ГОСУДАРСТВЕННЫЙ КОМИТЕТ РОССИЙСКОЙ ФЕДЕРАЦИИ
ПО ВЫСШЕМУ ОБРАЗОВАНИЮ

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. Н.И.ЛОБАЧЕВСКОГО

В.А.ТАЛАНОВ

МАТЕМАТИЧЕСКАЯ ЛОГИКА
И
МОДЕЛИ ВЫЧИСЛЕНИЙ

Учебное пособие

Издательство Нижегородского университета
Н.Новгород, 1994

ББК В I2
Т I6

Рецензенты:
профессор В.А.Буевич
профессор В.М.Галкин

Таланов В.А.
Т I6 Математическая логика и модели вычислений:Учебное пособие.- Нижний Новгород:ИЗД.ННГУ,1994 - II6 с.

В учебном пособии содержатся начала логики предикатов и теории алгоритмов. Она предназначена для студентов, обучающихся по специальности прикладная математика и информатика в рамках университетского учебного плана, содержащего курс "Дискретная математика". Основные понятия теории булевых функций из этого курса используются в ней без предварительного напоминания. Материал содержит примеры, иллюстрирующие основные понятия математической логики, и достаточно большое число упражнений.

Т $\frac{1602110000 - 279}{M187(03) - 94}$ ББК В I2

ISBN 5 - 230 - 04359 - 8

ВВЕДЕНИЕ

Математическая логика как самостоятельный предмет сформировалась к концу 19-го столетия. Ее развитию способствовали работы Буля (1815-1864), Фреге (1848-1925), Рассела (1872-1970), Гильберта (1862-1943). Различные ее элементы имелись в традиционной логике Аристотеля (384-322), в работах Лейбница (1646-1716) и других. Однако традиционная логика считается частью философии, в то время как математическая логика тесно связана с математикой по своим целям, методам и применениям.

В математике утверждения формулируются на обычном, естественном языке с использованием математических символов. Развитие современной математической логики позволяет формулировать математические утверждения с помощью некоторого числа специальных символов, не используя слов естественного языка. Ниже на базе небольшого числа таких символов мы опишем математическую модель языка, предназначенную для выражения математических утверждений. Эта модель, называемая логическим языком первого порядка, составляет основу для многих исследований в области математической логики.

Поскольку естественный язык представляет собой достаточно сложное постоянно развивающееся общественное явление, трудно от его формальных моделей ждать полной адекватности. Рассматриваемая нами модель языка предназначена для выражения утвердительных высказываний об об'ектах некоторой заданной совокупности и отношениях между ними. Предполагается, что каждому такому

высказыванию в определенных условиях можно дать истинностную оценку, то есть считать данное высказывание истинным или ложным.

Исследования в математической логике возникли в основном из вопросов, связанных с обоснованиями математики. Например, Фреге предложил строить математику на логических и теоретико-множественных принципах. Рассел пытался устранить противоречие, которое возникло в системе Фреге. Задача Гильберта состояла в том, чтобы показать, что обычно используемые методы математики не ведут к противоречию.

Часть I. ЛОГИЧЕСКИЙ ЯЗЫК ПЕРВОГО ПОРЯДКА

§1. Предварительные сведения

Пусть U - некоторое множество об'ектов, представляющих предмет нашего рассмотрения, будем называть его в дальнейшем универсом рассуждения.

Если f - символ, используемый для обозначения некоторой k-местной функции, отображающей U^k в U, и $a_1, a_2, \ldots, a_k$ - обозначения некоторых об'ектов из U, то выражение $f(a_1, a_2, \ldots, a_k)$ используется обычно как имя элемента из U, который соответствует набору элементов $a_1, a_2, \ldots, a_k$ при отображении f.

Имея имена об'ектов и функций, мы можем строить из них более сложные имена. Например, если g - имя одноместной, а h - двухместной функции и a, b - имена об'ектов из U, то, используя традиционным образом скобки, мы можем построить сложное имя h(g(a),h(b,g(a))). Чтобы определить значение этого имени, сначала следует определить значение имени g(a), затем h(b,g(a)) и, наконец, h(g(a),h(b,g(a))).

Наряду с символами-именами об'ектов будем использовать также специальные символы, называемые переменными. Если, например, x - переменная, то выражение h(g(x),h(b,g(a))) рассматривают как именную форму (схему) для образования имени. Подставляя в такие формы вместо переменных имена об'ектов, будем получать новые имена.

Выражения, построенные из функциональных символов, символов об'ектов и переменных, называются термами. Ниже мы дадим точное

определение терма.

Пусть теперь Q - имя некоторого k-местного отношения, $t_1, t_2, \ldots, t_k$ - имена об'ектов (простые или сложные), тогда выражение $Q(t_1, t_2, \ldots, t_k)$ рассматривается как элементарное (атомарное) высказывание :"Элементы с именами $t_1, t_2, \ldots, t_k$ находятся в отношении с именем Q". Истинностным значением этого высказывания является истина или ложь, в зависимости от того, находятся или не находятся элементы с именами $t_1, t_2, \ldots, t_k$ в отношении с именем Q.

Если в выражения $t_1, t_2, \ldots, t_k$ входят переменные, то выражение $Q(t_1, t_2, \ldots, t_k)$ рассматривают как элементарную высказывательную форму, которая может быть превращена в элементарное высказывание подстановкой имен об'ектов вместо переменных.

При образовании термов и элементарных высказываний имя функции и отношения мы располагали перед аргументами, заключенными в скобки (это так называемая префиксная запись), однако в математике, когда речь идет о двухместных функциях и отношениях, используется другая манера построения термов и высказываний, а именно, знак функции или отношения записывается между аргументами (инфиксная запись, знаки функций и отношений называются инфиксами).

Например, термы с двухместной функцией сложения записываются в виде t_1+t_2, а не в виде $+(t_1, t_2)$; аналогично, выражения элементарных высказываний $t_1=t_2$, $t_1<t_2$ и т.д. используются вместо выражений $=(t_1, t_2)$, $<(t_1, t_2)$ и т.д. При этом часто используются различные правила опускания скобок.

Из элементарных высказывательных форм и высказываний с помощью логических связок $\rightarrow$ (если..., то...), & (и), $\vee$ (или), $\neg$ (не) и кванторов $\forall$ (для любого) и $\exists$ (существует) с использованием скобок могут быть образованы сложные высказывания и высказывательные формы.

Чтобы предварительно продемонстрировать способы построения сложных высказываний, рассмотрим пример.

Выберем в качестве универса множество N, состоящее из натуральных чисел, включая 0, и рассмотрим известные арифметические функции $+, \times$; предикаты $=, <$; имена натуральных чисел $0, 1, 2, \ldots$ и переменные x, y, z.

Выражения $x+2$, $x+y$, $(x+2)\times 3$, $1+2$ будут термами в инфиксной записи, те же термы в префиксной записи выглядят следующим образом:

$$+(x,2),\quad +(x,y),\quad \times(+(x,2),3),\quad +(1,2).$$

Выражения элементарных высказываний и высказывательных форм $x<2$, $x+2<(x+2)\times y$, $2<2+1$ в префиксной записи будут такими: $<(x,2)$, $<(+(x,2),\times(+(x,2),y))$, $<(2,+(2,1))$.

Выражение $((x+2)<6)\&(2<(x+1))$ будет сложной высказывательной формой, которая при подстановке вместо x константы 2 превращается в истинное высказывание: "$2+2<6$ и $2<2+1$"; при подстановке константы 4 она превращается в ложное высказывание.

Выражение $\exists x[(x+2<6)\&(2<x+1)]$ можно перевести на русский язык следующим образом.

"Существует такое значение переменной x, при котором $x+2<6$ и $2<x+1$". Очевидно, это высказывание истинно, в качестве x можно взять константу 2 или 3.

Выражение $\forall x[(x+2<6)\&(2<x+1)]$ переводится на русский язык следующим образом.

"Для любого значения из N переменной x имеют место неравенства $x+2<6$ и $2<x+1$". Очевидно, это утверждение ложно.

Опираясь на вышеизложенное, в следующем параграфе дадим полное описание логического языка 1-го порядка. Оно включает в себя описание алфавита, определение переменной, терма и определение формулы.

§2. Синтаксис логического языка первого порядка

ОПИСАНИЕ АЛФАВИТА. Алфавит языка будет состоять из конечного числа символов, разбитых на группы.

1. Логические символы (связки и кванторы):

$$\& \quad \vee \quad \neg \quad \rightarrow \quad \forall \quad \exists \ .$$

2. Вспомогательные символы (запятая, скобки и символы для построения переменных):

$$, \quad [\quad (\quad] \quad) \quad x \quad '.$$

3. Нелогические символы. Набор σ нелогических символов, называемый нелогической сигнатурой, заранее не фиксируем, заметим только, что он может состоять из конечного числа так называемых предикатных и функциональных символов, предназначенных, соответственно, для обозначения отношений и функций.

Считаем также, что каждому нелогическому символу поставлено в соответствие неотрицательное целое число - его местность (арность). При этом k-местные функциональные символы используются для обозначения k-местных функций, а k-местные предикатные символы - для обозначения k-местных отношений.

В частности, 0-местные функциональные символы используются для обозначения констант из универса, а 0-местные предикатные символы - для обозначения постоянных высказываний, не зависящих от элементов универса. Можно считать, что 0-местные предикатные символы используются для обозначения 0-местных отношений, то есть подмножеств множества $U^0=\{\lambda\}$, так что существует всего два 0-местных отношения: $\emptyset$ и $\{\lambda\}$, где $\{\lambda\}$ - нульместный набор, а $\emptyset$ - пустое множество.

Кроме того, считаем, что σ всегда содержит явно неупоминаемый символ "=" для обозначения предиката равенства. Множество же функциональных символов может быть пустым.

Если $\sigma=\langle P_1,P_2,\ldots,P_k;f_1,f_2,\ldots,f_s\rangle$, где P_i - ν_i-местный предикатный символ, f_j - μ_j-местный функциональный символ ($i=1,2,\ldots,k$; $j=1,2,\ldots,s$), то набор

$$\tau=\langle\nu_1,\nu_2,\ldots,\nu_k;\mu_1,\mu_2,\ldots,\mu_s\rangle$$

называется типом сигнатуры σ ($k\geq1$, $s\geq0$).

Мы зафиксировали набор логических и вспомогательных символов, но не фиксируем набор σ. Выбор его иногда удобно связывать с конкретной теорией, язык которой мы собираемся моделировать. Так, например, при построении формальной арифметики обычно используются известные знаки арифметических отношений и функций: =, <, +, ×, 0, 1 и т.д.

Заметим, что выбор логических и вспомогательных символов также в большой мере случаен. В качестве логических связок можно было бы выбрать обозначения функций из любого функционально полного набора булевых функций, есть также возможность варьировать набор кванторов. Так, например, можно ограничиться следующим набором логических символов: $\neg$, $\rightarrow$, $\forall$, а остальные

определить через них.

ПОСТРОЕНИЕ ПЕРЕМЕННЫХ. Для обозначения переменных будем использовать выражения вида x, x', x'', x''' и т.д., построенные из двух символов: x и '. Очевидно, таким способом можно построить счетный набор переменных. На практике неудобно пользоваться выражениями с большим количеством штрихов, поэтому используются разные способы их кодирования, например, для обозначения переменных используют разные буквы с индексами и без них: x, y, z, x_1, y_1 и т.д. Здесь нам важно отметить, что для построения счетного набора переменных достаточно конечного числа символов. Множество построенных переменных обозначим через V (Var).

ПОСТРОЕНИЕ ТЕРМОВ осуществляется по следующим правилам.

1. Нульместный функциональный символ из σ или переменная являются термами.

2. Если f - k-местный функциональный символ из σ ($k \geq 1$) и $t_1, t_2, \ldots, t_k$ - термы, то выражение вида $f(t_1, t_2, \ldots, t_k)$ является термом.

Эти два правила полностью описывают множество выражений, называемых термами.

Термы используются как имена или формы для образования имен об'ектов универса.

Заметим, что в правилах построения термов мы не использовали инфиксную запись термов, хотя на практике будем использовать для наглядности термы вида (xfy), где x,y - переменные, а f - двухместный функциональный символ. При рассмотрении синтаксиса будем использовать лишь префиксную запись, имея в виду, что от записи вида (xfy) можно перейти к записи f(x,y).

Множество всех термов обозначим через Term (или через

Term(σ), когда необходимо указать явно нелогическую сигнатуру).

ПОСТРОЕНИЕ АТОМАРНЫХ (ЭЛЕМЕНТАРНЫХ) ФОРМУЛ. Если P - k-местный предикатный символ и $t_1, t_2, \ldots, t_k$ - термы, то выражение $P(t_1, t_2, \ldots, t_k)$ - атомарная формула. В частности, при k=0 список $t_1, t_2, \ldots, t_k$ - пустой, и скобки опускаются.

На практике наряду с такими выражениями будем использовать инфиксные выражения вида (xPy), где x,y - переменные, а P - двухместный предикатный символ.

ПОСТРОЕНИЕ ФОРМУЛ

1. Атомарная формула является формулой.

2. Если A и B - выражения, являющиеся формулами, то выражения вида [A&B], [A∨B], [A→B] являются формулами.

3. Если A - формула, то ¬A - формула.

4. Если y - переменная и A - формула, то выражения ∀yA и ∃yA -формулы.

Правила 1-4 полностью определяют множество выражений, являющихся формулами. Обозначим это множество через Form (или Form(σ)). На практике выражения ∀y и ∃y иногда для наглядности будем заключать в круглые скобки.

Введем некоторые синтаксические понятия, касающиеся формул.

Любая подпоследовательность следующих друг за другом символов в формуле A, которая сама является формулой, называется подформулой формулы A. Очевидно, что одна и та же подформула может иметь несколько <u>вхождений</u> в формулу A. Нетрудно доказать, что два различных вхождения подформулы не могут перекрываться.

Подформула формулы A, которая непосредственно следует за символами ∀y или ∃y, где y - переменная, называется <u>областью действия</u> соответствующего квантора. Вхождение переменной

y называется связанным, если оно находится в области действия квантора по y, и свободным в противном случае. Очевидно, одна и та же переменная может иметь в формуле свободные и связанные вхождения.

Формула, не имеющая свободных вхождений переменных, называется замкнутой формулой или предложением. Множество всех предложений обозначим через Sent (Sent(σ)).

На этом описание синтаксиса логического языка 1-го порядка заканчивается. Напомним, что мы определили множества Var, Term, Form, Sent, составляющие описание языка. Построенный язык будем обозначать через $\mathcal{L}(\sigma)$. Термы, формулы и предложения, построенные с помощью сигнатуры σ, будем называть также σ-термами, σ-формулами, σ-предложениями.

§3. Интерпретация формул логического языка первогопорядка

Мы уже говорили, что формулы используются для записи высказываний или высказывательных форм, но, чтобы формула превратилась в высказывание, необходимо предикатным и функциональным символам, входящим в формулу, поставить в соответствие конкретные отношения и функции на некотором множестве U, называемом областью интерпретации или универсом, а свободным переменным - конкретные об'екты из универса. При этом формула превращается в высказывание об этих отношениях, функциях и об'ектах.

Если свободным переменным не поставлены в соответствие никакие об'екты универса, то формулу можно рассматривать как высказывательную форму.

Рассмотрим следующий пример. Пусть $\sigma=\langle E,M,S\rangle$, где E - двухместный предикатный символ, а M и S, соответственно, двумхестный и одноместный функциональные символы.

Далее, пусть y,z,u,v - переменные, тогда выражение

$$\forall y\forall z\forall u\forall v[[E(u,S(y))\&E(v,S(z))]\rightarrow E(M(u,v),S(M(z,y)))],(1)$$

очевидно, является формулой (то есть выражением, построенным по правилам построения формул). Заметим, что это замкнутая формула.

Чтобы приписать какой-либо смысл этой формуле, выберем в качестве универса множество П точек плоскости. Символу S поставим в соответствие функцию, которая каждой точке $a\in П$ ставит в соответствие точку, симметричную точке a относительно некоторой заранее выбранной точки $o\in П$. Символу M поставим в соответствие функцию, которая каждым двум точкам $a,b\in П$ ставит в соответствие середину отрезка между a и b. Символу E ставим в соответствие отношение равенства.

При такой интерпретации нелогических символов наша формула представляет следующее геометрическое высказывание: "Если точка u симметрична точке y относительно точки o, а точка v симметрична точке z, то середина отрезка [u,v] симметрична середине отрезка [y,z] относительно o".

Заметим, что тот же самый смысл имеет и более короткая формула

$$\forall z\forall y\ E(M(S(z),S(y)),S(M(z,y))).(2)$$

Дадим еще одну интерпретацию нелогических символов E, M, S. В качестве универса рассмотрим множество действительных чисел без нуля. Проинтерпретируем S как функцию, ставящую в соответствие действительному, отличному от нуля, числу a число a^{-1}; M - как функцию, ставящую в соответствие паре чисел их произведение, а E

- как отношение равенства. Тогда формулы (1) и (2) выражают следующее: "Число, обратное произведению двух чисел, равно произведению чисел, обратных к ним".

В соответствии с приведенными выше примерами и раз'яснениями дадим полное определение интерпретации I логического языка первого порядка на универсе U.

Интерпретацией будем называть отображение I, заданное на $\sigma \cup V$, которое:

каждому n-местному предикатному символу $P \in \sigma$ ставит в соответствие отношение $P^{(I)} \subseteq U^n$;

каждому n-местному функциональному символу $f \in \sigma$ ставит в соответствие функцию $f^{(I)}: U^n \longrightarrow U$, в частности, при n=0 $f^{(I)} \in U$ и называется константой;

каждой переменной $y \in V$ ставит в соответствие элемент $y^{(I)} \in U$.

С помощью интерпретации I каждому терму $t \in Term(\sigma)$ ставится в соответствие об'ект

$$t^{(I)} = \begin{cases} y^{(I)}, \text{ если } t=y \text{ и } y \in V, \\ f^{(I)}(t_1^{(I)}, t_2^{(I)}, \ldots, t_n^{(I)}), \text{если } t=f(t_1, t_2, \ldots, t_n). \end{cases}$$

Каждой формуле $A \in Form(\sigma)$ ставится в соответствие истинностное значение $A^{(I)} \in \{0,1\}$, при этом 0 интерпретируется как ложь, а 1 - как истина. Часто вместо символов 0,1 используют, соответственно, символы f,t (от слов false, true).

Если A - атомарная формула вида $P(t_1, t_2, \ldots, t_n)$, то

$$A^{(I)} = \begin{cases} 1, \text{ если } (t_1^{(I)}, t_2^{(I)}, \ldots, t_n^{(I)}) \in P^{(I)}, \\ 0 \text{ в противном случае.} \end{cases}$$

Для сложных формул $A^{(I)}$ определяется рекурсивно с помощью булевых функций, обозначения которых совпадают с используемыми

нами логическими связками:

$$A^{(I)} = \begin{cases} A_1^{(I)} \& A_2^{(I)}, \text{ если } A=[A_1 \& A_2], \\ A_1^{(I)} \vee A_2^{(I)}, \text{ если } A=[A_1 \vee A_2], \\ \neg A_1^{(I)}, \text{ если } A = \neg A_1, \\ \neg A_1^{(I)} \vee A_2^{(I)}, \text{ если } A = [A_1 \rightarrow A_2], \\ \bigvee_{I' \in \psi_I} B^{(I')}, \text{ если } A= \exists y B, \\ \bigwedge_{I' \in \psi_I} B^{(I')}, \text{ если } A= \forall y B, \end{cases}$$

где ψ_I - множество интерпретаций, совпадающих с I всюду, кроме, быть может, значения $y^{(I)}$ переменной y при интерпретации I. Бесконечная диз'юнкция и кон'юнкция в последних двух случаях определяются естественным образом.

Заметим, что если A - замкнутая формула, то ее истинностное значение $A^{(I)}$ не зависит от того, как определено I на множестве V, поэтому особое значение имеет сужение отображения I на множество σ; такое сужение I_σ называется структурой или алгебраической системой сигнатуры σ.

Если σ состоит из символов $P_1, P_2, \ldots, P_k, f_1, f_2, \ldots, f_m$, то структуру $S=I_\sigma$ на универсе U изображают в виде набора

$$S=I_\sigma=\langle U; P_1^{(I)}, P_2^{(I)}, \ldots, P_k^{(I)}, f_1^{(I)}, \ldots, f_m^{(I)} \rangle.$$

Сужение I_σ отображения I на V называют означиванием переменных.

ОСНОВНЫЕ ПОНЯТИЯ, СВЯЗАННЫЕ С ИНТЕРПРЕТАЦИЕЙ

Формула A называется <u>истинной при интерпретации</u> I, если $A^{(I)}=1$.

Формула A называется <u>общезначимой</u> или тождественно истинной,

если она истинна при любой интерпретации.

Формула A называется выполнимой, если существует интерпретация, в которой она истинна, в противном случае она называется невыполнимой (или тождественно ложной).

Формула A логически следует из множества формул Г, если в любой интерпретации, в которой истинны все формулы из Г, истинна также формула A. Отношение логического следования будем обозначать знаком $\Rightarrow$ и писать $\Gamma \Rightarrow A$, если A логически следует из Г.

Формулы A и B называются логически равносильными, если $\{A\} \Rightarrow B$ и $\{B\} \Rightarrow A$. Логическую равносильность формул A и B обозначают $A \equiv B$. Вместо $\{A\} \Rightarrow B$ будем писать также $A \Rightarrow B$.

Пусть теперь Г - множество предложений ($\Gamma \subseteq Sent(\sigma)$), тогда структура, в которой истинны все предложения из Г, называется моделью множества Г.

УПРАЖНЕНИЯ

1. В структуре $(N;+,\times,0,1)$, где N множество натуральных чисел, включая 0, выразить следующие предикаты:

p(x,y,z) - "z является наибольшим общим делителем чисел "x,y";

p(x,y,z) - "z является наименьшим общим кратным чисел "x,y";

p(x,y) - "x меньше y";

p(x,y) - "x меньше или равно y";

p(x) - "x является квадратом целого числа";

p(x) - "x является четным числом";

p(x) - "x является нечетным числом";

p(x) - "x является простым числом";

p - "множество простых чисел бесконечно".

2. Пусть M - множество точек плоскости. Рассмотрим два предиката на M.

B(a,b,c) - "точки a,b,c лежат на одной прямой, причем b расположена между a и c";

D(a,b,c,d) - "расстояние от точки a до b равно расстоянию от c до d".

Выразить в структуре (M;B,D) следующие предикаты:

p(a,b,c,d) - "отрезки ab и cd имеют единственную общую точку";

p(a,b,c,d) - "отрезки ab и cd параллельны";

p(a,b,c) - "точки a,b,c являются вершинами равностороннего треугольника";

p(a,b,c) - "точки a,b,c лежат на одной прямой";

p(a,b,c) - "точки a и b симметричны относительно точки c";

p(a,b,c,d) - "точка a симметрична точке b относительно отрезка cd";

p(a,b,c) - "расстояние между точками a,b не превосходит расстояния между точками b,c";

p(a,b,c,d) - "расстояние между точками a,b не превосходит расстояния между точками c,d";

p(a,b,c,d) - "точки a,b,c,d являются вершинами ромба";

p(a,b,c,d,e,f) - "углы abc и def конгруентны";

p(a,b,c,d,e,f) - "треугольники с вершинами a,b,c и d,e,f подобны".

