

1 Применение графов для представления булевых функций

В этой главе мы свяжем два основных предыдущих раздела нашего курса: булевы функции и графы. Мы рассматривали два основных представления булевых функций: табличное и с помощью формул общего вида или формул специального вида, в частности, дизъюнктивных или конъюнктивных нормальных форм и многочленов Жегалкина. К сожалению, эти способы не позволяют эффективно представлять функции от большого числа переменных: таблица для функции от n переменных всегда содержит 2^n строк, многочлен Жегалкина может включать до 2^n слагаемых (и для большинства функций по порядку столько и включает). Такие представления нельзя реализовать на практике уже для n порядка нескольких десятков. Могло показаться, что сокращенные ДНФ, которые мы научились эффективно строить с помощью метода Блейка, и минимальные ДНФ, которые можно получить, удаляя из сокращенных «лишние» конъюнкции (впрочем, хороший алгоритм для такого удаления неизвестен), дают существенно более экономные представления булевых функций. Но в общем случае это не так. Для большинства булевых функций от n переменных минимальные ДНФ имеют экспоненциальный от n размер. В качестве примера конкретной простой функции с длиной ДНФ можно рассмотреть линейную функцию, определяющую нечетность суммы аргументов: $odd(X_1, X_2, \dots, X_n) = X_1 + X_2 + \dots + X_n$.

Задача 1. Докажите, что совершенная, сокращенная и минимальная ДНФ функции $odd(X_1, X_2, \dots, X_n)$ совпадают и состоят из 2^{n-1} элементарных конъюнкций длины n .

Те два представления булевых функций, которые мы рассматриваем в этом разделе: *логические схемы* (схемы из функциональных элементов) и *упорядоченные бинарные диаграммы решений* (УБДР), в общем случае тоже имеют экспоненциальные размеры от числа аргументов представляемых ими функций. Но во многих ситуациях они позволяют построить достаточно компактные представления естественно возникающих на практике булевых функций от сотен и даже тысяч аргументов.

1.1 Логические схемы (схемы из функциональных элементов)

Многие элементы в современной электронике являются устройствами, преобразующими некоторые входные сигналы (данные) в выходные. *Логические схемы*, в отечественной литературе чаще называемые *схемами из функциональных элементов*, представляют собой математическую модель таких устройств, в которых временем выполнения преобразования входов в выходы можно пренебречь.

1.1.1 Основные определения

Чтобы не усложнять определение, зафиксируем конкретный базис $B_0 = \{\wedge, \vee, \neg\}$ и определим схемы в этом базисе.

Определение 1. *Схемой из функциональных элементов в базисе B_0 называется размеченный ориентированный граф без циклов $S = (V, E)$, в котором*

- 1) *вершины, в которые не входят ребра, называются входами схемы и каждая из них помечена некоторой переменной (разным вершинам соответствуют разные переменные);*
- 2) *в каждую из остальных вершин входит одно или два ребра; вершины, в которые входит одно ребро помечены функцией \neg , а вершины, в которые входят по два ребра, — одной из функций \wedge или \vee . Такие вершины называются функциональными элементами.*

Как и для деревьев, для ориентированных графов без циклов можно естественным образом ввести понятие глубины.

Определение 2. Глубина вершины $v \in V$ в схеме $S = (V, E)$ — это максимальная длина пути из входов S в v .

Глубиной $D(S)$ схемы S назовем максимальную из глубин ее вершин.

Пусть входы схемы S помечены переменными x_1, \dots, x_n . С каждой вершиной $v \in V$ схемы S свяжем булеву функцию $f_v(x_1, \dots, x_n)$, реализуемую в этой вершине. Определим f_v индукцией по глубине v .

Определение 3. Базис: v имеет глубину 0. Тогда это входная вершина, которая помечена некоторой переменной x_i . Положим $f_v(x_1, \dots, x_n) = x_i$.

