

Применение графов для представления булевых функций

(Лекции по дискретной математике)

М.И. Дехтярь

Содержание

1	Введение	1
2	Логические схемы (схемы из функциональных элементов)	2
2.1	Основные определения	2
2.2	Схемы и линейные программы	4
2.3	Примеры схем	6
2.3.1	Сложение по модулю 2	6
2.3.2	Сумматор	7
2.4	Задачи	9
3	Упорядоченные бинарные диаграммы решений (УБДР)	9
3.1	Основные определения	9
3.2	Сокращенные УБДР	12
3.3	Построение сокращенных УБДР по формулам	16
3.4	Задачи	17

1 Введение

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 длинной ДНФ можно рассмотреть линейную функцию, определяющую нечетность суммы аргументов: $odd(X_1, X_2, \dots, X_n) = X_1 + X_2 + \dots + X_n$ (см. задачу 3.1 на стр. 17).

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

2 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.

2.1 Основные определения

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

Определение 2.1. *Логической схемой (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 2.2. 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 2.3.

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 2.4. 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 логических схем 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 2.5. 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 2.1. Рассмотрим схему S_1 с тремя входными переменными x, y и z , изображенную на рис. 1 и решим для нее проблему анализа.

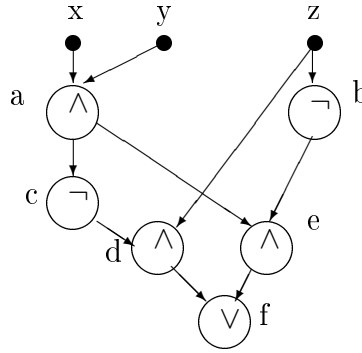


Рис. 1: Схема S_1

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.

2.2 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 $P_Z(\sigma_1, \dots, \sigma_n)$.

Определение 2.6. 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 2.1.

(1) По каждой логической схеме 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 логическую схему 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 $i = 1, \dots, t$ программа P_S вычисляет в переменной v_i функцию $f_{v_i}(x_1, \dots, x_n)$.

Пример 2.2. Применим конструкцию теоремы к схеме S_1 , представленной на рис. 1 на стр. 4. 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 на разных процессорах, так как результаты любой из них не используются в качестве аргументов другой.

2.3 Примеры схем

2.3.1 Сложение по модулю 2

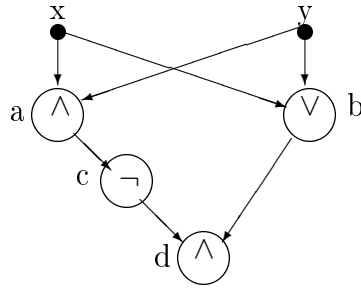
Рассмотрим схему S_+ на рис. 2.

В соответствии с определением вершины этой схемы реализуют следующие функции:

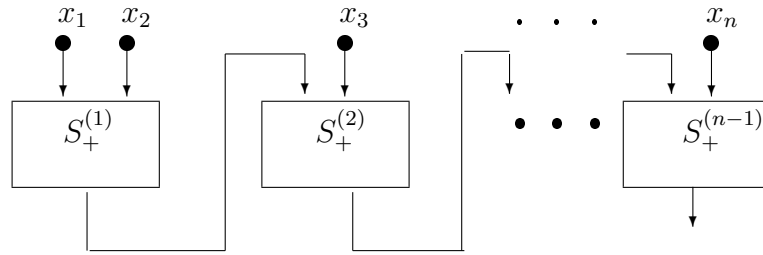
$$f_a(x, y) = x \wedge y, f_b(x, y) = x \vee y, f_c(x, y) = \neg f_a(x, y) = \neg(x \wedge y), f_d(x, y) = f_c(x, y) \wedge f_b(x, y) = \neg(x \wedge y) \wedge (x \vee y) = x + y.$$

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

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

Рис. 2: Схема S_+ для функции $x + y$

Используя схему S_+ нетрудно построить схему S_{odd} для реализации линейной функции-суммы n аргументов по модулю 2 $odd(X_1, X_2, \dots, X_n) = X_1 + X_2 + \dots + X_n$ (см. рис.3 на стр. 7).

Рис. 3: 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 2.2. *Существует схема S_{odd} , реализующая функцию $odd(X_1, X_2, \dots, X_n) = X_1 + X_2 + \dots + X_n$ со сложностью $L(S_{odd}) = 4(n - 1)$.*

2.3.2 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).