3. Пусть $A=\{a,b\}$ - алфавит из двух символов a и b, в дальнейшем будем их рассматривать как однобуквенные слова, λ - пустое слово. Рассмотрим структуру $(A^*;\hat{},a,b,\lambda)$, где A^* -

множество слов в алфавите A, ^ - двухместная операция конкатенации слов (под u^v понимаем результат приписывания слова v к слову u, вместо u^v будем писать uv, опуская знак конкатенации). Выразить в данной структуре следующие предикаты:

p(u,v) - "слово u является префиксом, то есть начальным фрагментом слова v";

p(u,v) - "слово u является суффиксом, то есть конечным фрагментом слова v";

p(u,v) - "слово u является фрагментом слова v".

При описании предикатов в структуре (N;+,×,0,1) будем использовать наряду с обычными кванторами, так называемые ограниченные кванторы общности и существования, а именно, выражения вида $(\exists x\leqslant y)$,$(\forall x\leqslant y)$, где x и y - переменные. При построении формул эти кванторы будут использоваться так же как обычные. Например, если A - формула, то выражения $(\exists x\leqslant y)A$ и $(\forall x\leqslant y)A$ - формулы, причем переменная y в них свободна. Семантику этих формул можно описать с помощью следующих соотношений:

$$(\exists x\leqslant y)A \equiv (\exists x)[x\leqslant y \ \&\ A],$$

$$(\forall x\leqslant y)A \equiv (\forall x)[x\leqslant y \rightarrow A].$$

Отношение на N назовем арифметическим, если оно выразимо формулой первого порядка в сигнатуре (+,×,0,1).

Отношение на N назовем конструктивно арифметическим, если оно выразимо формулой первого порядка, не содержащей неограниченных кванторов, в сигнатуре (+,×,0,1).

Приведем несколько примеров.

Отношение "$x\leqslant y$" конструктивно арифметическое так как выразимо формулой $(\exists z\leqslant y)[x+z=y]$.

Отношение "x делит y" конструктивно арифметическое так как выразимо формулой $(\exists z\leqslant y)[x\times z=y]$.

Легко видеть, что отношения $x\leq y$, $x>y$, $x\geq y$ конструктивно арифметические.

Для каждого простого числа p отношение "x является степенью простого числа p" конструктивно арифметическое, так как равносильно выражению

$$(\forall y\leq x)[\text{"y делит x"} \ \&\ y>1 \rightarrow \text{"p делит y"}].$$

Любую структуру S для сигнатуры σ будем для краткости называть σ-структурой. Интерпретацию любого предикатного символа $P\in\sigma$ в структуре S будем обозначать через P^S, аналогично, интерпретацию любого функционального символа $f\in\sigma$ - через f^S, а универс структуры S через U^S.

Пусть заданы две σ-структуры A и B. Отображение $\pi:U^A\to U^B$ называется изоморфизмом структуры A в струтуру B, если

(I) π - взаимно однозначно,

(2) для каждого n-местного предикатного символа $P\in\sigma$ и для любых $a_1,a_2,\ldots,a_n\in U^A$

$$(a_1,a_2,\ldots,a_n)\in P^A \Leftrightarrow (\pi(a_1),\ldots,\pi(a_n))\in P^B,$$

(3) для каждого n-местного функционального символа $f\in\sigma$ и для любых $a_1,a_2,\ldots,a_n\in U^A$

$$\pi(f^A(a_1,a_2,\ldots,a_n))=f^B(\pi(a_1),\ldots,\pi(a_n)).$$

ПРИМЕР. Пусть $\sigma=\{+,0\}$, где + -двухместный , а 0 - нульместный функциональные символы. Рассмотрим две σ-структуры $A=(U^A;+^A,0^A)$ и $B=(U^B;+^B,0^B)$, где U^A - множество целых чисел, U^B - множество четных чисел. Отображение $\pi:U^A\to U^B$, такое что для любого

$k \in U^4$ $\Pi(k)=2k$, является при естественной интерпретации символов +, 0 изоморфизмом структуры A на структуру B.

Если существует изоморфизм структуры A на структуру B, то говорим что структура A изоморфна структуре B (сокращенно $A \cong B$).

Очевидно отношение изоморфизма между структурами рефлексивно, симметрично и транзитивно.

Нетрудно доказать так же, что на изоморфных σ-структурах A и B истинны одни и те же σ-предложения.

Однако, если на σ-структурах A и B истинны одни и те же σ-предложения, то A и B не обязательно изоморфны.

Это замечание дает повод для следующего определения.

Две σ-структуры A и B называются элементарно эквивалентными, если на них истинны одни и те же σ-предложения (сокращенно $A \approx B$).

ПРИМЕРЫ. Пусть P и Q - одноместные предикатные символы. Рассмотрим формулы $A_1 = \forall x P(x)$, $A_2 = \forall x[P(x) \vee \neg P(x)]$, $A_3 = \forall x[P(x) \& \neg P(x)]$, $A_4 = \exists x[P(x) \vee Q(x)]$, $A_5 = [\exists x P(x) \vee \exists x Q(x)]$, $A_6 = \forall x[P(x) \vee Q(x)]$, $A_7 = \forall x P(x) \vee \forall x Q(x)$.

Все перечисленные выше формулы кроме A_3 выполнимы, формула A_3 невыполнима, формулы A_4 и A_5 логически равносильны, формулы A_6 и A_7 не являются логически равносильными, из формулы A_7 логически следует формула A_6, из формулы A_6 логически не следует формула A_7, формула A_2 общезначима.

УПРАЖНЕНИЕ

Привести примеры формул A и B, так чтобы выполнялись следующие утверждения:

a) A выполнима, $A \to B$ тождественно истинна, а B не тождественно истинна;

b) $A \to B$ тождественно истинна, а B не тождественно истинна;

c) $A \& B$ не выполнима, а $A \vee B$ тождественно истинна;

d) A выполнима, B выполнима, а $A \to B$ не выполнима;

e) A выполнима, B выполнима, а $A \vee B$ не выполнима;

f) A выполнима, B выполнима, а $A \& B$ не выполнима.

§ 4. Способы задания и подсчет числа структур на конечных универсах

Если универс конечен, то предикаты и функции на нем можно представить в виде таблиц, графов и других наглядных конструкций.

Пусть, например, универс U состоит из 6 элементов $a_1, a_2, \ldots, a_6$, и двухместное отношение P на U задано множеством пар $P=\{(a_i, a_j) \mid j$ делит $i\}$; очевидно, P содержит пары $(a_1, a_1), (a_2, a_1), (a_2, a_2), (a_3, a_1)$ и т.д.

Отношение P можно представить таблицей (рис.1), в которой на пересечении i-й строки и j-го столбца находится 1, если i делится на j, и 0 - в противном случае.

	a_1	a_2	a_3	a_4	a_5	a_6
a_1	1	0	0	0	0	0
a_2	1	1	0	0	0	0
a_3	1	0	1	0	0	0
a_4	1	1	0	1	0	0
a_5	1	0	0	0	1	0
a_6	1	1	1	0	0	1

Рис. 1

Это же отношение можно задать с помощью графа с шестью вершинами, в котором из вершины с номером i в вершину с номером j проведена дуга в том и только том случае, когда $(a_i, a_j) \in P$.

Очевидно, любая таблица размера $n \times n$, заполненная нулями и единицами, представляет некоторое двухместное отношение на множестве из n элементов, и число таких отношений равно числу различных таблиц.

Простые комбинаторные рассуждения показывают, что это число равно 2^{n^2}.

Подсчитывая все k-местные ($k=0,1,2,\ldots$) отношения на множестве из n элементов ($n=1,2,\ldots$), получим, что их число равно 2^{n^k}.

Функции, как и отношения, можно представлять таблицами. Например, таблица на рис.2 представляет одноместную функцию, для которой $f(a_1)=a_2$, $f(a_2)=a_2$, $f(a_3)=a_4$ и т.д. Очевидно, число таких различных таблиц равно $6^6=46656$, так как на каждое из 6 мест во второй строке можно поместить любой из 6 элементов. Аналогично, на универсе из n элементов можно задать n^n различных одноместных функций.

x	a_1	a_2	a_3	a_4	a_5	a_6
f(x)	a_2	a_2	a_4	a_4	a_6	a_5

Рис.2

Нетрудно также подсчитать число различных k-местных функций. Так как на множестве из n элементов можно построить n^k k-местных наборов аргументов, и на каждом наборе функция принимает одно из n значений, получаем, что это число равно n^{n^k}.

Заметим, что функцию, представленную на рис.2, можно задать также с помощью графа с 6 вершинами, проводя дугу из вершины с номером i в вершину с номером j, если $f(a_i)=a_j$.

Выпишем теперь формулу для числа всех структур сигнатуры $\sigma=\langle P_1,P_2,\ldots,P_k;\ f_1,f_2,\ldots,f_m\rangle$ на универсе из n элементов. Пусть тип сигнатуры σ представлен набором

$$\langle \nu_1,\nu_2,\ldots,\nu_k;\ \mu_1,\mu_2,\ldots,\mu_m\rangle,$$

тогда число различных структур сигнатуры σ на универсе из n элементов равно

$$2^{\sum_{i=1}^{k} n^{\nu_i}} \cdot n^{\sum_{i=1}^{m} n^{\mu_i}}.$$

Очевидно, множество всех структур сигнатуры σ на бесконечном универсе бесконечно.

УПРАЖНЕНИЯ

Пусть P,Q - одноместные, R,S - двухместные предикатные символы, f - одноместный функциональный символ, T - трехместный предикатный символ.

1. Задать с помощью таблиц или графов модели на универсе из n ($n=1,2,3,4$) элементов для следующих предложений:

а) $\forall x\, R(x,x)$,

б) $\forall x \forall y\, [R(x,y) \to R(y,x)]$,

в) $\forall x \forall y\, [R(x,y) \vee R(y,x)]$,

г) $\forall x \forall y\, [R(x,y) \to \neg R(y,x)]$,

д) $\forall x \forall y\, [R(x,y) \to S(x,y)]$,

е) $\forall x \forall y \exists z\, [R(x,z) \,\&\, R(z,y)]$,

ж) $\forall x\, P(f(x))$,

з) $\forall x\, R(x,f(x))$.

2. Доказать, что предложение

$$\exists x \forall y\, [[R(x,y) \,\&\, \neg R(y,x)] \to [R(x,x) \leftrightarrow R(y,y)]]$$

истинно в любой структуре, содержащей не более трех элементов.

3. Доказать, что предложение

$\forall x \exists y\ R(x,y)\ \&\ \forall x \forall y\ [R(x,y) \to \neg R(y,x)]\ \& \forall x \forall y \forall z\ [R(x,y) \& R(y,z) \to R(x,z]$

ложно во всех конечных структурах и истинно в некоторых бесконечных.

4. Доказать, что предложение

$\forall x\ R(x,x) \to \exists x \forall y\ [R(x,y) \vee \exists z\ [R(y,z)\ \&\ \neg R(x,z)]]$

истинно на всех конечных структурах и ложно на некоторых бесконечных.

5. Найти число структур соответствующей сигнатуры на универсе U_n, состоящем из n элементов, в которых истинны следующие предложения:

$\forall x\ R(x,x)$,

$\forall x\ \neg R(x,x)$,

$\exists x\ R(x,x)$,

$\exists x\ \neg R(x,x)$,

$\forall x \forall y\ [R(x,y)\ \&\ R(y,x)]$,

$\forall x \forall y\ [R(x,y) \vee R(y,x)]$,

$\exists x \exists y\ [R(x,y)\ \&\ \neg R(y,x)]$,

$\exists x \exists y\ [\neg R(x,y)\ \&\ \neg R(y,x)]$,

$\forall x \exists y\ R(x,y)$, $\exists x \forall y\ R(x,y)$,

$\forall x\ P(x)$, $\exists x\ P(x)$,

$\forall x\ [P(x) \vee \neg P(x)]$,

$\forall x\ [P(x)\ \&\ \neg P(x)]$,

$\forall x\ [P(x)\ \&\ Q(x)]$,

$\forall x\ [P(x) \vee Q(x)]$,

$\exists x\ [P(x)\ \&\ Q(x)]$,

$\exists x\ [P(x) \vee Q(x)]$,

$\exists x\ [P(x) \to Q(x)]$,

$\forall x\ [P(x) \vee \exists y\ R(x,y)]$,

$\exists x\ [\neg P(x)\ \&\ \forall y\ \neg R(x,y)]$,

$\forall x\ [P(x) \to \exists y\ R(x,y)]$,

$\forall x\ [P(x) \to \exists y\ R(y,x)]$,

$\forall x \forall y \forall z\ [P(x)\ \&\ Q(y) \vee \neg P(x)\ \&\ Q(z)]$,

$\forall x \exists y \exists z\ [P(x)\ \&\ Q(y)\ \&\ \neg Q(z)]$,

$\forall x\ P(f(x))$,

$\exists x\ P(f(x))$,

$\forall x \exists y\ [y \neq x\ \&\ R(x,y)]$,

$\forall x \forall y\ [x \neq y \to R(x,y)]$,

$\forall x\ (f(x) \neq x)$,

$\forall x\ (x = f(f(x)))$,

$\forall x\ [P(x) \to \forall y\ R(x,y)]$,

$\forall x \forall y \exists z\ T(x,y,z)$,

$\forall x \exists y \forall z\ T(x,y,z)$,

$\exists x \forall y \forall z\ T(x,y,z)$,

$\exists x \forall y \exists z\ T(x,y,z)$,

$\exists x \exists y \forall z\ T(x,y,z)$,

$\forall x \exists y \exists z\ T(x,y,z)$,

$\forall x\ [P(x) \to \exists y\ R(x,y)\ \&\ \exists y\ \neg R(x,y)]$.

§5. Исключающие кванторы

В дальнейшем нам удобно будет воспользоваться небольшой модификацией логического языка первого порядка, а именно, вместо обычных кванторов общности и существования введем, соответственно, так называемые исключающий квантор общности и исключающий квантор существования.

Точнее, в правилах построения формул правило 4 заменим на

следующее:

4'. Если A - формула, x - переменная, $y_1, y_2, \ldots, y_k$ - список, составленный из переменных и констант, то выражения

$$(\forall x | y_1, y_2, \ldots, y_k)A, \quad (\exists x | y_1, y_2, \ldots, y_k)A \qquad (I)$$

будут также формулами. Список $y_1, y_2, \ldots, y_k$ будем называть списком исключений или исключающим списком.

Истинностные значения этих формул считаем соответственно равными истинностным значениям формул

$$\forall x\ [x \neq y_1\ \&\ x \neq y_2\ \& \ldots \&\ x \neq y_k \to A],$$

$$\exists x\ [x \neq y_1\ \&\ x \neq y_2\ \& \ldots \&\ x \neq y_k\ \&\ A].$$

Используя сформулированные правила приписывания формулам истинностных значений, легко увидеть, что введенное расширение понятия квантора не увеличивает выразительных возможностей в том смысле, что для любой формулы расширенного языка существует логически равносильная ей формула с обычными кванторами. Такие расширения языка будем называть несущественными.

В правиле (4') мы не исключаем возможности пустого списка исключений, в этом случае считаем k=0, а в записи формул (1) опускаем вертикальную черту и скобки. В этом случае понятие исключающего квантора совпадает с понятием обычного квантора.

Если список исключений в формулах вида (1) состоит в точности из всех свободных переменных формулы A, кроме x, то формулы (1) сокращенно будем записывать соответственно в виде $\dot{\forall} xA$ и $\dot{\exists} xA$.

Переменную y_i из списка $y_1, y_2, \ldots, y_k$ в формуле вида (1) будем считать свободной.

Выражения вида $(\forall x | y_1, y_2, \ldots, y_k)$ и $(\exists x | y_1, y_2, \ldots, y_k)$ будем

называть исключающими кванторами общности и существования, соответственно.

Формулу будем называть г-формулой, если в списке исключений каждого квантора находятся все свободные переменные, расположенные в его области действия.

Заметим, что любая формула A из $Form(\sigma)$ логически равносильна некоторой г-формуле. Чтобы по формуле A построить логически равносильную ей г-формулу достаточно воспользоваться соотношениями

$$\forall xA \equiv (\forall x | y_1, y_2, \ldots, y_k)A \ \&_{i=1}^{k} A_x[y_i],$$

$$\exists xA \equiv (\exists x | y_1, y_2, \ldots, y_k)A \vee_{i=1}^{k} A_x[y_i],$$

где $y_1, y_2, \ldots, y_k$ - список переменных, включающий все свободные переменные формулы A, а $A_x[y_i]$ - формула, полученная из формулы A заменой свободных вхождений переменной x на y_i.

ПРИМЕР

Формула $\forall x \exists y\ R(x,y)$, где R - двухместный предикатный символ, логически равносильна г-формуле

$$(\forall x)[(\exists y | x)R(x,y) \vee R(x,x)].$$

УПРАЖНЕНИЕ

Для формулы $\forall x \forall y \forall z [R(x,y) \& R(y,z) \to R(x,z)]$ построить равносильную ей г-формулу.

ПРИМЕР

Формулой расширенного языка является выражение

$$(\forall x)(\exists y | x)R(x,y),$$

где x,y - переменные, а R - двухместный предикатный символ. Эту

же формулу можно сокращенно записать в виде $\forall x \dot{\exists} y R(x,y)$. В качестве универса для интерпретации выберем множество $U=\{a,b,c\}$ и два отношения R_1 и R_2, заданных соответственно таблицами (рис.3,4).

R_1	a	b	c
a	1	0	1
b	0	1	1
c	0	0	1

Рис.3

R_2	a	b	c
a	1	1	0
b	0	1	1
c	0	1	0

Рис.4

При интерпретации символа R с помощью отношения R_1 рассматриваемая формула ложна, а при интерпретации с помощью R_2 - истинна. Формула $\forall x \exists y\ R(x,y)$ истинна в обеих интерпретациях.

УПРАЖНЕНИЕ

Вычислить число моделей формулы $\forall x \dot{\exists} y\ R(x,y)$ на универсе из n элементов.

Часть 2. ЛОГИЧЕСКИЙ ВЫВОД И ЭЛЕМЕНТАРНЫЕ ТЕОРИИ

§I. Логический вывод

Введенное нами определение логического следования не дает никакого непосредственного способа выяснения, следует ли логически та или иная формула из множества формул Г.

В математической практике для выяснения справедливости такого рода утверждений используются доказательства.

Любое доказательство можно рассматривать как некоторую систему утверждений, каждое из которых либо с очевидностью является истинным, либо следует из доказанных утверждений. Опираясь на эти интуитивные представления о сущности доказательств, мы введем основные формальные понятия теории доказательств, такие, как правило вывода, доказательство, теорема, непротиворечивость множества предложений и т.п.

Чтобы иметь представление о правилах вывода, рассмотрим следующую ситуацию. Допустим, в некотором естественном математическом доказательстве на некотором этапе мы доказали утверждение A, затем доказали, что из A следует утверждение B, тогда мы делаем вывод, что истинно утверждение B.

Считая A и B предложениями логического языка, можно возвести такую ситуацию в ранг правила, по которому из истинности предложений (посылок) A и $[A \to B]$ в некоторой интерпретации можно в качестве заключения вывести истинность предложения B в той же интерпретации. Такое правило можно записать в виде

$$\frac{A,\ [A \to B]}{B}.$$

где над чертой выписаны формулы-посылки, а под чертой - формула-заключение. Приведенное выше правило популярно во многих логических исследованиях и имеет специальное название "модус поненс".

Мы будем рассматривать такие правила вывода, в которых заключением является не одна единственная формула, а несколько, при этом будем отличать случай, когда выводится одновременная истинность всех формул заключения, от случая, когда выводится утверждение о том, что истинна хотя бы одна из формул заключения. В первом случае будем записывать формулы заключения под чертой в виде столбца, а во втором - в виде строки, в которой формулы отделены друг от друга вертикальными черточками. Правила второго вида будем называть разветвляющими.

Примеры правил вывода:

$$\text{а)}\ \frac{[A\&B]}{\begin{matrix}A\\B\end{matrix}},\quad \text{б)}\ \frac{[A\to B]}{\neg A|B},\quad \text{в)}\ \frac{[A\vee B]}{A|B},\quad \text{г)}\ \frac{\neg[A\to B]}{\begin{matrix}A\\\neg B\end{matrix}},$$

где A и B - произвольные предложения логического языка.

Очевидно, посылки и заключения в приведенных примерах связаны, соответственно, следующими отношениями:

а) $[A\&B] \Rightarrow A$ и $[A\&B] \Rightarrow B$,

б) $[A\to B] \Rightarrow \neg A$ или $[A\to B] \Rightarrow B$,

в) $[A\vee B] \Rightarrow A$ или $[A\vee B] \Rightarrow B$,

г) $\neg[A\to B] \Rightarrow A$ и $\neg[A\to B] \Rightarrow \neg B$.

Эти отношения говорят о том, что приведенные правила корректны. Заметим, что если в правилах б) и в) в качестве A и B употребить формулы со свободными переменными, то корректность нарушится.

Одним из основных вопросов математической логики является вопрос существования и выбора правил вывода, с помощью которых можно было бы доказать любые истинные утверждения вида $\Gamma \Rightarrow A$, где A - некоторое предложение, а Γ - множество предложений, и были бы недоказуемы ложные утверждения такого вида; при этом слову "доказать" должен быть приписан четкий математический смысл.

Ниже мы покажем, что существует конечное число правил вывода и такое математическое понятие доказательства, для которых $\Gamma \Rightarrow A$ тогда и только тогда, когда существует доказательство предложения A из множества посылок Γ.

Для простоты рассмотрим случай, когда σ не содержит равенства и функциональных символов.

Чтобы зафиксировать набор правил вывода, заметим сначала, что любое предложение логического языка с исключающими кванторами, если оно не атомарно и не отрицание атомарного, имеет один из следующих одиннадцати видов:

$$[A\&B],\ [A\vee B],\ [A\to B],\ (\forall x|\alpha)C,\ (\exists x|\alpha)C,$$
$$\neg[A\&B],\ \neg[A\vee B],\ \neg[A\to B],\ \neg(\forall x|\alpha)C,\ \neg(\exists x|\alpha)C,\ \neg\neg A,$$

где A,B - предложения, C - формула с единственной свободной переменной x, α - состоящий из констант список исключений. Соответственно этому, в таблице 1 приведены одиннадцать правил вывода, где через $C_x[a]$ обозначен результат подстановки константы a в формулу C вместо переменной x. Логические связки, указанные в скобках рядом с правилами, можно использовать как названия соответствующих правил.

$$\frac{[A\&B]}{\begin{matrix}A\\B\end{matrix}}\ (\&) \qquad \frac{\neg[A\&B]}{\neg A|\neg B}\ (\neg\&)$$

$$\frac{[A\vee B]}{A|B}\ (\vee) \qquad \frac{\neg[A\vee B]}{\begin{matrix}\neg A\\\neg B\end{matrix}}\ (\neg\vee)$$

$$\frac{[A\to B]}{\neg A|B}\ (\to) \qquad \frac{\neg[A\vee B]}{\begin{matrix}\neg A\\\neg B\end{matrix}}\ (\neg\to)$$

$$\frac{(\forall x|\alpha)C}{\begin{matrix}(\forall x|\alpha,a)C\\C_x[a]\end{matrix}}\ (\forall) \qquad \frac{\neg(\forall x|\alpha)C}{(\exists x|\alpha)\neg C}\ (\neg\forall)$$

$$\frac{(\exists x|\alpha)C}{C_x[a]}\ (\exists) \qquad \frac{\neg(\exists x|\alpha)C}{(\forall x|\alpha)\neg C}\ (\neg\exists)$$

$$\frac{\neg\neg A}{A}\ (\neg\neg)$$

Таблица 1

Теперь рассмотрим вопрос о том, как строить доказательства с помощью выбранных правил вывода. Естественные математические доказательства можно разделить на два класса: прямые и косвенные, например, доказательства от противного. В математической логике нет необходимости использовать оба этих варианта. Мы покажем, что можно, например, обойтись только понятием доказательства от противного. Рассмотрим предварительно два примера.

