

# Оглавление

<b>1 Математическая логика</b>	<b>2</b>
1.1 Пропозициональные формулы . . . . .	2
1.1.1 Основные равносильности . . . . .	2
1.1.2 Дизъюнктивная и конъюнктивная формы . . . . .	3
1.2 Исчисления высказываний . . . . .	4
1.2.1 Секвенциальное исчисление высказываний . . . . .	4
1.2.2 Метод резолюций для исчисления высказываний . . . . .	7
1.3 Исчисления предикатов . . . . .	8
<b>2 Формальные теории</b>	<b>12</b>
<b>3 Теория алгоритмов</b>	<b>18</b>
3.1 Рекурсивные функции . . . . .	18
3.2 Машины Тьюринга . . . . .	19
3.2.1 Многоленточные МТ . . . . .	21
3.2.2 Многоголовчатые МТ . . . . .	21
3.2.3 Недетерминированные МТ . . . . .	22
3.3 Нормальные алгоритмы Маркова . . . . .	22
3.4 Код алгоритма. Применимость алгоритма к данным . . . . .	23
3.5 Массовые проблемы. Алгоритмическая разрешимость и неразрешимость . . . . .	24
3.6 Теория сложности алгоритмов . . . . .	26
3.6.1 Полиномиальная сводимость и полиномиальная эквивалентность . . . . .	27
3.6.2 NP-полные задачи . . . . .	28
3.6.3 Задача Выполнимость . . . . .	28

# Глава 1

## Математическая логика

### 1.1. Пропозициональные формулы

**Определение 1.** Высказыванием называется утверждение, относительно которого однозначно можно сказать, истинно оно или ложно.

**Определение 2.** Пропозициональной переменной называется переменная для высказываний.

**Определение 3.**

1. логические константы являются пропозициональными формулами;
2. пропозициональные переменные являются пропозициональными формулами;
3. если  $A$  — пропозициональная формула, то  $\neg A$  является пропозициональной формулой;
4. Если  $A$  и  $B$  — пропозициональные формулы,  $*$  — бинарная логическая связка, то  $(A * B)$  является пропозициональной формулой;
5. никакие другие выражения не являются пропозициональными формулами.

**Определение 4.** Пропозициональная формула называется тавтологией, если она истинна при всех наборах значений своих переменных.

**Определение 5.** Пропозициональная формула называется противоречием, если она ложна при всех наборах значений своих переменных.

**Определение 6.** Пропозициональная формула называется выполнимой, если она истинна хоть на одном наборе своих переменных.

**Определение 7.** Пропозициональные формулы называются равносильными, если их значения совпадают при всех наборах значений их переменных.

#### 1.1.1. Основные равносильности

Пусть  $T$  — тавтология,  $F$  — противоречие.

Идемпотентность	$A \& A \iff A$
	$A \vee A \iff A$
Коммутативность	$A \& B \iff B \& A$
	$A \vee B \iff B \vee A$
Ассоциативность	$(A \& B) \& C \iff A \& (B \& C)$
	$A \& B \& C$
	$(A \vee B) \vee C \iff A \vee (B \vee C)$
	$A \vee B \vee C$
Дистрибутивность	$A \& (B \vee C) \iff A \& B \vee A \& C$

	$A \vee B \ \& \ C$	$\iff$	$(A \vee B) \ \& \ (A \vee C)$
Поглощение	$A \ \& \ (A \vee B)$	$\iff$	$A$
	$A \vee A \ \& \ B$	$\iff$	$A$
Правило де Моргана	$\neg(A \ \& \ B)$	$\iff$	$\neg A \vee \neg B$
	$\neg(A \vee B)$	$\iff$	$\neg A \ \& \ \neg B$
Склейивание	$A \ \& \ B \vee \neg A \ \& \ C$	$\iff$	$A \ \& \ B \vee \neg A \ \& \ C \vee B \ \& \ C$
	$(A \vee B) \ \& \ (\neg A \vee C)$	$\iff$	$(A \vee B) \ \& \ (\neg A \vee C) \ \& \ (B \vee C)$
Двойное отрицание	$\neg\neg A$	$\iff$	$A$
	$A \rightarrow B$	$\iff$	$\neg A \vee B$
	$A \leftrightarrow B$	$\iff$	$(A \rightarrow B) \ \& \ (B \rightarrow A)$ $(\neg A \vee B) \ \& \ (\neg B \vee A)$ $(A \ \& \ B) \vee (\neg A \ \& \ \neg B)$
	$A \oplus B$	$\iff$	$\neg(A \leftrightarrow B)$ $(\neg A \ \& \ B) \vee (A \ \& \ \neg B)$ $(A \vee B) \ \& \ (\neg B \vee \neg A)$
	$A   B$	$\iff$	$\neg(A \ \& \ B)$
	$A \downarrow B$	$\iff$	$\neg(A \vee B)$
	$A \ \& \ T$	$\iff$	$A$
	$A \vee T$	$\iff$	$T$
	$A \ \& \ F$	$\iff$	$F$
	$A \vee F$	$\iff$	$A$
	$A \rightarrow T$	$\iff$	$T$
	$A \rightarrow F$	$\iff$	$\neg A$
	$T \rightarrow A$	$\iff$	$A$
	$F \rightarrow A$	$\iff$	$F$
	$A \leftrightarrow T$	$\iff$	$A$
	$A \leftrightarrow F$	$\iff$	$\neg A$

### 1.1.2. Дизъюнктивная и конъюнктивная формы

**Определение 8.** Элементарной конъюнкцией называется многократная конъюнкция переменных и их отрицаний.

**Определение 9.** Дизъюнктивной нормальной формой называется пропозициональная формула, являющаяся многократной дизъюнкцией элементарных конъюнкций.

**Теорема 1 (о ДНФ).** По всякой пропозициональной формуле, не являющейся противоречием, можно построить равносильную ей в ДНФ.

**Идея доказательства.** В таблице истинности для формулы есть хоть одна строка, в столбце значений которой стоит И. Для каждой такой строки, соответствующей набору значений  $(\alpha_1^i, \dots, \alpha_n^i)$ , выписываем элементарную конъюнкцию  $K_i = x_1^{\alpha_1^i} \ \& \ \dots \ \& \ x_n^{\alpha_n^i}$ , значение которой равно И на наборе  $(\alpha_1^i, \dots, \alpha_n^i)$  и только на нём. Формула в ДНФ  $K_1 \vee \dots \vee K_m$  принимает значение И на выбранных строках и только на них.

Так построенная формула называется *совершенной ДНФ*.

**Определение 10.** Элементарной дизъюнкцией называется многократная дизъюнкция переменных и их отрицаний.

**Определение 11.** Конъюнктивной нормальной формой называется пропозициональная формула, являющаяся многократной конъюнкцией элементарных дизъюнкций.

**Теорема 2 (о КНФ).** По всякой формуле, не являющейся тавтологией, можно построить равносильную ей в КНФ.

**Идея доказательства.** Отрицание формулы не является противоречием, следовательно можно построить формулу в ДНФ. Отрицание формулы в ДНФ, после применения правил де Моргана, является формулой в КНФ.

## 1.2. Исчисления высказываний

Для того, чтобы задать исчисление, достаточно задать

1. Алфавит — конечное множество символов  $A = \{ a_1, \dots, a_n \}$ .
2. Множество формул — слов в алфавите, для которых существует эффективная процедура проверки, является ли слово формулой.
3. Множество аксиом — эффективно проверяемое подмножество множества формул.
4. Правила вывода — конечное множество отношений над формулами, для каждого из которых все аргументы, кроме последнего, называются *посылками правила*, а последний аргумент — *заключением правила*.

**Определение 12.** Формула  $B$  непосредственно выводима из формул  $A_1, \dots, A_n$ , если имеется  $n + 1$ -местное отношение  $R$ , задающее одно из правил вывода, такое, что  $R(A_1, \dots, A_n, B)$ .

**Определение 13.** Выводом в исчислении называется последовательность формул, каждая из которых либо является аксиомой, либо непосредственно выводима из некоторых предыдущих.

**Определение 14.** Последняя формула любого вывода в заданном исчислении называется *выводимой* в нём (*теоремой* этого исчисления).

**Обозначение.**  $\models_C P - P$  выводима в исчислении  $C$

**Определение 15.** Формула  $B$  выводима из множества формул  $\Sigma$ , если найдётся последовательность формул, заканчивающихся формулой  $B$ , каждая из которых либо является аксиомой, либо входит в множество  $\Sigma$ , либо непосредственно выводима из некоторых предыдущих.

**Обозначение.**  $\Sigma \models_C P - P$  выводима из  $\Sigma$  в исчислении  $C$

**Определение 16.** Исчисление называется *полным*, если всякая истинная формула этого исчисления выводима в нём.

**Определение 17.** Исчисление называется *непротиворечивым*, если не существует такой формулы этого исчисления, что выводима она сама и её отрицание.

### 1.2.1. Секвенциальное исчисление высказываний

**Определение 18.** Секвенцией называется выражение вида  $\Gamma \vdash \Delta$ , где  $\Gamma, \Delta$  — списки (возможно, пустые) формул,  $\vdash$  — знак секвенции.

Г называется *антecedентом* секвенции,  $\Delta$  — *сукцедентом* (*консеквентом*) секвенции.