Шаг индукции. Пусть всем вершинам w глубины $\leq k$ уже сопоставлены функции f_w и пусть v — произвольная вершина глубины $k + 1$. Тогда

а) если v помечена \neg и в нее входит ребро (w, v) , то положим

$$f_v(x_1, \dots, x_n) = \neg f_w(x_1, \dots, x_n);$$

б) если v помечена \wedge и в нее входят два ребра (w_1, v) и (w_2, v) , то положим

$$f_v(x_1, \dots, x_n) = f_{w_1}(x_1, \dots, x_n) \wedge f_{w_2}(x_1, \dots, x_n);$$

в) если v помечена \vee и в нее входят два ребра (w_1, v) и (w_2, v) , то положим

$$f_v(x_1, \dots, x_n) = f_{w_1}(x_1, \dots, x_n) \vee f_{w_2}(x_1, \dots, x_n).$$

Нетрудно понять, что шаг индукции в этом определении корректен, так как, если в схеме S имеется ребро (w, v) и глубина вершины v равна $k + 1$, то глубина вершины w не превосходит k и для нее f_w уже определена по индукционному предположению.

Определение 4. Схема S реализует набор булевых функций g_1, g_2, \dots, g_m , если для каждого $i \in [1, m]$ в схеме существует такая вершина v_i , что $f_{v_i} = g_i$.

Замечание. Определение схем из функциональных элементов естественным образом можно распространить и на другие базисы. При этом, однако, надо для вершин, помеченных несимметричными функциями (например, импликацией) явно нумеровать входящие в них ребра, указывая, каким аргументам они соответствуют.

Определение 5. Сложность $L(S)$ схемы S — это число функциональных элементов в S . Сложность $L(f)$ булевой функции $f(x_1, \dots, x_n)$ — это наименьшая из сложностей схем, реализующих эту функцию.

Отношения между булевыми функциями и схемами естественно приводят к двум следующим основным проблемам.

Проблема анализа: по заданной схеме из функциональных элементов и выделенному подмножеству ее выходных вершин определить булевы функции, реализуемые в этих вершинах.

Проблема синтеза: по некоторому описанию булевой функции построить схему из функциональных элементов, реализующую эту функцию. При решении проблемы синтеза для исходной функции часто стараются построить схему минимальной или почти минимальной сложности.

Пример. Рассмотрим следующую схему S_1 с тремя входными переменными x, y, z :

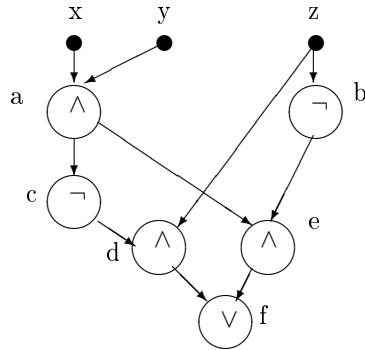


Схема S_1

Решим для этой схемы проблему анализа. В соответствии с данным выше определением вершины схемы S_1 реализуют следующие функции:

$f_a(x, y, z) = x \wedge y$, $f_b(x, y, z) = \neg z$, $f_c(x, y, z) = \neg f_a(x, y, z) = \neg(x \wedge y)$, $f_d(x, y, z) = f_c(x, y, z) \wedge z = \neg(x \wedge y) \wedge z$, $f_e(x, y, z) = f_a(x, y, z) \wedge f_b(x, y, z) = x \wedge y \wedge \neg z$ и, наконец, $f_f(x, y, z) = f_d(x, y, z) \vee f_e(x, y, z) = (\neg(x \wedge y) \wedge z) \vee ((x \wedge y) \wedge \neg z)$.

Глубина этой схемы $D(S_1) = 4$, а ее сложность $L(S_1) = 6$. В то же время формула для результирующей функции f_f содержит 7 функциональных знаков. За счет чего достигнута экономия? За счет того, что функция $(x \wedge y)$ в схеме S_1 вычисляется один раз в вершине a , а в формуле приходится вычислять ее дважды.

В этом и состоит основное преимущество вычислений булевых функций схемами: *каждую подформулу (подфункцию) достаточно вычислить один раз*, а затем полученное значение можно использовать сколько угодно раз в качестве аргумента для других подфункций.

1.1.2 Схемы и линейные программы

Указанное выше свойство характерно и для программ, в которых один раз вычисленное значение выражения можно использовать неоднократно. Рассмотрим один из простейших классов программ — *линейные или неветвящиеся программы*. Такие программы представляют последовательности присваиваний вида:

$X = F(X_1, \dots, X_k)$, где X, X_1, \dots, X_k — переменные, F — имя k -местной базисной функции.