ПРИМЕР 1. Пусть нам необходимо ответить на вопрос: является ли предложение

$$\exists z\ [S(z)\ \&\ \neg P(z)]$$

логическим следствием предложений

$$\forall x\ [P(x) \to \neg M(x)] \quad \text{и} \quad \exists y\ [S(y)\ \&\ M(y)]?$$

Предположим, что не является, тогда существует некоторая интерпретация I с областью U, в которой истинны формулы- посылки

1. $\exists y\ [S(y)\ \&\ M(y)]$,
2. $\forall x\ [P(x) \to \neg M(x)]$

и истинно отрицание формулы-заключения

3. $\neg\exists z\ [S(z)\ \&\ \neg P(z)]$.

Формула 1 утверждает, что существует некоторый элемент из U (введем для его обозначения константу a), для которого истинна формула

4. $[S(a)\ \&\ M(a)]$.

Формула 2 утверждает, что для всех элементов, в том числе и для a, выполняются формулы

5. $[P(a) \to \neg M(a)]$,
6. $(\forall x|a)\ [P(x) \to \neg M(x)]$.

Из истинности формулы 3 следует истинность формулы

7. $\forall z\ \neg[S(z)\ \&\ \neg P(z)]$.

Из истинности формулы 4 следует истинность формул

8. $S(a)$,
9. $M(a)$.

Из истинности формулы 5 следует, что возможны два случая. В первом случае утверждается истинность формулы

10.1. $\neg P(a)$,

а во втором случае - истинность формулы

10.2. $\neg M(a)$.

Но формула 10.2 противоречит формуле 9, поэтому остается только анализировать первый случай. Формулу 6 пропускаем, а из

истинности формулы 7 выводим истинность формул

11.1. $\neg[S(a) \& \neg P(a)]$,

11.2. $(\forall z|a)\ \neg[S(z) \& \neg P(z)]$.

Формулы 8, 9, 10.1 пропускаем. Формула 11.1 может быть истинна в двух случаях (это будут два подслучая случая 1), а именно, когда истинна формула

12.1.1. $\neg S(a)$

или истинна формула

12.1.2. $\neg\neg P(a)$.

Но 12.1.1 противоречит формуле 8, а 12.1.2 - формуле 10.1. Итак, сделанное нами предположение приводит к противоречию, следовательно,

$$\{\forall x\ [P(x) \to \neg M(x)],\ \exists y\ [S(y) \& M(y)]\} \Rightarrow \exists z\ [S(z) \& \neg P(z)].$$

Опуская комментарии, приведенное выше доказательство можно представить в виде дерева (рис.5), узлам которого приписаны формулы. Двойная черта в конце ветви означает, что ветвь содержит противоречие.

ПРИМЕР 2

Пусть требуется определить, является ли предложение

$$\exists z\ [P(z) \& \neg S(z)]$$

логическим следствием предложений

$$\exists x\ [P(x) \& \neg M(x)] \quad \text{и} \quad \exists y\ [\neg S(y) \& M(y)].$$

Строя рассуждения по образцу предыдущего примера, получим дерево, изображенное на рисунке 6.

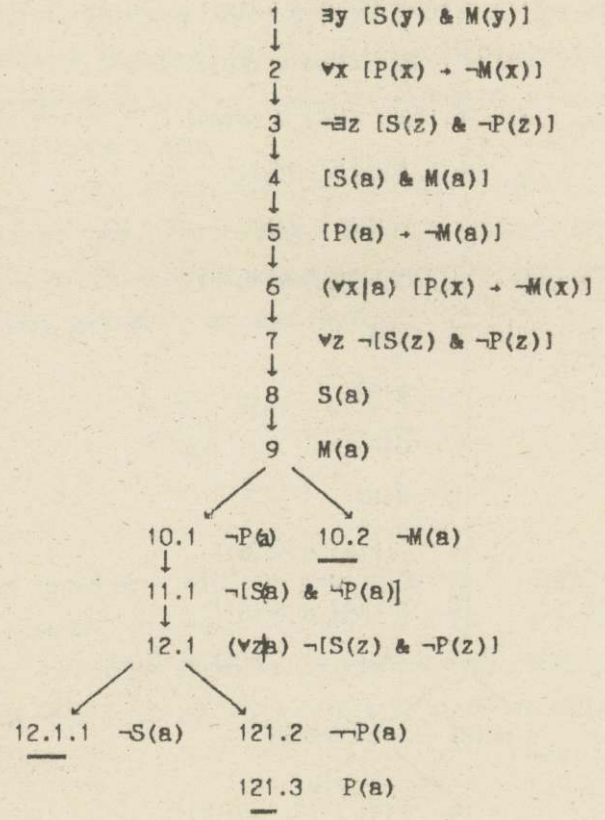


Рис.5

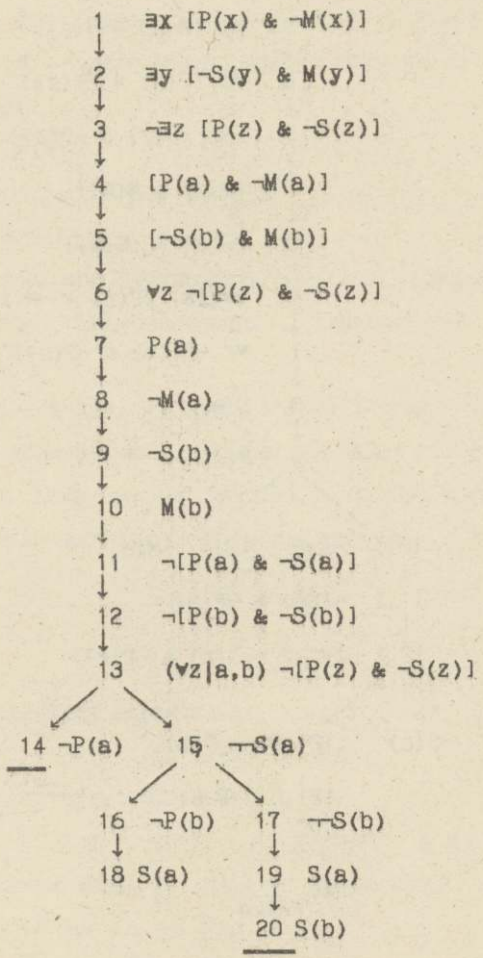


рис.6

При построении этого дерева мы ввели два параметра: a – на шаге 4 и b – на шаге 5. В итоге пришли к дереву, в котором все формулы, кроме формулы 13, не являющиеся атомарными и отрицаниями атомарных, использованы для расширения дерева. При этом ветвь, заканчивающаяся формулой 18, не содержит противоречия. Выбирая все атомарные формулы и отрицания атомарных формул, входящие в эту ветвь, получим список

$$S(a),\ \neg P(b),\ M(b),\ \neg S(b),\ \neg M(a),\ P(a).$$

Используя эти предложения, легко построить структуру на универсе {a,b}, в которой посылки истинны, а заключение ложно. Эту структуру зададим с помощью таблицы

x	P(x)	M(x)	S(x)
a	1	0	1
b	0	1	0

Легко проверить, что она удовлетворяет требуемому условию. Наличие такой структуры говорит о том, что предложение $\exists z\ [P(z)\ \&\ \neg S(z)]$ не является логическим следствием предложений $\exists x\ [P(x)\ \&\ \neg M(x)]$ и $\exists y\ [\neg S(y)\ \&\ M(y)]$. Такие структуры называют контрмоделями.

Пусть Г – множество предложений сигнатуры σ и A – некоторая формула той же сигнатуры.

Для доказательства утверждений вида $\Gamma \vdash A$ или поиска контрмоделей для таких утверждений будем строить поисковые корневые деревья, узлы которых будем помечать формулами сигнатуры $\sigma \cup D$, где $D=\{a_1, a_2, \ldots\}$ – счетное множество констант, такое, что

$\sigma \cap D = \varnothing$. Для простоты будем предполагать, что σ своих констант не имеет. Константы из D будут использоваться при применении $(\forall)$, $(\exists)$ - правил к узлам дерева; будем называть их параметрами.

Последовательность узлов, а также приписанных им формул, от корня до концевого узла будем называть ветвью. Ветвь назовем блокированной, если она содержит некоторую формулу B и ее отрицание $\neg B$.

Определение <u>поискового дерева</u> для утверждения $\Gamma \to A$ дадим с помощью правил его построения.

П1) Дерево, состоящее из одного узла, помеченного формулой $\neg A$, является поисковым деревом с корнем $\neg A$. Единственный узел этого дерева считаем неиспользованным.

П2) Если в дереве есть неиспользованный узел v, которому приписана формула, являющаяся посылкой одного из правил вывода, то каждая неблокированная ветвь W, проходящая через этот узел, расширяется следующим образом:

если правило разветвляющее, то из концевого узла ветви w проводятся две дуги, оканчивающиеся новыми вершинами, которым приписываются формулы-заключения данного правила;

если правило, соответствующее узлу v, - неразветвляющее, то к концевому узлу ветви W присоединяются последовательно один к другому новые узлы, помеченные формулами-заключениями.

Уточнения требуют случаи применения правил $(\forall)$ и $(\exists)$, поскольку они связаны с выбором параметра a. При использовании правила $(\forall)$ в качестве a выбирается параметр с наименьшим номером, не входящий в список α в посылке данного правила. При использовании правила $(\exists)$ выбирается параметр с наименьшим номером, не встречающийся в расширяемой ветви, в том числе и в

списке α. После расширения дерева считаем узел v использованным, а вновь построенные узлы - неиспользованными.

П3) Дерево также можно расширить, добавляя к концевым узлам неблокированных ветвей новый узел, помеченный очередной формулой из Γ или формулой вида $[A \vee \neg A]$, и считать его неиспользованным.

Очевидно, с помощью правил 1-3 поисковое дерево для утверждения $\Gamma \to A$ определяется неоднозначно. С помощью сформулированных правил для любого утверждения вида $\Gamma \to A$ может быть построена, вообще говоря, неоднозначно, последовательность $T_1, T_2, \ldots$ поисковых деревьев, в которой дерево T_1 построено по правилу П1), а каждое следующее получено из предыдущего по одному из правил П2) или П3). Любую такую последовательность будем называть поисковой. Если она бесконечна, то $\bigcup_{i=0}^{\infty} T_i$ также будем называть поисковым деревом.

ЛЕММА (о поисковых последовательностях)

Пусть $T_1, T_2, \ldots$ - поисковая последовательность для утверждения $\Gamma \to A$ и D_i - множество параметров, используемых в дереве T_i $(i=1,2,\ldots)$, тогда, если существует интерпретация I сигнатуры σ на универсе U, в которой истинны все формулы из множества Γ и формула $\neg A$, то для каждого $i=1,2,\ldots$ существует интерпретация I_i сигнатуры $\sigma \cup D_i$, являющаяся расширением I, такая, что в T_i есть ветвь, все формулы которой истинны в I_i.

ДОКАЗАТЕЛЬСТВО. Для $i=1$ утверждение леммы очевидно. Пусть существует интерпретация I_{i-1} сигнатуры $\sigma \cup D_{i-1}$, являющаяся расширением интерпретации I, такая, что в T_{i-1} есть ветвь W, все формулы которой истинны в I_{i-1}, и пусть T_i получено из T_{i-1} применением некоторого правила вывода к узлу v.

Если $v \in W$, то W будет искомой ветвью и в дереве T_i, при этом интерпретация символов из $D_i \setminus D_{i-1}$, если такие есть, может быть произвольной.

Пусть теперь $v \in W$. Если v имеет вид $(\forall x|\alpha)C$, и $D_i = D_{i-1}$, то формулы, присоединяемые к концевой вершине ветви в соответствии с правилом вывода, будут истинны в интерпретации $I_i = I_{i-1}$, если же D_i получено из D_{i-1} добавлением нового параметра a_ν, то берем в качестве $a_\nu^{(I)}$ произвольный элемент из U, тогда присоединяемые к W формулы $C_x[a_\nu]$, $(\forall x|\alpha, a_\nu)C$ будут истинными в полученной интерпретации I_i.

Пусть теперь v имеет вид $(\exists x|\alpha)C$ и при этом D_i получено из D_{i-1} добавлением параметра a_ν. Поскольку формула $(\exists x|\alpha)C$ истинна в интерпретации I_{i-1}, то существует интерпретация I', совпадающая с I_{i-1} повсюду, кроме, может быть, переменной x, такая, что $C^{(I')}=1$. Положим $a_\nu^{(I)} = x^{(I')}$, и тогда присоединяемая к W формула $C_x[a_\nu]$ будет истинной в I_i.

Рассмотрение остальных способов расширения дерева предоставляем читателю.

Если все ветви поискового дерева, построенного для $\Gamma \Rightarrow A$, блокированы, то такое дерево называется <u>доказательством</u> (в виде дерева) формулы A из посылок (гипотез) Γ. Заметим, что такое дерево нельзя расширить с помощью сформулированных выше правил П1-П3.

Если для утверждения $\Gamma \Rightarrow A$ существует дерево-доказательство, то будем говорить, что формула A <u>выводима</u> из множества гипотез Γ, и обозначать $\Gamma \vdash A$. Основанием для такого определения служит следующая теорема.

ТЕОРЕМА (о корректности дедуктики)

Если $\Gamma \vdash A$, то $\Gamma \Rightarrow A$.

ДОКАЗАТЕЛЬСТВО. Пусть $\Gamma \vdash A$ и T - соответствующее дерево-доказательство. Если бы A логически не следовала из Γ, то существовала бы интерпретация I сигнатуры σ, в которой были бы истинны все формулы из Γ и формула $\neg A$, но тогда по лемме о поисковой последовательности существовала бы ветвь в дереве T, все формулы которой были бы истинны в некотором расширении интерпретации I, что противоречит тому, что в T все ветви блокированы. Теорема доказана.

Ветвь W называется <u>насыщенной</u> относительно $D' \subseteq D$, если:

- вместе с любой посылкой любого разветвляющего правила она содержит хотя бы одну заключительную формулу данного правила;
- вместе с любой посылкой неразветвляющего правила она содержит все заключительные формулы данного правила;
- она содержит каждую формулу из Γ.

Уточнения, как и прежде, требуют правила $(\forall)$ и $(\exists)$, а именно, если "$(\forall x|\alpha)C$"$\in W$, то для любого $a \in D'$ "$C_x[a]$"$\in W$; если "$(\exists x|\alpha)C$"$\in W$, то по крайней мере для одного $a \in D'$ "$C_x[a_\nu]$"$\in W$.

Поисковое дерево называется полным относительно множества параметров D', если любая его ветвь либо насыщена относительно D', либо блокирована.

ЛЕММА (о существовании полного дерева)

Для любого утверждения вида $\Gamma \Rightarrow A$ существует полное поисковое дерево.

ДОКАЗАТЕЛЬСТВО. Уточним правила построения поисковых деревьев следующим образом. В правиле П2) в качестве v будем

выбирать узел минимальной глубины, удовлетворяющий остальным требованиям. Если хотя бы одно из правил П2) и П3) применимо, то обязательно его применяем.

Используя уточненные правила, либо через конечное число шагов получим дерево, в котором каждая ветвь блокирована или насыщена, либо процесс будет продолжаться бесконечно, тогда об'единение $\bigcup_{i=1}^{\infty} T_i$, где T_i - дерево, полученное на 1-ом шаге, будет искомым деревом, в котором каждая бесконечная ветвь будет насыщенной, а каждая конечная будет блокированной или насыщенной.

ТЕОРЕМА (о полноте дедуктики)

Если $\Gamma \Rightarrow A$, то $\Gamma \vdash A$.

ДОКАЗАТЕЛЬСТВО. Пусть A невыводима из Γ, тогда полное дерево T для утверждения $\Gamma \Rightarrow A$ содержит по крайней мере одну неблокированную насыщенную ветвь W. Пусть D' - список параметров, входящих в эту ветвь (он может быть как конечным, так и бесконечным).

Рассмотрим D' в качестве универса и зададим интерпретацию I сигнатуры σ на D' следующим образом. Для k-местного предикатного символа $R \in \sigma$ положим

$$R^{(I)} = \{(a_{i_1}, a_{i_2}, \ldots, a_{i_k}) \mid \text{"}R(a_{i_1}, a_{i_2}, \ldots, a_{i_k})\text{"} \in W\}.$$

Легко доказать индукцией по построению формулы, учитывая правила построения дерева, что при такой интерпретации все формулы, входящие в ветвь W, будут истинными. В том числе будут истинны все формулы из Γ и формула $\neg A$, следовательно, A логически не следует из Γ.

Об'единяя теоремы о корректности и полноте, получим следующую теорему.

ТЕОРЕМА (об адекватности дедуктики)

$\Gamma \Rightarrow A$ тогда и только тогда, когда $\Gamma \vdash A$.

ТЕОРЕМА (о компактности)

Если $\Gamma \Rightarrow A$, то существует конечное подмножество $\Gamma' \subseteq \Gamma$ такое, что $\Gamma' \Rightarrow A$.

Действительно, пусть $\Gamma \Rightarrow A$, тогда по теореме о полноте $\Gamma \vdash A$, следовательно, существует дерево-доказательство, в котором все ветви блокированы и, следовательно, конечны. Следовательно, число гипотез из Γ, содержащихся на этих ветвях, конечно, они и образуют требуемое конечное подмножество гипотез.

УПРАЖНЕНИЯ

I. Построить поисковые деревья для следующих утверждений:

а) $\{ \forall x[P(x) \to \forall y R(x,y)], \exists x \forall y \neg R(x,y), \forall x[p(x) \to \neg Q(x)] \} \Rightarrow \exists x[Q(x) \& \forall y \neg R(x,y)]$;

б) $\{ \forall x[P(x) \to \forall y R(x,y)], \exists x \forall y \neg R(x,y) \} \Rightarrow \exists x[\neg P(x) \& \forall y \neg R(x,y)]$;

в) $\{ \forall x[Q(x) \to \exists y[S(y) \& R(x,y)]], \forall x \forall y \forall z[R(x,y) \& R(y,z) \to R(x,z)], \exists x[P(x) \& \forall y[\neg S(y) \vee \neg R(x,y)]] \} \Rightarrow \exists x[P(x) \& \forall y[\neg Q(y) \vee \neg R(x,y)]]$;

г) $\{ \forall x[P(x) \to \neg Q(x)], \forall x \forall y[R(X,y) \to P(x) \& Q(y)] \} \Rightarrow \forall x \forall y[Q(x) \& P(y) \to \neg R(x,y)]$;

д) $\{ \forall x[P(x) \to \neg Q(x) \vee \neg R(x) \vee \neg S(x)], \forall x[Q(x) \to P(x) \vee \neg R(x) \vee \neg S(x)] \} \Rightarrow \forall x[R(x) \& Q(x) \to \neg S(x)]$;

е) $\{ \exists x[P(x) \& \forall y[S(y) \to R(x,y)], \forall x[P(x) \to \forall y[Q(y) \to \neg R(x,y)]] \} \Rightarrow \forall x[S(x) \to \neg Q(x)]$.

§2. Канонические формы предложений в логике первого порядка

Две различные формулы в языке $L(\sigma)$ могут оказаться логически равносильными. В качестве примера рассмотрим пару формул

$$A = [\exists x P(x) \vee \exists y Q(y)] \quad \text{и}$$

$$B = \exists x\, [P(x) \vee Q(x)],$$

где P и Q - одноместные предикатные символы. Легко построить доказательства для утверждений $A \to B$ и $B \to A$.

При рассмотрении классов логически равносильных формул удобно бывает в каждом классе выделить формулу какого-либо стандартного вида. В качестве такого стандартного вида часто используют так называемую префиксную (или предваренную нормальную) форму предложений из $L(\sigma)$.

Формула A называется <u>префиксной</u>, если она имеет вид

$$Q_1 y_1\, Q_2 y_2 \ldots Q_n y_n\, B,$$

где $Q_1, Q_2, \ldots, Q_n$ - кванторы, $y_1, y_2, \ldots, y_n$ - индивидные переменные, B - формула из $L(\sigma)$, не содержащая кванторов.

Существует алгоритм, который для каждой формулы из $L(\sigma)$ строит логически равносильную ей префиксную формулу.

Действие алгоритма основано на применении следующих правил вынесения кванторов и замены связанной переменной:

а) $\neg\exists x A \equiv \forall x \neg A$,

б) $\neg\forall x A \equiv \exists x \neg A$,

в) $\exists x A \& B \equiv \exists x\, [A \& B]$,

г) $\exists x A \vee B \equiv \exists x\, [A \vee B]$,

д) $\forall x A \& B \equiv \forall x\, [A \& B]$,

е) $\forall x A \vee B \equiv \forall x\, [A \vee B]$,

ж) $B \& \exists x A \equiv \exists x\, [B \& A]$,

з) $B \vee \exists x A \equiv \exists x\, [B \vee A]$,

и) $B \& \forall x A \equiv \forall x\, [B \& A]$,

к) $B \vee \forall x A \equiv \forall x\, [B \vee A]$,

л) $\forall x A \equiv \forall y\, A_x[y]$,

м) $\exists x A \equiv \exists y\, A_x[y]$,

где A и B - формулы, переменная x не имеет свободных вхождений в B, переменная y (в правилах л), м)) не имеет свободных вхождений в A.

Указанные выше равносильности легко доказываются методом построения деревьев.

УПРАЖНЕНИЯ

1. Доказать правила вынесения кванторов.

2. Для перечисленных ниже формул построить равносильные им префиксные формулы:

а) $\forall x\, P(x) \vee \forall x\, Q(x)$,

б) $\forall x\, P(x) \& \forall x\, Q(x)$,

в) $\exists x\, P(x) \vee \exists x\, Q(x)$,

г) $\exists x\, P(x) \& \exists x\, Q(x)$,

д) $\forall x\, P(x) \to \forall x\, Q(x)$,

е) $\exists x\, P(x) \to \forall x\, Q(x)$,

ж) $\forall x\, P(x) \to \exists x\, Q(x)$.

Формула называется сингулярной, если все ее нелогические символы являются одноместными предикатами. Сингулярная формула называется примарной, если она имеет вид

$$\forall x [\bar{P}_1(x) \vee \bar{P}_2(x) \vee \ldots \vee \bar{P}_k(x)]$$

или

$$\exists x [\bar{P}_1(x) \& \bar{P}_2(x) \& \ldots \& \bar{P}_k(x)],$$

где $P_1, P_2, \ldots, P_k$ - одноместные предикатные символы, а знак $\tilde{}$ означает, что перед соответствующим символом может находится знак отрицания - $\neg$.

Существует алгоритм, который любую замкнутую сингулярную формулу преобразует в логически равносильную ее булеву комбинацию примарных формул.

Действие алгоритма основано на следующем преобразовании. Находится подформула A, начинающаяся с квантора, в области действия которого расположена булева комбинация примарных и атомарных формул. Пусть она имеет вид $\forall xB$. Подформула B представляется в виде кон'юнктивной нормальной формы (к.н.ф.) вида $B_1 \& B_2 \& \ldots \& B_s$, где каждое B_i $(i=1,2,\ldots,s)$ есть диз'юнкция примарных и атомарных формул или их отрицаний. Подформула $\forall xB$ заменяется подформулой $\forall xB_1 \& \forall xB_2 \& \ldots \& \forall xB_s$. Далее, каждая подформула B_i представляется в виде

$$\tilde{C}_1 \vee \tilde{C}_2 \vee \ldots \vee \tilde{C}_t \vee \tilde{P}_1(x) \vee \tilde{P}_2(x) \vee \ldots \vee \tilde{P}_r(x),$$

где $C_i (i=1,2,\ldots,t)$ либо примарная, либо атомарная с переменной отличной от x, а $P_1, P_2, \ldots, P_r$ - одноместные предикатные символы. Затем подформула $\forall xB_i$ заменяется подформулой вида

$$\tilde{C}_1 \vee \tilde{C}_2 \vee \ldots \vee \tilde{C}_t \vee \forall x[\tilde{P}_1(x) \vee \tilde{P}_2(x) \vee \ldots \vee \tilde{P}_r(x)].$$

Если исходная подформула A имеет вид $\exists xB$, то поступают двойственным образом, используя вместо к.н.ф. диз'юнктивную нормальную форму (д.н.ф.).