Каждой секвенции можно поставить в соответствие формулу, которая называется *формулённым* образом секвенции:

$$\Phi(A_1 \dots A_n \vdash B_1 \dots B_m) = A_1 \& \dots \& A_n \rightarrow (B_1 \vee \dots \vee B_m)$$

Формальное описание:

1. Алфавит: множество символов для записи имён пропозициональных переменных, объединённое с множеством логических связок  $\{\&, \neg\}$ , символа секвенции и скобок.
2. Формулы исчисления: секвенции, содержащие пропозициональные формулы в выбранном алфавите.
3. Аксиомы исчисления: единственная схема аксиом

$$\Gamma_1 \ A \ \Gamma_2 \vdash \Delta_1 \ A \ \Delta_2$$

4. Правила вывода

$$\begin{array}{c}
(\vdash \neg) \frac{\Gamma_1 A \Gamma_2 \vdash \Delta_1 \Delta_2}{\Gamma_1 \Gamma_2 \vdash \Delta_1 \neg A \Delta_2} \\
(\vdash \&) \frac{\Gamma \vdash \Delta_1 A \Delta_2 \quad \Gamma \vdash \Delta_1 B \Delta_2}{\Gamma \vdash \Delta_1 (A \& B) \Delta_2} \\
(\vdash \vee) \frac{\Gamma \vdash \Delta_1 A B \Delta_2}{\Gamma \vdash \Delta_1 (A \vee B) \Delta_2} \\
(\vdash \rightarrow) \frac{\Gamma_1 A \Gamma_2 \vdash \Delta_1 B \Delta_2}{\Gamma_1 \Gamma_2 \vdash \Delta_1 (A \rightarrow B) \Delta_2} \\
(\vdash \leftrightarrow) \frac{\Gamma_1 A \Gamma_2 \vdash \Delta_1 B \Delta_2 \Gamma_1 B \Gamma_2 \vdash \Delta_1 A \Delta_2}{\Gamma_1 \Gamma_2 \vdash \Delta_1 (A \leftrightarrow B) \Delta_2} \\
(\vdash \oplus) \frac{\Gamma_1 \Gamma_2 \vdash \Delta_1 A B \Delta_2 \Gamma_1 A B \Gamma_2 \vdash \Delta_1 \Delta_2}{\Gamma_1 \Gamma_2 \vdash \Delta_1 (A \oplus B) \Delta_2}
\end{array}
\qquad
\begin{array}{c}
(\neg \vdash) \frac{\Gamma_1 \Gamma_2 \vdash \Delta_1 A \Delta_2}{\Gamma_1 \neg A \Gamma_2 \vdash \Delta_1 \Delta_2} \\
(\& \vdash) \frac{\Gamma_1 A B \Gamma_2 \vdash \Delta}{\Gamma_1 (A \& B) \Gamma_2 \vdash \Delta} \\
(\vee \vdash) \frac{\Gamma_1 A \Gamma_2 \vdash \Delta \quad \Gamma_1 B \Gamma_2 \vdash \Delta}{\Gamma_1 (A \vee B) \Gamma_2 \vdash \Delta} \\
(\rightarrow \vdash) \frac{\Gamma_1 \Gamma_2 \vdash \Delta_1 A \Delta_2 \quad \Gamma_1 B \Gamma_2 \vdash \Delta_1 \Delta_2}{\Gamma_1 (A \rightarrow B) \Gamma_2 \vdash \Delta_1 \Delta_2} \\
(\leftrightarrow \vdash) \frac{\Gamma_1 \Gamma_2 \vdash \Delta_1 A B \Delta_2 \quad \Gamma_1 A B \Gamma_2 \vdash \Delta_1 \Delta_2}{\Gamma_1 (A \leftrightarrow B) \Gamma_2 \vdash \Delta_1 \Delta_2} \\
(\oplus \vdash) \frac{\Gamma_1 A \Gamma_2 \vdash \Delta_1 B \Delta_2 \quad \Gamma_1 B \Gamma_2 \vdash \Delta_1 A \Delta_2}{\Gamma_1 (A \oplus B) \Gamma_2 \vdash \Delta_1 \Delta_2}
\end{array}$$

**Теорема 3.** Формульный образ аксиомы является тавтологией.

**Доказательство.** Пусть аксиома имеет вид  $P_1 \dots P_k A Q_1 \dots Q_l \vdash R_1 \dots R_m A T_1 \dots T_n$ . Её формульным образом является формула

$$P_1 \& \dots \& P_k \& A \& Q_1 \& \dots \& Q_l \vdash R_1 \vee \dots \vee R_m \vee A \vee T_1 \vee \dots \vee T_n,$$

которая равносильна элементарной дизъюнкции

$$\neg P_1 \vee \dots \vee \neg P_k \vee \neg A \vee \neg Q_1 \vee \dots \vee \neg Q_l \vee R_1 \vee \dots \vee R_m \vee A \vee T_1 \vee \dots \vee T_n,$$

содержащей контрапротивную пару  $\neg A \vee A$  и, следовательно, являющейся тавтологией.  $\square$

**Теорема 4.** Для каждого правила вывода формульный образ его заключения равносителен конъюнкции формульных образов посылок этого правила.

**Доказательство.** Доказательство проведём для правила  $(\vdash \&)$ . Формульные образы посылок:

$$P_1 \dots P_k \vdash R_1 \dots R_m A T_1 \dots T_n, \quad P_1 \dots P_k \vdash R_1 \dots R_m B T - 1 \dots T_n$$

равносильны соответственно

$$\neg P_1 \vee \dots \vee \neg P_k \vee R_1 \vee \dots \vee R_m \vee A \vee T_1 \vee \dots \vee T_n$$

$$\neg P_1 \vee \dots \vee \neg P_k \vee R_1 \vee \dots \vee R_m \vee B \vee T_1 \vee \dots \vee T_n$$

Воспользуемся коммутативностью дизъюнкции:

$$\neg P_1 \vee \dots \vee \neg P_k \vee R_1 \vee \dots \vee R_m \vee T_1 \vee \dots \vee T_n \vee A$$

$$\neg P_1 \vee \dots \vee \neg P_k \vee R_1 \vee \dots \vee R_m \vee T_1 \vee \dots \vee T_n \vee B$$

С учётом дистрибутивности дизъюнкции имеем

$$\neg P_1 \vee \dots \vee \neg P_k \vee R_1 \vee \dots \vee R_m \vee T_1 \vee \dots \vee T_n \vee A \& B,$$

что является формульным образом секвенции

$$P_1 \dots P_k \vdash R_1 \dots R_m A \& B T_1 \dots T_n$$

$\square$

**Теорема 5.** Секвенция выводима в секвенциальном исчислении тогда и только тогда, когда её формульный образ является тавтологией.

**Доказательство.**

- Необходимость:  $(\models S) \implies \Phi(S)$  является тавтологией

Доказательство по **индукции** по длине вывода секвенции  $S$ :

– **База.** Длина вывода = 1. Секвенция является аксиомой, и по теор. 3 её формульный образ – тавтология.

– **Переход.** Пусть формульный образ любой секвенции, длина вывода которой меньше  $n$  является тавтологией. Докажем, что формульный образ секвенции, длина вывода которой равна  $n$ , является тавтологией.

Рассмотрим последнее применение правила вывода, по которому была получена секвенция  $n$ . Пусть она была получена из  $S_1$  (или из  $S_1$  и  $S_2$ ). Длина вывода  $S_1$  (и  $S_2$ ) меньше  $n$ , т. к. они находятся раньше  $S$  в последовательности, определяющей вывод. По **индукционному предположению** формульный образ  $S_1$  (и  $S_2$ ) – тавтология. По теор. 4  $\Phi(S) \iff \Phi(S_1)$  (или  $\Phi(S) \iff \Phi(S_1) \& \Phi(S_2)$ ). Следовательно,  $\Phi(S)$  является тавтологией.

- Достаточность:  $\Phi(S)$  является тавтологией  $\implies (\models S)$

Доказательство по **индукции** по количеству логических связок в секвенции  $S$ :

– **База.** Секвенция не содержит логических связок. В этом случае  $S$  имеет вид  $p_1 \dots p_k \models q_1 \dots q_m$ , где  $p_1, \dots, p_k, q_1, \dots, q_m$  – пропозициональные переменные. Формульный образ этой секвенции равносителен  $\neg p_1 \vee \dots \vee \neg p_k \vee q_1 \vee \dots \vee q_m$  и является тавтологией. Если каждая переменная  $p_1, \dots, p_k$  отлична от любой переменной  $q_1, \dots, q_m$ , то на наборе значений (И, ..., И, Л, ..., Л) этот формульный образ имеет значение Л, что противоречит тому что он – тавтология. Значит, среди переменных  $p_1, \dots, p_k$  имеется переменная, входящая в  $q_1, \dots, q_m$ ,  $S$  является аксиомой и, следовательно, выводима.

– **Переход.** Пусть для всякой секвенции, содержащей  $n$  логических связок, и чей формульный образ является тавтологией, она выводима. Докажем, что всякая секвенция  $S$ , содержащая  $n+1$  логическую связку, и чей формульный образ является тавтологией, выводима.

Рассмотрим произвольную внешнюю связку  $*$  секвенции  $S$ . Секвенция  $S$  может быть получена по правилу  $(\vdash *)$  или  $(* \vdash)$  из одной  $S_1$  или двух  $S_1$  и  $S_2$  секвенций, содержащих  $n$  логических связок. При этом по теор. 4  $\Phi(S) \iff \Phi(S_1) \& \Phi(S_2)$  и, следовательно,  $\Phi(S_1) \& \Phi(S_2)$  – тавтология. Отсюда  $\Phi(S_1)$  – тавтология и  $\Phi(S_2)$  – тавтология. По **индукционному предположению** они выводимы.

□

**Определение 19.** Пропозициональная формула  $P$  называется *выводимой* в секвенциальном исчислении высказываний, если в нём выводима секвенция  $\vdash P$ .

**Следствие.** Секвенциальное исчисление высказываний полно и непротиворечиво.

**Доказательство.**

- **Полнота** секвенциального исчисления высказываний следует из достаточности:

Пусть  $P$  – тавтология. Так как  $P \iff \Phi(\vdash P)$ , то  $\Phi(\vdash P)$  тоже тавтология и, следовательно,  $\vdash P$  выводима.

- **Непротиворечивость** секвенциального исчисления высказываний следует из необходимости:

Пусть в секвенциальном исчислении высказываний выводимы секвенции  $\vdash P$  и  $\vdash \neg P$ . Тогда их формульные образы  $P$  и  $\neg P$  являются тавтологиями, что невозможно.

□

Кроме того, по аналогии с теоремой можно доказать следующее утверждение.

**Утверждение 1.** Формульный образ секвенции равносителен конъюнкции формульных образов секвенций, из которых она выводима.

**Определение 20.** Правило вывода называется *допустимым* в исчислении, если по всякому выводу, содержащему применение этого правила, можно построить вывод с той же конечной формулой, не содержащий применения этого правила.

**Утверждение 2.** Если в языке секвенциального исчисления высказываний имеется только две логические связки  $\neg$  и  $\&$  и основными правилами вывода в нём являются  $(\vdash \neg)$ ,  $(\neg, \vdash)$ ,  $(\vdash \&)$ ,  $(\&, \vdash)$ , то остальные правила вывода являются допустимыми в нём.

**Доказательство.** Покажем допустимость остальных правил вывода на примере правила  $(\rightarrow \vdash)$ .

Так как в языке нет логической связки  $\rightarrow$ , формула  $A \rightarrow B$  будет записана в виде  $\neg(A \& \neg B)$ . Поэтому правило  $(\rightarrow \vdash)$  примет вид

$$\frac{\Gamma_1 \Gamma_2 \vdash \Delta_1 A \Delta_2 \quad \Gamma_1 B \Gamma_2 \vdash \Delta_1 \Delta_2}{\Gamma_1 \neg(A \& \neg B) \Gamma_2 \vdash \Delta_1 \Delta_2}$$

Пусть имеется вывод с применением правила  $(\rightarrow \vdash)$ . Количество применений этого правила конечно, т. к. сам вывод конечен. Для каждого применения правила  $(\rightarrow \vdash)$  обозначим 3 секвенции в правиле  $S_1, S_2$  и  $S_3$  соответственно То есть в выводе имеется конечная последовательность секвенций вида  $S_1, \dots, S_2, \dots, S_3$ .

