.

[58] Gallier J. *Logic for Computer Science: Foundations of Automatic Theorem Proving.* Harper & Row Publishers, 1986.

[59] Hoare C. A. R. *An Axiomatic Basis for Computer Programming.* // Communications of the ACM, Vol. 12, No. 10, October 1969, Pp. 576–580, 583.

[60] Lassez J.-L., Maher M., Marriott K. *Unification Revisited.* // Lecture Notes in Computer Science, Vol. 306, 1988, Pp. 67–113.

[61] Nilsson U., Maluszynski J. *Logic, Programming and Prolog.* John Wiley & Sons, 1995.

[62] Sipser M. *Introduction to the Theory of Computation, Second Edition.* Course Technology, 2006.

[63] Troelstra A. S., Schwichtenberg H. *Basic Proof Theory, Second Edition.* Cambridge University Press, 2000.

Предметный указатель