В случае нашего базиса $B_0 = \{\wedge, \vee, \neg\}$ линейная программа состоит из присваиваний вида: $Z = X \wedge Y$, $Z = X \vee Y$ и $Z = \neg X$. Линейная программа P с выделенными входными переменными X_1, \dots, X_n порождает для каждого набора $\sigma_1, \dots, \sigma_n$ значений входных переменных естественный процесс вычисления: вначале переменным X_1, \dots, X_n присваиваются значения $\sigma_1, \dots, \sigma_n$, соответственно, а каждой из остальных переменных присваивается значение 0. Затем последовательно выполняются присваивания программы P , в результате чего каждая из переменных Z программы получит заключительное значение $P_Z(\sigma_1, \dots, \sigma_n)$.

Скажем, что программа P со входными переменными X_1, \dots, X_n вычисляет в выходной переменной Z функцию $F(X_1, \dots, X_n)$, если для любого набора значений входов $\sigma_1, \dots, \sigma_n$ после завершения работы $P_Z(\sigma_1, \dots, \sigma_n) = F(\sigma_1, \dots, \sigma_n)$.

Между схемами и линейными программами имеется тесная связь.

Теорема 1.

- (1) По каждой схеме из функциональных элементов S со входами x_1, \dots, x_n и функциональными элементами v_1, \dots, v_m можно эффективно построить линейную программу P_S со входными переменными x_1, \dots, x_n и рабочими переменными v_1, \dots, v_m , которая в любой переменной $v_i, i = 1, \dots, m$, вычисляет функцию $f_{v_i}(x_1, \dots, x_n)$.
- (2) По каждой линейной программе P со входными переменными X_1, \dots, X_n , вычисляющей в выходной переменной Z некоторую функцию $F(X_1, \dots, X_n)$ можно эффективно построить схему из функциональных элементов S_P со входами X_1, \dots, X_n , в которой имеется вершина v такая, что $f_v((X_1, \dots, X_n)) = F(X_1, \dots, X_n)$.

Доказательство. (1) Пусть S — схема со входами x_1, \dots, x_n и функциональными элементами v_1, \dots, v_m . Построим по ней линейную программу P_S со входными переменными x_1, \dots, x_n следующим образом. Упорядочим все входные и функциональные вершины S по глубине (вершины одной глубины в любом порядке): u_1, \dots, u_{n+m} . Программа P_S будет последовательностью m присваиваний.

- а) Пусть вершина u_{n+i} помечена \neg и в нее входит ребро из u_j . Тогда в качестве i -ой команды поместим в P_S присваивание $u_{n+i} = \neg u_j$.
- б) Пусть вершина u_{n+i} помечена $\circ \in \{\wedge, \vee\}$ и в нее входят ребра из u_j и u_k . Тогда в качестве i -ой команды поместим в P_S присваивание $u_{n+i} = u_j \circ u_k$.

Упорядочение вершин по глубине гарантирует, что $j < n + i$ и $k < n + i$. Поэтому при вычислении u_{n+i} значения аргументов уже получены и индукцией по глубине легко показать, что для каждого $i = 1, \dots, m$ программа P_S вычисляет в переменной v_i функцию $f_{v_i}(x_1, \dots, x_n)$.

Задача 2. Докажите пункт (2) теоремы.

Применим конструкцию теоремы к схеме S_1 . Ее вершины можно упорядочить по глубине так: $x, y, z, a, b, c, d, e, f$. Порождая команды по описанным выше правилам, получим следующую линейную программу P_{S_1} :

$a = x \wedge y;$
 $b = \neg z;$
 $c = \neg a;$
 $d = c \wedge z;$
 $e = a \wedge b;$
 $f = d \vee e$

Замечание. Число команд в линейной программе P_S , т.е. время ее выполнения, совпадает со сложностью $L(S)$ схемы S . Глубина схемы $D(S)$ также имеет смысл с точки зрения времени

вычисления. Именно, $D(S)$ — это время выполнения P_S на многопроцессорной системе. Действительно, все команды, соответствующие вершинам одинаковой глубины, можно выполнять параллельно, так как результаты любой из них не используются в качестве аргументов другой.