Но  $S_3$  может быть получена по правилу  $(\neg \vdash)$  из

$$S' \text{ вида } \Gamma_1 \Gamma_2 \vdash \Delta_1 (A \& \neg B) \Delta_2$$

$S'$  может быть получена по правилу  $(\vdash \&)$  из  $S_1$  и

$$S'' \text{ вида } \Gamma_1 \Gamma_2 \vdash \Delta_1 \neg B \Delta_2$$

$S''$  может быть получена по правилу  $(\vdash \neg)$  из  $S_2$ .

Таким образом для каждого применения правила  $(\rightarrow \vdash)$  последовательность секвенций  $S_1, \dots, S_2, \dots, S_3$  будет заменена на  $S_1, \dots, S_2, S'', \dots, S', S_3$ .  $\square$

**Утверждение 3.** Следующие правила являются допустимыми в секвенциальном исчислении высказываний:

- Правила перестановки.

$$\frac{\Gamma_1 P \Gamma_2 Q \Gamma_3 \vdash \Delta}{\Gamma_1 Q \Gamma_2 P \Gamma_3 \vdash \Delta} \quad \frac{\Gamma \vdash \Delta_1 P \Delta_2 Q \Delta_3}{\Gamma \vdash \Delta_1 Q \Delta_2 P \Delta_3}$$

- Правила добавления.

$$\frac{\Gamma_1 \Gamma_2 \vdash \Delta}{\Gamma_1 P \Gamma_2 \vdash \Delta} \quad \frac{\Gamma \vdash \Delta_1 \Delta_2}{\Gamma \vdash \Delta_1 P \Delta_2}$$

- Правила сокращения повторений.

$$\frac{\Gamma_1 P \Gamma_2 P \Gamma_3 \vdash \Delta}{\Gamma_1 P \Gamma_2 \Gamma_3 \vdash \Delta} \quad \frac{\Gamma \vdash \Delta_1 P \Delta_2 P \Delta_3}{\Gamma \vdash \Delta_1 P \Delta_2 \Delta_3}$$

- Правила исключения логических связок, например, для  $\&$ :

$$(\vdash \&_1^-) \frac{\Gamma \vdash \Delta_1 (A \& B) \Delta_2}{\Gamma \vdash \Delta_1 A \Delta_2} \quad (\vdash \&_2^-) \frac{\Gamma \vdash \Delta_1 (A \& B) \Delta_2}{\Gamma \vdash \Delta_1 B \Delta_2} \quad (\& \vdash) \frac{\Gamma_1 (A \& B) \Gamma_2 \vdash \Delta}{\Gamma_1 A B \Gamma_2 \vdash \Delta}$$

## 1.2.2. Метод резолюций для исчисления высказываний

Предложением в методе резолюций для исчисления высказываний называется элементарная дизъюнкция.

Формальное описание метода резолюций:

- Алфавит: множество символов для записи имён пропозициональных переменных, объединённое с множеством из двух логических связок  $\vee$  и  $\neg$ .
- Формулы: предложения.
- Аксиомы: отсутствуют.
- Правила вывода:

- правило сокращения повторений:

$$\frac{D_1 \vee x \vee D_2 \vee x \vee D_3}{D_1 \vee x \vee D_2 \vee D_3}$$

- правило резолюции

$$\frac{D_1 \vee x \vee D_2 \quad D_3 \vee \neg x \vee D_4}{D_1 \vee D_2 \vee D_3 \vee D_4}$$

**Определение 21.** Множество предложений называется *неудовлетворимым*, если из него выводимо пустое предложение *null*.

Следующая теорема может быть доказана **индукцией** по количеству применения правила резолюции (необходимость) и по количеству предложений (достаточность).

**Теорема 6.** Для того, чтобы множество предложений было неудовлетворимо необходимо и достаточно, чтобы их конъюнкция являлась противоречием.

### 1.3. Исчисления предикатов

**Определение 22.** Имя предмета называется *предметной константой*.

**Определение 23.** *Предметной переменной* называется переменная, которая в качестве своих значений может принимать предметные константы.

**Определение 24.**

1. Предметная константа является *термом*.
2. Предметная переменная является *термом*.
3. Если  $t_1, \dots, t_n$  — термы,  $f$  —  $n$ -местный функциональный символ, то выражение  $f(t_1, \dots, t_n)$  является *термом*.
4. Никакие другие выражения не являются *термами*.

**Определение 25.**

1. Если  $t_1, \dots, t_n$  — термы,  $P$  —  $n$ -местный предикатный символ, то  $P(t_1, \dots, t_n)$  является *атомарной формулой*.
2. Никакие другие выражения не являются *атомарными формулами*.

**Определение 26.**

1. Атомарная формула является *предикатной формулой*.
2. Если  $A$  — атомарная формула, то  $\neg A$  является *предикатной формулой*.
3. Если  $A, B$  — предикатные формулы,  $*$  — бинарная логическая связка, то  $(A * B)$  является *предикатной формулой*.
4. Если  $A$  — предикатная формула,  $x$  — предметная переменная, то  $\forall x A$  и  $\exists x A$  являются *предикатными формулами*.
5. Никакие другие выражения не являются *предикатными формулами*.

**Определение 27.** Кванторным комплексом называется выражения вида  $\forall x$  или  $\exists x$ , где  $x$  — имя предметной переменной.

**Определение 28.** Областью действия квантора называется формула, стоящая непосредственно за кванторным комплексом, содержащем вхождение этого квантора.

**Определение 29.** Вхождение предметной переменной в формулу называется *связанным*, если оно находится в кванторном комплексе или в области действия квантора по этой переменной.

Вхождения переменных, не являющихся связанными, называется *свободным*.

**Определение 30.** Формула без свободных переменных называется *замкнутой*.

**Определение 31.** Для того, чтобы задать интерпретацию формулы достаточно

- задать область интерпретации  $D$  — множество констант;
- каждому  $n$ -местному функциональному символу  $f$  поставить в соответствие конкретную функцию из  $D^n$  в  $D$ ;
- каждому  $n$ -местному предикату  $P$  поставить в соответствие конкретное отношение над  $D^n$ .

**Определение 32.** Формула называется *истинной* (*ложной*) в заданной интерпретации, если она истинна (ложна) на всех наборах значений из области интерпретации, подставляемых вместо свободных вхождений предметных переменных этой формулы.

**Определение 33.** Формула называется *выполнимой* в заданной интерпретации, если она истинна хоть на одном наборе значений из области интерпретации, подставляемых вместо свободных вхождений предметных переменных этой формулы.

**Определение 34.** Формула называется *общезначимой* (*противоречием*), если она истинна (ложна) в любой интерпретации.

**Определение 35.** Формула называется *выполнимой*, если она выполнима хоть в одной интерпретации.

**Определение 36.** Формула  $B$  логически следует из формул  $A_1, \dots, A_n$ , если в любой интерпретации на любом наборе значений свободных переменных, для которых все формулы  $A_1, \dots, A_n$  истинны, формула  $B$  тоже истинна.

**Обозначение.**  $A_1, \dots, A_n \Rightarrow B$

**Определение 37.** Формулы  $A$  и  $B$  называются *равносильными*, если в любой интерпретации на любом наборе значений свободных переменных их значения совпадают.

**Обозначение.**  $A \Leftrightarrow B$

Все равносильности, справедливые для пропозициональных формул, справедливы и для предикатных формул. Кроме того, имеют место равносильности для работы с кванторами. Пусть  $x$  не входит свободно в  $B$ , а  $y$  не входит в  $C$ .

$$\begin{array}{rcl} \neg \forall x A & \iff & \exists x \neg A \\ \neg \exists x A & \iff & \forall x \neg A \\ \forall x B & \iff & B \\ \exists x B & \iff & B \\ \forall x C & \iff & \forall y [C]_y^x \\ \exists x C & \iff & \exists y [C]_y^x \\ \forall x A \& B & \iff \forall x (A \& B) \\ \exists x A \& B & \iff \exists x (A \& B) \\ \forall x A \vee B & \iff & \forall x (A \vee B) \\ \exists x A \vee B & \iff & \exists x (A \vee B) \\ B \rightarrow \forall x A & \iff & \forall x (B \rightarrow A) \\ B \rightarrow \exists x A & \iff & \exists x (B \rightarrow A) \\ \forall x A \rightarrow B & \iff & \exists x (A \rightarrow B) \\ \exists x A \rightarrow B & \iff & \forall x (A \rightarrow B) \end{array}$$

**Определение 38.** Терм  $t$  называется *свободным для подстановки* в формулу  $A$  вместо свободных вхождений предметной переменной  $x$ , если он не содержит предметных переменных, в области действия кванторов по которым имеются свободные вхождения предметной переменной  $x$ .

### Следствия из определения.

1. Любой постоянный терм свободен для подстановки в любую формулу вместо свободных вхождений любой переменной.
2. Переменная  $x$  свободна для подстановки в любую формулу вместо своих свободных вхождений.
3. Если предметная переменная  $x$  не имеет свободных вхождений формулы  $A$ , то любой терм свободен для подстановки в формулу  $A$  вместо свободных вхождений переменной  $x$ .
4. Если терм не содержит предметных переменных, входящих в формулу  $A$ , то он свободен для подстановки в формулу  $A$  вместо свободных вхождений любой переменной.

**Определение 39.** Формула находится в *предварённой нормальной форме*, если она представляет собой последовательность кванторных комплексов, все переменные которых различны, и формулы, не содержащих кванторов.

**Теорема 7.** По всякой предикатной формуле  $P$  можно построить такую формулу  $Q$  в предварённой нормальной форме такую, что  $P \iff Q$ .

## Секвенциальное исчисление предикатов

Формальное описание секвенциального исчисления предикатов:

1. Алфавит: множество символов для записи имён предметных переменных и констант, функциональных и предикатных символов, объединённое с множеством логических связок, кванторов, запятой, символа секвенции и скобок.
2. Формулы исчисления: секвенции, содержащие предикатные формулы в выбранном алфавите.
3. Аксиомы исчисления: единственная схема аксиом:

$$\Gamma_1 A \Gamma_2 \vdash \Delta_1 A \Delta_2$$

4. Правила вывода для логических связок такие же, как в секвенциальном исчислении высказываний, но добавлены ещё четыре правила для кванторов и правила сокращения повторений.

$$(\vdash \exists) \frac{\Gamma \vdash \Delta_1 [A]_T^x \Delta_2}{\Gamma \vdash \Delta_1 \exists x A \Delta_2} \quad (\forall \vdash) \frac{\Gamma_1 [A]_T^x \Gamma_2 \vdash \Delta}{\Gamma_1 \forall x A \Gamma_2 \vdash \Delta},$$

где терм  $T$  свободен для подстановки в формулу  $A$  вместо свободных вхождений предметной переменной  $x$ .

$$(\exists \vdash) \frac{\Gamma_1 [A]_y^x \Gamma_2 \vdash \Delta}{\Gamma_1 \exists x A \Gamma_2 \vdash \Delta} \quad (\vdash \forall) \frac{\Gamma \vdash \Delta_1 [A]_y^x \Delta_2}{\Gamma \vdash \Delta_1 \forall x A \Delta_2},$$

где переменная  $y$  не входит свободно в заключение правила и свободна для подстановки в формулу  $A$  вместо свободных вхождений предметной переменной  $x$ .

$$\frac{\Gamma_1 P \Gamma_2 P \Gamma_3 \vdash \Delta}{\Gamma_1 P \Gamma_2 \Gamma_3 \vdash \Delta} \quad \frac{\Gamma \vdash \Delta_1 P \Delta_2 P \Delta_3}{\Gamma \vdash \Delta_1 P \Delta_2 \Delta_3}$$

**Определение 40.** Секвенция называется *чистой*, если ни одна переменная не имеет одновременно свободных и связанных вхождений.

