КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Лекции по дискретной математике

Глава 1

Алгебра логики

1.1 Булевы функции

Булевы функции названы в честь английского математика XIX века Дж. Буля, который впервые применил алгебраические методы для решения логических задач. Они образуют самый простой нетривиальный класс дискретных функций – их аргументы и значения могут принимать всего два значения (если мощность множества значений функции равна 1, то это тривиальная функция – константа). С другой стороны, этот класс достаточно богат, и его функции имеют много интересных свойств. Булевы функции находят применение в логике, электротехнике, многих разделах информатики.

Обозначим через B двухэлементное множество $\{0, 1\}$. Тогда B^n это множество всех двоичных последовательностей (наборов, векторов) длины n.

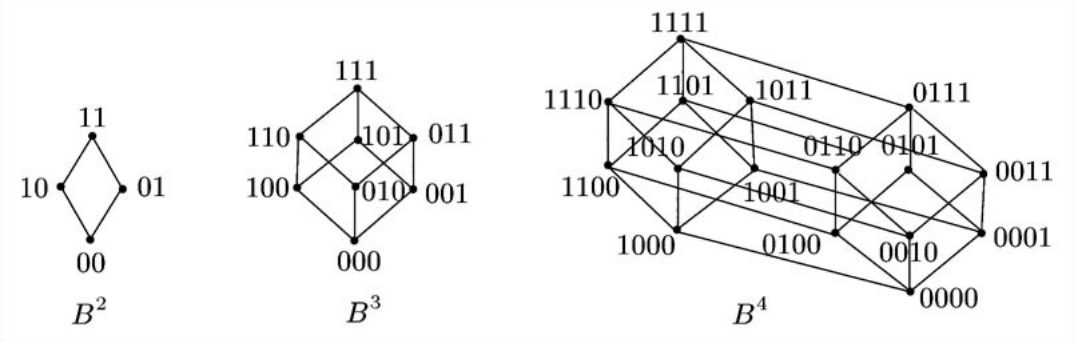
Определение 1.1.1. *Булевой функцией от n переменных (аргументов) называется любая функция $f(x_1, \ldots, x_n) : B^n \to B$.*

Каждый из ее аргументов x_i, $1 \leq i \leq n$, может принимать одно из двух значений 0 или 1, и значением функции на любом наборе из B^n также может быть 0 или 1. Обозначим через P_2^n множество всех булевых функций от n переменных, а также будем использовать обозначение P_2 для $\bigcup_{n\geq 0} P_2^n$, т.е. множества всех булевых функций.

Имеется несколько различных способов представления и интерпретации булевых функций. В этом разделе мы рассмотрим геометрическое и табличное представления, а также представление с помощью логических формул.

1.1.1 Геометрическое представление

B^n можно рассматривать как единичный n-мерный куб. Каждый набор из нулей и единиц длины n задает вершину этого куба. Ниже представлены единичные кубы B^n при $n = 2, 3, 4$.



При этом существует естественное взаимно-однозначное соответствие между подмножествами вершин n-мерных единичных кубов и булевыми функциями от n переменных: подмножеству $N \subseteq B^n$ соответствует его характеристическая функция

$$f_N(x_1, \ldots, x_n) = \begin{cases} 1, \text{ если } (x_1, \ldots, x_n) \in N \\ 0, \text{ в противном случае.} \end{cases}$$

Например, верхней грани куба B^3 (ее вершины выделены на рисунке) соответствует функция f : $f(0,0,1) = f(0,1,1) = f(1,0,1) = f(1,1,1) = 1$ и $f(0,0,0) = f(0,1,0) = f(1,0,0) = f(1,1,0) = 0$.

Очевидно, что указанное соответствие действительно взаимнооднозначное: каждая булевая функция f от n переменных задает подмножество $N_f = \{(\alpha_1, \ldots, \alpha_n) | f(\alpha_1, \ldots, \alpha_n) = 1\}$ вершин B^n . Например, функция, тождественно равная 0, задает пустое множество, а функция, тождественно равная 1, задает множество всех вершин B^n.

1.1.2 Табличное представление

Булевы функции от небольшого числа аргументов удобно представлять с помощью таблиц. Таблица для функции $f(x_1, \ldots, x_n)$ имеет $n+1$ столбец. В первых n столбцах указываются значения аргументов $x_1, \ldots, x_n$, а в $(n+1)$-ом столбце значение функции на этих аргументах – $f(x_1, \ldots, x_n)$.

x_1	x_2	$\ldots$	x_n	$f(x_1, x_2, \ldots, x_n)$
0	0	$\ldots$	0	$f(0,0,\ldots,0)$
0	0	$\ldots$	1	$f(0,0,\ldots,1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
1	1	$\ldots$	1	$f(1,1,\ldots,1)$

Если наборы рассматривать как записи чисел в двоичной системе счисления, то 1-ая строка представляет число 0, 2-ая – 1, 3-ая – 2, ..., а последняя – $(2^n - 1)$. Наборы аргументов в строках обычно располагаются в лексикографическом порядке (по возрастанию двоичных чисел).

При больших n табличное представление становится громоздким, например, для функции от 10 переменных потребуется таблица с 1024 строками. Но для малых n оно достаточно наглядно.

Используя табличное представление, нетрудно подсчитать число булевых функций от n переменных.

Теорема 1.1.1. $|P_2^n| = 2^{2^n}$.

Рассмотрим примеры элементарных булевых функций. Для $n = 1$:

x	0	1	x	$\overline{x}$
0	0	1	0	1
1	0	1	1	0

Функция 0 называется константой ноль, 1 – константой единица, функция x – тождественной, а функция $\overline{x}$ – отрицанием x.

Для $n = 2$:

x	y	$x\&y$	$x \vee y$	$x \to y$	$x \sim y$	$x \oplus y$	$x \mid y$	$x \downarrow y$
0	0	0	0	1	1	0	1	1
0	1	0	1	1	0	1	1	0
1	0	0	1	0	0	1	1	0
1	1	1	1	1	1	0	0	0

Приведем принятые названия указанных функций:

- $x\&y = x \wedge y = x \cdot y = xy$ – конъюнкция, логическое умножение, логическое "и";
- $x \vee y$ – дизъюнкция, логическое сложение, логическое "или";
- $x \to y$ – импликация, логическое следование;
- $x \sim y$ – эквивалентность;
- $x \oplus y$ – сложение по модулю 2, неэквивалентность, исключающее "или";
- $x \mid y$ – штрих Шеффера, отрицание конъюнкции;
- $x \downarrow y$ – стрелка Пирса, отрицание дизъюнкции.

Для $n = 3$ рассмотрим функцию "голосования" $m(x, y, z)$, значение которой равно значению большинства аргументов. Эта функция задается следующей таблицей:

x	y	z	$m(x, y, z)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Существенные и фиктивные переменные. В обычной алгебре справедливо равенство $x+y-y = x$, несмотря на то, что в левой части записана функция от двух переменных, а в правой – от одной. Таким образом, функции от разного числа переменных могут быть равными, что дает повод ввести понятие существенных и фиктивных переменных.

Определение 1.1.2. *Переменная x_i называется существенной переменной функции $f(x_1, \ldots, x_n)$, если существуют такие $\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n \in B$, что*

$$f(\alpha_1, \ldots, \alpha_{i-1}, 0, \alpha_{i+1}, \ldots, \alpha_n) \neq f(\alpha_1, \ldots, \alpha_{i-1}, 1, \alpha_{i+1}, \ldots, \alpha_n).$$

В противном случае переменная x_i называется фиктивной.

Если x_i – фиктивная переменная функции f, то функция f однозначно определяется некоторой функцией $g(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$, при этом $g(\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n) = f(\alpha_1, \ldots, \alpha_{i-1}, 0, \alpha_{i+1}, \ldots, \alpha_n) = f(\alpha_1, \ldots, \alpha_{i-1}, 1, \alpha_{i+1}, \ldots, \alpha_n)$ для любого набора значений α_i. Говорят, что g получается из f удалением фиктивной переменной x_i.

Кроме того, любую функцию можно дополнить произвольным числом фиктивных переменных. При этом ее таблица дополняется соответствующим числом новых строк и столбцов.

Определение 1.1.3. *Две функции алгебры логики называются равными, если одну из них можно получить из другой путем добавления или удаления некоторого числа фиктивных переменных.*

1.1.3 Формулы алгебры логики

Пусть задано некоторое множество булевых функций $A = \{f_1, f_2, \ldots\}$. Определим формулу над A следующим образом.

Определение 1.1.4. *По индукции:*

1. *Любая функция из A называется формулой над A.*
2. *Если $f \in A$ и для любого $1 \leq i \leq n$ H_i – либо переменная, либо формула над A, то выражение $f(H_1, H_2, \ldots, H_n)$ также является формулой над A.*
3. *Других формул нет.*

Примеры. $((x_1 \& x_2) \vee (x_1 \to x_3))$ является формулой, а $x\& \vee y$ – нет.

Обычно принимаются следующие соглашения для сокращения записи формул над множеством $\{^{-}, \&, \vee, \to, \oplus, \sim, |, \downarrow\}$:

- внешние скобки у формул можно опускать;
- ассоциативные скобки для $\&$, $\vee$ и $\oplus$ можно опускать;
- формулу $(H_1 \& H_2)$ можно записать в виде $(H_1 \cdot H_2)$ или $(H_1\, H_2)$;
- приоритет у связки $^{-}$ выше, чем у любой двухместной связки;
- связка $\&$ имеет более высокий приоритет по сравнению с $\vee$, $\oplus$, $\sim$ и $\to$.

Благодаря этому формулу $(((x \oplus (\overline{y})) \oplus z) \to ((x \& y) \vee z))$ можно записать в виде $(x \oplus \overline{y} \oplus z) \to (x\, y \vee z)$.

По аналогии с определением формулы вводится понятие функции, реализуемой формулой над A.

Определение 1.1.5. *По индукции:*

1. *Если формула $H = f(x_1, \ldots, x_n)$ для некоторой функции $f \in A$, то она реализует функцию f.*
2. *Пусть $H = f(H_1, \ldots, H_n)$, $f \in A$ и для любого $1 \leq i \leq n$ H_i – либо переменная, либо формула над A. Тогда формула H реализует функцию $f(g_1, g_2, \ldots, g_n)$, где g_i – либо переменная, либо функция, реализуемая формулой H_i.*

Определение 1.1.6. *Две формулы H_f и H_g называются эквивалентными, если реализуемые ими функции f и g равны, т.е. $H_f = H_g$, если $f = g$.*

Замечание. Для отношения эквивалентности формул мы не вводим специальное обозначение, а используем тот же, что и для равенства функций.

Основные эквивалентности.