Сумматор должен вычислять набор из $(n + 1)$ -ой результирующей функции:

$$c_i(a_0, \dots, a_{n-1}, b_0, \dots, b_{n-1}) \quad (i = 0, 1, \dots, n),$$

задающих соответствующие разряды суммы c .

$$\begin{array}{cccc}
a_{n-1} & \dots & a_1 & a_0 \\
+ & b_{n-1} & \dots & b_1 & b_0 \\
\hline
c_n & c_{n-1} & \dots & c_1 & c_0
\end{array}$$

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 совпадает с последним переносом: $c_n = p_n$.

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 (см. рис. 4), которая имеет три входа a_i, b_i и p_i ($1 \leq i \leq n-1$) и вычисляет c_i и p_{i+1} .

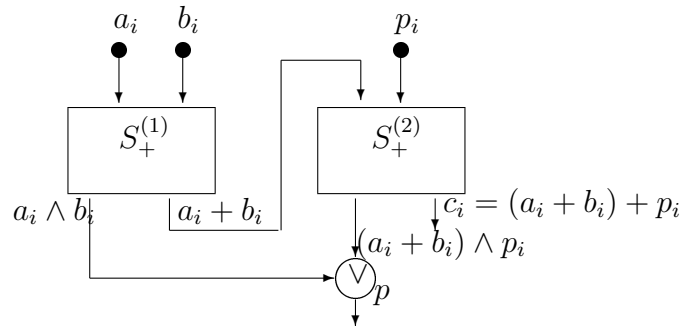


Рис. 4: Схема SUM_1

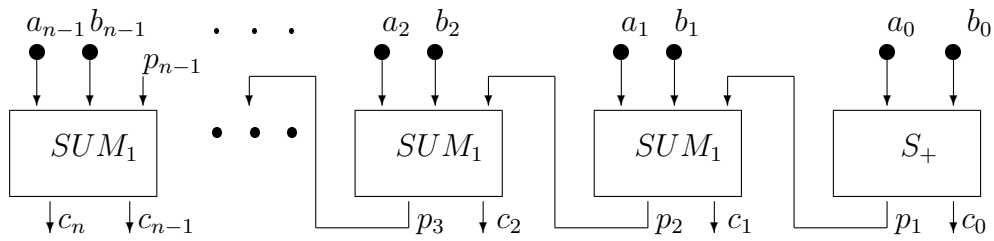
Действительно, из построения следует, что в вершине p этой схемы вычисляется функция $f_p = (a_i \wedge b_i) \vee ((a_i + b_i) \wedge p_i) = (a_i \wedge b_i) \vee (p_i \wedge a_i) \vee (p_i \wedge b_i) = p_{i+1}$. Из представленной схемы видно, что сложность одноразрядного сумматора $L(SUM_1) = 9$.

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

Таким образом мы установили следующее утверждение.

Теорема 2.3. Для каждого $n \geq 1$ существует схема SUM , реализующая операцию суммирования двух n -разрядных двоичных чисел и имеющая сложность $L(SUM_n) = 9n - 5$.

Замечание Логические схемы интенсивно исследовались в 50-е - 70-е годы прошлого столетия. В частности, К. Шеннон и О.Б. Лупанов установили оценки сложности схем для булевых функций от n аргументов. Оказалось, что любую такую

Рис. 5: Схема сумматора SUM_n

функцию можно реализовать со сложностью не большей (по порядку) $2^n/n$ и что “почти все” они имеют не меньшую сложность. При этом до сих пор не известна ни одна последовательность “конкретных” функций f_n , сложность которых по порядку превосходила бы линейную функцию.

2.4 Задачи

Задача 2.1. Докажите пункт (2) теоремы 2.1 на стр. 5.

Задача 2.2. Докажите, что минимальная схема для сложения имеет сложность $L(+) = 4$.

Задача 2.3. 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 2.4. Определите глубину схем S_+ , S_{odd} , SUM_1 и SUM_n .

Задача 2.5. 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 2.6. Постройте схему, определяющую результат голосования в комитете, состоящем из трех членов и председателя. В случае равенства голосов, голос председателя является решающим.

Задача 2.7. 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) $f_3 = (0011\ 1001)$.

3 Упорядоченные бинарные диаграммы решений (УБДР)

Указанный способ представления булевых функций с помощью специального подкласса ориентированных графов без циклов был предложен Р. Бриантом (R. Bryant) в 1986г. Его английское название — “Ordered binary decision diagram”, сокращенно — OBDD. Сейчас УБДР являются одним из основных средств реализации булевых функций от большого числа переменных в задачах искусственного интеллекта, проверки правильности электронных схем, программ, протоколов и т.п.

3.1 Основные определения

Одним из предшественников УБДР являются *бинарные деревья решений*.