**Теорема 8.** Чистая секвенция выводима в секвенциальном исчислении тогда и только тогда, когда её формульный образ общезначим.

**Теорема 9.** Секвенциальное исчисление полно и непротиворечиво.

## Метод резолюций для исчисления предикатов

**Определение 41.** Общим унифициатором формул  $Q$  и  $R$  называется такая подстановка  $\lambda = |_{t_1 \dots t_n}^{x_1 \dots x_n}$  термов  $t_1 \dots t_n$  вместо свободных вхождений предметных переменных  $x_1 \dots x_n$ , что результаты этой подстановки в формулы  $Q$  и  $R$  графически совпадают.

Формальное описание метода резолюций для исчисления предикатов:

1. Алфавит: множество символов для записи имён предметных переменных и констант, имён предикатных переменных и функциональных символов, объединённое с множеством из двух логических связок  $\vee$  и  $\neg$ , скобок, а также запятой.
2. Формулы: предложения.
3. Аксиомы: отсутствуют.
4. Правила вывода:

- правило резолюции:

$$\frac{D_1 \vee R_1 \vee D_2 \quad D_3 \vee \neg R_2 \vee D_4}{[D_1 \vee D_2 \vee D_3 \vee D_4]\lambda},$$

где  $\lambda$  — общий унифициатор формул  $R_1$  и  $R_2$ .

- правило сокращения повторений:

$$\frac{D_1 \vee R \vee D_2 \vee R \vee D_3}{D_1 \vee R \vee D_2 \vee D_3}$$

**Определение 42.** Множество предложений называется *неудовлетворимым*, если из него выводимо пустое предложение *nill*.

**Теорема 10.** Для того, чтобы множество предложений было неудовлетворимо необходимо и достаточно, чтобы их конъюнкция являлась противоречием.

**Определение 43.** Сколемовской константой (функцией) называется константа (функция), существование которой утверждается в формуле, но её значение (определение) может быть неизвестно.

## Глава 2

# Формальные теории

Для того, чтобы задать формальную теорию необходимо задать

- множество констант  $D$ ;
- множество функциональных символов  $F$ ;
- множество предикатных символов  $P$ ;
- множество аксиом, определяющих “смысл” констант, функциональных и предикатных символов.

При этом говорят, что задана *формальная теория в сигнатуре  $\langle D, F, P \rangle$* .

Кроме того, для определения ФТ требуется выбрать исчисление, в котором будут доказываться формулы ФТ. Следовательно, в определение ФТ входят аксиомы соответствующего исчисления.

### Формальные теории с равенством

**Определение 44.** ФТ называется *ФТ с равенством*, если множество предикатных символов содержит выделенных двухместный предикат  $=$  и для любой формулы  $A(x)$  выводимы следующие формулы

- $\forall x(x = x)$
- $\forall x \ y \left( x = y \rightarrow (A(x) \leftrightarrow A(y)) \right)$

Это равносильно тому, что ФТ с равенством содержит следующие **аксиомы для равенства**:

**ER** (рефлексивность равенства).  $\forall x(x = x)$ ;

**ES** (симметричность равенства).  $\forall x \ y(x = y \rightarrow y = x)$ ;

**ET** (транзитивность равенства).  $\forall x \ y \ z(x = y \ \& \ y = z \rightarrow x = z)$ .

и **аксиомы согласования с равенством** для любого функционального символа из сигнатуры:

$$\forall x \ y \left( x = y \rightarrow (f(\bar{u}, x, \bar{v}) = f(\bar{u}, y, \bar{v})) \right)$$

## Теории множеств

### Парадокс Рассела в наивной теории множеств

Рассмотрим множество

$$A = \{ x : \neg(x \in x) \}$$

Доказано, что ни  $A \in A$ , ни  $A \notin A$ . При этом,

$$\forall x(x \in A \leftrightarrow \neg(x \in x))$$

В частности, при  $x = A$ , получаем

$$A \in A \leftrightarrow \neg(A \in A)$$

Эта формула заведомо ложна. Следовательно, в наивной теории множеств доказуема ложная формула. Таким образом, наивная теория множеств противоречива.

## Теория типов Рассела

Рассматриваются множества различных типов:

1. Индивиды, т. е. множества, не имеющие элементов, обозначаются  $x_1, y_1, \dots$
2. Классы индивидов, т. е. множества, элементами которых являются индивиды, обозначаются  $x_2, y_2, \dots$

...

$i + 1$ . Классы объектов типа  $i$ , обозначаются  $x_{i+1}, y_{i+1}, \dots$

...

При таких ограничениях парадокс Рассела невозможен, т. к. нельзя написать  $x \in x$ , а только  $x_i \in x_{i+1}$ .

**Определение 45.** Упорядоченная пара двух множеств  $i$ -го типа — это множество  $i+2$ -го типа:  $(x_i, y_i) = \{\{x_i\}, \{x_i, y_i\}\}$ .

**Определение 46.** Бинарное отношение — это множество упорядоченных пар, т. е. бинарное отношение между множествами  $i$ -го типа — это множество  $i+3$ -го типа.

**Формальное определение теории типов.** Теория типов — это формальная теория в сигнатуре  $\langle ; ; \in, = \rangle$ . Так как в сигнатуре имеется предикат равенства, в **Т** присутствуют аксиомы для равенства и аксиомы согласования с равенством для предиката  $\in$ .

Собственные аксиомы теории типов:

1. Аксиома объёмности.

Два множества равны, если они равны поэлементно.

$$\forall y_{i+1} z_{i+1} (\forall x_i (x_i \in y_{i+1} \leftrightarrow x_i \in z_{i+1}) \rightarrow y_{i+1} = z_{i+1})$$

2. Схема аксиом выделения.

Для всякой формулы в терминах **Т** со свободное переменной  $i$ -го типа существует множество  $i+1$ -го типа, все элементы которого и только они удовлетворяют этой формуле.

Если  $F(x_i)$  — формула в терминах **Т** со свободной переменной  $x_i$ , то

$$\exists y_{i+1} \forall x_i (x_i \in y_{i+1} \leftrightarrow F(x_i))$$

3. Аксиома бесконечности.

В **Т** имеется бесконечное множество индивидов.

$$\exists w_4 (\forall x_1 \neg w_4(x_1, x_1) \quad \& \quad \forall x_1 \exists y_1 w_4(x_1, y_1) \quad \& \\ \forall x_1 \forall y_1 \forall z_1 (w_4(x_1, y_1) \& w_4(y_1, z_1) \rightarrow w_4(x_1, z_1)))$$

## Аксиоматическая теория множеств Цермело—Френкеля

**Формальное определение аксиоматической теории множеств** Аксиоматическая теория множеств Цермело—Френкеля **ZF** — это формальная теория в сигнатуре  $\langle ; ; \in, = \rangle$ . В **ZF** присутствуют аксиомы для равенства и аксиомы согласования с равенством для  $\in$ .

Собственные аксиомы **ZF**:

1. Аксиома объёмности.

Два множества равны, если они равны поэлементно.

$$\forall xy (\forall u (u \in x \leftrightarrow u \in y) \rightarrow x = y)$$

2. Аксиома пары.

Двухэлементное множество является множеством.

$$\forall xy \exists w \forall z (z \in w \leftrightarrow (z = x \wedge z = y))$$

$$w = \{ x, y \}$$

### 3. Аксиома суммы (объединения).

Для всякого множества множеств объединение всех входящих в него множеств является множеством.

$$\forall x \exists w \forall y (y \in w \leftrightarrow \exists z (z \in x \ \& \ y \in z))$$

$$w = \bigcup_{z \in x}$$

### 4. Аксиома булеана.

$$\forall x \exists w \forall y (y \in w \leftrightarrow \forall z (z \in y \rightarrow z \in x))$$

$$w = P(x) = 2^x$$

### 5. Схема аксиом выделения.

Для всякой формулы  $\varphi$  с одной свободной переменной  $v$

$$\forall x \exists w \forall y (y \in w \leftrightarrow y \in x \ \& \ [\varphi]_y^v)$$

$$w = \{ v \in x : \varphi(v) \}$$

### 6. Аксиома бесконечности.

Существуют бесконечные множества.

$$\exists y (\emptyset \in y \ \& \ \forall x (x \in y \rightarrow \{ x \} \in y))$$

### 7. Аксиома ограничения (фундирования).

У непустого множества имеется элемент, не пересекающийся с ним.

$$\forall y (\neg(y = \emptyset) \rightarrow \exists x (x \in y \ \& \ x \cap y = \emptyset))$$

### 8. Аксиома подстановки.

Образ множества является множеством (даже если функция  $F$  не является множеством).

$$\forall x \exists y \forall z (\exists u (u \in x \ \& \ (u, z) \in F) \leftrightarrow z \in y)$$

### 9. Аксиома выбора.

Для любого множества  $S$  непустых множеств существует функция  $f$ , определённая на всех элементах множества  $S$  такая, что

$$\forall S \exists f \forall x (x \in S \rightarrow f(x) \in x)$$

**Определение 47.** Ординалом называется множество, вполне упорядоченное отношением включения  $\subset$ , каждый элемент которого является его подмножеством.

## Конечные ординалы

- Пустое множество  $\emptyset$  является нулевым ординалом.
- Первый ординал  $\{ \emptyset \}$ .
- Второй ординал  $\{ \emptyset, \{ \emptyset \} \}$ .
- ...
- Пусть  $\{ X \}$  —  $n$ -й ординал. Тогда  $n + 1$ -й ординал  $\{ X, \{ X \} \}$ .

## Бесконечные ординалы

По аксиоме бесконечности

$$w = \left\{ \underbrace{\emptyset}_0, \underbrace{\{ \emptyset \}}_1, \underbrace{\{ \emptyset, \{ \emptyset \} \}}_2, \dots, \underbrace{\{ X \}}_n, \underbrace{\{ X, \{ X \} \}}_{n+1}, \dots \right\}$$

является множеством. Это *первый бесконечный ординал*.