1. Коммутативность:
   $x \circ y = y \circ x$
   для $\circ \in \{\&, \vee, \oplus, \sim, |, \downarrow\}$
2. Ассоциативность:
   $(x \circ y) \circ z = x \circ (y \circ z)$
   для $\circ \in \{\&, \vee, \oplus, \sim\}$
3. Дистрибутивность:
   $(x \vee y)z = xz \vee yz$
   $(xy) \vee z = (x \vee z)(y \vee z)$
   $(x \oplus y)z = xz \vee yz$
4. Законы поглощения:
   $x \vee x \cdot y = x$
   $x \cdot (x \vee y) = x$
5. Закон обобщенного склеивания
   $x\,y \vee \overline{x}\,z = x\,y \vee \overline{x}\,z \vee y\,z$
6. Законы де Моргана:
   $\overline{x \vee y} = \overline{x} \cdot \overline{y}$
   $\overline{x \cdot y} = \overline{x} \vee \overline{y}$
7. $x \vee x = x$
   $x \cdot x = x$
   $x \oplus x = 0$
   $x \vee \overline{x} = 1$
   $x \cdot \overline{x} = 0$
   $x \oplus \overline{x} = 1$
   $x \vee 1 = 1$
   $x \cdot 1 = x$
   $x \oplus 1 = \overline{x}$
   $x \vee 0 = x$
   $x \cdot 0 = 0$
   $x \oplus 0 = x$
8. $\overline{\overline{x}} = x$
   $x \mid y = \overline{x \cdot y}$
   $x \downarrow y = \overline{x \vee y}$
   $x \to y = \overline{x} \vee y$
   $x \sim y = (x \to y)(y \to x) = xy \vee \overline{x}\,\overline{y}$
   $x \oplus y = \overline{x \sim y} = x\overline{y} \vee \overline{x}y$

1.1.4 Булевы функции и логика высказываний

Как уже отмечалось, Дж. Буль ввел булевы функции для решения логических задач. В логике под высказыванием понимают некоторое повествовательное предложение, относительно которого можно сказать, истинно оно или ложно. Логика высказываний занимается выяснением истинности тех или иных высказываний, связью между истинностью различных высказываний и т.п.

Булевы функции могут служить полезным инструментом при решении многих логических задач.

Каждую переменную можно рассматривать как некоторое элементарное высказывание, принимающее одно из двух значений: 1 (истина) или 0 (ложь). Сложным высказываниям соответствуют формулы, построенные из элементарных высказываний с помощью логических связок. Вычисляя значения задаваемых ими функций, можно устанавливать зависимости истинностных значений сложных высказываний от значений входящих в них элементарных высказываний.

Важную роль в логике играют понятия тождественно истинной и выполнимой формулы.

Определение 1.1.7. *Булева формула H называется тождественно истинной (тождественно ложной), если она истинна (ложна) при любых значениях входящих в нее переменных, т.е. функция f_H тождественно равна 1 (тождественно равна 0).*

Определение 1.1.8. *Булева формула H называется выполнимой, если существует такой набор значений переменных, на котором она истинна, т.е. функция f_H равна 1 хотя бы на одном наборе аргументов.*

Как проверить тождественную истинность или выполнимость формулы H. На первый взгляд кажется, что ответ прост – построим по H таблицу для функции f_H , и, если в столбце значений стоят только единицы, то заключаем, что H тождественно истинна, если там есть хоть одна единица, то H выполнима. К сожалению, этот способ пригоден

только для формул с небольшим числом переменных. Уже для нескольких десятков и сотен переменных он не годится из-за большого размера получающейся таблицы – известно, что число 2^{90} превосходит количество атомов во всей видимой вселенной.

В математической логике построены аксиоматические системы, позволяющие формализовать человеческие рассуждения о выводимости одних тождественно истинных формул из других. В некоторых случаях они позволяют доказать тождественную истинность достаточно длинных формул, имеющих регулярную структуру. Но в общем случае они практически не применимы для произвольных формул с большим числом переменных.

В теории сложности алгоритмов имеется ряд результатов, которые свидетельствуют о том, что эффективных алгоритмов для проверки выполнимости или тождественной истинности произвольной булевой формулы не существует. Вместе с тем для некоторых подклассов формул эти задачи решаются достаточно эффективно.

1.2 Двойственные функции. Принцип двойственности

Определение 1.2.1. *Пусть* $f(x_1, \ldots, x_n) \in P_2$. *Тогда функция*

$$f^*(x_1, \ldots, x_n) = \overline{f}(\overline{x}_1, \ldots, \overline{x}_n)$$

называется двойственной к f.

Исходя из определения, таблица для функции f^* получается из таблицы для f записью значений в обратном порядке и заменой их на противоположные.

Пример.

x	y	f	f^*
0	0	0	1
0	1	0	0
1	0	1	1
1	1	0	1

Замечание. $f^{**} = f$.

Следующая теорема позволяет установить связь формул для произвольной функции и двойственной к ней.

Теорема 1.2.1. *Пусть*

$$F(x_1, \ldots, x_n) = f(f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)).$$

Тогда

$$F^* = f^*(f_1^*(x_1, \ldots, x_n), \ldots, f_m^*(x_1, \ldots, x_n)).$$

Следующее следствие из сформулированной теоремы называется *принципом двойственности.*

Следствие 1.2.1. *Пусть функция* F *задана формулой* $H_F[f_1, f_2, \ldots]$ *над* $A = \{f_1, f_2, \ldots\}$. *Тогда при замене в этой формуле вхождений* f_i *на* f_i^* *получится формула* $H_F[f_1^*, f_2^*, \ldots]$ *для* F^*.

Пример. Пусть $f = (x \cdot y \oplus 1) \vee \overline{z}$. Тогда $f^* = (x \vee y \sim 0) \cdot \overline{z}$.

Замечание. Если имеет место эквивалентность $H_1 = H_2$, то справедлива также эквивалентность $H_1^* = H_2^*$. Таким образом, для любой эквивалентности можно построить двойственную к ней.

Пример. $x \vee 0 = x \Rightarrow x \cdot 1 = x$.

1.3 Разложение булевой функции

Введем в рассмотрение следующую функцию x *в степени* σ.

Определение 1.3.1.

$$x^\sigma = \begin{cases} x, \sigma = 1 \\ \overline{x}, \sigma = 0 \end{cases}$$

или, что то же самое, $x^\sigma = x \cdot \sigma \vee \overline{x} \cdot \overline{\sigma}$.

Очевидно, что $x^\sigma = 1$ тогда и только тогда, когда $x = \sigma$.

Теорема 1.3.1. *Для любой булевой функции* $f(x_1, \ldots, x_n)$ *и любого целого* k $(1 \leq k \leq n)$ *справедливо следующее равенство:*

$$f(x_1, \ldots, x_n) = \bigvee_{(\sigma_1, \ldots, \sigma_k) \in B^k} x_1^{\sigma_1} \cdot x_2^{\sigma_2} \cdot \ldots \cdot x_k^{\sigma_k} \cdot f(\sigma_1, \ldots, \sigma_k, x_{k+1}, \ldots, x_n).$$

Это равенство называется разложением булевой функции по (первым) k переменным.

Следствие 1.3.1. *Разложение произвольной булевой функции по одной (первой) переменной имеет вид:*

$$f(x_1, \ldots, x_n) = \overline{x}_1 f(0, x_2, \ldots, x_n) \vee x_1 f(1, x_2, \ldots, x_n).$$

Следствие 1.3.2. *Для любой булевой функции, отличной от константы ноль, справедливо следующее представление:*

$$f(x_1, \ldots, x_n) = \bigvee_{(\sigma_1, \ldots, \sigma_n) | f(\sigma_1, \ldots, \sigma_n) = 1} x_1^{\sigma_1} \cdot x_2^{\sigma_2} \cdot \ldots \cdot x_n^{\sigma_n}.$$

Такое представление булевой функции называется ее совершенной дизъюнктивной нормальной формой (СДНФ). Оно позволяет по таблице построить формулу, реализующую данную функцию.

Пример.

x	y	$x \to y$
0	0	1
0	1	1
1	0	0
1	1	1

$$x \to y = x^0 \cdot y^0 \vee x^0 \cdot y^1 \vee x^1 \cdot y^1 = \overline{x}\,\overline{y} \vee \overline{x}\,y \vee x\,y$$

Теорема 1.3.2. *Для любой булевой функции, отличной от константы единица, справедливо следующее представление:*

$$f(x_1, \ldots, x_n) = \mathop{\&}_{(\sigma_1, \ldots, \sigma_n) | f(\sigma_1, \ldots, \sigma_n) = 0} x_1^{\overline{\sigma}_1} \vee x_2^{\overline{\sigma}_2} \vee \ldots \vee x_n^{\overline{\sigma}_n}.$$

Такое представление булевой функции является двойственным к СДНФ и называется совершенной конъюнктивной нормальной формой (СКНФ) функции.

Пример.

x	y	$x \to y$
0	0	1
0	1	1
1	0	0
1	1	1

$$x \to y = \overline{x} \vee y$$

1.4 Функционально полные системы

Определение 1.4.1. *Множество булевых функций $A = \{f_1, f_2, \ldots\}$ называется функционально полной системой (в P_2), если любую функцию из P_2 можно представить в виде формулы над A.*

Теорема 1.4.1. *Система $A = \{\&, \vee, \neg\}$ является полной.*

Теорема 1.4.2. *Если система $A = \{f_1, f_2, \ldots\}$ полна, и любая функция из A может быть представлена в виде формулы над некоторой системой $B = \{g_1, g_2, \ldots\}$, то система B также полна.*

1.4.1 Примеры полных систем

1. $A = P_2$
2. $A = \{\&, \vee, \neg\}$
3. $A = \{\&, \neg\}$
4. $A = \{\vee, \neg\}$
5. $A = \{\&, \oplus, 1\}$
6. $A = \{\to, 0\}$
7. $A = \{|\}$
8. $A = \{\downarrow, 0\}$

1.5 Полиномы Жегалкина

Определение 1.5.1. *Монотонной конъюнкцией над множеством переменных $X_n = \{x_1, \ldots, x_n\}$ называется любое выражение вида $x_{i_1} \cdot x_{i_2} \cdot \ldots \cdot x_{i_r}$, где $1 \le r \le n$, $1 \le i_j \le n$ для любого $j \in \{1, \ldots, n\}$, и все переменные различны, т.е. $i_j = i_k \iff j = k$.*

Определение 1.5.2. *Полиномом Жегалкина над множеством переменных $X_n = \{x_1, \ldots, x_n\}$ называется выражение вида*

$$K_1 \oplus K_2 \oplus \ldots \oplus K_l \oplus \sigma,$$

где $l \ge 0$, а все K_j – это различные монотонные конъюнкции над множеством переменных X_n.

Другими словами, полиномом Жегалкина – это сумма конъюнкций различных переменных и, возможно, константа.

Теорема 1.5.1. *Любая булева функция $f(x_1, \ldots, x_n)$ может быть единственным образом представлена в виде полинома Жегалкина над множеством переменных $X_n = \{x_1, \ldots, x_n\}$.*

1.6 Замыкание. Основные замкнутые классы

Определение 1.6.1. *Пусть $A \subseteq P_2$. Тогда замыканием A называется множество всех булевых функций, которые можно представить в виде формулы над A.*

Замыкание множества A принято обозначать через $[A]$.

Примеры. Рассмотрим некоторые множества булевых функций и их замыкания.

1. Пусть $A = P_2$. Тогда $[A] = P_2$.
2. Пусть $A = \{0\}$. Тогда $[A] = \{0\}$.
3. Пусть $A = \{\&, \vee, \neg\}$. Тогда $[A] = P_2$.
4. Пусть $A = \{1, \oplus\}$. Тогда $[A] = \{f \mid f = \alpha_1 x_1 \oplus \alpha_1 x_1 \oplus \ldots \oplus \alpha_n x_n \oplus \alpha_{n+1},$ где $\alpha_i -$ const$\} = L$ – класс линейных функций.