Определение 3.1. *Бинарное дерево решений (БДР) — это бинарное дерево $T = (V, E)$, все внутренние вершины которого помечены переменными, а листья — значениями 0 или 1. Из каждой внутренней вершины v выходят 2 ребра, одно помечено 0, другое — 1; вершина w_0 , в которую ведет ребро, помеченное 0, называется 0-сыном v , а вершина w_1 , в которую ведет ребро, помеченное 1, называется 1-сыном v .*

Такое дерево, вершины которого помечены переменными x_1, \dots, x_n реализует булеву функцию $f(x_1, \dots, x_n)$, если для каждого набора значений переменных $\sigma_1, \sigma_2, \dots, \sigma_n$ ветвь в дереве, соответствующая этому набору (из вершины x_i идем по ребру, помеченному σ_i), завершается листом с меткой $f(\sigma_1, \sigma_2, \dots, \sigma_n)$.

Пример 3.1. Например, рассмотрим изображенное ниже БДР T_1 (на всех рисунках предполагается, что ребра направлены сверху вниз).

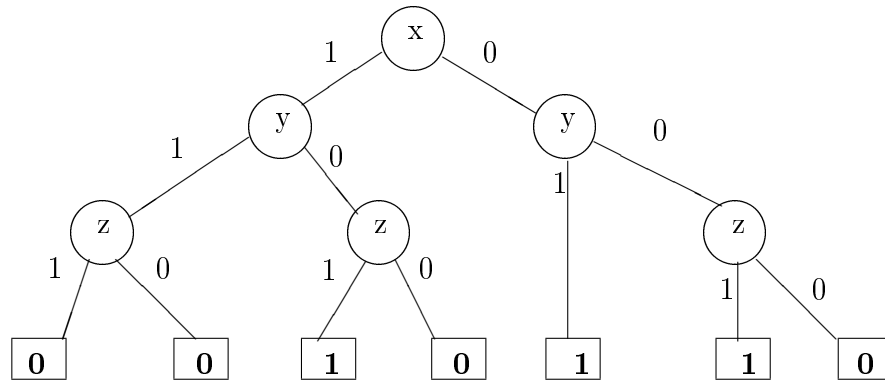


Рис. 6: Бинарное дерево решений T_1

По определению T_1 реализует функцию $f_1(x, y, z)$, представленную в таблице 1.

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

Таблица 1: Функция $f_1(x, y, z)$, реализуемая БДР T_1 на рис. 6

Нетрудно построить ДНФ этой функции: $f(x, y, z) = (\neg x \wedge y) \vee (\neg y \wedge z)$.

УБДР являются модификацией БДР, в которой все листья с одной меткой представлены одной вершиной, в каждую вершину может входить несколько ребер, возможен выбор порядка появления переменных на ветвях.

Определение 3.2. Пусть зафиксирован некоторый порядок n переменных $\pi : x_{\pi(1)}, \dots, x_{\pi(n)}$.

Упорядоченная бинарная диаграмма решений относительно порядка переменных π — это ориентированный граф без циклов с одним корнем, в котором

- 1) существует лишь две вершины, из которых не выходят ребра; они помечены константами 0 и 1 и называются **стоками**;
- 2) остальные (внутренние) вершины помечены переменными и из каждой из них выходят два ребра, одно помечено 0, другое — 1;
- 3) порядок, в котором переменные встречаются на любом пути из корня в сток, совместим с π , т.е. если из вершины, помеченной $x_{\pi(i)}$, есть путь в вершину, помеченную $x_{\pi(j)}$, то $i < j$.

Как и в случае БДР, УБДР реализует булеву функцию $f(x_1, \dots, x_n)$, если для каждого набора значений переменных $\sigma_1, \sigma_2, \dots, \sigma_n$ путь в диаграмме, начинающийся в корне и соответствующий этому набору (из вершины x_i идем по ребру, помеченному σ_i), завершается стоком с меткой $f(\sigma_1, \sigma_2, \dots, \sigma_n)$.