1.1.3 Сложение по модулю 2

Рассмотрим следующую схему S_+ :

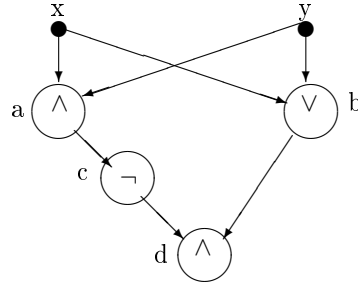


Схема S_+ для функции $x + y$

В соответствии с определением вершины этой схемы реализуют следующие функции:
 $f_a(x, y) = x \wedge y$, $f_b(x, y) = x \vee y$, $f_c(x, y) = \neg f_a(x, y) = \neg(x \wedge y)$, $f_d(x, y) = f_c(x, y) \wedge f_b(x, y) = \neg(x \wedge y) \wedge (x \vee y) = x + y$.

Таким образом, схема S_+ реализует (в вершине d) функцию $+$ сложения по модулю 2.

Из приведенного выше примера следует, что $L(S_+) = 4$ и $L(+) \leq 4$.

Задача 3. Докажите, что $L(+) = 4$.

Используя схему S_+ нетрудно построить схему S_{odd} для реализации линейной функции $odd(X_1, X_2, \dots, X_n) = X_1 + X_2 + \dots + X_n$.

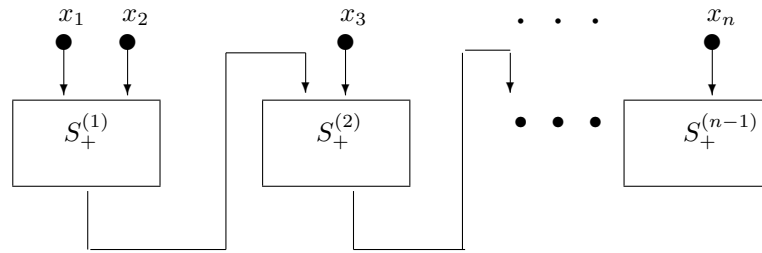


Схема S_{odd}

На этой схеме прямоугольники $S_+^{(1)}, S_+^{(2)}, \dots, S_+^{(n)}$ содержат копии схемы S_+ . При этом входами $S_+^{(1)}$ являются переменные x_1 и x_2 , а входами $S_+^{(i+1)}$ являются выход схемы $S_+^{(i)}$ и переменная x_{i+1} . По индукции легко показать, что вершина d в $S_+^{(i)}$ реализует функцию $(x_1 + x_2 + \dots + x_{i+1})$. Таким образом, нами установлена

Теорема 2. Существует схема S_{odd} , реализующая функцию $odd(X_1, X_2, \dots, X_n) = X_1 + X_2 + \dots + X_n$ со сложностью $L(S_{odd}) = 4(n - 1)$.

1.1.4 Сумматор

Сумматором порядка n называют схему, вычисляющую результат сложения двух n -разрядных двоичных чисел $a = \sum_{i=0}^{n-1} a_i 2^i$ и $b = \sum_{i=0}^{n-1} b_i 2^i$. Пусть $c = a + b = \sum_{i=0}^n c_i 2^i$ (здесь $a_i, b_i, c_i \in \{0, 1\}$ — соответствующие двоичные разряды этих чисел).

	a_{n-1}	\dots	a_1	a_0
$+$	b_{n-1}	\dots	b_1	b_0
<hr/>				
c_n	c_{n-1}	\dots	c_1	c_0

Сумматор должен вычислять набор из n результирующих функций: $c_i(a_0, \dots, a_{n-1}, b_0, \dots, b_{n-1})$ — разрядов суммы c . Обозначим через p_i бит переноса из $(i-1)$ -го разряда в i -ый. Тогда нетрудно видеть, что при $i = 0$