На основе понятия замыкания можно ввести эквивалентное определение функционально полной системы.

Определение 1.6.2. *Множество булевых функций A называется функционально полной системой, если $[A] = P_2$.*

Свойства замыкания.

1. $A \subseteq [A]$;
2. $\big[[A]\big] = [A]$;
3. $A_1 \subseteq A_2 \Rightarrow [A_1] \subseteq [A_2]$;
4. $[A_1] \cup [A_2] \subseteq [A_1 \cup A_2]$.

Определение 1.6.3. *Множество булевых функций A называется замкнутым классом, если $[A] = A$.*

Примеры. P_2, $\{0\}$ и L являются замкнутыми классами.

1.6.1 Класс T_0

Определение 1.6.4. *Класс функций, сохраняющих константу 0, определяется следующим образом:*

$$T_0 = \{f \in P_2 | f(0, 0, \ldots, 0) = 0\}.$$

Лемма 1.6.1. *Класс T_0 замкнут.*

Исходя из определения, для проверки принадлежности булевой функции классу T_0 достаточно построить первую строчку ее таблицы – значение функции на нулевом наборе должно быть нулевым.

Примеры. Функции 0, $x \cdot y$, $x \vee y$ принадлежат классу T_0, а 1, $x \to y$, $x \sim y$ – нет.

Обозначим через T_0^n подмножество функций класса T_0, зависящих от n переменных. Используя определение класса T_0, нетрудно подсчитать число функций в T_0^n.

Лемма 1.6.2. $|T_0^n| = 2^{2^n-1}$.

1.6.2 Класс T_1

Определение 1.6.5. *Класс функций, сохраняющих константу* 1, *определяется следующим образом:*

$$T_1 = \{f \in P_2 | f(1, 1, \ldots, 1) = 1\}.$$

Лемма 1.6.3. *Класс T_1 замкнут.*

Исходя из определения, для проверки принадлежности булевой функции классу T_1 достаточно построить последнюю строчку ее таблицы – значение функции на единичном наборе должно быть единичным.

Примеры. Функции 1, $x \to y$, $x \sim y$ принадлежат классу T_1, а 0, $x \oplus y$, $x \mid y$ – нет.

Обозначим через T_1^n подмножество функций класса T_1, зависящих от n переменных. Используя определение класса T_1, нетрудно подсчитать число функций в T_1^n.

Лемма 1.6.4. $|T_1^n| = 2^{2^n-1}$.

1.6.3 Класс S

Определение 1.6.6. *Булева функция $f(x_1, \ldots, x_n) \in P_2$ называется самодвойственной, если она совпадает со своей двойственной, т.е. $f = f^*$.*

Другими словами, самодвойственная функция на противоположных наборах принимает противоположные значения, т.е. таблица такой функции будет антисимметрична относительно центра.

Пример.

x	y	z	$m(x,y,z)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Определение 1.6.7. *Класс S определяется следующим образом:*

$$S = \{f \in P_2 | f = f^*\}.$$

Лемма 1.6.5. *Класс S замкнут.*

Исходя из определения, для проверки принадлежности булевой функции классу S достаточно построить ее двойственную – она должна совпадать с исходной.

Примеры. Функции x, $\overline{x}$ принадлежат классу S, а $x \cdot y$, $x \vee y$, $x \to y$ – нет.

Обозначим через S^n множество самодвойственных функций, зависящих от n переменных. Используя определение самодвойственной функции, нетрудно подсчитать число функций в S^n.

Лемма 1.6.6. $|S^n| = 2^{2^{n-1}}$.

Следующее утверждение называется *леммой о несамодвойственной функции.*

Лемма 1.6.7. *Если $f \notin S$, то подстановкой вместо всех ее переменных x, $\overline{x}$ можно получить константу.*

1.6.4 Класс M

Определение 1.6.8. *Пусть $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ и $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$, $\alpha, \beta \in B^n$. Тогда*

$$\alpha \prec \beta \iff \forall i\, (\alpha_i \leq \beta_i).$$

Замечание. Существуют наборы, для которых неприменимо отношение упорядоченности, определенное выше. Так, например, наборы $(0, 0, 1)$ и $(0, 1, 0)$ несравнимы.

Определение 1.6.9. *Булева функция $f(x_1, \ldots, x_n) \in P_2$ называется монотонной, если для любых двух наборов $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ и $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$ таких, что $\alpha \prec \beta$, выполняется $f(\alpha) \leq f(\beta)$.*

M обозначает класс всех монотонных функций из P_2.

Лемма 1.6.8. *Класс M замкнут.*

Исходя из определения, для проверки принадлежности булевой функции классу M достаточно построить ее таблицу и проверить отсутствие нарушений на сравнимых наборах в середине таблицы, отбросив первые нули и последние единицы.

Примеры. Функции 1, $x \cdot y$, $x \vee y$ принадлежат классу M, а $\overline{x}$, $x \oplus y$, $x \to y$ – нет.

Кроме того, проверить принадлежность булевых функций классу M можно при помощи следующей леммы.

Лемма 1.6.9. *Функция f, отличная от константы, монотонна тогда и только тогда, когда она представима в виде формулы над $\{\&, \vee\}$.*

Замечание. Константы 0 и 1 принадлежат классу M.

Следующее утверждение называется *леммой о немонотонной функции.*

Лемма 1.6.10. *Если $f \notin M$, то подстановкой вместо всех ее переменных 0, 1, x можно получить $\overline{x}$.*

1.6.5 Класс L

Определение 1.6.10. *Булева функция $f(x_1, \ldots, x_n) \in P_2$ называется линейной, если она представима в виде линейного полинома Жегалкина, т.е. полинома вида*

$$f(x_1, \ldots, x_n) = \alpha_0 \oplus \alpha_1 x_1 \oplus \alpha_2 x_2 \oplus \ldots \oplus \alpha_n x_n$$

для некоторого набора значений $\alpha_i \in \{0, 1\}$, $0 \leq i \leq n$.

Другими словами, в полиноме Жегалкина линейной функции отсутствует конъюнкция переменных.

L обозначает класс всех линейных функций из P_2.

Лемма 1.6.11. *Класс L замкнут.*

Исходя из определения, для проверки принадлежности булевой функции классу L достаточно построить ее полином Жегалкина – в нем не должно быть конъюнкций переменных.

Примеры. Функции $\overline{x}$, $x \oplus y$, $x \sim y$ принадлежат классу L, а $x \cdot y$, $x \vee y$, $x \to y$ – нет.

Кроме того, проверить принадлежность булевых функций классу L можно при помощи двух следующих лемм.

Лемма 1.6.12. *Если $f \in L$ и не является константой, то число нулей и единиц в ее таблице совпадает.*

Лемма 1.6.13. *Функция $f \in L$ тогда и только тогда, когда она на соседних (по существенным переменным) наборах принимает противоположные значения.*

Обозначим через L^n подмножество линейных функций, зависящих от n переменных. Используя определение линейной функции, нетрудно подсчитать число функций в L^n.

Лемма 1.6.14. $|L^n| = 2^{n+1}$.

Следующее утверждение называется *леммой о нелинейной функции.*

Лемма 1.6.15. *Если $f \notin L$, то подстановкой в нее 0, 1, x, $\overline{x}$, y, $\overline{y}$ можно получить $x \cdot y$ или $\overline{x \cdot y}$.*

Замечание. Замкнутые классы T_0, T_1, S, M, L попарно различны.

	T_0	T_1	S	M	L
0	+	−	−	+	+
1	−	+	−	+	+
$\overline{x}$	−	−	+	−	+
$x \oplus y \oplus z$	+	+	+	−	+
$m(x, y, z)$	+	+	+	+	−

1.7 Теорема Поста и ее следствия

Следующая теорема дает критерий проверки функциональной полноты для произвольной системы булевых функций и называется *теоремой Поста о функциональной полноте.*

Теорема 1.7.1. *Система булевых функций $A = \{f_1, f_2, \ldots\}$ полна в P_2 тогда и только тогда, когда она не содержится целиком ни в одном из пяти основных замкнутых классов T_0, T_1, S, M, L.*

Примеры.

1. Система $A = \{x \cdot y, x \vee y, \overline{x}\}$ полна в P_2, т.к. не содержится целиком ни в одном из пяти основных замкнутых классов.

	T_0	T_1	S	M	L
$x \cdot y$	+	+	−	+	−
$x \vee y$	+	+	−	+	−
$\overline{x}$	−	−	+	−	+

2. Система $A = \{x \to y, x \sim y\}$ не полна, т.к. целиком содержится в классе T_1.

	T_0	T_1	S	M	L
$x \to y$	−	+	−	−	−
$x \sim y$	−	+	−	−	+

3. Система $A = \{x \mid y\}$ полна в P_2, т.к. не содержится целиком ни в одном из пяти основных замкнутых классов.

	T_0	T_1	S	M	L
$x \mid y$	−	−	−	−	−

Следующие следствия показывает, почему классы T_0, T_1, S, M, L принято называть *основными.*

Следствие 1.7.1. *Пусть A – некоторый замкнутый класс, отличный от P_2. Тогда класс A содержится в одном из классов T_0, T_1, S, M, L (или совпадает с одним из них).*

Определение 1.7.1. *Класс функций A называется предполным классом, если:*

1. *A – не полный класс;*
2. *Для любой булевой функции $f \notin A$, класс $A \cup \{f\}$ является полным в P_2.*

Следствие 1.7.2. *В алгебре логики существует в точности пять предполных классов: T_0, T_1, S, M, L.*

1.7.1 Базисы алгебры логики

Определение 1.7.2. *Система булевых функций $A = \{f_1, f_2, \ldots\}$ называется базисом (в P_2), если:*

1. *A – полный класс;*
2. *Для любой булевой функции $f \in A$, класс $A \backslash \{f\}$ не является полным в P_2.*

Примеры.

1. Система $A = \{x \cdot y, x \vee y, \overline{x}\}$ не является базисом, т.к. при удалении из нее конъюнкции или дизъюнкции она останется полной.
2. Система $A = \{x \to y, x \sim y\}$ не является базисом, т.к. не полна.
3. Система $A = \{x \mid y\}$ является базисом.
4. Система $A = \{0, 1, x \oplus y \oplus z, m(x, y, z)\}$ также является базисом.

Теорема 1.7.2. *Максимальное число функций в базисе алгебры логики равно 4.*

1.8 Минимизация булевых функций

В данном разделе рассматривается задача построения минимальной формулы для заданной булевой функции, а конкретнее – ее минимальной дизъюнктивной нормальной формы.

Определение 1.8.1. *Элементарной конъюнкцией K над множеством переменных $X_n = \{x_1, \ldots, x_n\}$ назовем формулу вида*

$$K = x_{i_1}^{\sigma_{i_1}} \cdot x_{i_2}^{\sigma_{i_2}} \cdot \ldots \cdot x_{i_r}^{\sigma_{i_r}},$$

т.е. конъюнкцию различных переменных или их отрицаний.