Разработка деталей алгоритма и его доказательство предоставляется читателю.

Процесс приведения сингулярной формулы к виду булевой комбинации примарных является в некотором смысле обратным к

процессу приведения к префиксному виду и характеризуется продвижением кванторов вглубь формулы.

УПРАЖНЕНИЕ

Представить следующие сингулярные формулы в виде булевой комбинации примарных формул:

а) $\exists x \forall y[[P(x) \& \neg Q(y)] \to R(y)]$,

б) $\forall x \exists y[[P(x) \vee P(y) \& Q(y)]$,

в) $\forall x \exists y \forall z[P(x) \& Q(z) \vee \neg P(y) \& \neg Q(z)]$.

Если формула A содержит хотя бы один более чем одноместный предикатный символ, процесс продвижения кванторов внутрь формулы может не привести к такому же простому виду как в случае сингулярных формул. Однако, максимально возможное продвижение кванторов внутрь может привести формулу A к логически равносильной ей антипрефиксной формуле. Введем соответствующие определения.

Вхождение атомарной части в формулу A назовем свободным если оно не находится в области действия квантора ни по одной из своих переменных.

Формулу назовем <u>атомарно замкнутой</u>, если она не имеет свободных атомарных частей.

Формулу называют <u>антипрефиксной</u>, если каждая ее подформула, начинающаяся с квантора имеет один из следующих двух видов

$$(\exists x | y_1 y_2 \ldots y_k)[C \vee D], \qquad (1)$$

$$(\forall x | y_1 y_2 \ldots y_k)[C \& D], \qquad (2)$$

где C - бескванторная формула, каждая атомарная часть которой содержит переменную x, а D - атомарно-замкнутая формула. Одна из

формул C,D может быть тождественно истинной или тождественно ложной, а список переменных $x, y_1, y_2, \ldots y_k$ содержит в себе все свободные переменные формул C,D.

Нетрудно показать, что каждая атомарно замкнутая формула логически равносильна булевой комбинации формул вида (I),(2).

Процесс приведения к антипрефиксному виду начинается с выбора самой внутренней подформулы, начинающейся с квантора. Если эта формула имеет вид $\forall xA$, то A приводится к кон'юнктивной нормальной форме и квантор $\forall x$ распределяется по кон'юнктивным членам. Если же найденная подформула имеет вид $\exists xA$, то A приводится к диз'юнктивной нормальной форме и квантор $\exists x$ распределяется по диз'юнктивным членам. После этих преобразований найденная подформула приводится к антипрефиксному виду. Читателю предоставляется возможность завершить формулировку процедуры приводящей любую формулу к антипрефиксному виду.

УПРАЖНЕНИЕ

Привести к антипрефиксному виду формулы:

1) $\forall x \forall y \forall z [R(x,x) \& [R(x,y) \to R(y,x)] \& [R(x,y) \& R(y,z) \to R(x,z)]]$.

2) $\forall x \forall y \forall z [R(x,y) \& R(y,z) \to R(x,z)]$.

§ 3. Моделирование математических теорий

В математике можно выделить два способа формирования теорий. Один из них называется аксиоматическим и заключается в том, что некоторые утверждения из каких-либо содержательных соображений об'являются аксиомами. Развитие теории в этом случае состоит в изучении ее интерпретаций и в получении следствий из аксиом.

Примерами таких теорий являются теории групп, колец, полей и др.

Второй способ формирования теории начинается с изучения какой-либо конкретной математической структуры или класса структур и тогда, естественно, возникает вопрос о полной аксиоматизации этого класса, т.е. о выборе множества аксиом так, чтобы множество следствий из этих аксиом совпадало с множеством истинных в рассматриваемом классе структур утверждений.

Естественным методологическим требованием к множеству аксиом при этом является требование его алгоритмической разрешимости. Это требование заключается в том, чтобы существовал способ, позволяющий по любому утверждению отвечать на вопрос, является ли оно аксиомой. Известно, например, что для многих существенных для математики структур в языке, содержащем логику первого порядка, не существует полной разрешимой аксиоматики. Примером такой структуры является множество натуральных чисел с обычными операциями +, × и предикатом равенства.

Множество предложений, истинных в заданной структуре S, называют теорией структуры S и обозначают Th(S).

Опираясь на наши рассуждения, введем следующие определения.

Пусть σ - нелогическая сигнатура и Г - множество предложений в языке $\mathfrak{L}(\sigma)$. Пару $T=(\sigma,\Gamma)$ будем называть аксиоматической теорией, а множество Г - множеством аксиом этой теории. Логические следствия из аксиом, выраженные в сигнатуре σ, называются теоремами теории T. Часто аксиоматическую теорию отождествляют с множеством ее теорем, то есть с логическим замыканием множества Г. Логическое замыкание множества Г обозначим через [Г].

В общем случае под элементарной теорией будем понимать любое

логически замкнутое множество предложений рассматриваемой сигнатуры.

§ 4. Примеры формализации математических теорий

1. Теория групп

В качестве первого примера рассмотрим сигнатуру $\sigma=\{=,\cdot,e\}$, где = - предикат равенства, $\cdot$ - двуместный функциональный символ, e - константа.

Пусть Г состоит из предложений

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

$\forall x((x\cdot e)=x)$,

$\forall x\exists y((x\cdot y)=e)$.

Множество всех теорем теории $G=(\sigma,\Gamma)$ составляет так называемую элементарную теорию групп, являющуюся фрагментом математической теории групп, полную формализацию которой можно провести, например, в рамках теории множеств. Этот вопрос остается за рамками данного пособия.

2. Теория отношения эквивалентности

Это теория структур с одним универсом и одним бинарным отношением R, удовлетворяющим аксиомам

1) $(\forall x)R(x,x)$,

2) $(\forall x)(\forall y)[R(x,y) \to R(y,x)]$,

3) $(\forall x)(\forall y)(\forall z)[R(x,y) \,\&\, R(y,z) \to R(x,z)]$.

3. Теория упорядоченных множеств

Структура $(A,<\ )$, где инфикс - '<' обозначает бинарное отношение, удовлетворяющее аксиомам

1) $(\forall x)\neg(x<x)$,

2) $(\forall x)(\forall y)(\forall z)[(x<y) \,\&\, (y<z) \to (x<z)]$,

3) $(\forall x)(\forall y)[(x<y) \vee (x=y) \vee (y<x)]$,

называется упорядоченным множеством. Примером такой структуры может быть множество вещественных чисел с обычным отношением '<'.

4. Теория полей

Пусть P - произвольное множество; '+','×' - бинарные операции на P, а '0','1' - выделенные элементы из P. Теория полей это теория структур вида $(P;+,\times,0,1)$, задаваемая аксиомами

1) $(\forall x)(\forall y)(\forall z)[(x+y)+z=x+(y+z)]$,

2) $(\forall x)(\forall y)(\forall z)[(x\times y)\times z=x\times(y\times z)]$,

3) $(\forall x)(\exists y)(x+y=0)$,

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

5) $\neg(0=1)$,

6) $(\forall x)(\forall y)(\forall z)[x\times(y+z)=(x\times y)+(x\times z)]$,

7) $(\forall x)(x+0=x)$,

8) $(\forall x)(x\times 1=x)$,

9) $(\forall x)[\neg(x=0) \to (\exists y)(x\times y=1)]$,

10) $(\forall x)(\forall y)(x\times y=y\times x)$.

5. Теория упорядоченных полей

Это теория структур вида $(P;+,\times,0,1,<)$, аксиомами которой являются аксиомы теории упорядоченных множеств, аксиомы теории полей и еще две аксиомы

$(\forall x)(\forall y)(\forall z)[(x<y) \to (x+z)<(y+z)]$,

$(\forall x)(\forall y)(\forall z)[(x<y) \,\&\, (0<z) \to (x\times z)<(y\times z)]$,

связывающие бинарные операции с отношением порядка.

Упорядоченное поле называется пифагоровым, если оно удовлетворяет аксиоме

$$(\forall x)(\forall y)(\forall z)(x\times x+y\times y=z\times z).$$

Упорядоченное поле называется евклидовым, если оно удовлетворяет аксиоме

$$(\forall x)(\exists y)(x\geq 0\rightarrow x=y\times y)).$$

6. Евклидова планарная геометрия

В качестве универса рассмотрим множество M точек евклидовой плоскости и на этом множестве - два предиката D и B, заданные следующим образом.

D(a,b,c,d) - "расстояние между точками a,b равно расстоянию между точками c,d".

B(a,b,c) - "точки a,b,c лежат на одной прямой, причем b расположена между a и c".

С помощью введенных предикатов средствами логики первого порядка можно формулировать геометрические утверждения, составляющие предмет изучения в школьной планиметрии.

Например, формула

$$a\neq b \;\&\; a\neq d \;\&\; b\neq d \;\&\; D(a,b,c,d) \;\&\; D(a,c,b,d)$$

выражает тот факт, что точки a,b,c,d являются вершинами параллелограмма. Заметим, что порядок обхода вершин этой формулой не определяется.

А.Тарским написана система аксиом, из которой логически следуют все истинные утверждения планарной геометрии. В приведенных ниже 16 аксиомах опущены кванторы общности по всем свободным переменным, а запись $(\exists p,q\ldots)$ следует рассматривать как $(\exists p)(\exists q)\ldots$.

1) $D(a,b,b,a)$,

2) $D(a,b,c,c) \rightarrow a=b$,

3) $D(a,b,c,d) \;\&\; D(a,b,e,f) \rightarrow D(c,d,e,f)$,

4) $(\exists p)[D(a,c,b,c) \;\&\; D(b,c,b,d) \;\&\; D(b,d,a,d) \rightarrow D(a,p,b,p) \;\&\; D(c,p,d,p)]$,

5) $(\exists p,q)[q\neq c \;\&\; q\neq p \;\&\; [[D(a,q,a,c) \;\&\; D(b,q,b,c)] \vee [D(a,p,a,q) \;\&\; D(b,p,b,q) \;\&\; D(c,p,c,q)]]$,

6) $(\exists p,q)[p\neq q \;\&\; [[D(a,p,b,p) \;\&\; D(b,p,c,p)] \vee [D(a,p,a,q) \;\&\; D(b,p,b,q) \;\&\; D(c,p,c,q)]]]$,

7) $D(a,p,c,p) \;\&\; D(b,p,d,p) \;\&\; D(a,q,c,q) \;\&\; D(b,q,d,q) \rightarrow a=c \vee b=d \vee p=q \vee D(a,b,c,d)$,

8) $a\neq b \;\&\; c\neq d \;\&\; p\neq q \;\&\; D(a,p,b,p) \;\&\; D(a,q,b,q) \;\&\; D(c,p,d,p) \;\&\; D(c,q,d,q) \;\&\; D(a,s,b,s) \rightarrow D(c,s,d,s)$,

9) $(\exists a,b,c,d)[a\neq b \;\&\; D(a,b,b,c) \;\&\; D(b,c,c,d) \;\&\; D(c,d,d,a) \;\&\; D(a,c,b,d)]$,

10) $(\exists b)[D(a,p,a,q) \;\&\; a\neq c \rightarrow D(b,p,b,q) \;\&\; D(a,b,a,c)]$,

11) $B(a,b,c) \rightarrow B(c,b,a)$,

12) $B(a,b,d) \;\&\; B(b,c,d) \rightarrow B(a,b,c)$,

13) $(\exists u,v)[B(a,b,c) \rightarrow u\neq v \;\&\; D(a,u,a,v) \;\&\; D(b,u,b,v) \;\&\; D(c,u,c,v)]$,

14) $u\neq v \;\&\; D(a,u,a,v) \;\&\; D(b,u,b,v) \;\&\; D(c,u,c,v) \rightarrow B(a,b,c) \vee B(b,c,a) \vee B(c,a,b)$,

15) $(\exists q)[B(a,p,d) \;\&\; B(b,d,c) \rightarrow [B(b,p,q) \vee B(p,q,b) \vee B(q,b,p)] \;\&\; [B(a,q,c) \vee B(q,c,a) \vee B(c,a,q)]]$,

16) $(\exists p,q)[a\neq b \;\&\; b\neq d \;\&\; B(a,b,c) \;\&\; D(b,a,b,c) \;\&\; B(a,b,d) \rightarrow B(a,p,q) \;\&\; D(p,a,p,q) \;\&\; D(d,a,d,q) \;\&\; D(p,a,p,c)]$.

§5. Свойства элементарных теорий

Теория T заданной сигнатуры σ называется полной, если для любого предложения $A \in Sent(\sigma)$ либо A либо $\neg A$ принадлежит теории T.

Если рассматриваемая теория T является теорией Th(S) некоторой σ-структуры S, то T полна, так как любое σ-предложение либо истинно либо ложно в структуре S.

Если же теория задана аксиоматически, то ее полнота не гарантирована. Например, теория групп не полна, так как предложение, выражающее коммутативность группы, и его отрицание не являются следствием групповых аксиом, поскольку в математике имеются как примеры коммутативных, так и некоммутативных групп.

Теория $T \subset Sent(\sigma)$ называется алгоритмически разрешимой, если существует алгоритм, который по любому предложению $A \in Sent(\sigma)$ отвечает на вопрос принадлежит ли A теории T.

Для доказательства разрешимости многих теорий применим так называемый метод элиминации кванторов. Этот метод применим к теориям, для которых может быть выполнен элементарный шаг элиминации, то есть для любой формулы A вида

$$\exists x[A_1(x) \& A_2(x) \& \ldots \& A_n(x)],$$

где $A_i(x)$ - атомарная формула или отрицание атомарной формулы, может быть построена бескванторная формула B, такая что $A \leftrightarrow B$ истинна в рассматриваемой теории.

Если это условие выполнено, то к каждому предложению $A \in Sent(\sigma)$, где σ - сигнатура рассматриваемой теории, применима следующая процедура.

I) Превратить с помощью правил де Моргана самые внутренние кванторы в кванторы существования, если они не являются таковыми.

2) Привести область действия каждого из этих кванторов к диз'юнктивной нормальной форме.

3) Распределить кванторы существования по диз'юнктивным членам.

4) Элиминировать их с помощью элементарных шагов элиминации.

5) Если результат все еще не является бескванторным, то повторить описанные шаги, в противном случае рассматриваемое выражение превратилось в истину или ложь.

Рассмотрим метод элиминации кванторов в применении к теории плотного линейного порядка без концевых точек. Это теория с одним нелогическим символом $\leq$, удовлетворяющая аксиомам

1) $\forall x \forall y \forall z[(x \leq y) \& (y \leq z) \to (x \leq z)]$,

2) $\forall x \forall y[(x \leq y) \& (y \leq x) \to (x = y)]$,

3) $\forall x(x \leq x)$,

4) $\forall x \forall y[(x \leq y) \vee (y \leq x)]$,

5) $\forall x \forall y[(x \leq y) \& \neg(x = y) \to \exists z[(x \leq z) \& (z \leq y) \& \neg(x = z) \& \neg(y = z)]]$,

6) $\exists x \exists y \neg(x = y)$.

7) $\forall x \exists y[(x \leq y) \& \neg(x = y)]$,

8) $\forall x \exists y[(y \leq x) \& \neg(x = y)]$.

Заметим, что единственной, с точностью до изоморфизма, счетной моделью этой теории является структура $(Q; \leq)$, где Q - множество рациональных чисел.

Основным случаем в процессе элиминации кванторов является случай, когда формула, начинающаяся с самого внутреннего квантора, имеет вид

$$\exists z\ [\mathop{\&}_{i=1}^{k}(x_i \leq z)\ \&\ \mathop{\&}_{j=1}^{m}(z \leq y_j)].$$

В процессе элиминации она заменяется логически равносильной ей бескванторной формулой

$$\mathop{\&}_{i=1}^{k} \mathop{\&}_{j=1}^{m} (x_i \le y_j)$$

Читателю предоставляется возможность разработать в деталях алгоритм элиминации кванторов для рассматриваемой теории.

§6. Некоторые замечания о возможностях формализации математических теорий

При определении понятия структуры мы полагали, что функциональные символы интерпретируются как всюду определенные функции. В некоторых случаях это приводит к затруднениям. Например, рассматривая операцию деления в области R вещественных чисел как функцию, замечаем, что она не всюду определена из-за невозможности деления на нуль. Можно предложить два выхода из этого положения: либо доопределить эту функцию, либо рассматривать операцию деления как трехместное отношение

$$\{(a,b,c) \mid b \ne 0 \ \& \ a/b=c\}.$$

Радикальным выходом было бы развивать семантику с использованием частичных функций, однако, это привело бы к усложненной системе и, как мы только что видели, не внесло бы ничего нового.

Другим затруднением является то, что многие важные математические структуры содержат элементы различных типов. Например, линейное векторное пространство состоит из векторов и скаляров.

В таких случаях можно было бы рассматривать так называемые многосортные структуры и, соответственно, языки, либо воспользоваться техникой об'единения сортов. Продемонстрируем обе эти возможности на упомянутом выше примере линейного векторного

пространства.

(1) Рассмотрим векторное пространство как структуру с двумя универсами P и V

$$(P,V;+,\times,0,1,\oplus,\theta,\cdot),$$

где P - поле скаляров, V - множество векторов, $\langle V,\oplus,\theta\rangle$ - аддитивная коммутативная группа, $\langle P,+,\times,0,1\rangle$ - поле, а $\cdot$ - отображение $P\times V \to V$.

Напомним, что $\oplus$ - операция сложения векторов, θ - нулевой вектор, а $\cdot$ - операция умножения векторов на скаляры.

Чтобы описывать такую двухсортную структуру, введем двухсортный язык, который будет отличаться от введенного ранее тем, что в нем будут использоваться переменные двух сортов: переменные одного сорта $x_1, x_2, \ldots$, значениями которых будут скаляры, и переменные другого сорта $v_1, v_2, \ldots$, значениями которых будут векторы. Для иллюстрации приведем некоторые аксиомы теории линейных векторных пространств.

Ассоциативность сложения скаляров

$$(\forall x_1)(\forall x_2)(\forall x_3)[(x_1+(x_2+x_3))=((x_1+x_2)+x_3)].$$

Ассоциативность сложения векторов

$$(\forall v_1)(\forall v_2)(\forall v_3)[(v_1\oplus(v_2\oplus v_3))=((v_1\oplus v_2)\oplus v_3)].$$

(2) То же самое векторное пространство можно рассматривать как структуру с одним универсом $P\cup V$, применяя, так называемую, редукцию сортов. Чтобы различать векторы и скаляры в этом "смешанном" универсе, вводят два одноместных предиката P,V, считая P(x) - истинным тогда и только тогда, когда $x\in P$, а V(x) - тогда и только тогда, когда $x\in V$. В аксиомах для каждой из переменных должно быть указано, какому из предикатов P,V она удовлетворяет. Так одна из аксиом будет выглядеть следующим

образом

$$(\forall x)(\forall y)(\forall z)[P(x)\&V(y)\&V(z)\to x\cdot(y\circ z)=x\cdot y\circ x\cdot z].$$

Рассмотрим пример, в котором встречаются качественно другие трудности формализуемости математических теорий. Группой с вращением называется группа, в которой каждый элемент имеет конечный порядок, то есть для каждого элемента a из группы найдется натуральное число n такое, что a^n равно нейтральному элементу e. Это свойство можно было бы описать бесконечной формулой

$$(\forall x)[x=e \vee x\times x=e \vee x\times x\times x=e \vee \ldots],$$

однако, в логике первого порядка допускаются только конечные формулы.

Более того, известно что в логике первого порядка не существует множества формул, моделями которых являлись бы в точности группы с вращением.

Следующая проблема возникает из желания описать арифметическую структуру $(N;+,\times,0,1)$, где N – множество натуральных чисел, системой аксиом с точностью до изоморфизма, то есть написать такую систему аксиом, чтобы упомянутая выше структура была единственной с точностью до изоморфизма ее моделью.

Для простоты рассмотрим структуру $(N;S,0)$, где S – одноместная функция на N, такая что $S(x)=x+1$.

Рассматриваемая структура удовлетворяет системе аксиом Пеано:

(I) 0 не является значением функции S ни при каком значении аргумента;

(2) S – ин'ективное отображение;

(3) для каждого подмножества $M\subseteq N$: если $0\in M$ и для каждого x из M $S(x)$ так же принадлежит M, то $M=N$ (это так называемая аксиома индукции).

Первые две аксиомы легко формализуются в рамках логики первого порядка в сигнатуре $\{S,0\}$ в виде следующих формул:

(P1) $(\forall x)\neg(S(x)=0)$,

(P2) $(\forall x)(\forall y)[S(x)=S(y) \to x=y]$.

Третья аксиома могла бы быть легко формализована, если наряду с предикатными константами ввести в рассмотрение предикатные переменные и использовать кванторы не только по индивидным переменным, но и по предикатным. Полученный таким способом язык называется логическим языком второго порядка.

В рамках логики второго порядка аксиома (3) может выглядеть как

(P3) $\forall M[M(0)\&\forall x[M(x)\to M(S(x))]\to\forall y M(y)]$,

где M, предикатная переменная.

Известная теорема Дедекинда утверждает, что любая структура сигнатуры $\{S,0\}$ изоморфна структуре $(N;S,0)$, однако, известно, что в рамках логики первого порядка не существует системы аксиом, описывающей структуру $(N;S,0)$ однозначно с точностью до изоморфизма.

Не приводит к успеху попытка заменить аксиому (P3) бесконечным множеством аксиом вида

$$A(0)\&\forall x[A(x)\to A(S(x))]\to\forall x A(x),$$

где $A(x)$ – произвольная формула логики первого порядка, имеющая свободное вхождение переменной x, а формулы $A(0)$, $A(S(x))$ получены из A заменой свободных вхождений x на 0 и на $S(x)$, соответственно. Заметим, что множество таких формул счетно, а

множество подмножеств M, о которых идет речь в аксиоме (P3), континуально.

§7. Расширение элементарных теорий

В процессе изучения теории часто полезным оказывается ее расширение или сужение. Теория $T'=(\sigma',\Gamma')$ называется расширением теории $T=(\sigma,\Gamma)$, если $\sigma\subseteq\sigma'$ и $[\Gamma]\subseteq[\Gamma']$. Расширение T' называется консервативным, если $[\Gamma'] \cap Form(\sigma) = [\Gamma]$.

Консервативные расширения теории возникают, например, когда с помощью определений в теорию вводятся новые нелогические символы.

Например, в рассмотренной выше элементарной теории групп G можно ввести новый одноместный функциональный символ S для выражения операции обращения, а к множеству аксиом добавить предложение

$\forall x\forall y\ [y=S(x)\leftrightarrow X\cdot y=e)$.

Легко показать, что полученное расширение теории G является консервативным. Введение функционального символа S в теорию G служит примером явного определения.

В общем виде явным определением нового k-местного функционального символа $f\notin\sigma$ в теории $T=(\sigma,\Gamma)$ называется предложение вида

$$\forall x_1\forall x_2\ldots\forall x_k\forall y[y=f(x_1,x_2,\ldots,x_k)\leftrightarrow B],$$

где $B\in Form(\sigma)$, перемнные $x_1,x_2,\ldots,x_k,y$ попарно различны и этот список совпадает со списком свободных переменных формулы B.