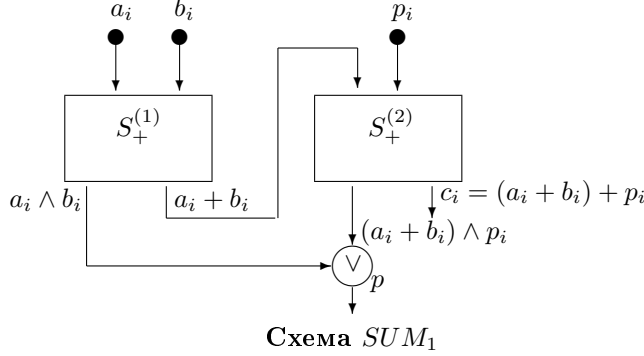
$$c_0 = a_0 + b_0 \text{ и } p_1 = a_0 \wedge b_0,$$

а при $1 \leq i \leq n-1$

$$c_i = p_i + a_i + b_i \text{ и } p_{i+1} = (a_i \wedge b_i) \vee (p_i \wedge a_i) \vee (p_i \wedge b_i).$$

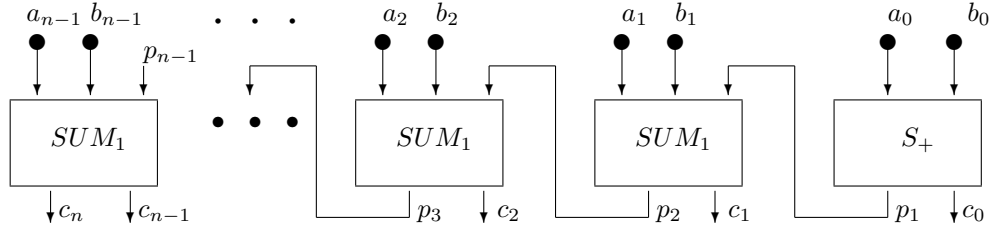
Старший разряд $c_n = p_n$.

Рассмотрим теперь построенную выше схему S_+ как схему, вычисляющую набор из двух функций: $x \wedge y$ (в вершине a) и $x + y$ (в вершине d). Используя два экземпляра этой схемы $S_+^{(1)}$ и $S_+^{(2)}$, можно легко реализовать схему одноразрядного сумматора SUM_1 , которая для $1 \leq i \leq n-1$ с тремя входами a_i, b_i и p_i вычисляет c_i и p_{i+1} :



Действительно, из построения следует, что в вершине p этой схемы вычисляется функция $f_p = (a_i \wedge b_i) \vee ((a_i + b_i) \wedge p_i) = (a_i \wedge b_i) \vee (p_i \wedge a_i) \vee (p_i \wedge b_i) = p_{i+1}$. Из представленной схемы видно, что сложность одноразрядного сумматора $L(SUM_1) = 9$.

Теперь из S_+ и одноразрядных сумматоров SUM_1 соберем схему SUM_n для n -разрядного сумматора.



Таким образом мы установили следующее утверждение.

Теорема 3. Для каждого $n \geq 1$ существует схема SUM , реализующая операцию суммирования двух n -разрядных двоичных чисел и имеющая сложность $L(SUM_n) = 9n - 5$.

Задача 4. Используя схему SUM_n , постройте схему, реализующую операцию вычитания двух n -разрядных двоичных чисел: $d = a - b$ (при условии, что $a \geq b$). Оцените сложность полученной схемы.

Задача 5. Определите глубину схем S_+ , S_{odd} , SUM_1 и SUM_n .

Задача 6. Два игрока независимо выбирают одно из четырех чисел от 0 до 3. Первый игрок выигрывает, если выбранные числа совпадают. Постройте схему, определяющую выигрыш 1-го игрока. Ее входы x_1, x_2 представляют число, выбранное 1-ым игроком, а y_1, y_2 — число, выбранное 2-ым игроком. Реализуемая функция $F(x_1, x_2, y_1, y_2)$ равна 1 тогда и только тогда, когда $x_1 = y_1$ и $x_2 = y_2$.

Задача 7. Постройте схему, определяющую результат голосования в комитете, состоящем из трех членов и председателя. В случае равенства голосов, голос председателя является решающим.

Задача 8. Пусть наборы аргументов булевой функции от трех аргументов упорядочены лексикографически, а ее значения задаются последовательностью 8 нулей и единиц. Постройте схемы, реализующие следующие функции.

а) $f_1 = (1111 \ 1011)$,

б) $f_2 = (1001 \ 1001)$,

в) $f = (0011 \ 1001)$.