Далее последовательность бесконечных ординалов строится следующим образом.

$$\begin{aligned} 2w &= \left\{ \underbrace{w}_w, \underbrace{w \cup \{ w \}}_{w+1}, \underbrace{w \cup \{ w, \{ w \} \}}_{w+2}, \dots, \underbrace{w \cup \{ X \}}_{w+n}, \underbrace{w \cup \{ X, \{ X \} \}}_{w+n+1}, \dots \right\} \\ 3w &= \left\{ \underbrace{2w}_{2w}, \underbrace{2w \cup \{ w \}}_{2w+1}, \underbrace{2w \cup \{ w, \{ w \} \}}_{2w+2}, \dots, \underbrace{2w \cup \{ X \}}_{2w+n}, \underbrace{2w \cup \{ X, \{ X \} \}}_{2w+n+1}, \dots \right\} \\ &\quad \dots \\ (k+1)w &= \left\{ \underbrace{kw}_{kw}, \underbrace{kw \cup \{ w \}}_{kw+1}, \underbrace{kw \cup \{ w, \{ w \} \}}_{kw+2}, \dots, \underbrace{kw \cup \{ X \}}_{kw+n}, \underbrace{kw \cup \{ X, \{ X \} \}}_{kw+n+1}, \dots \right\} \\ &\quad \dots \\ w^2 &= \bigcup_{k=1}^w kw \end{aligned}$$

Аналогичным образом строятся  $w^2 + 1, w^2 + 2, \dots, w^3, \dots, w^w = \varepsilon_0$ .

Ординал  $\varepsilon_0$  — первый несчётный ординал. К нему таким же образом можно прибавлять по единице.

**Примечание.** Вообще, во всех источниках

$$\varepsilon_0 = w^{w^{\cdot^{\cdot^{\cdot}}}}$$

## Формальная арифметика

Формальная арифметика — это ФТ в сигнатуре  $\langle 0; S, +, \cdot, \wedge, = \rangle$ .

Так как в сигнатуре присутствует знак  $=$ , то для каждого функционального символа требуются аксиомы согласования с равенством. Например, для  $S$  она имеет вид

$$\forall x y (x = y \rightarrow (S(x) = S(y)))$$

Для двухместного функционального символа можно написать две аксиомы

$$\forall x y z (x = y \rightarrow (x + z = y + z)) \quad \forall x y z (x = y \rightarrow (z + x = z + y))$$

или одну аксиому

$$\forall x y u v (x = y \& u = v \rightarrow (x + u = y + v))$$

### Собственные аксиомы FA.

1.  $\forall x \neg(S(x) = 0)$
2.  $\forall xy (S(x) = S(y) \rightarrow x = y)$
3.  $\forall x (x + 0 = x) \quad \forall xy (x + S(y) = S(x + y))$
4.  $\forall x (x \cdot 0 = 0) \quad \forall xy (x \cdot S(y) = x \cdot y + x)$
5.  $\forall x (x^0 = S(0)) \quad \forall xy (x^{S(y)} = x^y \cdot x)$
6. *Аксиома индукции.* Для любой формулы  $A(x)$

$$A(0) \quad \& \quad \forall x (A(x) \rightarrow A(S(x))) \quad \rightarrow \quad \forall x A(x)$$

### Полнота и непротиворечивость

**Определение 48.** Номером буквы  $a_i$  называется цифра  $i$   $(p+1)$ -чной системы счисления.

**Обозначение.**  $\#a_i = i$

**Определение 49.** Номером слова  $a_{i_1} \dots a_{i_n}$  называется число  $(\overline{i_1 \dots i_n})_{p+1}$ .

Поскольку вывод — это последовательность формул, то вывод — это тоже слово в том же алфавите и имеет номер.

Гёдель ввёл в рассмотрение предикат  $\models (X, \#\varphi)$  — “число  $X$  является номером вывода формулы  $\varphi$ ”, для которого явно выписал формулу на языке FA и доказал, что выводимость такой формулы равносильна её истинности.

Кроме того, Гёдель рассмотрел формулу

$$G(\#\varphi) \iff \forall X (\models (X, \#\varphi) \rightarrow \exists Y (Y < X \ \& \ \models (Y, \#\neg\varphi)))$$

Для этой формулы справедливо следующее утверждение.

**Утверждение 4.** Если формальная арифметика непротиворечива, то  $\varphi$  выводима тогда и только тогда, когда  $G(\#\varphi)$  ложна.

Обозначим номер  $G(\#\varphi)$  как  $\#G$ .

## Теоремы Гёделя о неполноте

**Теорема 11 (первая Гёделя).**

1. Если FA непротиворечива, то она не полна.
2. Если FA непротиворечива, то в ней существует замкнутая формула, для которой не выводима ни она сама, ни её отрицание.
3. Если FA непротиворечива, то в ней не выводимы ни формула  $G(\#G)$ , ни её отрицание  $\neg G(\#G)$ .

**Лемма 1.** FA противоречива тогда и только тогда, когда в ней выводима формула  $S(0) = 0$ .

**Доказательство.**

•  $\implies$

Это можно доказать, например, в секвенциальном исчислении предикатов с использованием правил добавления и сечения.

•  $\impliedby$

В FA имеется аксиома  $\forall x \neg(S(x) = 0)$  и, следовательно,  $\neg(S(0) = 0)$  выводима.

□

**Теорема 12 (вторая Гёделя).**

1. Если средствами FA можно доказать, что она непротиворечива, то она противоречива.
2. Если в FA выводима формула

$$\forall x \neg \models (x, \#(S(0) = 0)),$$

то в ней выводима формула  $S(0) = 0$ .

## Консервативное расширение FA

Построим нестандартное расширение \*FA, в котором присутствуют бесконечно большие натуральные числа. Для этого заменим аксиому 6 на \*6 и введём схему аксиом \*7, декларирующую, что  $w$  больше любого натурального числа.

\*6. Для всякой формулы  $A(x)$  с одной свободной переменной  $x$

$$A(0) \ \& \ \forall x (\forall y (y < x \ \& \ A(y)) \rightarrow A(x)) \rightarrow \forall x A(x)$$

\*7.

$$w > 0$$

$$w > S(0)$$

...

$$w > \underbrace{S(\dots S(}_{n} 0) \dots )$$

...

Для нестандартного расширения  ${}^*FA$  можно доказать аналог принципа переноса Лейбница, который означает консервативность такого расширения.

**Теорема 13.** Всякая замкнутая формула аксиоматической теории чисел одновременно выводима в FA и  ${}^*FA$ .

**Доказательство.**

- Пусть замкнутая формула выводима в FA. Это означает, что в построенном выводе нет константы  $w$ , не используется аксиома \*7, и может использоваться аксиома 6, которая следует из \*6. Следовательно, имеется вывод и в  ${}^*FA$ .
- Пусть замкнутая формула выводима в  ${}^*FA$ . В выводе может присутствовать константа  $w$ . Найдём самое большое натуральное число  $n$ , присутствующее в выводе. Все вхождения  $w$  в выводе заменим на константу  $n + 1$ .

Если в выводе присутствует аксиома \*7 вида  $k < w$ , она заменится на выводимую формулу  $k < n + 1$ . Использование трансфинитной индукции \*6 заменим на использование аксиомы индукции 6.

□

# Глава 3

## Теория алгоритмов

Один из способов задания конструктивных объектов — формулы Бэкуса.

**Определение 50.** Формулы Бэкуса имеют вид

$\langle \text{понятие} \rangle ::= \langle \text{понятие}_1 \rangle | \langle \text{понятие}_2 \rangle | \langle \text{понятие}_i \rangle \langle \text{понятие}_j \rangle | \dots$

### 3.1. Рекурсивные функции

**Определение 51.** Простейшими называются функции натурального аргумента, определяемые равенствами

$$S(x) = x + 1, \quad O(x) = 0, \quad I_n^m(x_1, \dots, x_n) = x_m$$

**Определение 52.** Функция  $f$  от  $n+1$  переменных получена их функции  $g$  от  $n$  переменных и функции  $h$  от  $n+2$  переменных с помощью *оператора примитивной рекурсии*, если

$$\begin{cases} f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n) \\ f(x_1, \dots, x_n, y+1) = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y)) \end{cases}$$

**Определение 53.** Оператор неограниченной минимизации  $\mu$  определяется как

$$g(x_1, \dots, x_n) = \mu y \{ f(x_1, \dots, x_n, y) = 0 \}$$

(наименьшее  $y$ , для которого  $f(x_1, \dots, x_n, y) = 0$ )

**Определение 54.** Оператор ограниченной минимизации  $\mu_{\leq z}$  определяется как

$$g(x_1, \dots, x_n, z) = \mu y_{\leq z} \{ f(x_1, \dots, x_n, y) = 0 \}$$

(наименьшее  $y$ , не превосходящее  $z$  и для которого  $f(x_1, \dots, x_n, y) = 0$ , иначе 0)

**Определение 55.** Функция называется *частично рекурсивной*, если она может быть получена из простейших с помощью применения операторов подстановки, примитивной рекурсии и ограниченной минимизации.

**Определение 56.** Обобщённый оператор минимизации определяется как

$$\mu^* y \{ f(\bar{x}, y) = 0 \} = \begin{cases} \mu y \{ f(\bar{x}, y) = 0 \}, & \text{если такой } y \text{ существует,} \\ 0 & \text{иначе} \end{cases}$$

**Определение 57.** Функция называется *общерекурсивной*, если она может быть получена из простейших с помощью применения операторов подстановки, примитивной рекурсии и обобщённой минимизации.

**Тезис Чёрча** Всякая интуитивно вычислимая функция является общерекурсивной.

## 3.2. Машины Тьюринга

**Определение 58.** Машина Тьюринга задаётся тройкой  $\langle A, Q, P \rangle$ , где

- $A = \{a_1, \dots, a_n\}$  — внешний алфавит (содержащий пустой символ \*);
- $Q = \{q_0, q_1, \dots, q_k\}$  — внутренний алфавит, в котором выделены начальное и заключительное состояния  $q_1$  и  $q_0$ ;
- $P$  — программа.

**Определение 59.** Команда машины Тьюринга имеет вид

$$q_r a_i \rightarrow q_t S a_j,$$

где  $i, j = 1, \dots, n$ ,  $r = 1, \dots, k$ ,  $t = 0, 1, \dots, k$ ,  $S \in L, R, \underline{\phantom{x}}$ .