Чтобы такое определение было корректным, необходимо, чтобы предложение

$$A = \forall x_1\forall x_2\ldots\forall x_k\exists! yB$$

было теоремой в теории T (выражение $\exists! y$ означает "существует единственное y", избавиться от этого выражения можно, заменив предложение A предложением

$$\forall x_1\forall x_2\ldots\forall x_k\exists y\forall z[y=z\leftrightarrow B_y[z]]).$$

Явным определением k-местного предикатного символа $P\notin\sigma$ в теории $T=(\sigma,\Gamma)$ называется предложение вида

$$\forall x_1\forall x_2\ldots\forall x_k[P(x_1,x_2,\ldots,x_k)\leftrightarrow B],$$

где $B\in Form(\sigma)$, и список попарно различных переменных $x_1,x_2,\ldots,x_k$ совпадает со списком свободных переменных формулы B.

Легко доказать, что расширения теорий с помощью явных опеределений являются консервативными, что согласуется с интуитивным представлением о роли определений в математических теориях.

Явная определимость нелогических символов в теориях используется не только для расширения теорий, но, когда это необходимо, и для исключения из теорий избыточных символов. Семантическим методом установления неопределимости нелогического символа теории через остальные является метод Падоа, который заключается в следующем. Чтобы доказать, что нелогический символ $P\in\sigma$ не определяется в теории $T=(\sigma,\Gamma)$ через остальные, достаточно найти две модели сигнатуры σ, отличающиеся друг от друга только интерпретацией символа P, в которых истинны все предложения из Γ.

В качестве примера рассотрим сигнатуру $\sigma=\{R,P\}$, где R,P – двухместные предикатные символы, и множество Γ, состоящее из аксиом

$\forall x\forall y\forall z\ [P(x,y)\&P(y,z)\to P(x,z)]$,

$\forall x\forall y\forall z\ [R(x,y)\&R(y,z)\to R(x,z)]$,

$\forall x\forall y\ [[P(x,y)\vee P(y,x)\vee R(x,y)]\&\neg[P(x,y)\&P(y,x)\&R(x,y)]]$.

Рассмотрим универс $U=\{1,2\}$ и две структуры на этом универсе $S_1=(U;<,=)$ и $S_2=(U;>,=)$. На обеих структурах все формулы из Γ истинны, причем символ R интерпретируется одинаково, как равенство, а символ P интерпретируется по-разному, следовательно P не определяется через R.

Неконсервативные расширения используются, когда на рассматриваемый класс структур необходимо наложить дополнительные требования. Например, добавляя к аксиомам теории G аксиому коммутативности

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

получим неконсервативное расширение, а именно, элементарную теорию коммутативных групп.

Две теории $T=(\sigma,\Gamma)$ и $T'=(\sigma',\Gamma')$ называются эквивалентными, если $\sigma=\sigma'$ и $[\Gamma]=[\Gamma']$. T и T' называются слабо эквивалентными, если некоторое расширение теории T с помощью определений эквивалентно некоторому расширению теории T' с помощью определений.

УПРАЖНЕНИЯ

I) Рассмотрим теорию G' сигнатуры $\{=,:\}$ с аксиомами

$$\forall x \forall y \forall z\ ((x:z):(y:z)=x:y),$$

$$\forall x \forall y\ ((x:(y:y))=x),$$

$$\forall x \forall y \forall z\ ((x:x):(y:z)=z:y),$$

где ":" - двухместный функциональный символ. Доказать, что теория G' слабо эквивалентна теории G.

2) Показать, что для каждой теории T существует теория T', не содержащая функциональных символов, слабо эквивалентная теории T.

Часть 3. ПРИБЛИЖЕННОЕ ВЫРАЖЕНИЕ СВОЙСТВ СТРУКТУР В ЛОГИЧЕСКИХ ЯЗЫКАХ

§ I. Доля выполнимости логических формул

В этом разделе мы рассматриваем логический язык $\mathfrak{L}(\sigma)$ первого порядка, предполагая, что нелогическая сигнатура может содержать предикат равенства и не содержит функциональных и нульместных предикатных символов. Кванторы могут быть как исключающими, так и неисключающими.

Пусть A - предложение языка $\mathfrak{L}(\sigma)$.

Долей выполнимости предложения A на универсе $U=\{1,2,\dots,n\}$ называется отношение $\gamma_n(A)$ числа структур сигнатуры σ на универсе U_n, в которых истинно предложение A, к числу всех структур сигнатуры σ на U_n.

ПРИМЕР. Пусть $A = \forall x \forall y [R(x,y) \to R(y,x)]$, где R - двухместный предикатный символ. Любую структуру сигнатуры $\sigma=\{R\}$ на универсе U_n, в которой истинна формула A, можно представить матрицей размера $n \times n$ из нулей и единиц, симметричной относительно диагонали. На саму диагональ формула A, очевидно, ограничений не накладывает. Чтобы сосчитать число таких матриц, достаточно подсчитать число способов, которыми можно заполнить диагональные и наддиагональные элементы. Поскольку число таких элементов в матрице равно $(n^2+n)/2$, а число всех элементов в матрице равно n^2, получим, что $\gamma_n(A)=2^{(n^2+n)/2}/2^{n^2}$. Заметим, что $\lim\limits_{n\to\infty} \gamma_n(A) = 0$.

В 1966 году Ю.В.Глебским было доказано, что для любого

предложения языка $\mathcal{L}(\sigma)$, где σ не содержит функциональных и нульместных предикатных символов, предел доли выполнимости при $n\to\infty$ существует и равен 0 или I.

Предел доли выполнимости $\gamma_n(A)$ предложения A при $n\to\infty$ будем называть асимптотическим значением истинности предложения A и обозначать через $\gamma(A)$.

§ 2. Разрешимость свойства асимптотической истины в логике первого порядка

Формулу A будем называть атомарно замкнутой, если каждая ее атомарная подформула вида $P(x_1,x_2,\dots,x_k)$, где P - предикатный символ, находится в области действия квантора хотя бы одной из своих переменных $x_1,x_2,\dots,x_k$.

Алгоритм вычисления асимптотического значения истинности для атомарно замкнутых формул основан на использовании так называемой антипрефиксной формы. Для ее определения удобно расширить понятие формулы, введя два дополнительных символа t и f и считая их, соответственно, тождественно истинной и тождественно ложной формулами.

Для доказательства (0-I)-закона в множестве $Sent(\sigma)$, где σ - сигнатура, не содержащая функциональных символов и нульместных предикатов, распространим понятие доли выполнимости с множества предложений на множество формул со свободными переменными.

Пусть $A\in Form(\sigma)$. Через $\sigma(A)=\{P_1,P_2,\dots,P_k\}$ обозначим набор всех предикатных символов, встречающихся в формуле A, а через $Var(A)=\{x_1,\dots,x_s\}$ - множество всех ее свободных переменных.

Об'емом выполнимости формулы A на универсе $U_n=\{1,2,\dots,n\}$

$(n\geq s)$ при фиксированном означивании $g:Var(A)\to U_n$ свободных переменных называется величина $V_n^g(A)$, равная числу всевозможных структур, в которых формула A истинна при означивании g.

Долей выполнимости формулы A на универсе U_n при фиксированном означивании g свободных переменных называется величина $\gamma_n^g(A)=V_n^g(A)/V_n^g(A\vee\neg A)$. Заметим, что знаменатель в этом отношении равен 2^m, где $m=\sum_{i=1}^{k} n^{\nu_i}$, а ν_i - число аргументов в предикате P_i.

Если означивание g ин'ективно, то доля $\gamma_n^g(A)$ называется нормальной. Нетрудно показать, что при двух различных ин'ективных означиваниях g_1 и g_2 выполняется равенство $\gamma_n^{g_1}(A)=\gamma_n^{g_2}(A)$, поэтому в обозначении нормальной доли верхний индекс g будем опускать.

При рассмотрении величины $\gamma_n(A)$ всегда будем полагать, что $n\geq|Var(A)|$.

Отметим некоторые простые свойства нормальной доли выполнимости.

ЛЕММА I

Ia) Если формула вида $[A\to B]$ общезначима, то $\gamma_n(A)\leq\gamma_n(B)$.

Ib) Если формулы A и B логически равносильны, то $\gamma_n(A)=\gamma_n(B)$.

Ic) Для любой формулы A $\gamma_n(\neg A)=1-\gamma_n(A)$.

Id) Если формула вида $\neg[A\&B]$ общезначима, то $\gamma_n([A\vee B])=\gamma_n(A)+\gamma_n(B)$.

Ie) Для любых формул A и B $\gamma_n(A)\leq\gamma_n(A\vee B)\leq\gamma_n(A)+\gamma_n(B)$.

If) Если формулы A и B не содержат общих предикатных символов, то $\gamma_n(A\&B)=\gamma_n(A)\cdot\gamma_n(B)$.

Ig) Если формула A имеет вид $P(x_1,x_2,\dots,x_k)$, где P - k-местный предикатный символ, то $\gamma_n(A)=1/2$.

Напомним некоторые используемые в дальнейшем сведения о числовых последовательностях.

Говорят, что числовая последовательность $\{\alpha_n\}$ экспоненциально сходится к числу a, если существует число θ $(0\le\theta<1)$ и натуральное число N, что при всех $n>N$ $|\alpha_n-a|<\theta^n$. (Обозначение: $\alpha_n \overset{exp}{\to} a$).

Заметим, что если $\alpha_n \overset{exp}{\to} a$, то $n\cdot\alpha_n \overset{exp}{\to} a$. Используя соотношения Ia)-Ig), легко доказать следующие утверждения для любых Г-формул A и B.

ЛЕММА 2

2a) Если $\gamma_n(A)\overset{exp}{\to}0$ и $\gamma_n(B)\overset{exp}{\to}0$, то $\gamma_n(A\vee B)\overset{exp}{\to}0$.

2b) Если $\gamma_n(A)\overset{exp}{\to}1$ или $\gamma_n(B)\overset{exp}{\to}1$, то $\gamma_n(A\vee B)\overset{exp}{\to}1$.

2c) Если $\gamma_n(A)\overset{exp}{\to}1$ и $\gamma_n(B)\overset{exp}{\to}1$, то $\gamma_n(A\&B)\overset{exp}{\to}1$.

2d) Если $\gamma_n(A)\overset{exp}{\to}0$ или $\gamma_n(B)\overset{exp}{\to}0$, то $\gamma_n(A\&B)\overset{exp}{\to}0$.

2e) Если $\gamma_n(A)\overset{exp}{\to}0$, то $\gamma_n(\neg A)\overset{exp}{\to}1$.

2f) Если $\gamma_n(A)\overset{exp}{\to}1$, то $\gamma_n(\neg A)\overset{exp}{\to}0$.

2g) Если $\gamma_n((\forall x|x_1,x_2,\dots,x_k)A)\overset{exp}{\to}0$ и $\gamma_n(B)\overset{exp}{\to}0$, то $\gamma_n((\forall x|x_1,x_2,\dots,x_k)[A\vee B])\overset{exp}{\to}0$.

2h) Если $\gamma_n(\exists x|x_1,x_2,\dots,x_k)A)\overset{exp}{\to}1$ и $\gamma_n(B)\overset{exp}{\to}1$, то $\gamma_n((\exists x|x_1,x_2,\dots,x_k)[A\&B])\overset{exp}{\to}1$.

2i) $\gamma_n((\exists x|x_1,x_2,\dots,x_k)A)\le n\cdot\gamma_n(A)$.

В утверждениях 2g, 2h, 2i списки исключений в кванторах содержат все свободные переменные, находящиеся в области действия соответствующих кванторов.

Формулу назовем атомарно замкнутой, если каждая ее атомарная подформула находится в области действия квантора по крайней мере по одной из своих переменных.

ТЕОРЕМА

Для каждой атомарно замкнутой Г-формулы A, не содержащей функциональных и 0-местных предикатных символов, либо $\gamma_n(A)\overset{exp}{\to}0$, либо $\gamma_n(A)\overset{exp}{\to}1$.

Доказательство проведем индукцией по числу m вхождений исключающих кванторов в формулу A.

При m=1 формула A имеет вид $(\forall x/x_1x_2\dots x_k)B$ или $(\exists x/x_1x_2\dots x_k)B$, где B - бескванторная формула, каждая атомарная часть которой зависит от x.

В первом случае проведя, комбинаторный подсчет, имеем при $n\ge k$

$$\gamma_n((\forall x/x_1x_2\dots x_k)B)=(S/2^m)^{n-k},$$

где m - число различных атомарных подформул в формуле B, а S - число слагаемых в совершенной диз'юнктивной нормальной форме формулы B.

Следовательно, $\gamma_n(A)=1$ при $S=2^m$ (в этом случае B - тождественно истинная формула), а при $S<2^m$ имеем $\gamma_n(A)\overset{exp}{\to}0$.

Во втором случае

$$\gamma_n((\exists x/x_1x_2\dots x_k)B)=1-\gamma_n((\forall x/x_1x_2\dots x_k)\neg B)$$

и вопрос сводится к предыдущему случаю.

Предположим теперь, что для Г-формул с числом кванторов $\le m$ утверждение теоремы справедливо, и рассмотрим формулу A с числом кванторов m+I.

Если A имеет вид $[B\vee C]$ или $[B\&C]$, то, используя предположение индукции и лемму 2, получим утверждение теоремы. Если A начинается со знака отрицания, то, продвигая его внутрь формулы по правилам де Моргана, получим либо формулу уже рассмоpенного вида, либо формулу, начинающуюся с квантора. Таким

образом осталось рассмотреть случаи, когда A имеет вид $(\forall x/x_1,x_2,\dots,x_k)B$ или $(\exists x/x_1,x_2,\dots,x_k)B$. Остановимся на втором случае.

Используя правила пронесения кванторов, формулу B можно привести к виду

$$\bigvee_{i=1}^{s}[C_i\&D_i],$$

где C_i - бескванторные формулы, все атомарные части которых зависят от x, а формулы D_i не содержат свободных атомарных частей и в каждую из них входит не более m кванторов. По предположению индукции утверждение теоремы справедливо для всех D_i. Рассмотрим два случая

а) Для каждого $i=1,2,\dots,s$ C_i не является тождественно истинной или $\gamma_n(D_i)\overset{exp}{\to}0$.

в) Для некоторого $i_0=1,2,\dots,s$ C_{i_0} - тождественно истинна и $\gamma_n(D_{i_0})\overset{exp}{\to}1$.

В первом случае, используя последовательно леммы I и 2, имеем

$$\gamma_n(A)=\gamma_n((\exists x/x_1,x_2,\dots,x_k)B)\le n\cdot\gamma_n(\bigvee_{i=1}^{s}[C_i\&D_i])\le n\cdot\sum_{i=1}^{s}\gamma_n(C_i\&D_i)\overset{exp}{\to}0.$$

Во втором случае -

$$\gamma_n(A)=\gamma_n((\exists x/x_1,x_2\dots x_k)B)\ge\gamma_n((\exists x/x_1,x_2\dots x_k)[C_{i_0}\&D_{i_0}])\overset{exp}{\to}1.$$

Таким образом, если A имеет вид $(\exists x/x_1,x_2,\dots,x_k)B$, утверждение теоремы доказано. Доказательство для формул вида $(\forall x/x_1,x_2,\dots,x_k)B$ получается переходом к отрицанию. Теорема доказана.

ТЕОРЕМА ((0-I)-закон для логики первого порядка)

Для всякого предложения A, не содержащего функциональных и

нульместных предикатных символов, либо $\gamma_n(A)\overset{exp}{\to}0$, либо $\gamma_n(A)\overset{exp}{\to}1$.

Для доказательства построим сначала предваренную нормальную форму предложения A, затем заменяем все кванторы, начиная с внутреннего, исключающими. После таких преобразований придем к г-формуле, логически равносильной формуле A и удовлетворяющей условию теоремы I.

Покажем теперь, что (0-I)-закон не имеет места, если в предложение входят функциональные символы. Действительно, рассмотрим формулу $A=\forall x[f(x)\ne x]$, где f - одноместный функциональный символ. Непосредственный подсчет показывает, что $\gamma_n(A)=\frac{(n-1)^n}{n^n}\to 1/e$, где e - основание натуральных логарифмов.

§ 3. О приближенной выразимости свойств структур в логических языках

В дальнейшем нам потребуются некоторые новые обозначения. Пусть $\mathfrak{U}_n=\mathfrak{U}_n(\sigma)$ - множество всех структур сигнатуры σ на универсе $U_n=\{1,2,\dots,n\}$, $\mathfrak{U}=\bigcup_{n=1}^{\infty}\mathfrak{U}_n$.

Для выражения свойств структур из множества $\mathfrak{U}$ можно использовать как предложения естественного языка, так и формальных языков таких, например, как язык $\mathfrak{L}(\sigma)$. Если $\mathfrak{L}$ - один из таких языков, то будем считать заданным отношение выполнимости (истинности) на произведении $\mathfrak{U}\times\mathfrak{L}$. При расширениях языка $\mathfrak{L}$ будем доопределять соответствующим образом и отношение выполнимости.

Для любого предложения A языка $\mathfrak{L}$ множество структур, на которых истинно A, будем обозначать через $[[A]]$.

Свойство $C\subseteq\mathfrak{U}$ назовем выразимым в языке $\mathfrak{L}$ предложением A, если $[[A]]=C$.

Если $C \subseteq \mathcal{U}$, то через C_n обозначим пересечение $C \cap \mathcal{U}_n$.

Свойство $C \subseteq \mathcal{U}$ назовем слабо выразимым в языке $\mathfrak{L}$ предложением A, если $C_n = [[A]]_n$ при любом n.

Свойство $C \subseteq \mathcal{U}$ назовем асимптотически выразимым в языке $\mathfrak{L}$ предложением A, если

$$\frac{|C_n \cap [[A]]_n|}{|C_n \cup [[A]]_n|} \to 1 \text{ при } n \to \infty.$$

Предложение A в таком случае назовем приближением свойства C в языке $\mathfrak{L}$. Предложение A назовем приближением свойства C снизу (сверху), если при этом $[[A]]_n \subseteq C_n$ ($C_n \subseteq [[A]]_n$).

Ограниченность возможностей для выражения свойств структур формулами логики первого порядка (или, что то же самое, конечным множеством таких формул) можно проиллюстрировать на следующем примере.

Пусть R - двухместное отношение на некотором универсе U. Будем называть его связным, если для любых элементов $x,y \in U$ существует конечная последовательность $z_1, z_2, \ldots, z_k$ элементов из U такая, что $(x,z_1) \in R, (z_k,y) \in R$, и для любого $i=1,2,\ldots,k-1$ $(z_i, z_{i+1}) \in R$.

Легко описать свойство связности отношения R бесконечной формулой. Действительно, пусть $A_0 = R(x,y)$ и для каждого $k=1,2,\ldots$

$A_k = \exists z_1 \exists z_2 \ldots \exists z_k [R(x,z_1) \& R(z_1,z_2) \& \ldots \& R(z_{k-1},z_k) \& R(z_k,y)]$,

тогда бесконечная формула

$$\forall x \forall y [A_0 \vee A_1 \vee A_2 \vee \ldots]$$

при естественном понимании бесконечной диз'юнкции выражает свойство связности отношения R.

Свойство связности отношения R можно выразить также формулой

второго порядка

$$\forall P [\exists x P(x) \& \exists x \neg P(x) \to \exists x \exists y [P(x) \& \neg P(y) \& R(x,y)]],$$

где P - одноместная предикатная переменная. Однако на вопрос, существует ли формула первого порядка, выражающая свойство связности, дадим отрицательный ответ. Для этого используем теорему Робинсона о бесконечно возрастающих цепях формул.

Бесконечную последовательность $Y = \{Y_1, Y_2, \ldots\}$ формул первого порядка назовем строго возрастающей, если $Y_{i+1} \Rightarrow Y_i$ для $i=1,2,\ldots$, но не выполняется $Y_i \Rightarrow Y_{i+1}$.

ТЕОРЕМА

Для любой строго возрастающей последовательности Y формул первого порядка не существует формулы логически равносильной этой последовательности.

Действительно, пусть имеется формула A такая, что $A \equiv Y$, тогда по теореме компактности существует конечное подмножество $Y' \subseteq Y$ такое, что $Y' \Rightarrow A$. Следовательно, для некоторого i $Y_i \Rightarrow A$. С другой стороны, $A \Rightarrow Y$ и, следовательно, $A \Rightarrow Y_{i+1}$. Но тогда $Y_i \Rightarrow Y_{i+1}$, что противоречит условию строгого возрастания цепи Y.

Рассмотрим теперь последовательность формул

$$Y_k = \neg [A_0 \vee A_1 \vee \ \ldots \ \vee A_k],$$

где $k=0,1,2,\ldots$. Очевидно, эта последовательность строго возрастающая и выражает несуществование пути между x и y, откуда и следует невыразимость связности отношения R.

УПРАЖНЕНИЯ

I. Найти в $\mathfrak{L}(\sigma)$ предложение, приближающее снизу свойство связности и сильной связности графа.

2. Найти в $\mathfrak{L}(\sigma)$ приближенное выражение для свойства планарности графа.

ЧАСТЬ 4. РЕЛЯЦИОННЫЙ ЯЗЫК

§I. Синтаксис реляционного языка

Нелогическими символами реляционного языка являются предикатные символы. Каждому такому символу в качестве типа приписана пара неотрицательных целых чисел. В дальнейшем, при задании интерпретации I построенного языка на универсе D_I, каждому предикатному символу R типа (m,n) будет ставиться в соответствие отношение $I(R) \subseteq D_I^m \times D_I^n$.

Основными грамматическими элементами языка будут термы и формулы. Дадим соответствующие определения.

Пусть $\sigma=\{R_j, j\in J\}$ - нелогическая сигнатура, (m_j,n_j) - тип предикатного символа R_j.

ОПРЕДЕЛЕНИЕ ТЕРМА

I) Любой предикатный символ типа (m,n) является термом того же типа.

2) Если T_1, T_2 - термы типа (m,n), то выражения $[T_1 \cup T_2]$, $[T_1 \cap T_2]$ - термы типа (m,n).

3) Если T - терм типа (m,n), то T^{-1} - терм типа (n,m), $\bar{T}$ - терм типа (m,n).

4) Если T_1 - терм типа (m,n), а T_2 - терм типа (n,k), то $[T_1 \cdot T_2]$ - терм типа (m,k).

5) Если T_i - терм типа (m,n_i), $i=1,2,\ldots,k$, то выражение $\langle T_1,T_2,\ldots,T_k\rangle$ - терм типа $(m, \sum_{i=1}^{k} n_i)$.

Считаем, что среди предикатных символов имеются специальные

символы:

а) $U^{m,n}$ - для обозначения универсального отношения типа (m,n),

б) $\Omega^{m,n}$ - для обозначения пустого отношения типа (m,n),

в) $E^{m,i}$ - для обозначения отношения селекции типа (m,1).

ОПРЕДЕЛЕНИЕ ФОРМУЛЫ

Если T_1, T_2 - термы типа (m,n), то выражения вида $T_1 \subseteq T_2$ и $T_1 = T_2$ называются формулами.

Выражения $T_1=T_2$ используются для обозначения множества, состоящего из двух формул $T_1 \subseteq T_2$ и $T_2 \subseteq T_1$.

Если Г - множество формул и $T_1 \subseteq T_2$ - формула, то выражение $\Gamma \to T_1 \subseteq T_2$ называем утверждением.

§2. Семантика реляционного языка

Интерпретация I языка состоит из универса D_I и для каждого предикатного символа $R_j \in \sigma$ типа (m_j,n_j) соответствующего ему отношения $I(R_j) \subseteq D_I^{m_j} \times D_I^{n_j}$.