Число переменных r, входящих в элементарную конъюнкцию, называется *рангом* элементарной конъюнкции.

Определение 1.8.2. *Дизъюнктивной нормальной формой (ДНФ) над множеством переменных X_n назовем дизъюнкцию различных элементарных конъюнкций над X_n:*

$$K_1 \vee K_2 \vee \ldots \vee K_m.$$

Число элементарных конъюнкций m, входящих в ДНФ, называется *длиной* ДНФ.

Замечание. Рассмотренная ранее СДНФ является частным случаем ДНФ, когда все элементарные конъюнкции имеют ранг n (число переменных).

Замечание. Для одной и той же булевой функции может существовать несколько ДНФ.

Пример.

x	y	z	f
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

1. СДНФ$= \overline{x}\,\overline{y}\,\overline{z} \vee \overline{x}\,\overline{y}\,z \vee \overline{x}\,y\,\overline{z} \vee x\,\overline{y}\,z \vee x\,y\,z$
2. ДНФ$_1 = \overline{x}\,\overline{z} \vee x\,z \vee \overline{x}\,\overline{y}$
3. ДНФ$_2 = \overline{x}\,\overline{z} \vee x\,z \vee \overline{y}\,z$
4. ДНФ$_2 = x\,y\,z \vee \overline{y}\,z \vee \overline{x}\,\overline{z}$

Таким образом, задача минимизации булевой функции состоит в построении наиболее простой ДНФ для заданной функции. Введем критерии оценки сложности ДНФ.

Определение 1.8.3. *Минимальной ДНФ для функции f назовем ДНФ, реализующую функцию f и содержащую наименьшее число вхождений переменных по сравнению с любой другой ДНФ, реализующей данную функцию.*

Пример. В примере выше ДНФ$_1$ и ДНФ$_2$ оказываются минимальными ДНФ, реализующими функцию f.

Определение 1.8.4. *Кратчайшей ДНФ для функции f назовем ДНФ, реализующую функцию f и имеющую наименьшую длину по сравнению с любой другой ДНФ, реализующей данную функцию.*

Пример. В рассмотренном ранее примере ДНФ$_1$, ДНФ$_2$ и ДНФ$_3$ оказываются кратчайшими ДНФ, реализующими функцию f.

С учетом введенных определений, задача минимизации формулируется как поиск минимальной и кратчайшей ДНФ для заданной функции.

1.8.1 Тривиальный алгоритм построения минимальной и кратчайшей ДНФ

Подсчитаем число различных ДНФ над множеством переменных $X_n = \{x_1, \ldots, x_n\}$.

Лемма 1.8.1. *Число различных ДНФ над множеством переменных X_n равно 2^{3^n}.*

Т.к. число различных ДНФ конечно, можно найти минимальную и кратчайшую ДНФ полностью перебрав все варианты. Таким образом, можно сформулировать тривиальный алгоритм построения минимальной и кратчайшей ДНФ:

1. Строим все возможные ДНФ и сортируем их в порядке возрастания сложности.
2. Перебираем ДНФ от простых к сложным до тех пор, пока таблица очередной ДНФ не совпадет с таблицей заданной функции f.

Найденная ДНФ и будет простейшей ДНФ, реализующей f.

Указанный алгоритм можно заметно улучшить, если исключить ДНФ, содержащие элементарные конъюнкции, которые заведомо не могут войти в ДНФ для функции f.

Определение 1.8.5. *Элементарная конъюнкция K называется допустимой конъюнкцией для функции f, если $K \leq f$.*

Лемма 1.8.2. *В любой ДНФ функции f содержатся только допустимые конъюнкции для f.*

1.8.2 Геометрическая интерпретация построения минимальной и кратчайшей ДНФ

Геометрическая интерпретация построения минимальной и кратчайшей ДНФ основывается на представлении булевой функции $f(x_1, \ldots, x_n)$ в виде подмножества N_f вершин n-мерного единичного куба B^n.

Отметим некоторые свойства, которыми обладает множество N_f.

1. Пусть $f = f_1 \cdot f_2$. Тогда $N_f = N_{f_1} \cap N_{f_2}$.
2. Пусть $f = f_1 \vee f_2$. Тогда $N_f = N_{f_1} \cup N_{f_2}$.
3. Пусть $f = \overline{f}_1$. Тогда $N_f = B^n \backslash N_{f_1}$.
4. Пусть $f_1 \leq f_2$. Тогда $N_{f_1} \subseteq N_{f_2}$.

Определение 1.8.6. *Пусть* $K = x_{i_1}^{\sigma_{i_1}} \cdot x_{i_2}^{\sigma_{i_2}} \cdot \ldots \cdot x_{i_r}^{\sigma_{i_r}}$ *– это элементарная конъюнкция ранга* r *над множеством переменных* $X_n = \{x_1, \ldots, x_n\}$, *а* N_K *– множество вершин* B^n, *где* $K = 1$. *Тогда* N_K *называется интервалом ранга* r, *соответствующим элементарной конъюнкции* K.

По определению N_K содержит те наборы, где $x_{i_1} = \sigma_{i_1}, x_{i_2} = \sigma_{i_2}, \ldots, x_{i_r} = \sigma_{i_r}$, а значения остальных переменных из X_n произвольны.

Примеры. Рассмотрим примеры различных интервалов.

1. Пусть $K = \overline{x}\, y\, z$. Тогда $N_K = \{(0,1,1)\}$ – вершина куба B^3.
2. Пусть $K = \overline{x}\, y$. Тогда $N_K = \{(0,1,0),(0,1,1)\}$ – ребро куба B^3.
3. Пусть $K = \overline{x}$. Тогда $N_K = \{(0,0,0),(0,0,1),(0,1,0),(0,1,1)\}$ – грань куба B^3.

Рассмотрим некоторую ДНФ для функции f

$$f = K_1 \vee K_2 \vee \ldots \vee K_m.$$

Согласно свойствам множества N_f, имеем

$$N_f = \bigcup_{i=1}^{m} N_{K_i}.$$

Таким образом, каждой ДНФ функции f соответствует некоторое покрытие множества N_f интервалами N_{K_i} такими, что $\forall i\ N_{K_i} \subseteq N_f$.

Справедливо и обратное: по произвольному покрытию множества N_f интервалами N_{K_i}, удовлетворяющими условию $N_{K_i} \subseteq N_f$, можно построить ДНФ для функции f.

В силу вышесказанного, геометрическая интерпретация построения минимальной и кратчайшей ДНФ состоит в отыскании минимального и кратчайшего покрытия множества N_f интервалами N_{K_i}.

1.8.3 Сокращенные ДНФ

Определение 1.8.7. *Интервал* N_K *называется максимальным интервалом, если*

1. $N_K \subseteq N_f$;
2. *Не существует интервала* $N_{K'}$ *такого, что* $N_K \subset N_{K'} \subseteq N_f$.

Очевидно, что любой интервал $N_K \subseteq N_f$ содержится в некотором максимальном интервале или совпадает с ним.

Определение 1.8.8. *Сокращенной ДНФ функции* f *(*$Д_c(f)$*) называется ДНФ, соответствующая покрытию множества* N_f *всеми максимальными интервалами.*

Следующие две теоремы связывают построение сокращенной ДНФ с задачей минимизации булевых функций.

Теорема 1.8.1. *Любая минимальная ДНФ функции* f *может быть получена из сокращенной ДНФ отбрасыванием некоторых элементарных конъюнкций.*

Теорема 1.8.2. *Существует кратчайшая ДНФ функции f, получаемая из сокращенной ДНФ отбрасыванием некоторых элементарных конъюнкций.*

В общем случае сокращенная ДНФ для f не является ее минимальной и кратчайшей ДНФ, однако справедлива следующая теорема.

Теорема 1.8.3. *Сокращенная ДНФ для монотонной функции*

1. *Не содержит отрицаний;*
2. *Является ее единственной минимальной и кратчайшей ДНФ.*

Метод Блейка для построения сокращенной ДНФ из произвольной ДНФ состоит в применении правил обобщенного склеивания $x\,K_1 \vee \overline{x}\,K_2 = x\,K_1 \vee \overline{x}\,K_2 \vee K_1\,K_2$ и поглощения $K_1 \vee K_1\,K_2 = K_1$:

1. На первом этапе производится операция обобщенного склеивания до тех пор, пока это возможно.
2. На втором этапе применяется правило поглощения.

Теорема 1.8.4. *В результате применения метода Блейка получается сокращенная ДНФ.*

1.8.4 Тупиковые ДНФ

Определение 1.8.9. *Покрытие множества N_f максимальными интервалами называется неприводимым, если после удаления из него любого интервала оно перестает быть покрытием.*

Определение 1.8.10. *ДНФ функции f называется тупиковой ДНФ, если она соответствует неприводимому покрытию.*

Замечание. Любая тупиковая ДНФ получается из сокращенной отбрасыванием некоторых элементраных конъюнкций. При этом тупиковая ДНФ для функции f строится неоднозначно.

Теорема 1.8.5. *Любая минимальная ДНФ функции f является ее тупиковой ДНФ.*

Теорема 1.8.6. *Существует кратчайшая ДНФ функции f, являющаяся ее тупиковой ДНФ.*

1.8.5 Методы построения тупиковых ДНФ

Определение 1.8.11. *Будем говорить, что элементарная конъюнкция K поглощается ДНФ для функции $f = K_1 \vee \ldots \vee K_m$, если $K \le f$ или $N_K \subset \bigcup\limits_{i=1}^{m} N_{K_i}$.*

Теорема 1.8.7. *Для того, чтобы элементарная конъюнкция K поглощалась ДНФ необходимо и достаточно, чтобы каждая ее элементарная конъюнкция K_i представлялась в виде $K_i' \cdot K_i''$, причем:*

1. $K \le \mathop{\&}\limits_{i=1}^{m} K_i'$;

2. $\bigcup_{i=1}^{m} K_i'' = 1$.

Определение 1.8.12. *Элементарные конъюнкции K_1 и K_2 называются ортогональными, если их произведение равно нулю, или, что то же самое, соответствующие им интервалы не пересекаются ($N_{K_1} \cap N_{K_2} = \emptyset$).*

Очевидно, что для исследования того, поглощается ли элементарная конъюнкция K ДНФ, достаточно исследовать только те элементарные конъюнкции из ДНФ, которые не ортогональны K.

Теорема 1.8.8. *Элементарные конъюнкции K_1 и K_2 ортогональны тогда и только тогда, когда в одной из них содержится некоторая переменная, а в другой ее отрицание.*

ДНФ Квайна.

Определение 1.8.13. *Максимальный интервал $N_K \subset N_f$ называется ядровым интервалом функции f, если существует вершина $\alpha \in B^n$, принадлежащая N_K и не принадлежащая никакому другому максимальному интервалу.*

Определение 1.8.14. *Множество всех ядровых интервалов функции f называется ее ядром.*

Очевидно, что интервалы, входящие в ядро, входят в каждое неприводимое покрытие. Соответствующие им элементарные конъюнкции входят в каждую тупиковую ДНФ. С другой стороны, интервалы, покрываемые ядром, не входят ни в одно неприводимое покрытие.