Из этого определения непосредственно следует, что каждая внутренняя вершина диаграммы v , помеченная переменной $x_{\pi(k)}$, является корнем поддиаграммы, которая включает все вершины диаграммы, достижимые из v и реализует некоторую функцию $f_v(x_{\pi(k)}, x_{\pi(k+1)}, \dots, x_{\pi(n)})$ от $(n - k + 1)$ переменных $x_{\pi(k)}, x_{\pi(k+1)}, \dots, x_{\pi(n)}$. При этом ее 0-сын w_0 является корнем поддиаграммы, реализующей функцию $f_{w_0}(x_{\pi(k+1)}, \dots, x_{\pi(n)}) = f_v(0, x_{\pi(k+1)}, \dots, x_{\pi(n)})$, а 1-сын w_1 — корень поддиаграммы, реализующей функцию $f_{w_1}(x_{\pi(k+1)}, \dots, x_{\pi(n)}) = f_v(1, x_{\pi(k+1)}, \dots, x_{\pi(n)})$. Пусть диаграмма реализует функцию

$f(x_1, \dots, x_n) = f'(x_{\pi(1)}, \dots, x_{\pi(n)})$ и $\sigma_{\pi(1)}, \dots, \sigma_{\pi(k-1)}$ — это набор значений переменных $x_{\pi(1)}, \dots, x_{\pi(k-1)}$, который соответствует пути из корня в вершину v (таких наборов может быть несколько). Тогда $f_v(x_{\pi(k)}, \dots, x_{\pi(n)}) = f'(\sigma_{\pi(1)}, \dots, \sigma_{\pi(k-1)}, x_{\pi(k)}, \dots, x_{\pi(n)})$.

Пример 3.2. Реализуем с помощью УБДР функцию $f_1(x, y, z)$, представленную выше в примере 3.1 с помощью БДР T_1 и таблицы 1.

Вначале зафиксируем порядок переменных: $x < y < z$. Объединив листья с одинаковыми метками и две z -вершины с одинаковыми потомками, получим УБДР D_1 , приведенную на рис. 7.

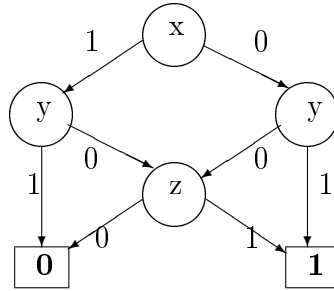


Рис. 7: УБДР D_1 для функции $f_1(x, y, z)$

Ясно, что реализация функции $f_1(x, y, z)$ с помощью УБДР D_1 намного компактнее, чем с помощью БДР T_1 .

Под сложностью $L(D)$ УБДР D будем понимать число внутренних вершин в D . Например, $L(D_1) = 4$. Может ли сложность диаграммы для некоторой функции зависеть от порядка переменных? Да! Рассмотрим порядок переменных $y < x < z$. Как показывает следующий рисунок относительно этого порядка функцию $f(x, y, z)$ можно реализовать УБДР D_2 со сложностью $L(D_2) = 3$.

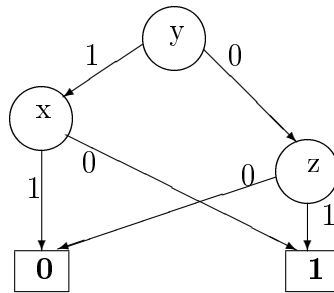


Рис. 8: УБДР D_2 для функции $f_1(x, y, z)$

3.2 Сокращенные УБДР

Когда порядок переменных зафиксирован, то достаточно просто можно по произвольной УБДР для функции построить минимальную УБДР, реализующую данную функцию.

Определение 3.3. УБДР называется **сокращенной**, если

- 1) из любой внутренней вершины v ее 0-сын и 1-сын не совпадают;
- 2) нет такой пары внутренних вершин u и v , для которых поддиаграммы с корнями u и v являются изоморфными (т.е. взаимно однозначно отображаются друг на друга с сохранением всех меток).

Смысл этого определения понятен: если из некоторой вершины v оба ребра ведут в одну вершину, то такая вершина v не нужна, а если имеются две вершины с одинаковыми поддиаграммами, то их можно слить. Определим два типа эквивалентных преобразований УБДР.

Правило сокращения: если 0-сын и 1-сын вершины v совпадают и равны w , то удалить v , перенаправив все входящие в нее ребра в вершину w .

Правило слияния: если вершины v и w помечены одной переменной x и имеют одинаковых 0-сыновей и 1-сыновей, то удалить вершину v , перенаправив все входящие в нее ребра в вершину w .

На следующем рисунке показаны преобразования по этим правилам.

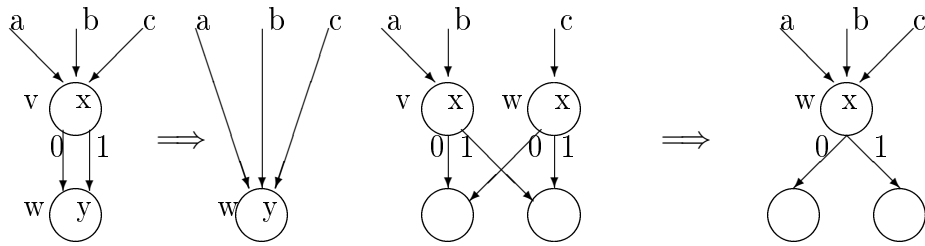


Рис. 9: Правило сокращения

Правило слияния

Следующая простая теорема показывает, что применимость этих двух правил является критерием несокращаемости УБДР.