**Определение 60.** Команды называются *согласованными*, если они имеют различные левые части или полностью совпадают.

**Определение 61.** Программой машины Тьюринга называется конечное непустое множество согласованных команд.

**Определение 62.** Конфигурацией машины Тьюринга называется слово вида

$$b_1 \dots b_{p-1} q_r b_p \dots b_l,$$

где

- $b_1, \dots, b_l$  — слово в алфавите  $A$ , записанное на ленте;
- слева и справа от этого слова на ленте находятся только пустые символы;
- машина находится в состоянии  $q_r$  и обозревает  $p$ -й символ этого слова;
- на концах конфигурации находится не более чем по одному пустому символу.

**Определение 63.** Протоколом работы машины Тьюринга называется последовательность конфигураций, первая из которых является начальной, а каждая следующая получена из предыдущей одной из команд.

Машина Тьюринга заканчивает работу над данными  $X$ , если она пришла в состояние  $q_0$ , или ни одна из команд не может быть применена к текущей конфигурации.

**Тезис Тьюринга—Чёрча.** Всякая интуитивно вычислимая функция может быть вычислена на машине Тьюринга.

**Лемма 2.** По всякой машине Тьюринга  $M$ , которая по данным  $X$  в алфавите  $A$  вычисляет значение функции  $f(X)$ , можно построить машину Тьюринга  $M_1$ , которая по данным  $X$  вычисляет значение функции  $f(X)$  и заканчивает работу в конфигурации  $q_0 f(X)$ .

**Доказательство.**

1. Пометим начало слова символом, не входящим в алфавит  $A$  (например, символом #) и вернёмся в начало исходных данных.

$$q_1 a_i \rightarrow q_1 L a_i$$

$$q_1 * \rightarrow q_2 R \#$$

2. В программе машины  $M$  каждое вхождение состояния  $q_i$  заменяем на  $q_{i+1}$ .

3. Возможны два случая завершения работы программы:

- Машина Тьюринга завершила работу в конфигурации  $\#Y'q_0Y''$ , где  $Y'Y''$  совпадает с  $f(X)$ .

Заменяем в тексте программы состояние  $q_0$  на  $q_{k+2}$  и добавляем команды

$$q_{k+2}a_i \rightarrow q_{k+2}La_i, \quad a_i \in A$$

$$q_{k+2}\# \rightarrow q_0R*$$

- Машина Тьюринга завершила работу в конфигурации  $\#Y'q_ta_iY'''$ , где  $Y'a_iY'''$  совпадает с  $f(X)$  и в программе отсутствует команда с левой частью  $q_ta_i$ .

Для каждого состояния  $q_t$  и каждого символа  $a_i$  из алфавита, для которых в программе отсутствует команда с левой частью  $q_ta_i$ , добавляем команду

$$q_ta_i \rightarrow q_{k+2}La_i$$

и команды, указанные в предыдущем пункте.

□

**Теорема 14.** Пусть МТ  $M_1$  по данным  $X$  в алфавите  $A_1$  вычисляет значение функции  $g(X)$  в алфавите  $A_2$ , МТ  $M_2$  по данным  $Y$  в алфавите  $A_2$  вычисляет значение функции  $f(Y)$ .

Тогда существует МТ  $M_3$ , которая по данным  $X$  вычисляет значение функции  $f(g(X))$ .

**Доказательство.** Пусть  $M_i$  ( $i = 1, 2$ ) имеет программу  $P_i$  и использует состояния  $\{q_0, q_1, \dots, q_{k_i}\}$ .

В соответствии с леммой можно считать, что  $M_1$  заканчивает работу в конфигурации  $q_0g(X)$ .

В программе  $P_1$  состояние  $q_0$  заменяется на  $q_{k_1+1}$ , получаем программу  $P'_1$ .

В программе  $P_2$  состояния  $q_i$  ( $i = 1, \dots, k_2$ ) заменяются на  $q_{k_1+i}$ , получаем программу  $P'_2$ .

Машина с программой  $P'_1 \cup P'_2$  вычисляет  $f(g(X))$ .

□

**Определение 64.** Машина Тьюринга называется машиной Тьюринга с односторонне-ограниченной лентой, если она не использует ячейки, расположенные левее ячейки с первым символом исходных данных (левосторонне-ограниченная), или расположенные правее ячейки с последним символом исходных данных (правосторонне-ограниченная).

**Лемма 3.** По всякой МТ можно построить МТ с односторонне-ограниченной лентой, у которой на любых исходных данных результат её работы совпадает с результатом работы  $M$ .

**Теорема 15.** Пусть машина Тьюринга  $M_1$  по данным  $X_1, \dots, X_i, \dots, X_n$  в алфавите  $A_1$  вычисляет значение функции  $f(X_1, \dots, X_i, \dots, X_n)$  в алфавите  $A_2$ , машина Тьюринга  $M_2$  по данным  $Y_1, \dots, Y_m$  в алфавите  $A_2$  вычисляет значение функции  $g(Y_1, \dots, Y_m)$ .

Тогда существует машина Тьюринга  $M_3$ , которая по данным  $X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_n, Y_1, \dots, Y_m$  вычисляет значение функции  $f(X_1, \dots, X_{i-1}, g(Y_1, \dots, Y_m), X_{i+1}, \dots, X_n)$ .

**Доказательство.**

1. По машине  $M_2$  строим левосторонне-ограниченную машину  $M_3$  с программой  $P_3$ .
2. МТ с программой  $P_0$  с состояниями  $q_0, q_1, \dots, q_{k_0}$  отмечает место между  $X_{i-1}$  и  $X_{i+1}$  и подводит головку к первому символу данных  $Y_1, \dots, Y_m$ . Заменив  $q_0$  на  $q_{k_0+1}$  получаем программу  $P'_0$ .
3. В программе  $P_3$  заменяется  $q_i$  на  $q_{k_0+i}$  при  $i \neq 0$ , а  $q_0$  на  $q_{k_0+k_3+1}$  и получаем программу  $P'_3$ .
4. Машина с программой  $P_4$  с состояниями  $q_0, q_1, \dots, q_{k_4}$  вставляет результат работы программы  $P'_3$  между  $X_{i-1}$  и  $X_{i+1}$  и подводит головку к первому символу, записанному на ленте. Заменив  $q_i$  на  $q_{k_0+k_3+i}$  при  $i \neq 0$ , а  $q_0$  на  $q_{k_0+k_3+k_4+1}$  получаем программу  $P'_4$ .
5. В программе  $P_1$  для машины  $M_1$  заменяется  $q_i$  при  $i \neq 0$  на  $q_{k_0+k_3+k_4+i}$  и получаем программу  $P'_1$ .  
Машина с программой  $P'_0 \cup P'_3 \cup P'_4 \cup P'_1$  вычисляет  $f(X_1, \dots, X_{i-1}, g(Y_1, \dots, Y_m), X_{i+1}, \dots, X_n)$

□

### 3.2.1. Многоленточные МТ

**Определение 65.** Команда  $k$ -ленточной машины Тьюринга имеет вид

$$q_r \begin{pmatrix} a_{i_1} \\ \vdots \\ a_{i_k} \end{pmatrix} \rightarrow q_t \begin{pmatrix} S_1 \\ \vdots \\ S_k \end{pmatrix} \begin{pmatrix} a_{j_1} \\ \vdots \\ a_{j_k} \end{pmatrix}, \quad S_1, \dots, S_k \in \{L, R, \_\}$$

Обычно предполагается, что 1-я лента — это входная лента для записи исходных данных,  $k$ -я лента — это выходная лента для записи результата, остальные ленты — это рабочие ленты.

**Теорема 16.** По всякой  $k$ -ленточной машине Тьюринга  $M_k$ , заканчивающей работу с исходными данными  $X$  за  $t$  шагов, можно построить одноленточную машину Тьюринга  $M_1$ , результат работы которой с исходными данными  $X$  совпадает с результатом работы и число шагов которой составляет  $O(t^2)$ .

**Доказательство.** Будем считать, что первая лента входная, а последняя — выходная.

На  $i$ -м шаге ( $i = 1, \dots, t$ ) длина записи на  $j$ -ой ленте  $M_k$  ( $j = 2, \dots, k$ ) может увеличиться не более чем на единицу (стать равной  $i$ ), причём запись нового символа может происходить как в середине слова, так и на одном его концов. При этом содержимое лент имеет вид

$$\begin{pmatrix} X \\ X_i^2 \\ \vdots \\ X_i^k \end{pmatrix}$$

Конфигурация моделирующей машины  $M_1$  в этот момент имеет вид

$$X * q_l X_i^2 * \dots * X_i^k$$

при некотором  $l$ .

Для моделирования  $i$ -го шага  $M_k$  машина  $M_1$  должна для каждого ( $j = 2, \dots, k$ ):

- сдвинуть головку на символ, обозреваемый  $M_k$  на  $j$ -ой ленте (не более, чем  $\|X_i^{j-1}\| + \|X_i^j\|$  шагов);
- произвести действие, которое  $M_k$  производит со словом  $X_i^j$  (1 шаг);
- в случае необходимости переместить всё содержимое ленты правее положения головки на одну ячейку вправо (не более, чем  $4 \sum_{j'=j}^k \|X_i^{j'}\|$ );
- вернуться в исходное положение (не более, чем  $\sum_{j'=j}^k \|X_i^{j'}\|$  шагов);

Всего при моделировании действия  $M_k$  с одним символом на  $i$ -ом шаге  $M_1$  совершает не более

$$\begin{aligned} \sum_{j=2}^k (\|X_i^{j-1}\| + \|X_i^j\| + 1 + 4 \sum_{j'=j}^k \|X_i^{j'}\| + \sum_{j'=j}^k \|X_i^{j'}\|) &\leq \sum_{j=2}^k (i + i + 1 + 5 \sum_{j'=j}^k i) = \sum_{j=2}^k (5(k-j)i + 2i + 1) = \\ &= \frac{1}{2} \cdot 5i(k-1)(k-2) + 2i(k-1) + 2(k-1) = \frac{3}{i}(k-1)(3k-4) + 2(k-1) \end{aligned}$$

Просуммировав полученное выражение по  $i = 1, \dots, t$ , имеем

$$\frac{3}{2} \sum_{i=1}^t i((k-1)(3k-4) + 2(k-1)) = \frac{3}{2}(k-1)(3k-4) \frac{t(t-1)}{2} + 2(k-1) = O(k^2 t^2) = O(t^2)$$

□

### 3.2.2. Многоголовчатые МТ

**Определение 66.** Команда  $m$ -головчатой машины Тьюринга имеет вид

$$q_r(a_{i_1}, \dots, a_{i_m}) \rightarrow q_t(A_1, \dots, S_m)(a_{j_1}, \dots, a_{j_m}), \quad S_1, \dots, S_m \in \{L, R, \_\}$$

### 3.2.3. Недетерминированные МТ

**Лемма 4.** Если недетерминированная машина Тьюринга, проверяющая предикат  $\exists Y P(X, Y)$  заканчивает работу с исходными данными  $X$  за  $t$  шагов, то длина “претендента”  $Y$  не превосходит  $t$ .