При этом должны выполняться следующие соотношения:

а) $I(U^{m,n}) = D_I^m \times D_I^n$,

б) $I(\Omega^{m,n}) = \emptyset$,

в) $I(E^{m,i}) = \{((a_1,a_2,\ldots,a_m),a_i) \mid a_j \in D, j=1,2,\ldots,m\}$.

$(i \le m)$

Интерпретация множества предикатных символов распространяется на термы следующим образом.

$I(T_1 \cup T_2) = I(T_1) \cup I(T_2)$,

$I(T_1 \cap T_2) = I(T_1) \cap I(T_2)$,

$I(T^{-1})=\{(a,b)\mid(b,a)\in I(T)\}$,

$I(\bar{T})=\{(a,b)\mid(a,b)\notin I(T)\}$,

$I(T_1\cdot T_2)=\{(a,b)\mid\exists c\ (a,c)\in I(T_1),\ (c,b)\in I(T_2)\}$,

$I(\langle T_1,T_2,\ldots T_k\rangle)=\{(a,(b_1,b_2,\ldots,b_k))\mid(a,b_i)\in I(T_i)\}$.

Формула $T_1\leq T_2$ истинна в интерпретации I, если $I(T_1)\leq I(T_2)$. Этот факт обозначается выражением $\vdash_I T_1\leq T_2$.

Утверждение $\theta\to(T_1\leq T_2)$ считается верным, если при любой интерпретации I, в которой истинны все формулы из θ, истинна также формула $T_1\leq T_2$.

Известно, что реляционный язык семантически эквивалентен логическому языку первого порядка с равенством. Так, например, утверждение, выраженное в языке с кванторами предложением $\forall x[P(x)\to Q(x)]$, где P и Q - одноместные предикатные символы, может быть выражено в реляционном языке формулой $P\leq Q$, где P и Q предикатные символы типа (I,O)(или типа (O,I)).

§3. Система доказательств в реляционном языке

Опишем метод, используемый для доказательства утверждений. В описываемой системе доказательств вводятся в рассмотрение индивидные параметры, что позволяет приблизить ее к обычному математическому выводу.

Приводимая система доказательств подобна методу построения деревьев в логическом языке первого порядка.

Пусть P - пронумерованное множество символов, называемое множеством <u>индивидных параметров</u>. <u>Параметр</u> - это набор индивидных параметров. Параметры будем обозначать буквами α, β, γ, а i-тая компонента параметра α обозначается α_i. <u>Длина</u> параметра α,

обозначаемая $|\alpha|$, есть число индивидных параметров в α. Параметр длины 0 обозначается λ.

Метками узлов при построении деревьев будут выражения вида

$$(\alpha,\beta)\in T \text{ или } (\alpha,\beta)\notin T,$$

где T имеет тип (m,n) и $|\alpha|=m$, $|\beta|=n$. Вместо $(\alpha,\lambda)\in T$ и $(\lambda,\alpha)\in T$ обычно сокращенно записывают $\alpha\in T$. Аналогично вместо $(\alpha,\lambda)\notin T$, $(\lambda,\alpha)\notin T$ пишут $\alpha\notin T$.

Параметры представляют наборы элементов универсального множества , а метка $(\alpha,\beta)\in T$ означает, что упорядоченная пара (α,β) удовлетворяет предикату T.

Чтобы доказать утверждение $T_1\leq T_2$, мы начнем с дерева, состоящего из двух последовательных узлов с метками

$$(\alpha,\beta)\in T_1,$$
$$(\alpha,\beta)\notin T_2.$$

Это соответствует утверждению "предположим пара (α,β) удовлетворяет предикату T_1 и не удовлетворяет предикату T_2". Цель расширения дерева - показать, что в качестве значений для α и β нельзя выбрать никакие наборы элементов универса.

Правила расширения дерева перечислены в таблице 2.

Таблица 2

$\dfrac{(\alpha,\beta)\in T_1\cup T_2}{(\alpha,\beta)\in T_1 \mid (\alpha,\beta)\in T_2}$	$\dfrac{(\alpha,\beta)\in T_1\cup T_2}{\begin{array}{l}(\alpha,\beta)\in T_1\\(\alpha,\beta)\in T_2\end{array}}$
$\dfrac{(\alpha,\beta)\in T\cap T_2}{\begin{array}{l}(\alpha,\beta)\in T_1\\(\alpha,\beta)\in T_2\end{array}}$	$\dfrac{(\alpha,\beta)\in T_1\cap T_2}{(\alpha,\beta)\in T_1 \mid (\alpha,\beta)\in T_2}$
$\dfrac{(\alpha,\beta)\in\langle T_1,T_2,\ldots,T_k\rangle}{\begin{array}{l}(\alpha,\beta_1)\in T_1\\(\alpha,\beta_2)\in T_2\\\ldots\\(\alpha,\beta_k)\in T_k\end{array}}$	$\dfrac{(\alpha,\beta)\in\langle T_1,T_2,\ldots,T_k\rangle}{(\alpha,\beta_1)\in T_1 \mid (\alpha,\beta_2)\in T_2 \mid \ldots \mid (\alpha,\beta_k)\in T_k}$
$\dfrac{(\alpha,\beta)\in T^{-1}}{(\beta,\alpha)\in T}$	$\dfrac{(\alpha,\beta)\in T^{-1}}{(\beta,\alpha)\in T}$
$\dfrac{(\alpha,\beta)\in T_1\cdot T_2}{\begin{array}{l}(\alpha,\gamma)\in T_1\\(\gamma,\beta)\in T_2\end{array}}$	$\dfrac{(\alpha,\beta)\in T_1\cdot T_2}{(\alpha,\gamma')\in T_1 \mid (\gamma',\beta)\in T_2}$
$\dfrac{(\alpha,\beta)\in U}{(\alpha,\beta)\in\Omega}$	$\dfrac{(\alpha,\beta)\in\Omega}{(\alpha,\beta)\in\Omega}$
$\dfrac{(\alpha,\beta)\in E^{1,1}}{(\beta,\alpha)\in E^{1,1}}$	$\dfrac{(\alpha,\beta)\in E^{m,i}}{(\alpha_i,\beta)\in E^{1,1}}$
$\dfrac{\varnothing}{(\alpha,\beta)T \mid (\alpha,\beta)\in T}$	$\dfrac{\begin{array}{l}L(\alpha)\\(\alpha,\beta)\in E^{1,1}\end{array}}{\begin{array}{l}L(\alpha)\\L(\beta)\end{array}}$

Например, для использования правила с посылкой $(\alpha,\beta)\in T_1\cdot T_2$ ветвь должна содержать эту посылку. Применяя данное правило, продолжаем ветвь присоединением последовательно двух новых вершин, одну с меткой $(\alpha,\gamma)\in T_1$, другую с меткой $(\gamma,\beta)\in T_2$, где γ – параметр, не встречавшийся ранее в дереве. При использовании правила с посылкой $(\alpha,\beta)\in T_1\cap T_2$, дерево может быть продолжено добавлением к концу ветви двух потомков: одного с меткой $(\alpha,\beta)\in T_1$ и другого с меткой $(\alpha,\beta)\in T_2$.

Аксиомы (гипотезы) могут быть введены как дополнительные правила вывода. Например, аксиома вида $T_1\subseteq T_2$ может быть выражена правилом

$$\frac{(\alpha,\beta)\in T_1}{(\alpha,\beta)\in T_2}.$$

Каждая ветвь дерева соответствует некоторой возможной интерпретации. Эта возможность становится нереализуемой, когда для какого-либо предиката R и параметров α и β метки $(\alpha,\beta)\in R$ и $(\alpha,\beta)\notin R$ одновременно появляются в этой ветви, так как в интерпретации R не может сразу содержаться и не содержаться пара (α,β). Такая ветвь называется <u>блокированной</u>. Дерево является доказательством, если в нем каждая ветвь блокирована. Это означает, что любая "возможность" отвергнута и, следовательно, не существует пары (α,β), удовлетворяющей T_1 и не удовлетворяющей T_2, тогда $T_1\subseteq T_2$ истинно в любой интерпретации.

ОПРЕДЕЛЕНИЕ. Пусть Γ – множество формул. Формула $T_1\subseteq T_2$ выводима из Γ (обозначаем $\Gamma \vdash T_1\subseteq T_2$), если дерево, состоящее из двух последовательных вершин с метками $(\alpha,\beta)\in T_1$, $(\alpha,\beta)\notin T_2$, может

быть продолжено до блокированного дерева использованием правил вывода из таблицы 2 вместе с правилами, соответствующими формулам из Г. Если Г - пусто, мы пишем $\vdash T_1 \leq T_2$.

Описанная система доказательств в реляционном языке является дедуктивно полной также, как и аналогичная система в языке с кванторами.

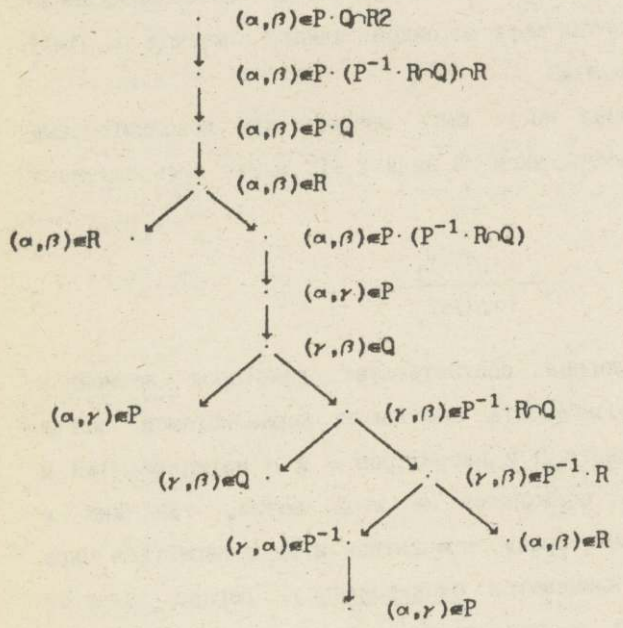


рис.7

В качестве примера на рис.7 дано доказательство формулы

$$P\cdot Q\cap R\leq P\cdot(P^{-1}\cdot R\cap Q)\cap R,$$

где P,Q,R - предикатные символы подходящих типов, и в записи формулы использованы традиционные соглашения об опускании скобок. Все ветви построенного дерева блокированы, поэтому оно является доказательством исходной формулы.

Часть 5. МОДЕЛИ ВЫЧИСЛЕНИЙ

§I. Исторические сведения

К концу I9 - началу 20 века в алгебре накопилось некоторое число задач, для которых математики, несмотря на упорные попытки, не могли предложить методов решения. Одной из таких задач является задача о разрешимости диофантова уравнения. В докладе Д.Гильберта, прочитанном на II международном конгрессе математиков в августе I900 года, она звучит следующим образом: "Пусть задано диофантово уравнение с произвольными неизвестными и целыми рациональными числовыми коэффициентами. Указать способ, при помощи которого возможно после конечного числа операций установить, разрешимо ли это уравнение в целых рациональных числах".

Неудачные попытки математиков решить эту и другие подобные задачи привели к мысли о том, что метода для их решения может и не существовать. Но, чтобы доказывать подобного рода утверждения, необходимо иметь математическое определение метода, то есть для интуитивных понятий разрешимости и вычислимости необходимо иметь их формальные эквиваленты. Одним из важнейших достижений математики 20 века является формирование таких эквивалентов, то есть математических понятий, которые раскрывают сущность интуитивных представлений о том, что такое метод (алгоритм) решения той или иной задачи, что такое вычислимая функция.

Важность этих понятий вытекает не только из общенаучных проблем развития математики, но также из практических задач

общества, использующего вычислительную технику в производстве, экономике, инженерных исследованиях и нуждающегося в адекватном представлении о возможностях вычислительной машины.

Приведем краткие исторические сведения о возникновении теории алгоритмов.

В 1932-35 годах А.Черч и С.К.Клини ввели понятие λ-определимой функции, которое сыграло важную роль в определении об'ема интуитивного понятия вычислимой функции.

В 1934 году Гедель, на основе идей Эрбрана, рассмотрел класс функций, названных общерекурсивными, а в 1936 году Черч и Клини доказали, что этот класс совпадает с классом λ-определимых функций.

В 1936 году А.Тьюринг ввел свое понятие вычислимой функции, а в 1937 году доказал, что оно совпадает с понятием λ-определимой функции.

В 1943 году Пост, основываясь на своей неопубликованной работе 1920-22 годов, опубликовал еще один формальный эквивалент понятия вычислимой функции.

Следующую эквивалентную формулировку дает теория алгоритмов Маркова (1951 год). Любое из упомянутых здесь уточнений интуитивного понятия вычислимой функции можно принять за математическое определение вычислимой функции. Основными доводами для такого принятия являются эквивалентность различных формулировок и естественно- исторический опыт, показывающий, что все изучавшиеся до сих пор функции, которые принято считать вычислимыми, являются таковыми в смысле любого из упомянутых выше определений.

Еще один подход к моделированию вычислений развивается в

рамках так называемого логического программирования. При таком подходе условия решаемой задачи и ее цель формулируются с помощью предложений формального логического языка. Затем осуществляется автоматический поиск доказательства цели, сопровождаемый вычислением неизвестных в задаче величин.

Библиографию по затронутым выше вопросам можно найти в книгах [3,4].

Заметим, что разрабатываемые в настоящее время алгоритмические языки, составляющие математическое обеспечение современных вычислительных машин, также можно использовать для определения понятия вычислимости. Более того, можно проследить тесную связь между упомянутыми здесь теоретическими моделями вычислений и реальным программированием для ЭВМ. Так λ-исчисление Черча является прообразом функционального программирования, реализованного в известном программистам языке ЛИСП, разработанном в 1961 году Дж.Маккарти, а модель Поста содержит идеи, реализованные в операторных языках типа Фортран, Алгол. Методы логического программирования реализованы в настоящее время в нескольких версиях языка Пролог.

Однако, реальные языки программирования из-за своей громоздкости и избыточности выразительных средств мало пригодны для теоретического анализа понятия вычислимости. Отметим также модели вычислительных устройств, разработанные Шепердсоном и Смерджисом (1963) и получившие название РАМ (равнодоступная адресная машина) и РАСП (равнодоступная адресная машина с хранимой программой). Эти модели в большей степени чем, например, модель Тьюринга отражают структуру современных вычислительных устройств.

§2. Тьюрингова модель переработки информации

ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ (модель памяти)

Считаем, что информация представляется словами (т.е. конечными последовательностями), составленными из букв конечного алфавита $A=\{a_1,a_2,\dots,a_t\}$, и записывается на неограниченной в обе стороны ленте, разделенной на ячейки. Слово записывается в идущих подряд ячейках по одной букве в ячейке.

В ячейку может быть ничего не записано, в этом случае говорим, что ячейка содержит пробел. Для обозначения пробела используем символ #. Конечную последовательность, составленную из символов алфавита A и символа пробела, называем псевдословом. Считаем, что слева от первой буквы псевдослова и справа от последней записаны пробелы, кроме того, один из символов псевдослова будем помечать стрелкой.

Множество всех конечных последовательностей символов из алфавита A обозначается через A^*.

Если псевдослово имеет вид $X\#u_n\#\dots\#u_1\overset{\downarrow}{\#}$, где $u_i\in A^*$, $X\in(A\cup\{\#\})^*$, то u_1 называем его первым словом, u_2 - вторым и т.д. Слова u_i могут быть и пустыми. Пустые слова не занимают место на ленте. Будем считать, что между двумя подряд идущими пробелами записано пустое слово.

Поскольку на ленте в каждый момент времени будет находиться не более чем конечное число символов, отличных от пробела, постольку для любого n в псевдослове будет определено его n-ое слово.

ПРЕОБРАЗОВАТЕЛЬ ИНФОРМАЦИИ

Преобразователь информации можно представить как некоторое устройство с лентой и головкой, обозревающей в каждый момент времени одну из ячеек ленты, которое по заранее намеченному плану (программе) может выполнять операции следующего вида:

- напечатать один из символов алфавита в обозреваемой ячейке;
- сдвинуться по ленте на одну ячейку влево;
- сдвинуться по ленте на одну ячейку вправо;
- ничего не делать до следующего такта времени.

ОПРЕДЕЛЕНИЕ ПРОГРАММЫ

Программу преобразования информации будем представлять в виде ориентированного графа, вершины которого помечены символами из множества $\tilde{A}\cup\{r,l,s\}$, ($\tilde{A}=A\cup\{\#\}$), а дуги символами из множества $\tilde{A}$ так, что разным дугам, выходящим из одной вершины, приписаны разные символы. Одна вершина графа выделена в качестве входной. Предполагаем, что $\tilde{A}\cap\{r,l,s\}=\emptyset$.

Действие программы осуществляется следующим образом. В начальный момент головка вычислителя обозревает одну из ячеек ленты. Просматривается входная вершина программы. Если ей приписан символ r, l или s, то головка вычислителя сдвигается по ленте на одну ячейку соответственно вправо, влево или остается на месте; если же ей приписан символ из алфавита $\tilde{A}$, то этот символ печатается в обозреваемой ячейке, старое содержимое ячейки при этом стирается. После того, как выполнено действие, соответствующее вершине q, в графе отыскивается выходящая из q дуга, помеченная той буквой, которая находится в данный момент в

обозреваемой ячейке. Следующим выполняется действие, соответствующее вершине, в которую ведет найденная дуга. Процесс продолжается до тех пор, пока не будет достигнута вершина, из которой не выходит дуга, помеченная буквой, обозреваемой в данный момент. Если такой момент не наступает, то программа работает бесконечно.

Вершину v, для которой найдется хотя бы одна буква из $\tilde{A}$, не используемая в качестве метки на дугах, выходящих из v, будем называть выходной.

Согласно данному описанию, программу можно задать как набор:

$$\Pi=\langle Q,A,q_0,\phi,\psi\rangle,$$

в котором

Q - множество вершин графа;

A - алфавит символов, печатающихся на ленте;

q_0 - входная вершина ($q_0 \in Q$);

ϕ - отображение Q в $\tilde{A}\cup\{r,l,s\}$;

ψ - частичное отображение $A\times Q$ в Q.

Множество выходных вершин программы П обозначим через V_Π.

ПРИМЕР

Пусть A={0,1} и на ленте записано псевдослово $\alpha_1\alpha_2\ldots\alpha_k\overset{\downarrow}{\#}$, где $\alpha_i \in A$, $k\geq 1$, а стрелка над # показывает положение головки в начальный момент. Рассматривая слово $\alpha_1\alpha_2\ldots\alpha_k$ как двоичную запись натурального числа n, составить программу, которая на ленте оставляет псевдослово $\beta_1\beta_2\ldots\beta_m\overset{\downarrow}{\#}$, являющееся двоичной записью числа n+I.

Нетрудно увидеть, что поставленную задачу решает программа, представленная на рисунке 8.

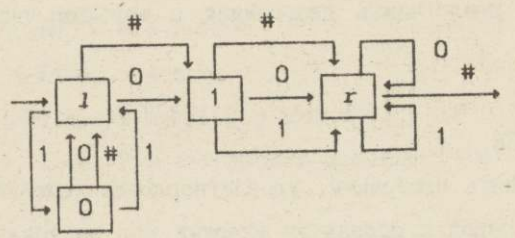


рис.8

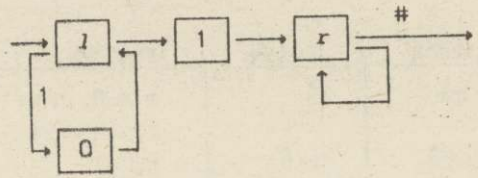


рис.9

Здесь входная и выходная вершины помечены соответственно входящей и выходящей стрелками.

Чтобы не загромождать чертежи большим количеством стрелок и надписей, введем следующие соглашения: если из вершины q в вершину q' ведет несколько дуг, будем заменять их одной дугой с надписанными над ней буквами, соответствующими заменяемым дугам; одну из дуг выходящих из данной вершины будем оставлять ненадписанной, считая при этом, что она помечена всеми буквами алфавита $\tilde{A}$, которые не использованы на других дугах, выходящих из вершины q. Такая дуга может оказаться единственной, выходящей из вершины q.

Используя эти соглашения, программу, представленную на рис.8, можно представить диаграммой с меньшим числом дуг (см. рис.9).

УПРАЖНЕНИЯ

I. Составить программы, удовлетворяющие условиям из таблицы 3, в которой через $\bar{m}$ обозначен унарный код натурального числа m (то есть слово, состоящее из m символов "1", а через $\bar{\bar{m}}$ - его бинарное представление.

	Алфавит	Вход	Выход	Условие
1	{1}	$\bar{m}\overset{\downarrow}{\#}$	$\bar{n}\overset{\downarrow}{\#}$	$m,n\in N;\ n=m+1$
2	-"-	-"-	-"-	$m,n\in N; n=2m$
3	-"-	$\bar{m}\#\bar{n}\overset{\downarrow}{\#}$	$\bar{n}\#\bar{m}\overset{\downarrow}{\#}$	$m,n\in N$
4	-"-	$\bar{m}\#\bar{n}\overset{\downarrow}{\#}$	$\bar{m}\#\bar{n}\#\bar{p}\overset{\downarrow}{\#}$	$m,n,p\in N;\ p=m+n$
5	-"-	$\bar{m}\#\bar{n}\overset{\downarrow}{\#}$	$\bar{m}\#\bar{n}\#\bar{p}\overset{\downarrow}{\#}$	$m,n,p\in N; p=m\cdot n$
6	-"-	$\bar{m}\#\bar{n}\overset{\downarrow}{\#}$	$\bar{m}\#\bar{n}\#\bar{p}\overset{\downarrow}{\#}$	$m,n,p\in N; p=m^n$
7	{0,1}	$\bar{m}\overset{\downarrow}{\#}$	$\bar{m}\#\bar{\bar{n}}\overset{\downarrow}{\#}$	$m,n\in N; m=n$
8	-"-	$\bar{m}\overset{\downarrow}{\#}$	$\bar{m}\#\bar{n}\overset{\downarrow}{\#}$	$m,n\in N; m=n$
9	-"-	$\bar{m}\#\bar{n}\overset{\downarrow}{\#}$	$\bar{m}\#\bar{n}\#\bar{p}\overset{\downarrow}{\#}$	$m,n,p\in N; p=m+n$
10	-"-	-"-	-"-	-"- ; $p=m\cdot n$
11	{1}	$\bar{m}\overset{\downarrow}{\#}$	$\bar{m}\#\bar{n}\overset{\downarrow}{\#}$	$m,n\in N; n=\mathrm{entier}(m/2)$
12	{0,1}	$\bar{m}\overset{\downarrow}{\#}$	$\bar{m}\#\bar{\bar{n}}\overset{\downarrow}{\#}$	$m,n\in N; n=\mathrm{entier}(m/2)$
13	{1}	$\bar{m}\#\bar{n}\overset{\downarrow}{\#}$	$\bar{m}\#\bar{n}\#\bar{p}\overset{\downarrow}{\#}$	$m,n,p\in N; p=m-n$
14	{0,1}	$\bar{m}\#\bar{n}\overset{\downarrow}{\#}$	$\bar{m}\#\bar{\bar{n}}\#\bar{p}\overset{\downarrow}{\#}$	$m,n,p\in N; p=m-n$
15	{1}	$\bar{m}\overset{\downarrow}{\#}$	$\bar{m}\#\bar{m}\overset{\downarrow}{\#}$	$m\in N$
16	{0,1}	$\bar{m}\overset{\downarrow}{\#}$	$\bar{m}\#\bar{\bar{m}}\overset{\downarrow}{\#}$	$m\in N$
17	{0,1}	$\bar{m}\overset{\downarrow}{\#}$	$\bar{m}\#\bar{\bar{n}}\overset{\downarrow}{\#}$	$m,n\in N; n=\mathrm{entier}(\log m)$
18	{1}	$\bar{m}\overset{\downarrow}{\#}$	$\bar{m}\#\bar{\bar{n}}\overset{\downarrow}{\#}$	$m,n\in N; n=2^m$

Таблица 3

2. Составить над алфавитом A программу с двумя выходами, которая бы псевдослово X перерабатывала само в себя и

осуществляла I-й выход, если X удовлетворяет условию P, и 2-й выход - в противном случае.