Определение 1.8.15. *ДНФ, получаемая из сокращенной ДНФ отбрасыванием максимальных интервалов, которые покрываются ядром, называется ДНФ Квайна.*

ДНФ типа ΣT.

Определение 1.8.16. *ДНФ, соответствующая покрытию множества N_f совокупностью максимальных интервалов, входящих по крайней мере в одно из неприводимых покрытий, называется ДНФ типа ΣT ($Д_{\Sigma T}$).*

Другими словами, $Д_{\Sigma T}$ – это объединение (дизъюнкция) всех тупиковых ДНФ.

Определение 1.8.17. *Пусть $\alpha \in N_f$. Тогда множество всех максимальных интервалов, покрывающих вершину α, называется пучком (Π_α).*

Определение 1.8.18. *Пусть α – вершина B^n, принадлежащая N_f и некоторому интервалу N_K. Вершину α назовем регулярной относительно N_f и N_K, если существует вершина $\alpha' \in N_f \backslash N_K$ такая, что ($\Pi_{\alpha'} \subset \Pi_\alpha$).*

Определение 1.8.19. *Максимальный интервал N_K называется регулярным, если каждая его вершина является регулярной относительно N_f и N_K.*

Теорема 1.8.9. *Для того, чтобы элементарная конъюнкция не принадлежала $Д_{\Sigma T}$, необходимо и достаточно, чтобы соответствующий интервал был регулярным.*

Теорема 1.8.10. *$Д_{\Sigma T}$ получается из ДНФ Квайна быть может отбрасыванием некоторых элементарных конъюнкций.*

Глава 2

Основы теории графов

2.1 Основные определения

Определение 2.1.1. *Графом называется произвольное множество элементов V и произвольное семейство E пар из V. Обозначение: $G = (V, E)$.*

В дальнейшем будем рассматривать конечные графы, то есть графы с конечным множеством элементов и конечным семейством пар.

Определение 2.1.2. *Если элементы из E рассматривать как неупорядоченные пары, то граф называется неориентированным, а пары называются ребрами. Если же элементы из E рассматривать как упорядоченные, то граф называется ориентированным, а пары — дугами.*

Определение 2.1.3. *Пара вида (a, a) называется петлей, если пара (a, b) встречается во множестве E несколько раз, то она называется кратным ребром (кратной дугой).*

В дальнейшем граф без петель и кратных ребер будем называть простым графом (или просто графом), граф без петель — мультиграфом, а мультиграф, в котором разрешены петли — псевдографом.

Определение 2.1.4. *Две вершины графа называются смежными, если они соединены ребром.*

Определение 2.1.5. *Говорят, что вершина и ребро инцидентны, если ребро содержит вершину.*

Определение 2.1.6. *Степенью вершины $deg(v)$ называется количество ребер, инцидентных данной вершине. Для псевдографа петля учитывается дважды.*

Если для некоторой вершины v выполняется $deg(v) = 1$, то она называется висячей. Если же $deg(v) = 0$, то вершину v называют изолированной.

Относительно степеней вершин в графе справедливы следующие утверждения.

Лемма 2.1.1. *В любом графе (псевдографе) справедливо следующее соотношение:*

$$\sum_{i=1}^{m} deg(v_i) = 2q,$$

где m — число вершин, а q — число ребер.

Лемма 2.1.2. *В любом графе число вершин нечетной степени четно.*

Определение 2.1.7. *Два графа $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$ называются изоморфными, если существует взаимно однозначное отображение $\varphi : V_1 \to V_2$ такое, что $(u, v) \in E_1 \iff (\varphi(u), \varphi(v)) \in E_2$.*

Другими словами, в изоморфных графах сохраняется отношение инцидентности между соответствующими парами вершин. Неформально, изоморфизм сводится к переименованию вершин.

Определение 2.1.8. *Граф $G_1 = (V_1, E_1)$ называется подграфом графа $G = (V, E)$, если $V_1 \subseteq V$ и $E_1 \subseteq E$.*

2.2 Геометрическая реализация графов

Определение 2.2.1. *Пусть задан некоторый неориентированный граф $G = (V, E)$. Пусть любой вершине v_i графа G сопоставлена некоторая точка a_i, причем $a_i \neq a_j$ $(i \neq j)$, а любому ребру $e = (v_i, v_j)$ сопоставлена некоторая непрерывная кривая L, соединяющая точки a_i и a_j и не проходящая через другие точки a_k $(k \neq i, j)$. Тогда если все кривые, сопоставленные ребрам, не имеют общих точек, кроме концевых, то говорят, что задана геометрическая реализация графа G.*

Теорема 2.2.1. *Для любого графа существует его реализация в трехмерном пространстве.*

Определение 2.2.2. *Граф называется планарным, если существует его геометрическая реализация в двумерном пространстве.*

Определение 2.2.3. *Графом K_5 называется граф с пятью вершинами, в котором каждая пара вершин соединена ребром.*

Теорема 2.2.2. *Граф K_5 не планарен.*

Определение 2.2.4. *Графом $K_{3,3}$ называется граф с шестью вершинами a_1, a_2, a_3, b_1, b_2, b_3, в котором каждая вершина a_i соединена ребром с каждой вершиной b_j и других ребер нет.*

Теорема 2.2.3. *Граф $K_{3,3}$ не планарен.*

Определение 2.2.5. *Подразделением ребра (a, b) называется операция, состоящая в следующих действиях:*

1. *удаление (a, b),*
2. *добавление новой вершины c,*
3. *добавление ребер (a, c) и (c, b).*

Определение 2.2.6. *Граф H называется подразделением графа G, если H можно получить из G путем конечного числа подразделений его ребер.*

Определение 2.2.7. *Два графа называются гомеоморфными, если существуют их подразделения, которые изоморфны.*

Теорема 2.2.4 (Понтрягина-Куратовского)**.** *Граф является планарным тогда и только тогда, когда он не содержит ни одного подграфа, гомеоморфного графам K_5 или $K_{3,3}$.*

2.3 Маршруты. Цепи. Циклы

Определение 2.3.1. *Маршрутом в графе $G = (V, E)$ называется любая последовательность вида $v_0e_1v_1e_2v_2 \dots e_kv_k$, где $v_i \in V$, $e_i \in E$.*

Число k называется длиной маршрута, а v_0 и v_k — это концевые вершины маршрута.

Определение 2.3.2. *Если $v_0 \neq v_k$, то маршрут называется открытым. Если же $v_0 = v_k$, маршрут называется замкнутым.*

Определение 2.3.3. *Маршрут называется цепью, если все его ребра различны.*

Определение 2.3.4. *Открытая цепь в графе называется путем.*

Определение 2.3.5. *Замкнутая цепь в графе называется циклом.*

Определение 2.3.6. *Если все вершины в цепи различны, то цепь называется простой.*

Аналогично определяются простой путь и простой цикл.

Замечание. Если в графе существует маршрут между v_0 и v_k, то существует и простой путь между этими вершинами.

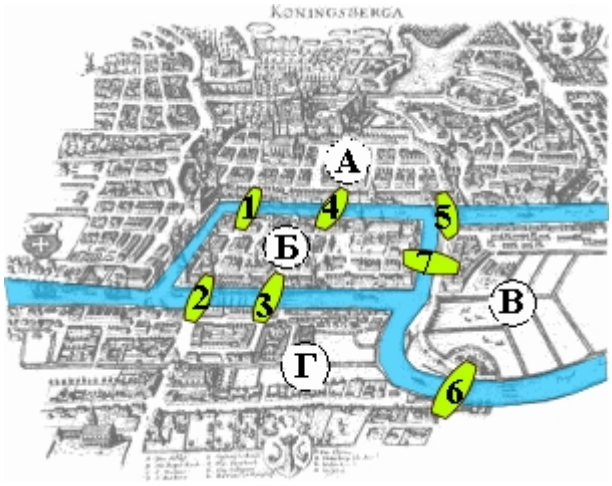
Определение 2.3.7. *Граф G называется связным графом, если в нем существует путь между каждой парой вершин.*

Если граф $G = (V, E)$ не является связным графом, то его можно единственным образом разбить на связные подграфы G_1, G_2, …, G_l, где $G_i = (V_i, E_i)$ и $V_i \subseteq V$, $E_i \subseteq E$, причем множества V_i попарно не пересекаются, а E_i содержат все ребра графа G, соединяющие вершины из V_i.

В этом случае подграфы G_1, G_2, …, G_l называются компонентами связности исходного графа G.

2.4 Эйлеровы графы

Пример. Задача о кенигсбергских мостах.



Старинная карта Кенигсберга.
Буквами обозначены части города:
А - Альтштадт, Б - Кнайпхоф,
В - Ломзе, Г - Форштадт.
Цифрами обозначены мосты
(в порядке строительства):
1 - Лавочный, 2 - Зелёный,
3 - Рабочий, 4 - Кузнечный,
5 - Деревянный, 6 - Высокий,
7 - Медовый.

Издавна среди жителей Кенигсберга была распространена такая загадка: как пройти по всем мостам, не проходя ни по одному из них дважды? Многие кенигсбержцы пытались решить эту задачу, как теоретически, так и практически, во время прогулок. Но никому это не удавалось, однако не удавалось и доказать, что это даже теоретически невозможно.

В 1736 году задача о семи мостах заинтересовала выдающегося математика, члена Петербургской академии наук Леонарда Эйлера, который смог найти правило, пользуясь которым легко определить, можно ли пройти по всем мостам, не проходя дважды ни по одному из них (в случае семи мостов Кенигсберга это невозможно).

На графе мостам соответствуют ребра графа, а частям города – вершины графа.

Определение 2.4.1. *Эйлеровым циклом в графе G называется цикл, содержащий все ребра графа G.*

Определение 2.4.2. *Эйлеровым графом называется граф, содержащий эйлеров цикл.*

Теорема 2.4.1. *Граф $G = (V, E)$ является эйлеровым тогда и только тогда, когда он связен и степень каждой его вершины четная.*

2.5 Гамильтоновы графы

Определение 2.5.1. *Гамильтоновым циклом в графе G называется простой цикл (без повторений вершин), содержащий все вершины графа G.*

Определение 2.5.2. *Гамильтоновым графом называется граф, содержащий гамильтонов цикл.*

В отличие от проверки наличия эйлерова цикла, для задачи поиска гамильтонова цикла не известно эффективного решения. Тем не менее, существует достаточное условие, позволяющее установить наличие гамильтонова цикла.

Теорема 2.5.1 (Теорема Дирака)**.** *Граф $G = (V, E)$ является гамильтоновым, если для любой вершины $v_i \in V$ выполняется $deg(v_i) \geq n/2$.*

2.6 Способы задания графов

Из определения графа следует, что каждый граф $G = (V, E)$ можно задать, непосредственно перечислив его множество вершин V и множество ребер E. Однако такое представление неудобно для решения многих задач о графах. Например, чтобы проверить наличие ребра между двумя вершинами, придется, вообще говоря, просмотреть все множество E. Хорошее представление, по крайней мере, должно позволять легко переходить от вершины к ее соседу и перечислять всех ее соседей.

Матрица (таблица) смежности.