**Доказательство.** Утверждение леммы следует из того, что длина записи слова не может быть больше, чем число шагов, затраченных на его выписывание.  $\square$

**Лемма 5.** Если недетерминированная МТ, проверяющая предикат  $\exists Y P(X, Y)$  заканчивает работу с исходными данными  $X$  за  $t$  шагов, то количество “претендентов”  $Y$  не превосходит  $2^{O(t)}$ .

**Доказательство.** Пусть  $A = \{a_1, \dots, a_k\}$  — внешний алфавит недетерминированной машины Тьюринга. Количество “претендентов”  $m$  не превосходит количества слов в этом алфавите, длина которых не превосходит  $t$ , т. е.

$$m \leq \sum_{i=0}^t k^i = \frac{k^{t+1} - 1}{k - 1} \leq k^{t+1} = 2^{(t+1) \log k} = 2^{O(t)}$$

$\square$

**Теорема 17.** По всякой недетерминированной машине Тьюринга, проверяющей предикат  $\exists Y P(X, Y)$  и заканчивающей работу с исходными данными  $X$  за  $t$  шагов, можно построить одноленточную машину Тьюринга, результат работы которой с исходными данными  $X$  совпадает с результатом работы исходной и число шагов которой составляет  $2^{O(t)}$ .

**Доказательство.** Работу детерминированной машины Тьюринга организуем следующим образом:

1. порождаем  $Y_1$ , проверяем  $P(X, Y_1) \leq t$  шагов;
2. порождаем  $Y_m$ , проверяем  $P(X, Y_m) \leq t$  шагов.

Общее число шагов не превосходит  $m \cdot t \leq t \cdot 2^{O(t)} = 2^{O(t+\log t)} = 2^{O(t)}$ .  $\square$

**Следствие (из доказательства).** Если недетерминированная машина Тьюринга, проверяющая предикат  $\exists Y P(X, Y)$  заканчивает работу с исходными данными  $X$  за  $t$  шагов, то можно построить одноленточную машину Тьюринга, результат работы которой с исходными данными  $X$  совпадает с результатом работы исходной и число используемых ячеек которой не превосходит  $t$ .

**Доказательство.** Для доказательства этого следствия достаточно в доказательстве теоремы после каждой проверки  $P(X, Y_m)$  возвращать головку машины Тьюринга в исходное положение. При этом число шагов может увеличиться вдвое.  $\square$

## 3.3. Нормальные алгоритмы Маркова

**Определение 67.** В обрабатываемом слове  $R$  находят первое вхождение слова  $P$  (если таковое имеется) и, не изменяя остальных частей слова  $R$ , это вхождение заменяют в нём словом  $Q$ . Полученное слово называется результатом применения *марковской подстановки*  $P \rightarrow Q$  к слову  $R$ . Если же вхождения  $P$  в слово  $R$  нет, то считается, что марковская подстановка  $P \rightarrow Q$  неприменима к слову  $R$ .

Марковская подстановка вида  $P \rightarrow \cdot Q$  называется *заключительной*.

**Определение 68.** Упорядоченный конечный список подстановок

$$\begin{cases} P_1 \rightarrow [\cdot]Q_1, \\ \dots \\ P_r \rightarrow [\cdot]Q_r, \end{cases}$$

в алфавите  $A$  называется *записью нормального алгоритма* в  $A$ .

**Принцип нормализации Маркова** Всякая интуитивно вычислимая функция может быть вычислена с помощью нормального алгоритма.

**Теорема 18.** Следующие классы функций, заданных на натуральных числах и принимающих натуральные значения, совпадают:

1. класс всех функций, вычислимых по Тьюрингу;
2. класс всех частично рекурсивных функций;
3. класс всех нормально вычислимых функций.

### 3.4. Код алгоритма. Применимость алгоритма к данным

**Определение 69.** Алгоритм  $A$  называется *применимым к данным*  $P$ , если он заканчивает работу над данными  $P$  за конечное число шагов.

**Обозначение.**  $!A(P)$

**Определение 70.** Алгоритм  $A$  называется *самоприменимым*, если он применим к собственному коду.

$$!A(\#A)$$

**Определение 71.** Алгоритм  $A$  называется *самоаннулируемым*, если результат его применения к собственному коду равен пустому слову (нулю).

$$A(\#A) = \Lambda$$

**Определение 72.** Алгоритм  $U$  называется *универсальным*, если для любого алгоритма  $A$  и исходных данных  $P$ , к которым он применим,  $U$  применим к  $\#A$  и  $P$  и результаты их работы совпадают.

$$\forall A \ P(!A(P) \rightarrow !U(\#A, P) \ \& \ U(\#A, P) = A(P))$$

**Определение 73.** Алгоритм  $B$  называется *продолжением алгоритма*  $A$  ( $A \subset B$ ), если для любых исходных данных  $P$ , к которым применим алгоритм  $A$ , алгоритм  $B$  тоже применим к ним и результаты их работы совпадают.

$$\forall P (!A(P) \rightarrow !B(P) \ \& \ A(P) = B(P))$$

**Теорема 19.** Не существует такого алгоритма  $B$ , который применим к кодам тех и только тех алгоритмов, которые не являются самоприменимыми.

$$\neg \exists B \ \forall A (!B(\#A) \leftrightarrow \neg !A(\#A))$$

**Доказательство.** Предположим, что такой алгоритм  $B_0$  существует

$$\forall A (!B_0(\#A) \leftrightarrow \neg !A(\#A))$$

Тогда при  $A = B_0$  верно

$$!B_0(\#B_0) \leftrightarrow \neg !B_0(\#B_0),$$

что невозможно. □

**Теорема 20.** Не существует такого алгоритма  $B$ , который равен нулю на кодах тех и только тех алгоритмов, которые не являются самоаннулируемыми.

$$\neg B \ \forall A (B(\#A) = \Lambda \leftrightarrow A(\#A) \neq \Lambda)$$

**Доказательство.** Предположим, что такой алгоритм  $B_0$  существует

$$\forall A (B_0(\#A) = \Lambda \leftrightarrow A(\#A) \neq \Lambda)$$

Тогда при  $A = B_0$  верно

$$B_0(\#B_0) = \Lambda \leftrightarrow B_0(\#B_0) \neq \Lambda,$$

что невозможно.  $\square$

**Теорема 21.** Не существует всюду применимого продолжения универсального алгоритма.

**Доказательство.** Предположим, что такой алгоритм  $B_0$  существует.

Построим алгоритм  $C$ , определяемый равенством  $\forall x(C(x) = B_0(x, x)\|a_1)$ , т. е. к результату его работы приписан символ  $a_1$ . Этот алгоритм всюду применим и следовательно, применим к собственному коду и  $C(\#C) = B_0(\#C, \#C)\|a_1$ .

Так как  $B_0$  — продолжение универсального алгоритма, то верно, что

$$\forall A P \left( !A(P) \rightarrow !B_0(\#A, P) \& B_0(\#A, P) = A(P) \right),$$

в частности, при  $A = C, P = \#C$

$$B_0(\#C, \#C) = C(\#C),$$

что противоречит полученному ранее значению для  $C(\#C)$ .  $\square$

### 3.5. Массовые проблемы. Алгоритмическая разрешимость и неразрешимость

**Определение 74.** Массовой проблемой называется задача вида

$$(?x) \varphi(x),$$

где  $\varphi(x)$  — формула какого-либо формализованного языка со свободной переменной  $x$ .

**Определение 75.** Массовая проблема  $(?x) \varphi(x)$  называется алгоритмически разрешимой, если существует всюду применимый алгоритм  $B$ , равный пустому слову  $\Lambda$  на тех и только тех значениях параметра  $x$ , для которых верна формула  $\varphi(x)$

$$\exists B \forall x(B(x) = \Lambda \leftrightarrow \varphi(x))$$

**Теорема 22.** Массовая проблема самоприменимости алгоритма

$$(?A) !A(\#A)$$

алгоритмически неразрешима.

**Доказательство.** Предположим, что  $(?A) !A(\#A)$  алгоритмически разрешима, т. е. имеется всюду применимый алгоритм  $B_0$  такой, что

$$\forall A (B_0(\#A) = \Lambda \leftrightarrow !A(\#A))$$

Построим алгоритм  $C$ , который применим к данным  $x$  тогда и только тогда, когда  $B_0(x) \neq \Lambda$ . Доказательство проведём для машины Тьюринга.

Пусть  $M_1$  — программа машины Тьюринга, вычисляющей  $B_0$  и имеющей состояния  $q_0, q_1, \dots, q_{k_1}$ .

Заменим в программе  $M_1$  состояние  $q_0$  на  $q_{k_1+1}$  и добавим команды

$$q_{k_1+1}a \rightarrow q_0a, \quad a \in A, \quad a \neq *$$

$$q_{k_1+1}* \rightarrow q_{k_1+2}*$$

Эта машина Тьюринга остановится, если  $B_0(P) \neq \Lambda$  (т. е.  $\neg !A(\#A)$ ), и головка будет бесконечно стоять на ячейке, в которой записан пустой символ, если  $B_0(P) = \Lambda$  (т. е.  $!A(\#A)$ ). Но по теор. 19 такой машины не существует.  $\square$

**Теорема 23.** Массовая проблема самоаннулируемости алгоритма

$$(\exists A) !A(\#A)$$

алгоритмически неразрешима.

**Доказательство.** Предположим, что  $(\exists A) A(\#A) = 0$  алгоритмически разрешима, т. е. имеется алгоритм  $B_0$  такой, что

$$\forall A (B_0(\#A) = \Lambda \leftrightarrow A(\#A) = 0)$$

Построим алгоритм  $C$ , который равен нулю на тех и только тех данных  $x$ , для которых  $B_0(x) \neq 0$ . Доказательство проведём для машины Тьюринга.

Пусть  $M_1$  — программа машины Тьюринга, вычисляющей  $B_0$  и имеющей состояния  $q_0, q_1, \dots, q_k$  и обрабатывающей слова в алфавите  $\{a_1, \dots, a_n, *\}$ .

Заменим в программе  $M_1$  состояние  $q_0$  на  $q_{k+1}$  и добавим команды

$$\begin{aligned} q_{k+1}* &\rightarrow q_0 a_1 \\ q_{k+1}a_i &\rightarrow q_{k+2}R*, \quad i \in \{1, \dots, n\} \\ q_{k+2}a_i &\rightarrow q_{k+2}R*, \quad i \in \{1, \dots, n\} \\ q_{k+2}* &\rightarrow q_0* \end{aligned}$$

Эта машина Тьюринга даёт в ответе  $\Lambda$ , если  $B_0(P) \neq \Lambda$  (т. е.  $A(\#A) = \Lambda$ ) и непустое значение, если  $B_0(P) = \Lambda$  (т. е.  $A(\#A) \neq 0$ ). Но по теор. 20 такой машины Тьюринга не существует.  $\square$

**Теорема 24.** Массовая проблема применимости алгоритма к данным алгоритмически неразрешима ни в одной из следующих формулировок

1.  $(\exists A P) !A(P)$ ;
2.  $(\exists A) !A(P)$ ;
3.  $(\exists P) !A(P)$ .