а) $A=\{1\}$, $X=\bar{m}\overset{\downarrow}{\#}$, $P \leftrightarrow$ m-четно;

б) $A=\{1\}$, $X=\bar{m}\#\bar{n}\overset{\downarrow}{\#}$, $P \leftrightarrow m=n$;

в) $A=\{1,+,=\}$, $X = \bar{m}+\bar{n}=\bar{k}\overset{\downarrow}{\#}$, $P \leftrightarrow m+n=k$;

г) $A=\{0,1\}$, $X=U\overset{\downarrow}{\#}$, $P \leftrightarrow$ U содержит подслово "101".

3. Изучите работу программы п (рис. I0) над алфавитом A и ответьте на вопросы:

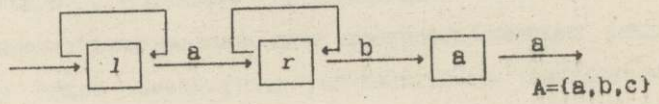


рис. I0

а) на каких псевдословах останавливается п?

б) в какие псевдослова перерабатываются псевдослова

$X_1 = abbabb\overset{\downarrow}{\#}$, $X_2 = acc\#bcbb\overset{\downarrow}{\#}$, $X_3 = bbb\overset{\downarrow}{\#}$, $X_4 = baac\overset{\downarrow}{\#}$?

в) на каких псевдословах программа не останавливается, а читающая головка уходит вправо (влево)?

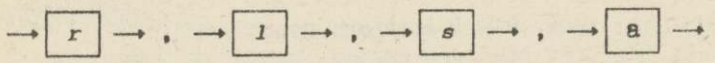
Ввиду большой близости введенного нами понятия программы с понятием машины Тьюринга мы будем называть наши программы тьюринговыми.

§3. Алгебра программ

Для записи программ в виде аналитических выражений введем обозначения для некоторых элементарных программ и операции,

позволяющие строить из элементарных программ более сложные.

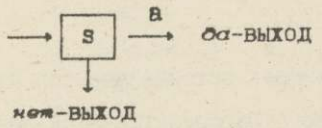
Элементарными вычисляющими программами будем называть программы вида



обозначать их будем соответственно символами r,l,s,a где $a \in \tilde{A}$.

Программы, у которых множество выходов разбито на два непустых подмножества (подмножество *да*-выходов и подмножество *нет*-выходов) назовем бинарными распознающими программами.

Элементарными распознающими программами будем считать программы вида



обозначать такую программу будем через <a>.

ПРАВИЛА КОМПОЗИЦИИ

Введем несколько правил, которые позволят нам из уже построенных программ строить более сложные.

I) Если $T_1, T_2, \ldots T_k$ - программы, то выражение $[T_1, T_2, \ldots T_k]$ обозначает программу, которая получена следующим образом. Все выходы программы T_i соединены дугой с входом программы T_{i+1} $(i=1,2,\ldots,k-1)$, каждая такая дуга помечена буквами из $\tilde{A}$, которые не использованы на других дугах, выходящих из рассматриваемой выходной вершины (в дальнейшем при соединении выходов одной программы с входом другой будем пользоваться этим правилом). Входом в полученную программу является вход программы T_1, а выходами - выходы программы T_k. Таким образом, программа $[T_1, T_2, \ldots T_k]$ предписывает последовательное выполнение программ $T_1, T_2, \ldots T_k$.

2) Если P - бинарная распознающая программа, а T - произвольная, то выражение

(*если* P) T

означает программу, полученную следующим образом. Все *да*-выходы программы P соединяются с входом программы T. Входом в полученную программу является вход в программу P, а выходом - выходы программы T и *нет*-выходы программы P. Программы такого вида называются охраняемыми и говорят в таких случаях, что программа T охраняется программой P.

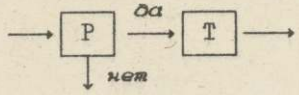


Схема программы: (*если* P)T

3) Если дан набор охраняемых программ вида: (если P_i)T_i $(i=1,2,\ldots,k)$, то выражение вида

(*если* P_1)$T_1 \vee \ldots \vee$(*если* P_k)T_k

обозначает программу, полученную следующим образом. *Да*-выходы программы P_i соединяются с входом T_i $(i=1,2,\ldots,k)$; *нет*-выходы программы P_i соединяются с входом T_{i+1} $(i=1,2,\ldots,k-1)$. Входом в полученную программу является вход в программу P_1, а выходом -

выходы программ $T_1, T_2, \ldots, T_k$ и *нет*-выходы программы P_k.

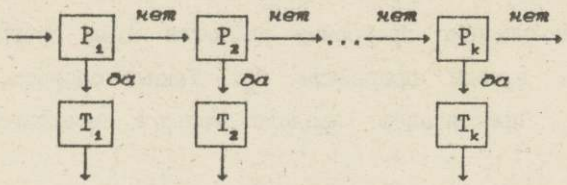


Схема программы: (*если* P_1)$T_1 \vee \ldots \vee$(*если* P_k)T_k

4) Если P - бинарная распознающая программа, а T - любая, то выражение

(*пока* P)T

обозначает программу, полученную следующим образом. Да-выходы программы P соединяются с входом программы T. Все выходы программы T соединены с входом в P. Входом в полученную программу является вход в P, а выходом *нет*-выходы программы P.

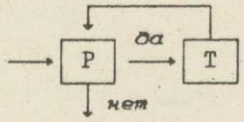


Схема программы: (*пока* P)T

5) Если P бинарная распознающая программа, а T - любая, то выражение

T(*до* P)

обозначает программу, полученную соединением *нет*-выходов программы P с входом в T, а выходов T - с входом в P. Входом в полученную программу является вход в T, а выходами - *да*-выходы

программы P.

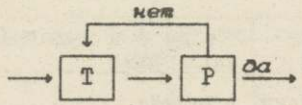


Схема программы: T(*до* P)

Сокращения: Программы вида $T_1 \vee T_2 \vee \ldots \vee T_k$ будем сокращенно записывать в виде $\bigvee_{i=1}^{k} T_i$, а программы вида $[T_1, T_2, \ldots, T_k]$ в случае, если $T_1 = T_2 = \ldots = T_k = T$ - в виде T^k. В контексте со словами "*если*", "*пока*", "*до*" угловые скобки в записи элементарного распознавающего оператора <a> будем опускать.

§4. Начальное математическое обеспечение

Приведем несколько программ, которым дадим обозначения и которые в дальнейшем используем для построения более сложных программ. Они будут составлять начальное математическое обеспечение программирования.

В таблице 4 приведены их схемы в предположении, что алфавитА состоит из символов $a_1, a_2, \ldots, a_l$; а символ # обозначен через a_0. Кроме того считаем, что X,Y - произвольные псевдослова над алфавитом A; $u_1, u_2, \ldots, u_n, u$ - слова в алфавите A; a - произвольный символ из $A \cup \{\#\}$; u^{-1} - слово, полученное из слова u изменением порядка символов на противоположный; n=1,2,... . Программы R, L, описанные в первых двух строках таблицы, используются в последующих строчках.

Обозн. прогр.	Назначение программы	Входное и выходное псевдослова, программа
L	Сдвиг головки влево до пробела	$X\#U\overset{\downarrow}{a}Y$; $X\overset{\downarrow}{\#}UaY$; $l(\infty\#)$.
R	Сдвиг головки вправо до пробела	$X\overset{\downarrow}{a}U\#Y$; $XaU\overset{\downarrow}{\#}Y$; $r(\infty\#)$.
K_n	Копирование n-го слова	$X\#U_n\#U_{n-1}\#\ldots\#U_1\overset{\downarrow}{\#}$; $X\#U_n\#U_{n-1}\#\ldots\#U_1\#U_n\overset{\downarrow}{\#}$; $L^n, r, [\bigvee_{i=1}^{t}(\text{если } a_i)[\#, R^{n+1}, a_i, L^{n+1}, a_i, r]$ $(\infty\#), R^n$.
S	Удаление буквы со сдвигом	$X\overset{\downarrow}{a}U\#Y$; $XU\#\overset{\downarrow}{\#}Y$; $[r, \bigvee_{j=0}^{t}(\text{если } a_j)[l, a_j], r](\infty\#)$.
Z_n	Циклический сдвиг n слов	$X\#U_n\#U_{n-1}\#\ldots\#U_2\#U_1\overset{\downarrow}{\#}$; $X\#U_{n-1}\#U_{n-2}\#\ldots\#U_1\#U_n\overset{\downarrow}{\#}$; $r, [L^{n+1}, r, \bigvee_{i=0}^{t}(\text{если } a_i)[S^n, a_i]](\infty\#)$.
Λ_n	Удаление n-го слова	$X\#U_n\#U_{n-1}\#\ldots\#U_1\overset{\downarrow}{\#}$; $X\#U_{n-1}\#U_{n-2}\#\ldots\#U_1\overset{\downarrow}{\#}$; $Z_n, [\#, l](\infty\#)$.
I	Запись слова в обратном порядке	$X\#U\overset{\downarrow}{\#}$; $X\#U\#U^{-1}\overset{\downarrow}{\#}$; $l, [\bigvee_{i=1}^{t}(\text{если } a_i)[\#, R^2, a_i, L^2, a_i, l]](\infty\#), R^2$.

Таблица 4

§5. Методика доказательства правильности алгоритмов с помощью индуктивных утверждений

Рассматриваемая методика предназначена для доказательства правильности алгоритмов, представленных в виде графов, вершинам которых поставлены в соответствие элементарные операторы над памятью, а дугам переходы от оператора к оператору. Одну из вершин назовем входной, ей соответствует оператор, с которого начинается выполнение алгоритма, а выходных вершин может быть несколько. Считаем, что входная и выходная вершины помечены, соответственно, входящей и выходящей стрелками. Такие представления алгоритмов называют блок-схемами.

Доказать правильность алгоритма это значит доказать утверждение вида:

"Если входные данные удовлетворяют входному условию, то алгоритм через конечное число шагов завершает работу и выходные данные удовлетворяют требуемому выходному условию."

На практике такое утвеждение часто разбивают на два.

(1) "Если входные данные удовлетворяют входному условию и алгоритм через конечное число шагов завершает работу, то выходные данные удовлетворяют требуемому выходному условию."

(2) "Если входные данные удовлетворяют входному условию, то алгоритм через конечное число шагов завершает работу."

Алгоритм, для которого доказано утверждение (1) называется частично правильным или частично корректным. Если же доказаны утверждения (1) и (2) то алгоритм называется правильильным или корректным.

Заметим, что когда доказательство утверждения (2) представляет непреодолимые трудности, то ограничиваются доказателством утверждения (1). Таковы, например, итерациионные алгоритмы, для которых не известна область сходимости. В таком случае, если алгоритм в приемлемое время завершает свою работу, то правильность ответа гарантируется.

Остановимся на доказательстве частичной корректности. Методика заключается в следующем.

1. Для контроля за ходом вычислений выбираются, так называемые,контрольные дуги. К числу контрольных обязательно относят входную и все выходные дуги, а также некоторое количество других дуг так, чтобы в блок-схеме алгоритма оказались "разрезанными" все циклы.

2. Для каждой контрольной дуги формулируется индуктивное условие, которому предположительно должно удовлетворять содержимое памяти алгоритма при каждом его прохождении через рассматриваемую дугу. Считаем что все контрольные дуги (в дальнешем будем называть их контрольными точками) и соответствующие им индуктивные утверждения пронумерованы.

3. Для каждой пары i,j контрольных точек, для которых в блок-схеме имеется путь из i в j, минующий другие контрольные точки, выбираются все такие пути и для каждого выбранного пути доказывается утверждение (индуктивный шаг):

"Если при очередном проходе через точку i выполнялось индуктивное предположение P_i и если реализуется рассматриваемый путь, то при достижении точки j будет выполняться условие P_j".
Если все эти шаги доказаны, то, используя принцип математической индукции, можно утверждать частичную корректность алгоритма.

§6. Вычислимость и разрешимость

Упорядоченный набор из n слов в алфавите A наывается n-местным набором над A. Множество всех n-местных наборов над A обозначим через $(A^*)^n$.

Любое подмножество R множества $(A^*)^n$ называется n-местным словарным отношением.

Любое, возможно, частичное отображение $f:(A^*)^n \to A^*$ называется n-местной словарной функцией. Область определения функции f обозначается через Def(f).

Результатом работы программы T на входном псевдослове X называется псевдослово T(x), которое появляется на ленте в момент остановки программы; если программа работает бесконечно, то результат не определен.

Программу, которая в процессе работы над любым псевдословом X не сдвигает головку левее пробела, расположенного слева от n-го слова псевдослова X, будем называть n-программой.

Словарное n-местное отношение R называется <u>полуразрешимым</u>, если существует n-программа T, которая останавливается в точности на всех псевдословах, имеющих вид

$$X\#u_n\#u_{n-1}\#\ldots\#u_1\overset{\downarrow}{\#}$$

где $(u_1,u_2,\ldots,u_n)\in R$.

Словарное n-местное отношение R называется <u>разрешимым</u>, если R и $\bar{R}$ полуразрешимы (под $\bar{R}$ здесь понимается множество $(A^*)^n\backslash R$).

Словарная n-местная функция $f:(A^*)^n \to A^*$ называется <u>вычислимой</u> по Тьюрингу, если существует n-программа T такая, что

$$T(u_n\#u_{n-1}\#\ldots\#u_1\overset{\downarrow}{\#})=\begin{cases} u_n\#u_{n-1}\#\ldots\#u_1\#f(u_1,u_2,\ldots,u_n)\overset{\downarrow}{\#}, \\ \qquad \text{если } (u_1,u_2,\ldots,u_n)\in Def(f); \\ \text{не определено, в противном случае.} \end{cases}$$

Вычислимые по Тьюрингу функции уместно было бы назвать полувычислимыми, а полувычислимые с разрешимой областью определения - вычислимыми, но это противоречит установившимся традициям.

ВЫЧИСЛЕНИЕ ЧИСЛОВЫХ ФУНКЦИЙ

Чтобы вычислять значения числовых функций с помощью тьюринговых программ, необходимо выбрать способ кодирования на ленте аргументов и значений функции. Мы рассматриваем функции из N^n в N, где N - множество натуральных чисел, включая 0, а $n\geq 1$. Значения функции и ее аргументов будем записывать в бинарном, унарном или каком-либо ином коде, для этого нам потребуется соответственно алфавит A={1}, A={0,1} и т.д.

Значения аргументов перед вычислением должны быть представлены на ленте в виде псевдослова

$$\#x_n\#x_{n-1}\#\ldots\#x_1\overset{\downarrow}{\#}, \qquad (1)$$

где x_i - код i-го аргумента (i=1,2,...,n).

После вычисления содержимое ленты должно иметь вид:

$$\#x_n\#x_{n-1}\#\ldots\#x_1\#y\overset{\downarrow}{\#}, \qquad (2)$$

где y - код значения функции при заданных значениях аргумента.

УПРАЖНЕНИЯ

1. Составить программу, перерабатывающую псевдослово $\#u\overset{\downarrow}{\#}$ в псевдослово $\#u\#v\overset{\downarrow}{\#}$, где u - бинарный, а v - унарный код некоторого

числа из N.

2. Составить программу сложения и умножения чисел в унарном и бинарном кодах.

3. Составить программу для удвоения числа в бинарном и унарном кодах.

4. Составить программу деления нацело натуральных чисел в унарном коде.

§7. Частично-рекурсивные функции

Пытаясь определить об'ем интуитивного понятия вычислимой функции А.Черч в 1936 году рассмотрел класс так называемых рекурсивных функций, а Клини расширил его до класса частично-рекурсивных функций. В то же время впервые была высказана естественно научная гипотеза о том, что интуитивное понятие вычислимой частичной функции совпадает с понятием частично рекурсивной функции. Эту гипотезу называют тезисом Черча. Здесь мы напомним понятие частично-рекурсивной функции и покажем, что любая частично-рекурсивная функция вычислима по Тьюрингу. Набор аргументов $x_1,x_2,\ldots,x_n$ обозначим через $\bar{x}$.

Говорят, что n-местная функция $f(\bar{x})$ получена суперпозицией из n-местных функций $g_1,g_2,\ldots,g_m$ и m-местной функции h, если

$$f(\bar{x})=h(g_1(\bar{x}),g_2(\bar{x}),\ldots,g_m(\bar{x})).$$

Говорят, что (n+1)-местная функция $f(\bar{x},y)$ получена примитивной рекурсией из (n+2)-местной функции g и n-местной функции h, если

$$f(\bar{x},y)=\begin{cases} h(\bar{x}), \text{ если } y=0; \\ g(f(\bar{x},y-1),\bar{x},y-1), \text{ если } y>0. \end{cases}$$

Говорят, что n-местная функция $f(\bar{x})$ получена минимизацией из (n+1)-местной функции g, если

$$f(\bar{x})=\begin{cases} k, & \text{если } g(k,\bar{x})=0 \text{ и при всех } k'<k \\ & g(k',\bar{x}) \text{ определена и не равна } 0; \\ \text{неопределена в противном случае.} & \end{cases}$$

Обозначают $f(\bar{x})$ через $\mu y(g(y,\bar{x})=0)$ или $\min\{y \mid g(y,\bar{x})=0\}$.

Заметим, что суперпозиция и примитивная рекурсия, примененные к всюду определенным функциям, дают всюду определенные функции, тогда как минимизация, примененная к всюду определенной функции, может дать частичную функцию.

Числовая функция $f:N^n \to N$ называется частично рекурсивной, если она является одной из базисных функций:

а) $O(y)=0$ (при всех $y \in N$),

б) $S(y)=y+1$ (при всех $y \in N$),

в) $I^n_m(x_1,x_2,\ldots,x_n)=x_m$ $(1 \le m \le n,\ n=1,2,\ldots)$

или получена из них с помощью конечного числа применений суперпозиции, примитивной рекурсии и минимизации.

ТЕОРЕМА

Любая частично-рекурсивная функция вычислима по Тьюрингу.

Доказательство теоремы заключено в следующих четырех леммах, с использованием унарного кодирования чисел.

ЛЕММА I (о базисных функциях)

Базисные функции $O(y)$, $S(y)$, $I^n_m(x_1,x_2,\ldots,x_n)$ вычислимы по Тьюрингу.

Действительно, функцию $S(y)$ вычисляет программа $[K_1,1,r]$,

функцию $O(y)$ вычисляет программа r, функцию $I^n_m(x_1,x_2,\ldots,x_n)$ вычисляет программа K_m (см. таблицу 4).

ЛЕММА 2 (о суперпозиции)

Если функции

$$g_1(\bar{x}),g_2(\bar{x}),\ldots,g_m(\bar{x}) \text{ и } h(y_1,y_2,\ldots,y_m)$$

вычислимы соответственно программами $G_1,G_2,\ldots,G_m$ и H, то функцию $f(\bar{x})=h(g_1(\bar{x}),g_2(\bar{x}),\ldots,g_m(\bar{x}))$ вычисляет программа:

$$[G_m,Z^n_{n+1},G_{m-1},Z^n_{n+1},\ldots,G_1,Z^n_{n+1},Z^m_{n+m},H,\Lambda^m_2].$$

Чтобы убедиться в справедливости, достаточно выписать псевдослова, которые появляются на ленте после выполнения отдельных частей программы. В начале на ленте находится псевдослово

$$\#x_n\#x_{n-1}\#\ldots\#x_1\overset{\downarrow}{\#},$$

после выполнения G_m будет

$$\#x_n\#x_{n-1}\#\ldots\#x_1\#g_m\overset{\downarrow}{\#},$$

после $[G_m,Z^n_{n+1}]$ -

$$\#g_m\#x_n\#x_{n-1}\#\ldots\#x_1\overset{\downarrow}{\#},$$

после $[G_m,Z^n_{n+1},G_{m-1}]$ -

$$\#g_m\#x_n\#x_{n-1}\#\ldots\#x_1\#g_{m-1}\overset{\downarrow}{\#},$$

и т.д. после $[G_m,Z^n_{n+1},G_{m-1},Z^n_{n+1},\ldots,G_1,Z^n_{n+1}]$ -

$$\#g_m\#g_{m-1}\#\ldots\#g_1\#x_n\#x_{n-1}\#\ldots\#x_1\overset{\downarrow}{\#},$$

после $[G_m,Z^n_{n+1},G_{m-1},Z^n_{n+1},\ldots,G_1,Z^n_{n+1},Z^m_{n+m},H]$ -

$$\#x_n\#x_{n-1}\#\ldots\#x_1\#g_m\#g_{m-1}\#\ldots\#g_1\#f\overset{\downarrow}{\#},$$

и, наконец, после выполнения всей программы получим

$$\#x_n\#x_{n-1}\#\ldots\#x_1\#f\overset{\downarrow}{\#}.$$

ЛЕММА 3. (о примитивной рекурсии)

Если функции $h(x_1,x_2,\ldots,x_n)$ и $g(y_1,y_2,\ldots,y_{n+2})$ вычислимы соответственно с помощью программ H и G, то функция $f(x_1,x_2,\ldots,x_n,y)$, полученная по схеме примитивной рекурсии вида

$$f(\bar{x},y)=\begin{cases} h(\bar{x}), \text{ если } y=0; \\ g(f(\bar{x},y-1),\bar{x},y-1), \text{если} y>0, \end{cases}$$

вычислима программой

$$[H,L^{n+1},1,(\textit{пока } 1)[\#,R^{n+2},G,\Lambda_2,L^{n+2},1,1],R^{n+2}].$$

ЛЕММА 4. (о минимизации)

Если функция $g(x_1,x_2,\ldots,x_n)$ вычислима программой G, то функция $f(x_1,x_2,\ldots,x_n)$, полученная по схеме минимизации

$$f(\bar{x})=\begin{cases} k, \text{если } g(k,\bar{x})=0 \text{ и при всех } k'<k \\ \quad g(k',\bar{x}) \text{ определена и не равна } 0; \\ \text{неопределена, в противном случае,} \end{cases}$$

вычисляется программой

$$[r,G,1,(\textit{пока } 1)[r,\Lambda_1,1,r,G.1].$$

Для доказательства лемм 3 и 4 воспользуемся методом индуктивных утверждений.

ДОКАЗАТЕЛЬСТВО ЛЕММЫ 3.

Представим программу, предлагаемую для вычисления функции $f(x_1,x_2,\ldots,x_n,y)$, блок-схемой, изображенной на рисунке II.

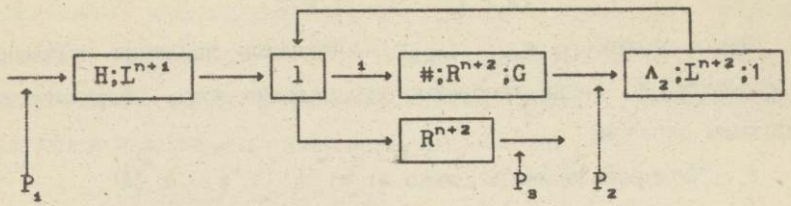


рис.II

Пунктирными стрелками показаны контрольные дуги, для которых будут сформулированы соответствующие индуктивные утверждения P_1,P_2,P_3.