Определение 2.6.1. *Матрицей смежности ориентированного (или неориентированного) графа $G = (V, E)$ с n вершинами $V = \{v_1, \ldots, v_n\}$ называется булева матрица A_G размера $n \times n$ с элементами*

$$a_{i,j} = \begin{cases} 1, & \text{если } (v_i, v_j) \in E \\ 0, & \text{в противном случае.} \end{cases}$$

Это представление позволяет легко проверять наличие ребер между заданными парами вершин. Для поиска всех соседей, в которые ведут ребра из вершины v_i, необходимо просмотреть соответствующую ей i-ю строку матрицы A_G, а чтобы найти вершины, из которых ребра идут в v_i, необходимо просмотреть ее i-ый столбец. Требуемая для A_G память – порядка n^2 битов – не может быть уменьшена для графов, у которых "много" ребер. Но для разреженных графов с числом ребер существенно меньшим по порядку n^2 в матрице смежности много "ненужных" нулей. Для таких графов более эффективными могут оказаться другие представления.

Матрица (таблица) инцидентности.

Определение 2.6.2. *Матрицей инцидентности ориентированного (или неориентированного) графа $G = (V, E)$ с n вершинами $V = \{v_1, \ldots, v_n\}$ и m ребрами $E = \{e_1, \ldots, e_m\}$ называется матрица B_G размера $n \times m$ с элементами*

$$b_{i,j} = \begin{cases} 1, \text{ если для некоторого } k \text{ ребро } (v_i, v_k) \in E \\ -1, \text{ если для некоторого } k \text{ ребро } (v_k, v_i) \in E \\ 2, \text{ если ребро } (v_i, v_i) \in E \\ 0, \text{ в противном случае.} \end{cases}$$

Таким образом, в матрице инцидентности B_G любому ребру $e_j = (v_i, v_k)$ соответствует j-ый столбец, в котором в i-ой строке стоит 1, а в k-ой – (-1). Ребра-петли выделяются числом 2. Для проверки наличия ребра между двумя вершинами v_i и v_k требуется просмотреть i-ю и k-ую строки B_G, поиск всех соседей вершины требует просмотра соответствующей строки. Если $m \gg n$, то это требует существенно больше времени, чем при использовании матрицы смежности. Поэтому при практическом решении задач на графах матрица инцидентности почти не используется.

Списки смежности.

Определение 2.6.3. *Пусть $G = (V, E)$ – ориентированный граф, v – вершина из V. Список смежности L_v для вершины v включает все смежные с ней вершины, т.е.*

$$L_v = w_1, w_2, \ldots, w_k, \text{ где } \forall\ i\ (v, w_i) \in E.$$

Представление графа $G = (V, E)$ с n вершинами $V = \{v_1, \ldots, v_n\}$ с помощью списков смежности состоит из списков смежности всех вершин: $L_{v_1}, L_{v_2}, \ldots, L_{v_n}$.

Размер этого представления сравним с суммой числа вершин и ребер графа. Оно позволяет легко переходить по ребрам от вершины к ее соседям. В программах списки смежности представляются списковыми структурами, которые легко реализуются во всех языках программирования.

2.7 Деревья

Определение 2.7.1. *Граф G называется ациклическим, если он не содержит циклов.*

Определение 2.7.2. *Деревом называется связный ациклический граф.*

Определение 2.7.3. *Деревом графа G называется связный ациклический подграф графа G.*

Определение 2.7.4. *Дерево графа G называется остовным деревом, если оно содержит все вершины графа G.*

Теорема 2.7.1. *Пусть $G = (V, E)$ – это граф, содержащий n вершин и m ребер. Следующие утверждения эквивалентны:*

1. *Граф G является деревом.*
2. *Между любой парой вершин в графе G существует единственный путь.*
3. *Граф G является связным графом и $m = n - 1$.*
4. *Граф G является ациклическим графом и $m = n - 1$.*
5. *G — ациклический граф, и при соединении любой пары вершин в графе новым ребром возникнет в точности 1 цикл.*

2.8 Примеры задач из теории графов

Задача коммивояжера (коммивояжер - бродячий торговец) является одной из самых известных задач комбинаторной оптимизации. Задача заключается в отыскании самого выгодного маршрута, проходящего через указанные города хотя бы по одному разу с последующим возвратом в исходный город. В условиях задачи указываются критерий выгодности маршрута (кратчайший, самый дешевый, совокупный критерий и т. п.) и соответствующие матрицы расстояний, стоимости и т. п. Как правило, указывается, что маршрут должен проходить через каждый город только один раз – в таком случае выбор осуществляется среди гамильтоновых циклов.

Для общей постановки задачи, впрочем как и для большинства ее частных случаев, не найдено эффективного решения.

Глава 3

Схемы из функциональных элементов

3.1 Основные определения. Примеры

Определение 3.1.1. *Вершины ориентированного графа, в которые не входит ни одной дуги, называются истоками.*

Определение 3.1.2. *Ориентированный граф называется ациклическим, если в нем нет ориентированных циклов.*

Определение 3.1.3. *В ациклическом ориентированном графе глубиной вершины v называется максимальное число дуг в ориентированном пути из какого-нибудь истока в вершину v.*

Если в ациклическом ориентированном графе есть дуга (v_1, v_2), то глубина v_2 больше глубины v_1.

Определение 3.1.4. *Систему $A = \{g_1, g_2, \ldots, g_m\}$, где все g_i – функции алгебры логики, будем называть базисом функциональных элементов.*

Определение 3.1.5. *Схемой из функциональных элементов в базисе A называется ациклический ориентированный граф, в котором:*

1. *каждому истоку приписана некоторая переменная, причем разным истокам приписаны разные переменные (истоки при этом называются входами схемы, а приписанные им переменные – входными переменными);*
2. *каждой вершине, в которую входят $k \geq 1$ дуг, приписана функция из базиса A, зависящая от k переменных (вершина с приписанной функцией при этом называется функциональным элементом);*
3. *некоторые вершины выделены как выходы (истоки одновременно могут являться выходами).*

Индукцией по глубине q вершины v определяется функция f_v, реализуемая в данной вершине. Если $q = 0$, то есть v – исток, и v приписана переменная x_i, то $f_v \equiv x_i$. Пусть реализуемые функции уже определены для всех вершин глубины меньшей, чем q_0, и глубина v равна q_0. Пусть в v входят дуги $e_1, e_2, \ldots, e_k$ из вершин $v_1, v_2, \ldots, v_k$ и в них реализуются функции $f_1, f_2, \ldots, f_k$. Пусть вершине v приписана функция $g(x_1, \ldots, x_k)$. Тогда в v реализуется функция $f_v = g(f_1, f_2, \ldots, f_k)$.

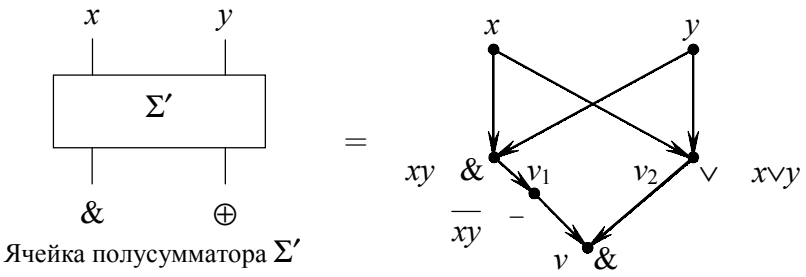
Другими словами, каждый вход функционального элемента – аргумент функции, выход – булева функция от аргументов. Возможные соединения функциональных элементов соответствуют булевым функциям и их суперпозициям.

Определение 3.1.6. *Будем говорить, что схема представляет систему функций, реализуемых в ее выходах.*

Определение 3.1.7. *Сложностью схемы из функциональных элементов называется число функциональных элементов в схеме.*

В дальнейшем по умолчанию будем подразумевать под базисом функциональных элементов систему $A = \{\&, \vee, \neg\}$. Очевидно, что с помощью такого базиса можно реализовать любую булеву функцию, т.к. система $\{\&, \vee, \neg\}$ функционально полна в P_2. Однако сложность реализации большинства функций имеет порядок $\frac{2^n}{n}$.

Пример. Полусумматор. Схема Σ' с двумя входами x и y и выходами $f_1 = \overline{xy}(x \vee y) = x \oplus y$ и $f_2 = x \cdot y$. Сложность (число элементов) полусумматора равна 4.



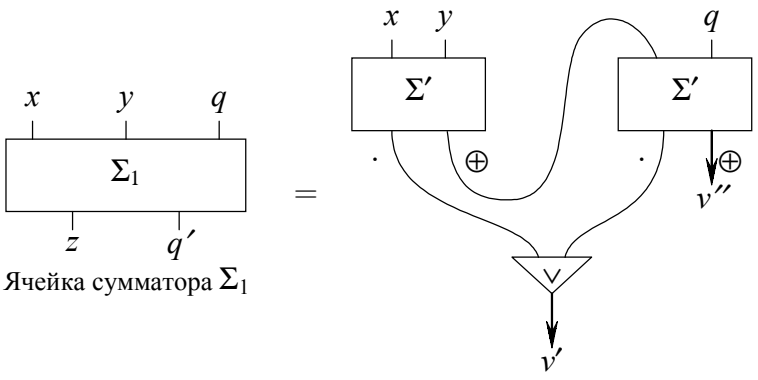
Ячейка полусумматора Σ'

Пример. Сумматор. Пусть есть 2 n-разрядных числа, и требуется найти их сумму (в дальнейших обозначениях x_i, y_i – разряды чисел, а q_i – биты переноса).

q_{n+1}	q_n	q_{n-1}	$\dots$	q_2	
	x_n	x_{n-1}	$\dots$	x_2	x_1
$+$	y_n	y_{n-1}	$\dots$	y_2	y_1
z_{n+1}	z_n	z_{n-1}	$\dots$	z_2	z_1

При $i = 2, 3, \dots, n$ задача решается системой функций

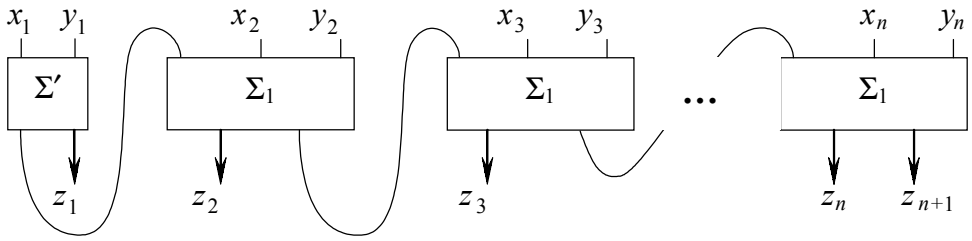
$$\begin{cases} z_i = x_i \oplus y_i \oplus q_i, \\ q_{i+1} = m(x_i, y_i, q_i). \end{cases}$$



Ячейка сумматора Σ_1

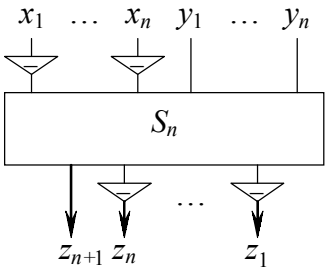
Сложность схемы, реализующей ячейку сумматора равна 9.

Очевидно, что $z_1 = x_1 \oplus y_1$, $q_2 = x_1 y_1$, $z_{n+1} = q_{n+1}$. Поэтому можно построить n-разрядный сумматор сложности $9n - 5$.