**Замечание.** Во второй и третьей формулировках имеются свободные переменные  $P$  и  $A$  соответственно. По ним предполагается квантор существования.

Из алгоритмической неразрешимости проблемы в третьей формулировке следует алгоритмическая неразрешимость проблемы в первой формулировке.

Кроме того, если доказана алгоритмическая неразрешимость проблемы в третьей формулировке для конкретного алгоритма, и в качестве данных  $P$  взяты те данные, для которых невозможно определить применимость к ним этого алгоритма, то тем самым будет доказана алгоритмическая неразрешимость проблемы во второй формулировке.

Поэтому докажем алгоритмическую неразрешимость проблемы в третьей формулировке для универсального алгоритма.

**Лемма 6.** Массовая проблема применимости универсального алгоритма к данным  $(\exists P) !U(P)$  алгоритмически неразрешима.

**Доказательство.** Предположим, что  $(\exists P) !U(P)$  алгоритмически разрешима, т. е. имеется алгоритм  $B_0$  такой, что

$$\forall P (B_0(P) = \Lambda \leftrightarrow !U(P))$$

Построим алгоритм  $C$ , который для кода любого алгоритма  $A$  и исходных данных  $P$  в качестве ответа выдаёт  $A(P)$ , если  $B_0(\#A, P) = \Lambda$ , и останавливается иначе. Доказательство проведём для машины Тьюринга.

Пусть  $M_1$  — программа машины Тьюринга, вычисляющей  $B_0$  и имеющей состояния  $q_0, q_1, \dots, q_{k_1}$ . Без потери общности можно считать, что при работе этой машины головка не сдвигается левее своего начального положения.

Пусть  $M_2$  — программа машины Тьюринга вычисляющей  $U$  и имеющей состояния  $q_0, q_1, \dots, q_{k_2}$ .

Можно построить машину Тьюринга, которая дублирует входное слово и останавливается в начале его второго экземпляра. Пусть эта машина имеет программу  $M_0$  и имеет состояния  $q_0, q_1, \dots, q_{k_0}$ .

Заменим в программе  $M_0$  состояние  $q_0$  на  $q_{k_0+1}$ , в программе  $M_1$  состояния  $q_i$  на  $q_{k_0+i}$  и  $q_0$  на

$q_{k_0+k_1+1}$  и добавим команды

$$\begin{aligned} q_{k_0+k_1+1}\Lambda &\rightarrow q_{k_0+k_1+2}L* \\ q_{k_0+k_1+1}a &\rightarrow q_0a, \quad a \in A, \quad a \neq * \\ q_{k_0+k_1+2}a &\rightarrow q_{k_0+k_1+2}LA, \quad a \in A, \quad a \neq * \\ q_{k_0+k_1+2}* &\rightarrow q_{k_0+k_1+3}R \end{aligned}$$

Добавим также программу  $M_2$ , в которой заменим состояния  $q_i$  ( $i \neq 0$ ) на  $q_{k_0+k_1+2+i}$ .

Эта машина Тьюринга является всюду применимым продолжением универсальной машины Тьюринга. Но по теор. 21 такой машины Тьюринга не существует.  $\square$

**Теорема 25.** Массовая проблема проверки общезначимости предикатной формулы алгоритмически неразрешима.

**Примечание.** Обычно говорят, что исчисление предикатов алгоритмически неразрешимо.

**Доказательство.** Построим по программе  $M$  универсального алгоритма и исходным данным  $P$  предикатную формулу, которая истинна тогда и только тогда, когда универсальный алгоритм применим к данным  $P$ .

Для этого достаточно рассмотреть исходные предикаты

$$O(i, k) \iff \text{"на } i\text{-м шаге машина } M \text{ находится в состоянии } q_k\text{"}$$

$$H(i, k) \iff \text{"на } i\text{-м шаге машина } M \text{ обозревает } j\text{-ю ячейку"}$$

$$S(i, j, h) \iff \text{"на } i\text{-м шаге в } j\text{-й ячейке записан символ } a_h\text{"}$$

С помощью этих предикатов можно описать весь процесс работы универсального алгоритма над исходными данными. Тот факт, что  $!U(P)$  запишется формулой, в посылке импликации которой стоит описание работы  $U$  над  $P$ , а в заключении — формула  $\exists i O(i, 0)$ .

Если исчисление предикатов алгоритмически разрешимо, то существует алгоритм, который по каждой такой формуле проверяет, общезначима ли она. Тем самым построен алгоритм, проверяющий применимость универсального алгоритма к данным.  $\square$

## 3.6. Теория сложности алгоритмов

**Определение 76.** Под *вычислительной сложностью* алгоритма понимают функцию, зависящую от длины записи исходных данных и характеризующую

- число шагов работы алгоритма над исходными данными (*временная сложность*);
- объём памяти, необходимой для работы алгоритма над исходными данными (*ёмкостная* или *зональная сложность*).

**Определение 77.** Сложностью  $S_A(P)$  алгоритма  $A$  при работе над данными  $P$  называется число шагов или объём памяти, затраченные в процессе работы алгоритма  $A$  над данными  $P$ .

**Определение 78.** Верхней (нижней) оценкой сложности алгоритма  $A$  при работе над данными длины  $n$  называется

$$S_A^U(n) = \max_{P: \|P\|=n} \{ S_A(P) \}$$

$$S_A^L(n) = \min_{P: \|P\|=n} \{ S_A(P) \}$$

**Определение 79.** Точной верхней оценкой сложности задачи  $Z$  с исходными данными длины  $n$  называется

$$S_Z^U(n) = \min_{A: A \text{ решает } Z} \{ S_A^U(n) \}$$

В качестве следствия теор. 17 можно доказать следующую теорему.

**Теорема 26.** Если задача вида  $\exists Y P(X, Y)$  принадлежит классу **NP**, то существует решающая её одноленточная машина Тьюринга, число шагов которой составляет  $2^{p(n)}$ , где  $p(n)$  — полином от длины записи исходных данных  $n = \|X\|$ .

### 3.6.1. Полиномиальная сводимость и полиномиальная эквивалентность

**Определение 80.** Задача  $Z_1$  вида  $\exists Y P_1(X, Y)$  при  $X \in D_1$  полиномиально сводится к задаче  $Z_2$  вида  $\exists Y P_2(X, Y)$  при  $X \in D_2$ , если существует функция  $f$ , отображающая  $D_1$  в  $D_2$ , такая, что

1. существует машина Тьюринга, вычисляющая функцию  $f$  не более чем за полиномиальное от длины записи исходных данных число шагов ( $f \in \mathbf{FP}$ );
2. задача  $Z_1$  имеет решение с исходными данными  $X$  тогда и только тогда, когда задача  $Z_2$  имеет решение с исходными данными  $f(X)$

$$\forall X \in D_1 \left( \exists Y P_1(X, Y) \leftrightarrow \exists Y P_2(f(X), Y) \right)$$

**Обозначение.**  $Z_1 \propto Z_2$

Далее под задачей  $Z_i$  будем подразумевать задачу вида  $\exists Y P_i(X, Y)$  при  $X \in D_i$ .

**Лемма 7.** Отношение полиномиальной сводимости рефлексивно и транзитивно.

**Лемма 8.** Если  $Z_1 \propto Z_2$  и  $Z_2 \in \mathbf{P}$ , то  $Z_1 \in \mathbf{P}$ .

#### Пример полиномиальной сводимости

**Задача 1 (Гамильтонов цикл).** **Дано:** граф  $G = (V, E)$ .

**Вопрос:** существует ли в  $G$  гамильтонов цикл?

$$\exists(v_{i_1}, \dots, v_{i_n}) \left( \{v_{i_1}, \dots, v_{i_n}\} = V \& \{v_{i_1}, v_{i_2}\} \in E \& \dots \& \{v_{i_n}, v_{i_1}\} \in E \right)$$

✉

**Задача 2 (Коммивояжёр).** **Дано:**  $C = \{c_1, \dots, c_n\}$  — множество городов,  $d_{ij} \in \mathbb{Z}_+$  — расстояния между  $c_i$  и  $c_j$ ,  $B \in \mathbb{Z}_+$ .

**Вопрос:** существует ли маршрут, проходящий через все города, длина которого меньше  $B$ ?

$$\exists(c_{i_1}, \dots, c_{i_m}) \left( \{c_{i_1}, \dots, c_{i_m}\} = C \& \sum_{j=1}^{m-1} d_{i_j i_{j+1}} + d_{i_m i_1} \leq B \right)$$

✉

Покажем, что **ГЦ**  $\propto$  **Коммивояжёр**. Для этого предъявим полиномиальный по времени алгоритм, который по графу  $G = (V, E)$  строит исходные данные  $C, d_{ij}$  и  $B$  с требуемыми свойствами.

$$C = V, \quad B = n, \quad d_{ij} = \begin{cases} 1, & \text{если } \{v_i, v_j\} \in E, \\ 2, & \text{иначе} \end{cases}$$

В графе есть гамильтонов цикл тогда и только тогда, когда маршрут проходит теми городами, расстояние между которыми равно 1.

**Определение 81.** Задача  $Z_1$  полиномиально эквивалентна задаче  $Z_2$ , если  $Z_1 \propto Z_2$  и  $Z_2 \propto Z_1$ .

**Обозначение.**  $Z_1 \sim_p Z_2$

**Теорема 27.** Отношение полиномиальной эквивалентности является отношением эквивалентности.

Отношение  $\sim_p$  разбивает класс **NP** на классы эквивалентности. Один из них — класс **P**.

### 3.6.2. NP-полные задачи

**Определение 82.** Задача  $Z$  называется *NP-полной*, если она сама принадлежит классу **NP**, и любая другая из класса **NP** полиномиально сводится к ней.

**Теорема 28.** Класс NP-полных задач образует класс эквивалентности в классе **NP**.

Если в определении NP-полной задачи убрать требование её принадлежности классу **NP**, то получим определение *NP-трудной* задачи.

### 3.6.3. Задача Выполнимость

Первым примером NP-полной задачи является следующая задача.

**Задача 3 (Выполнимость).** **Дано:**  $U = \{ u_1, \dots, u_n \}$  — множество пропозициональных переменных,  $C = \{ c_1, \dots, c_m \}$  — множество предложений над  $U$ .

**Вопрос:** выполнимо ли множество  $C$ , т. е. существует ли набор значений переменных из  $U$ , для которого истинны все предложения из  $C$ ?

$$\exists u_1, \dots, u_n (c_1 \ \& \ \dots \ \& \ c_m)$$