Утверждение P_1 соответствует входной дуге и поэтому должно описывать содержимое ленты в начальный момент. Утверждение P_3 соответствует выходной дуге и должно описывать содержимое ленты в момент завершения работы программы. Утверждение P_2 относится к дуге, разрезающей единственный имеющийся в блок-схеме цикл, поэтому должно быть сформулировано так, чтобы ему удовлетворяло содержимое ленты каждый раз, когда в программе реализуется переход по рассматриваемой дуге.

Напомним, что основное требование, пред'являемое к утверждениям P_1, P_2, P_3 заключается в том, чтобы была возможность доказательства индуктивных шагов:

(i1) $P_1 \to P_2$, если реализуется путь $[H;\ L^{n+1};\ 1;\ \#;\ R^{n+2};\ G]$;

(i2) $P_1 \to P_3$, если реализуется путь $[H;\ L^{n+1};\ 1;\ R^{n+2}]$;

(i3) $P_2 \to P_2$, если реализуется путь

$$[\Lambda_2;\ L^{n+2};\ 1;\ 1;\ \#;\ R^{n+2};\ G];$$

(i4) $P_2 \to P_3$, если реализуется путь

$[\Lambda_2;\ L^{n+2};\ 1;\ 1;\ R^{n+2}]$.

Пусть теперь $x_1, x_2, \ldots, x_n, y$ - исходные значения аргументов из множества N, тогда требуемые утверждения можно сформулировать следующим образом.

P_1. "Содержимое ленты равно $\#1^{y}\#1^{x_n}\#1^{x_{n-1}}\#\ldots\#1^{x_1}\underset{\downarrow}{\#}$".

P_2. "Существуют $y_1 \geq 1$, $y_2, g_1, g_2 \geq 0$ такие, что содержимое ленты равно $\#1^{y_2}\#1^{y_1}\#1^{x_n}\#1^{x_{n-1}}\#\ldots\#1^{x_1}\#1^{g_1}\#1^{g_2}\underset{\downarrow}{\#}$ и $y_1+y_2+1=y$, $g_1=g(f(x_1,x_2,\ldots,x_n,y_1-1),x_1,x_2,\ldots,x_n,y_1-1)$, $g_2=g(f(x_1,x_2,\ldots,x_n,y_1),x_1,x_2,\ldots,x_n,y_1)$".

P_3."Содержимое ленты равно $\#1^{y}\#1^{x_n}\#1^{x_{n-1}}\#\ldots\#1^{x_1}\#1^{z}\underset{\downarrow}{\#}$ и $z=f(x_1,x_2,\ldots,x_n,y)$".

Доказательство индуктивных шагов легко получить, выписывая содержимое ленты после каждого оператора в соответствующем пути. Читателю предоставляется возможность это проделать и убедиться в правильности программы, предлагаемой в формулировке леммы 3. Не забудьте доказать завершаемость исследуемой программы.

ДОКАЗАТЕЛЬСТВО ЛЕММЫ 4

Представим программу, предлагаемую для вычисления функции $f(x_1,x_2,\ldots,x_n)$ блок-схемой (рис I2)

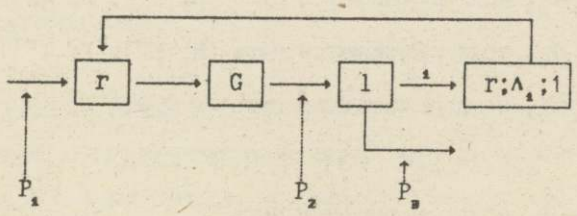


рис.I2

с указанными контрольными точками. Пусть $x_1, x_2, \ldots, x_n$ - исходные значения аргументов, тогда для доказательства частичной корректности предлагаемой программы можно воспользоваться следующими индуктивными утверждениями.

P_1. "Содержимое ленты равно

$$\#1^{x_n}\#1^{x_{n-1}}\#\ldots\#1^{x_1}\underset{\downarrow}{\#}".$$

P_2. "Существуют $k, z \geq 0$ такие, что содержимое ленты равно

$$\#1^{x_n}\#1^{x_{n-1}}\#\ldots\#1^{x_1}\#1^{k}\#1^{z}\underset{\downarrow}{\#},$$

где $z=g(k,x_1,x_2,\ldots,x_n)$".

P_3. "Содержимое ленты равно

$$\#1^{x_n}\#1^{x_{n-1}}\#\ldots\#1^{x_1}\#1^{z}\underset{\downarrow}{\#},$$

где $z=f(x_1,x_2,\ldots,x_n)$".

Требуется доказать следующие индуктивные шаги.

(i1) $P_1 \to P_2$, если реализуется путь [r;G].

(i2) $P_2 \to P_2$, если реализуется путь $[1;r;\Lambda_1;1;r;G]$.

i3) $P_2 \to P_3$, если реализуется путь [1] и головка остановится на символе #.

Доказательство индуктивных утверждений и завершаемости программы предоставляется читателю в качестве упражнений.

Заметим, что в доказательствах лемм 3 и 4 при рисовании блок-схем мы несущественно отступили от текстов программ, данных в их формулировках, а при доказательстве леммы 2 не выписывали индуктивные утверждения, так как в представлении программы блок-схемой нет циклов. Обращаем внимание на то, что правильность блоков из которых составлены программы мы не подвергаем сомнению.

§8. Универсальная тьюрингова программа

В этом параграфе мы убедимся в существовании универсальной тьюринговой программы U, которая может имитировать любую программу, работающую над алфавитом $A=\{a_1,a_2,\dots,a_t\}$. Эта программа U, получив на входе псевдослово, содержащее в определенном виде код произвольной программы T и псевдослово X, оставит на ленте код программы T и псевдослово T(X) - результат работы программы T на псевдослове X.

Опишем сначала способ кодирования моделируемой программы T. Будем представлять ее в виде последовательности команд, разделенных символами C или Г. Символ Г будет расположен перед активной (т.е. моделируемой в данный момент) командой и перед всеми командами, расположенными левее ее. Символ C будет стоять перед всеми командами, расположенными правее активной и после последней команды. В начальный момент символ Г будет расположен только перед первой командой.

Команда с номером i будет иметь вид

$$d_i a_0 \alpha_{i0} a_1 \alpha_{i1} \dots a_t \alpha_{it},$$

где d_i - символ из алфавита $\tilde{A}\cup\{r,l,s\}$, описывающий действие команды, а α_{ij} - слово, определяющее команду программы T, которая должна выполняться следующей в случае, когда после выполнения действия d_i обозревается символ a_j $(j=0,1,\dots,t)$. Если α_{ij} состоит из k знаков +(-), то следующая команда расположена на k команд правее (левее) текущей; если α_{ij} состоит из одного знака 0, то следующей выполняется та же самая команда; если α_{ij} состоит из одного знака v, то следующей команды нет, а текущая является выходной.

После символа C, расположенного за последней командой программы T, помещаем символ "b" из псевдослова X, который должен обозреваться программой T в начальный момент (в процессе работы программы U на месте этого символа будет размещаться символ, обозреваемый перед очередной командой программы T).

За символом "b" помещаем символ §, а затем псевдослово X', полученное из псевдослова X заменой символа "b" на новый вспомогательный символ γ. Символ § ставим также перед первой командой программы.

Итак, алфавит универсальной программы U кроме основных символов $a_0,a_1,\dots,a_t$ содержит отличные от них вспомогательные символы r,l,s,c,Г,γ,+,-,0,v,§, которые будем обозначать также через $a_{t+1},a_{t+2},\dots,a_{t+11}$.

Входным псевдословом программы U, согласно нашему описанию, будет псевдослово

$$\S Г q_1 c q_2 c \dots c q_s c b \S X',$$

где q_i - i-тая команда программы T.

Для составления программы U построим сначала программы D,P,Q, выполняющие отдельные ее этапы.

Программа D

$$r,\bigvee_{i=0}^{t+3}(\text{если } a_i)[r(\text{до } \S),l,\bigvee_{j=0}^{t}(\text{если } a_j)[r(\text{до}\gamma),a_j],a_i],$$

$$(\text{если } \S)[l(\text{до}\S),[\bigvee_{i=0}^{t+10}(\text{если } a_i)[l,a_i,r^2]](\text{до}\S),\#,l,\S),$$

обозревая в начальный момент символ Г активной команды, выполняет в соответствии с ней преобразование псевдослова и останавливается на символе, над которым будет проведено действие следующей моделируемой командой. Часть программы D, расположенная за

условием (*если* §), выполняется в том случае, когда очередное моделируемое действие сдвигает головку за левый конец псевдослова x. В этом случае код программы сдвигается влево на одну ячейку, давая пространство для работы программы T.

Программа P

$$\bigvee_{i=0}^{t} (\textit{если}\ a_i)[\gamma, l(\textit{до}\,\S), l, a_i, l(\textit{до}\,\Gamma), r(\textit{до}\,a_i)],$$

обозревая символ "a_i" в псевдослове X, заменяет его на γ, затем помещает a_i на место символа "b" и сдвигает головку до символа a_i в активной команде, определяющего очередную моделируемую команду.

Программа Q

Γ,(*если* +)[(*пока* +)[§,r(*до* c),Γ,l(*до* §),+,r],r(*до*c),l(*до*Γ)],

(*если* −)[(*пока* −)[§,l(*до*Γ),c,r(*до* §),−,r],l(*до*Γ),

(*если* 0)[l(*до*Γ)],

обозревая в начальный момент символ a_i, на котором остановилась программа P, выполняет следующие действия. Если после a_i стоят k знаков "+", то k символов "c", расположенных правее активной команды, заменяются на Γ. Если после a_i стоят k знаков "−", то k символов Γ, расположенных левее активной команды, заменяются на c. В обоих этих случаях, а также в случае, когда после a_i стоит 0, головка передвигается на символ Γ перед активной командой. Если после a_i стоит символ v, то головка останавливается на этом символе v.

Используя программы D,P,Q, универсальную программу U можно записать теперь следующим образом

[D,P,Q](*до* v).

§9. Пример невычислимой функции

Пусть $\mathfrak{F}_n$ - множество всех функций из N в N таких, что каждая из них определена в точке 0 и вычислима программой, состоящей не более чем из n команд. Рассмотрим функцию $\sigma(n)=\max\limits_{\phi\in\mathfrak{F}_n}\phi(0)$. Очевидно, $\sigma(n)$ всюду определена и монотонна.

ТЕОРЕМА

Для любой всюду определенной вычислимой функции f: N→N существует $k\in N$ такое, что при любом $m\geq k$ $f(m)<\sigma(m)$.

Действительно, пусть f(n) - произвольная всюду определенная функция из N в N, тогда, очевидно, функция

$$F(n)=\max(f(3n),f(3n+1),f(3n+2))+1$$

будет также всюду определенной и вычислимой. Пусть в унарном коде ее вычисляет программа T_F, состоящая из k команд. Рассмотрим программу $\tilde{T}=[[1,r]^n,T_F]$, которая сначала записывает на ленте число n, а затем работает как T_F. Очевидно, она состоит из 2n+k команд и вычисляет некоторую функцию $\tilde{F}\in\mathfrak{F}_{2n+k}$.

По определению $\tilde{F}$ и σ имеем $F(n)=\tilde{F}(0)\leq\sigma(2n+k)$. Используя монотонность функции σ, получим $F(n)\leq\sigma(3n)$ при всех $n\geq k$, следовательно, $f(3n+i)<\sigma(3n+i)$, $(i=0,1,2)$. Отсюда следует, что при $m\geq 3k$ $f(m)<\sigma(m)$, что и требовалось доказать.

СЛЕДСТВИЕ. Функция $\sigma(n)$ невычислима.

Заметим, что при любом фиксированном n значение $\sigma(n)$можно попытаться вычислить путем перебора всех программ длины $\leq n$ и вычисления для каждой из них времени работы до момента остановки или доказательства ее незавершаемости. Но вопрос о завершаемости

программ в общем виде алгоритмически неразрешим.

Уточним основные моменты этого утверждения.

Пусть T - тьюрингова программа, работающая в алфавите A, и пусть kod(T) - слово в алфавите A, кодирующее программу T (на деталях кодирования не останавливаемся).

Программа T называется самоприменимой, если при подаче ей на вход ее собственного кода, она через конечное число шагов остановится; в противном случае программа называется несамоприменимой. Пусть M - множество кодов самоприменимых программ.

ТЕОРЕМА. Множество M кодов самоприменимых программ алгоритмически неразрешимо.

ДОКАЗАТЕЛЬСТВО. Пусть M - алгоритмически разрешимо. Тогда существуют две программы T_1 и T_2 такие, что T_1 останавливается только на словах из M, а T_2 - только на словах из $\bar{M} = A^*\backslash M$. Тогда, если T_2 на своем собственном коде остановится, то по определению M $kod(T_2)\in M$, а по определению T_2 $kod(T_2)\notin M$. Если же T_2 на своем собственном коде не остановится, то по определению M $kod(T_2)\notin M$, а по определению T_2 $kod(T_2)\in M$. Итак, в любом случае имеем противоречие.

Еще одним примером алгоритмически неразрешимого множества является множество M_1 кодов программ, которые останавливаются при пустом входе. Легко показать, что если бы M_1 было разрешимым, то множество M тоже было бы разрешимым.

УПРАЖНЕНИЯ

I) Построить программу, моделирующую параллельную пошаговую

работу двух программ до тех пор, пока одна из них не достигнет выхода.

Указание. Разбить ленту на четные и нечетные ячейки. В четные заносить код первой машины и обрабатываемое слово, в нечетные - код и обрабатываемое слово второй машины. Использовать программы D,P,Q, в которых всюду элементарная программа r заменена на r^2, а l - на l^2.

2) Доказать, что словарный n-местный предикат $R\subseteq(A^*)^n$ разрешим тогда и только тогда, когда существует прогамма T, которая при любом входном псевдослове X останавливается и в момент остановки обозревает символ I, если $(x_1,\dots,x_n)\in R$ и символ 0, если $(x_1,\dots,x_n)\notin R$. Через $x_1,\dots,x_n$ обозначены первые n слов псевдослова X.

3) Доказать, что множество кодов программ, которые останавливаются при пустом входе, алгоритмически неразрешимо.

§10. Об измерении алгоритмической сложности задач

При практическом решении многих интересных задач с помощью вычислительных автоматов существенную роль играет время работы и об'ем памяти, которые требуются для их решения соответствующими алгоритмами. Упомянутые характеристики называются, соответственно, временной и пространственной сложностью алгоритма.

Временной и пространственной сложностью задачи в классе алгоритмов (или автоматов) называется время и об'ем памяти, требующиеся "лучшему" в данном классе алгоритму (автомату), для решения рассматриваемой задачи.

Вопрос о нахождении сложностных характеристик задач весьма труден. Трудности как правило связаны с логической сложностью рассматриваемых задач и обилием алгоритмов в любом универсальном классе.

Рассмотрим некоторые подробности на примере тьюринговых программ.

Считаем, что задача представлена двухместным словарным предикатом $R(u,v)$ над некоторым алфавитом A и заключается в нахождении по заданному слову $u \in A^*$ слова $v \in A^*$ такого, что $R(u,v)$ истинно.

Слово v будем считать решением задачи R при входном слове u. Чтобы пустоту слова v трактовать как отсутствие решения, наложим на R следующее ограничение

$$R(\lambda,\lambda),\ \forall u \exists v R(u,v),$$

$$\forall u[R(u,\lambda) \to \forall v[v \neq \lambda \to \neg R(u,v)]].$$

Результат работы тьюринговой программы T на входном слове u обозначим $T(u)$, считая $T(u)$ равным выходному слову.

Будем говорить, что тьюрингова программа T решает задачу R, если на любом входном слове u она останавливается через конечное число шагов и $\forall u R(u,T(u))$.

Через $time(T,u)$ будем обозначать число элементарных тьюринговых команд, которые будут выполнены программой T от начального момента до момента остановки при работе на входном слове u. Если на входе u программа T выполняет бесконечное число шагов, то считаем $time(T,u)=\infty$.

Величину $time(T,u)$ будем называть так же временем работы программы T на слове u.

В большинстве случаев величина $time(T,u)$ существенно зависит

от длины слова u, поэтому представляет интерес функция $t(T,n)=\max\limits_{|u|=n} time(T,u)$, где максимум вычисляется по всем словам длины n.

Заметим, что эта ситуация не является общей; в некоторых случаях величина $time(T,u)$ не зависит от длины слова u. Например, пусть требуется определить четность числа, представленного в двоичном коде. Для этого достаточно посмотреть на его младший разряд.

Временной сложностью задачи R называют функцию

$$f_R(n)=\min_T t(T,n),$$

где min вычисляется по всем программам T, решающим задачу R.

Точное нахождение функции $f_R(n)$ наталкивается, как правило, на непреодолимые трудности, поэтому ограничивается нахождением для нее верхней и нижней оценочных функций. Чтобы найти какую-нибудь верхнюю оценочную функцию достаточно спроектировать некоторую программу T и оценить сверху функцию $t(T,n)$, что часто оказывается посильной задачей. Качество верхней оценочной функции зависит от того, насколько хитроумна спроектированная программа.

Что касается нижних оценок, то всегда можно временную сложность оценить нулевой функцией, однако, получение нетривиальных оценок каждый раз представляет собой сложную математическую задачу. Известно, например, что временная сложность задачи распознавания симметрии слова тьюринговыми программами оценивается снизу функцией cn^2 при некоторой константе c.

Для многих интересных с практической точки зрения задач известные верхние оценки носят экспоненциальный характер, что свидетельствует о больших затратах времени при выполнении

соответствующих алгоритмов на вычислительных установках.

Если для задачи R имеется полиномиальная от n верхняя оценочная функция, то говорят, что R разрешима в полиномиальное время.

В последние десятилетия обнаружено большое число задач $R(u,v)$, для которых существование верхних полиномиальных оценочных функций не установлено, однако задача распознавания на паре слов u,v, является ли слово v решением задачи R на входе u, решается за полиномиальное от длины слова u время и, кроме того, для каждого u существует ответ v, длина которого огрничена некоторым полиномом от длины u. Это так называемые задачи с проверяемым за полиномиальное время ответом.

Такой задачей является, например, задача о выполнимости булевых формул, которая заключается в следующем. По заданой булевой формуле найти набор значений переменных, при которых соответствующая формуле булева функция принимает значение 1. Имея минимальный программистский опыт, легко убедиться в возможности проверки ответа к этой задаче за полиномиальное время.

Задачи с проверяемым за полиномиальное время ответом называются переборными с гарантированным экспоненциальным перебором. Действительно, пусть $R(u,v)$ такая задача и длина возможного ответа v ограничена полиномом p от длины входа u, то есть $|v|\le p(|u|)$. Пусть, далее q - полином, являющийся верхней оценкой времени работы проверяющей ответ программы, тогда, перебирая всевозможные $2^{p(|u|)}$ слов длины $p(|u|)$ и затрачивая на проверку каждого не более $q(|u|)$ тактов времени, получим алгоритм с верхней оценкой $q(u)2^{p(|u|)}$.

Задача R полиномиально сводится к задаче R', если существуют

работающие полиномиальное время тьюринговы программы T_1 и T_2 такие, что

$$\forall u \forall v [R'(T_1(u),v') \to R(u,T_2(v'))].$$

Из этого определения следует, что для решения задачи R на входе u достаточно применить к слову u программу T_1, затем найти ответ v' задачи R' на входе $u'=T_1(u)$ и, наконец, применить к v' программу T_2, получив ответ $v=T_2(v')$ задачи R на входе u. Таким образом, имеем следующую схему получения ответа v:

$$\boxed{u} \xrightarrow{T_1} \boxed{u'} \xrightarrow{R'} \boxed{v'} \xrightarrow{T_2} \boxed{v}$$

Оказывается, что среди задач с полиномиально проверяемым ответом существует задача, к которой полиномиально сводится любая другая задача с полиномиально проверяемым ответом. Такие задачи получили название универсальных переборных задач.

Исторически существование универсальных переборных задач обнаружено в 1971 году американским математиком Куком, когда он доказал, что задача выполнимости булевой формулы является универсальной переборной задачей.

В то же время было доказано, что и многие другие широко известные задачи являются универсальными переборными задачами.

Круг таких задач в настоящее время постоянно расширяется. По данному вопросу имеется обширная литература.

ЛИТЕРАТУРА

I. Булос Дж., Джеффри Р. Вычислимость и логика/ Пер. с англ. - М.: Мир, 1994.

2. Ершов Ю.Л., Палютин Е.А. Математическая логика. М.: Наука, 1979.

3. Катлэнд Н. Вычислимость. Введение в теорию рекурсивных функций/ Пер. с англ.- М.: Мир, 1983.

4. Мальцев А.И. Алгоритмы и рекурсивные функции. М.: Наука,1965.

5. Эббингауз Г.-Д., Манн Ф.-К., Хермес Г. Машины Тьюринга и рекурсивные функции. М.: Мир, 1972.

6. Яблонский С.В. Введение в дискретную математику. М.: Наука, 1979.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3

Часть I. ЛОГИЧЕСКИЙ ЯЗЫК ПЕРВОГО ПОРЯДКА 5

§I. Предварительные сведения 5

§2. Синтаксис логического языка первого порядка . . 8

§3. Интерпретация формул логического языка первого порядка I2

§4. Способы задания и подсчет числа структур на конечных универсах 2I

§5. Исключающие кванторы 25

Часть 2. ЛОГИЧЕСКИЙ ВЫВОД И ЭЛЕМЕНТАРНЫЕ ТЕОРИИ 29

§I. Логический вывод 29

§2. Канонические формы предложений в логике первого порядка 44

§3. Моделирование математических теорий 48

§4. Примеры формализации математических теорий . . 50

§5. Свойства элементарных теорий 54

§6. Некоторые замечания о возможностях формализации математических теорий 56

§7. Расширение элементарных теорий 60

Часть 3. ПРИБЛИЖЕННОЕ ВЫРАЖЕНИЕ СВОЙСТВ СТРУКТУР В ЛОГИЧЕСКИХ ЯЗЫКАХ 63

§I. Доля выполнимости логических формул 63

§2. Разрешимость свойства асимптотической истины в логике первого порядка 64

§3. О приближенной выразимости свойств структур в логических языках 69

Часть 4. РЕЛЯЦИОННЫЙ ЯЗЫК 72
§I. Синтаксис реляционного языка 72
§2. Семантика реляционного языка 73
§3. Система доказательств в реляционном языке . . . 74
Часть 5. МОДЕЛИ ВЫЧИСЛЕНИЙ 79
§I. Исторические сведения 79
§2. Тьюрингова модель переработки информации . . . 82
§3. Алгебра программ 87
§4. Начальное математическое обеспечение 9I
§5. Методика доказательства правильности алгоритмов с помощью индуктивных утверждений 93
§6. Вычислимость и разрешимость 95
§7. Частично-рекурсивные функции 97
§8. Универсальная тьюрингова программа I04
§9. Пример невычислимой функции I07
§I0. Об измерении алгоритмической сложности задач. I09
Литература II4

Владимир Александрович ТАЛАНОВ

МАТЕМАТИЧЕСКАЯ ЛОГИКА И МОДЕЛИ ВЫЧИСЛЕНИЙ

Учебное пособие

Редактор О. В. Широкова

ИБ 298

Темплан 1994 г. Позиция 45

Подписано в печать 24.03.95. Формат 60×84 1/16.
Бумага писчая. Печать офсетная.
Усл.-печ. л. 7,5. Уч.-изд. л. 7,0.
Тираж 600 экз. Заказ 597. С 171

Издательство Нижегородского государственного университета
им. Н. И. Лобачевского
603600, Нижний Новгород, пр. Гагарина, 23.

Типография ННГУ, Нижний Новгород, ул. Б. Покровская, 37.

В. А. ТАЛАНОВ

МАТЕМАТИЧЕСКАЯ ЛОГИКА И МОДЕЛИ ВЫЧИСЛЕНИЙ