Сумматор S_n

Пример. Вычитатель. Используя соотношение $(\overline{x}_n, \overline{x}_{n-1}, \ldots, \overline{x}_1)_2 = 2^n - 1 - (x_n, x_{n-1}, \ldots, x_1)_2$, можно построить схему для двоичного вычитателя из $2n$ инверторов и 1 сумматора. Таким образом, сложность схемы для вычитателя равна $11n - 5$.



Вычитатель W_n

Глава 4

Основы теории кодирования

4.1 Алфавитное кодирование

Пусть A – конечное непустое множество символов. Будем называть такое множество алфавитом. Конечная последовательность символов из A называется словом в алфавите A. Обозначим через A^s множество всех слов в алфавите A, имеющих длину s, а через Λ – пустое слово. Введем также обозначение

$$A^* = \{\Lambda\} \cup A^1 \cup A^2 \cup \ldots \cup A^s \cup \ldots$$

Определение 4.1.1. *Пусть $A = \{a_1, a_2, \ldots, a_r\}$ – исходный алфавит, а $B = \{b_1, b_2, \ldots, b_q\}$ – кодирующий алфавит. Тогда однозначное отображение $\varphi : A^* \to B^*$ называется кодированием.*

Определение 4.1.2. *Алфавитным кодированием называется отображение $\varphi : A \to B^*$ такое, что $\varphi : a_i \to B_i$, при этом $\varphi : a_{i_1}a_{i_2}\ldots a_{i_s} \to B_{i_1}B_{i_2}\ldots B_{i_s}$. Множество $\{B_1, B_2, \ldots, B_r\}$ при этом называется множеством кодовых слов (или просто кодом).*

Определение 4.1.3. *Кодирование $\varphi : A^* \to B^*$ называется взаимно-однозначным (декодируемым, разделимым), если для любых слов $\alpha_1, \alpha_2 \in A^*$ из $\alpha_1 \neq \alpha_2$ следует, что $\varphi(\alpha_1) \neq \varphi(\alpha_2)$.*

Определение 4.1.4. *Кодирование (код) называется равномерным, если длины всех его кодовых слов одинаковы.*

Лемма 4.1.1. *Любой равномерный код без повторяющихся кодовых слов является взаимно однозначным.*

Определение 4.1.5. *Код называется префиксным, если никакое кодовое слово не является началом другого.*

Лемма 4.1.2. *Любое префиксное кодирование является взаимно однозначным.*

Определение 4.1.6. *Код называется суффиксным (постфиксным), если никакое кодовое слово не является концом другого.*

Лемма 4.1.3. *Любое суффиксное кодирование является взаимно однозначным.*

Заметим, что взаимная однозначность кодирования означает выполнение условий определения 4.1.3 для слов произвольной длины, поэтому тривиальная проверка декодируемости (перебором исходных сообщений) является потенциально бесконечной.

Однако, теорема Маркова позволяет произвести проверку однозначности на конечном множестве.

4.1.1 Теорема Маркова

Пусть исходный алфавит A состоит из r букв, а кодирующий алфавит B содержит q символов. Рассмотрим некоторое алфавитное кодирование $\varphi : A \to B^*$, заданное множеством кодовых слов $\{B_1, B_2, \ldots, B_r\}$.

Обозначим через l_i длину слова B_i и пусть $L = \sum_{i=1}^{r} l_i$.

Рассмотрим все возможные разбиения кодовых слов

$$B_i = \beta' B_{j_1} B_{j_2} \ldots B_{j_w} \beta'',$$

где B_{j_k} – элементарные коды, а β', β'' – некоторые слова в алфавите B (возможно пустые), не являющиеся кодовыми словами, причем никакой элементарный код не является префиксом β' или суффиксом β''.

Пусть $W = \max w$.

Справедлива следующая теорема.

Теорема 4.1.1. *Однозначность кодирования φ может быть проверена на множестве исходных сообщений длины не более N, где*

$$N \le \left\lceil \frac{(W+1)(L-r+2)}{2} \right\rceil.$$

Эта теорема позволяет построить практический алгоритм распознавания однозначности.

Алгоритм распознавания однозначности кодирования.

1. Для каждого элементарного кода B_i рассматриваются все разбиения вида

$$B_i = \beta' B_{j_1} B_{j_2} \ldots B_{j_w} \beta''.$$

2. Строится множество $S(\Sigma) = \{\Lambda, \beta_1, \ldots, \beta_t\}$, где β_j – это некоторые слова в алфавите B, которые являются одновременно началом какого-то нетривиального разбиения и окончанием другого.

3. Строится граф $G(\Sigma)$, вершинами которого являются элементы множества $S(\Sigma)$: если существует разложение

$$B_i = \beta' B_{j_1} B_{j_2} \ldots B_{j_w} \beta'',$$

то вершины β' и β'' соединяются ориентированным ребром, которое помечается цепочкой $B_{j_1} B_{j_2} \ldots B_{j_w}$.

Построенный граф связан с исходным кодированием следующим образом.

Теорема 4.1.2. *Алфавитное кодирование со схемой Σ является неоднозначным тогда и только тогда, когда граф $G(\Sigma)$ содержит ориентированный цикл, проходящий через вершину Λ.*

4.1.2 Неравенство Макмиллана

Пусть исходный алфавит A состоит из r букв, а кодирующий алфавит B содержит q символов. Рассмотрим некоторое алфавитное кодирование $\varphi : A \to B^*$, заданное множеством кодовых слов $\{B_1, B_2, \ldots, B_r\}$. Обозначим через l_i длину слова B_i.

Справедлива следующая теорема.

Теорема 4.1.3. *Если φ является взаимно-однозначным кодированием, то*

$$\sum_{i=1}^{r} \frac{1}{q^{l_i}} \leq 1.$$

Замечание. Обратное, вообще говоря, не верно.

На основе теоремы Макмиллана можно доказать следующее утверждение.

Теорема 4.1.4. *Если код $\{B_1, B_2, \ldots, B_r\}$ является взаимно-однозначным, то существует префиксный код $\{B'_1, B'_2, \ldots, B'_r\}$, причем $l(B_i) = l(B'_i)$.*

Данная теорема имеет важное следствие.

Следствие 4.1.1. *Неравенство Макмиллана является необходимым и достаточным условием существования префиксного кода с заданными длинами кодовых слов.*

Глава 5

Ограниченно-детерминированные функции

5.1 Детерминированные функции

Термин "детерминированный" означает однозначную определенность. Детерминированность в решении какой-либо практической задачи или в алгоритме означает, что способ решения задачи определен однозначно в виде последовательности шагов. На любом шаге не допускаются никакие двусмысленности или неопределенности.

Введем формальное определение *детерминированной функции*.

Пусть A – непустой конечный алфавит. Его элементы называются буквами (или символами). Словом длины s в алфавите A называется последовательность вида $x(1)x(2)\dots x(s)$, составленная из букв алфавита A (здесь $s \geq 1$). Кратко эта последовательность обозначается так: x^s. Множество всех слов длины s в алфавите A ($s \geq 1$) обозначается через A^s.

Бесконечные последовательности $x^\omega = x(1)x(2)\dots x(t)\dots$, составленные из букв алфавита A, называются бесконечными словами в алфавите A. Множество всех бесконечных слов в алфавите A обозначается через A^ω.

Слово α, получающееся приписыванием слова α_2 справа к конечному слову α_1, называется соединением слов α_1 и α_2 и обозначается через $\alpha_1\alpha_2$. Слово α_1 называют при этом префиксом (или началом), а слово α_2 – суффиксом (или окончанием).

Пусть A и B – конечные непустые алфавиты. Будем рассматривать отображения вида $f : A^\omega \to B^\omega$. Алфавиты A и B называют соответственно входным и выходным алфавитами отображения f, слова (последовательности) из множества A^ω называют входными, а из множества B^ω – выходными.

Определение 5.1.1. *Отображение $f : A^\omega \to B^\omega$ называется детерминированной функцией (сокращенно: д. функцией), если оно удовлетворяет следующему условию: для всякого $s \geq 1$ и любого входного слова x^ω (из A^ω) s-й символ выходного слова $y^\omega = f(x^\omega)$ является однозначной функцией первых s символов слова x^ω.*

Примеры. Инверсия, конъюнкция, сложение, умножение, константа.

Понятие д. функции от n аргументов возникает естественным образом при рассмотрении д. функций вида $f : (A_1 \times A_2 \times \dots \times A_n)^\omega \to B^\omega$.

Иногда бывает удобно считать, что д. функция $f : A^\omega \to B^\omega$ реализуется некоторым дискретным устройством (автоматом), работающим в дискретные моменты времени $t = 1, 2, \dots$. На вход этого устройства в каждый момент t подается сигнал $x(t)$, а на выходе

появляется сигнал $y(t)$. Если $A = A_1 \times A_2 \times \ldots \times A_n$ и $B = B_1 \times B_2 \times \ldots \times B_m$, то можно считать, что автомат реализует m детерминированных функций, каждая из которых зависит от n переменных.

5.2 Представление д. функций деревьями

При описании д. функций бывает удобно пользоваться теоретико-графовым языком, а именно бесконечными информативными деревьями. Пусть A – алфавит, состоящий из r букв ($r \geq 1$).

Построим бесконечное ориентированное корневое дерево, удовлетворяющее следующим условиям:

1. из каждой вершины дерева, включая и корень, выходят ровно r дуг (т.е. ориентированных ребер);

2. в каждую вершину дерева, отличную от корня, входит только одна дуга, а в корень дерева не входит ни одной;

3. каждой дуге дерева приписана некоторая буква алфавита A, причем разным дугам, выходящим из одной и той же вершины дерева (в частности, из корня), приписаны разные буквы.

Ветвями дерева назовем связанные множества дуг, исходящие из корневой вершины. Заметим, что каждой бесконечной ветви в дереве соответствует вполне определенное слово из множества A^ω.

Припишем каждой дуге некоторую букву из алфавита B. Таким образом, всякой ветви в дереве отвечает слово из множества B^ω, составленное из букв, приписанных дугам этой ветви. Поэтому можно считать, что такое дерево задает (реализует) вполне определенное отображение $f : A^\omega \to B^\omega$.

Очевидно, что для любой д. функции реализующее ее дерево строится однозначно.

5.3 Вес д. функции. Ограниченно-детерминированные функции

Поддеревом назовем некоторую вершину дерева вместе со всем, что из нее "растет". Нетрудно видеть, что любое поддерево в дереве для д. функции также задает некоторую д. функцию.

Определение 5.3.1. *Д. функции f_1 и f_2 называются различимыми, если существует такое слово α^s ($s \geq 1$), что $f_1(\alpha^s) \neq f_2(\alpha^s)$. Если же $f_1(\alpha^s) = f_2(\alpha^s)$ для любых слов α^s ($s = 1, 2, \ldots$), т.е. если равенство $f_1(x^\omega) = f_2(x^\omega)$ выполняется при всяком входном слове x^ω, то функции f_1 и f_2 называются эквивалентными (или неразличимыми) д. функциями.*

Заметим, что эквивалентные д. функции задаются одинаковыми (эквивалентными) деревьями.

Указанное отношение эквивалентности позволяет разбить множество всех вершин дерева на классы эквивалентности.

Определение 5.3.2. *Количество попарно неэквивалентных вершин в дереве д. функции называется весом дерева.*

Учитывая то, что д. функция однозначно задается своим деревом, вес дерева называется также *весом д. функции.*

Примеры. Отрицание, конъюнкция, сложение.

Рассмотрим другой подход, приводящий к эквивалентному определению веса д. функции.

Определение 5.3.3. *Пусть f и g – д. функции. Если существует такое слово α^s, что $f(\alpha^s x^\omega) = f(\alpha^s)g(x^\omega)$ для любого слова $x^\omega \in A^\omega$, то функция g называется остаточной функцией функции f, порожденной словом α^s, и обозначается через f_{α^s}.*

Заметим, что остаточной функции f_{α^s} соответствует поддерево в дереве для f, растущее из такой вершины, в которой оканчивается последовательность ребер, исходящая из корня и соответствующая слову α^s.

Если остаточные функции g_1 и g_2 эквивалентны, то соответствующие им вершины и растущие из этих вершин поддеревья также эквивалентны, т.е. относятся к одному классу эквивалентности.

Определение 5.3.4. *Множество всех остаточных функций функции f, эквивалентных функции f_{α^s}, образует класс эквивалентности, называемый состоянием функции f, содержащим остаточную функцию f_{α^s}.*

Другими словами, состояния функции f описывают классы эквивалентности в дереве для f. Поэтому нетрудно видеть, что следующее определение эквивалентно понятию веса дерева.

Определение 5.3.5. *Количество попарно различных состояний называется весом функции.*

Вес д. функции можно интерпретировать как размер необходимой памяти, требуемой для программной реализации д. функции. Если вес бесконечен, то такую функцию не удастся реализовать программно – потребуется “бесконечная память”.

Пример. Д. функция $f(\alpha(1)\alpha(2)\alpha(3)\ldots) = \alpha(1)\alpha(1)\alpha(2)\alpha(2)\alpha(3)\alpha(3)\ldots$ имеет бесконечный вес.

$f(\overbrace{00\ldots0}^{k}x^\omega) = \overbrace{00\ldots0}^{k}g_k(x^\omega)$

$f(\underbrace{00\ldots00}_{k+1}x^\omega) = \underbrace{00\ldots00}_{k+1}g_{k+1}(x^\omega)$

Заметим, что все g_k не эквивалентны между собой, т.к., например, $g_k(\underbrace{1\ldots11}_{k+1}) = \underbrace{0\ldots0}_{k}1$, а $g_{k+1}(\underbrace{1\ldots11}_{k+1}) = \underbrace{0\ldots00}_{k+1}$. При этом все вершины самой левой ветви в дереве для f не эквивалентны между собой.

Определение 5.3.6. *Функция f называется ограниченно-детерминированной (сокращенно о.-д. функцией), если она имеет конечный вес.*

5.4 Способы задания о.-д. функций

Усеченное дерево. Вес всякой о.-д. функции конечен, поэтому для ее задания можно ограничиться не полным, а усеченным деревом, оставив лишь часть без повторений и пометив номер класса эквивалентности каждой вершины. Такого усеченного дерева достаточно, чтобы восстановить полное дерево.

Диаграмма Мура. Пусть $Q = \{q_0, q_1, \ldots, q_v\}$ – множество всех состояний о.-д. функции $f : A^\omega \to B^\omega$. Сопоставим функции f ориентированный граф с w вершинами, помеченными различными состояниями из множества Q. Будем соединять дугой вершину q_i с q_j тогда и только тогда, когда дуга, соединяющая q_i с q_j, присутствует в дереве для f, причем будем помечать дуги в графе так же, как они помечены в дереве. Дополнительно отметим звездочкой вершину графа, соответствующую корню дерева для f.

Построенный граф называется диаграммой Мура для о.-д. функции f. Неформально, диаграмма Мура есть по-другому нарисованное усеченное дерево – ее легко получить, "склеив" вершины дерева, соответствующие одному и тому же состоянию.

Пример. Конъюнкция, сложение.

Заметим, что диаграмма Мура описывает работу некоторого устройства (автомата). На каждом такте работы поведение автомата определяется текущим состоянием и входным сигналом – автомат вырабатывает выходной сигнал и переходит в новое состояние.

Канонические таблицы и канонические уравнения. Все дуги диаграммы Мура ведут из некоторого состояния $q(t-1)$ в состояние $q(t)$ и помечены $x(t), y(t)$. Они описывают следующий процесс: если текущим состоянием является $q(t-1)$ и на вход поступил $x(t)$, то автомат переходит в состояние $q(t)$, а на выход подается $y(t)$.

Этот процесс можно описать, связав с диаграммой Мура две функции:

- $F : A \times Q \to B$ – функция выходов,
- $G : A \times Q \to Q$ – функция переходов.

Система уравнений

$$\begin{cases} y(t) = F(x(t), q(t-1)), \\ q(t) = G(x(t), q(t-1)), \\ q(0) = q_0, \end{cases}$$

называется каноническими уравнениями функции f с начальным состоянием q_0.

Для о.-д. функции f отображения $F(x(t), q(t-1))$ и $G(x(t), q(t-1))$ и аргументы, от которых они зависят, принимают конечное число значений, поэтому возможно табличное задание о.-д. функции с помощью так называемой канонической таблицы. Отметим, что каждой дуге диаграммы Мура соответствует единственная строка канонической таблицы.

Пример. Сложение.

Функции F и G являются, вообще говоря, частичными, т. е. не всюду определенными. Обычно их доопределяют так, чтобы правые части уравнений имели по возможности более простой вид.

Пример. Конъюнкция.

Чтобы построить систему канонических уравнений по диаграмме Мура, сначала строят каноническую таблицу, а затем описывают функции F и G с помощью булевских формул (например, с помощью СДНФ, СКНФ или полинома Жегалкина).

Схемы из функциональных элементов. Т.к. система канонических уравнений представляет собой совокупность булевых функций, о.-д. функции также можно задавать схемами из функциональных элементов для булевых функций. Чтобы реализовать временной аспект работы автомата в модель схем из функциональных элементов дополнительно вводится элемент единичной задержки, реализующий следующую о.-д. функцию:

$$\begin{cases} y(t) = q(t-1), \\ q(t) = q(t-1), \\ q(0) = q_0. \end{cases}$$

В схеме из функциональных элементов и элементов задержки допускаются ориентированные циклы, но любой ориентированный цикл должен проходить хотя бы через одну задержку.

Считаем, что в каждый момент времени $t = 1, 2, \ldots$ на вход схемы поступает $x(t)$, а на выход выдается значение $y(t)$. В начальный момент времени на вход $q(t-1)$ подается значение q_0, а затем туда через элемент единичной задержки поступает новое значение функции переходов $q(t) = G(x(t), q(t-1))$.

Пример. Сложение.

Глава 6

Алгоритмы и вычислимость

6.1 Понятие алгоритма

Под алгоритмом, как правило, понимают последовательность действий, приводящую к конечному результату. Однако такое определение не является строго математическим, что необходимо для построения формальных теорий.

Понятие алгоритма было строго формализовано Аланом Тьюрингом, который ввел в рассмотрение математическую модель под названием *машина Тьюринга* – абстрактное вычислительное устройство, напоминающее современный компьютер с упрощенным набором элементарных операций и идеализированной бесконечной памятью.

Машина Тьюринга M состоит из бесконечной в обе стороны ленты, головки чтения/записи, перемещающейся по ленте, и управляющего устройства.

Лента разбита на ячейки, в каждой из которых в любой дискретный момент времени записан один из символов *внешнего алфавита* $\Sigma = \{a_0, a_1, \ldots, a_r\}$. Символ $a_0 = \Lambda$ называется *пустым*. Если все ячейки справа или слева от текущей содержат пустой символ, то эта часть ленты называется пустой.

Головка в каждый момент времени обозревает одну из ячеек ленты и за один такт может сдвигаться на одну ячейку влево или вправо.

Управляющее устройство в каждый момент находится в одном из *состояний* из некоторого конечного множества $Q = \{q_0, q_1, \ldots, q_l\}$. Состояние q_1 называется *начальным* – в нем машина начинает свою работу, а q_0 называется *заключительным* состоянием – при переходе в это состояние машина завершает свою работу.

Машина работает по программе P, состоящей из последовательности инструкций вида

$$P_k : q_i a_j \to a'_j q'_i \left\{ \begin{array}{c} L \\ R \\ S \end{array} \right\}.$$

Инструкция P_k выполняется следующим образом: если машина M в текущий момент времени находится в состоянии q_i, на ленте в текущей ячейке находится символ a_j, то вместо него на ленту записывается символ a'_j, машина переходит в состояние q'_i, а головка сдвигается на одну ячейку влево (L), вправо (R) или остается на месте (S).

Таким образом, любая машина Тьюринга полностью описывается пятеркой вида

$$M =< \Sigma, Q, P, q_0, q_1 > .$$

Определение 6.1.1. *Конфигурация машины Тьюринга – это конечное слово, описывающее содержимое непустой части ленты, положение головки на ленте и текущее состояние.*

Если текущим состоянием является q_1, то такая конфигурация называется *начальной*, а если q_0, то *конечной*.

Таким образом, машина Тьюринга, получая на вход некоторое слово, начинает свою работу в некоторой начальной конфигурации и работает по тактам – выполняет очередную инструкцию и переходит в новую конфигурацию. Если машина оказывается в заключительной конфигурации, то завершает свою работу.

В некоторых случаях машина может работать бесконечно долго. Тогда говорят, что машина *зацикливается*.

Очевидно, что каждая машина Тьюринга описывает выполнение некоторого алгоритма в терминах набора элементарных команд. С другой стороны, как будет показано ниже, любой алгоритм можно описать некоторой машиной Тьюринга. Другими словами, модель машины Тьюринга формализует понятие алгоритма, неформально: "алгоритм = машина Тьюринга".

6.2 Вычислимость и тезис Тьюринга-Черча

Определение 6.2.1. *Говорят, что машина Тьюринга M вычисляет некоторую функцию $f : U \to V$, если для любого $u \in U$ машина M, начиная работу в начальной конфигурации со словом u на ленте, через конечное число шагов переходит в заключительную конфигурацию со словом v на ленте, где $v = f(u)$. Если функция f не определена на u, то машина M зациклится на слове u (будет работать бесконечно долго).*

Если существует машина Тьюринга, вычисляющая некоторую функцию f, то такая функция называется *вычислимой по Тьюрингу*.

Несмотря на видимую простоту, машина Тьюринга является мощным вычислительным средством. Более того, все, что можно вычислить на современном компьютере, можно реализовать и на машине Тьюринга. В целом, вычислительные возможности машин Тьюринга описываются следующим тезисом.

Тезис Тьюринга-Черча. Любая интуитивно вычислимая функция вычислима по Тьюрингу.

Данное предложение нельзя строго доказать или опровергнуть, т.к. в нем присутствует неформальное понятие "интуитивно вычислимая функция", описывающее функции, вычислимые хоть по какому-то алгоритму. Тем не менее, существует масса доводов в пользу этого утверждения. Более того, весь накопленный математический опыт подтверждает этот тезис.