Теорема 3.1. УБДР D является сокращенной тогда и только тогда, когда к ней не применимо ни правило слияния, ни правило сокращения.

Доказательство. \Rightarrow Если к D применимо правило сокращения, то не выполнено условие (1) из определения сокращенной УБДР, а если к D применимо правило слияния, то поддиаграммы с корнями v и w являются изоморфными и не выполнено условие (2).

\Leftarrow Пусть к УБДР D нельзя применить правило сокращения. Тогда в ней нет вершин с совпадающими 0- и 1-сыновьями и выполнено условие (1). Пусть к УБДР D нельзя применить правило слияния. Тогда из следующей леммы можно заключить, что в D нет пары вершин, поддиаграммы которых являются изоморфными, и следовательно, выполнено условие (2).

Лемма 3.1. *Если в D есть такая пара вершин u и v , для которых поддиаграммы с корнями v и w являются изоморфными, то в D имеется и пара вершин v', w' с попарно одинаковыми 0- и 1-сыновьями u , следовательно, к D применимо правило слияния.*

Доказательство леммы проведем индукцией по высоте h поддиаграмм D_v и D_w с корнями v и w , соответственно (так как D_v и D_w изоморфны, то их высоты, т.е. длины максимальных путей из корней до стоков, одинаковы).

Базис: $h = 1$. В этом случае 0- и 1-сыновьями вершин v и w являются одинаковые стоки.

Шаг индукции. Предположим, что утверждение верно для $h = k$. Пусть D_v и D_w — поддиаграммы высоты $h = k + 1$. Пусть v_0 и w_0 — это 0-сыновья вершин v и w , соответственно, а v_1 и w_1 — их 1-сыновья. Если $v_0 = w_0$ и $v_1 = w_1$, то в качестве v', w' подходят сами v и w . Если же для некоторого $i \in \{0, 1\}$ $v_i \neq w_i$, то поддиаграммы D_{v_i} и D_{w_i} с корнями v_i и w_i являются изоморфными и имеют высоту k . Тогда по предположению индукции утверждение леммы выполнено. \square

Из теоремы 3.1 непосредственно следует, что применяя к произвольной УБДР правила сокращения и слияния, мы, в конце концов, получим сокращенную УБДР. Чтобы эта процедура работала эффективно, нужно применять правила в порядке “снизу-вверх”. Мы опишем этот алгоритм для “естественного” порядка переменных: x_1, \dots, x_n .

Алгоритм СОКРАЩЕНИЕ-УБДР

Вход: УБДР D для функции $f(x_1, \dots, x_n)$.

Выход: сокращенная УБДР для f .

1. Занумеруем множество вершин D : $V = \{v_1, v_2, \dots, v_m\}$;
2. **ДЛЯ** $i = n, n - 1, \dots, 1$ **ВЫПОЛНЯТЬ**
3. {
4. $V(i) = \{v \mid v \text{ помечена переменной } x_i\}$;
- /* Применение правила сокращения:
5. **ДЛЯ КАЖДОЙ** $v \in V(i)$ **ВЫПОЛНЯТЬ**
6. **ЕСЛИ** $(0\text{-сын } v) = (1\text{-сын } v) = w$ **ТО**
7. { удалить v из $V(i)$;
8. перенаправить все ребра, входящие в v , в вершину w ;

9. удалить v из D }
10. **ИНАЧЕ** $key(v) = (j, k)$, где v_j — это 0-сын v , а v_k — 1-сын v ;
- /* Применение правила слияния:
11. Отсортировать $V(i)$ по ключу $key(v)$: пусть в этом порядке $V(i) = \{u_1, \dots, u_{k_i}\}$;
12. $тек_ключ = (0, 0)$;
13. **ДЛЯ** $j = 1, \dots, k_i$ **ВЫПОЛНЯТЬ**
14. **ЕСЛИ** $тек_ключ = key(u_j)$ **ТО**
15. { удалить u_j из $V(i)$;
16. перенаправить все ребра, входящие в u_j , в $тек_вершина$;
17. удалить u_j из D }
18. **ИНАЧЕ** { $тек_вершина = u_j$; $тек_ключ = key(u_j)$ }
19. }

Пример 3.3. Рассмотрим пример применения алгоритма СОКРАЩЕНИЕ-УБДР, показанный на следующем рисунке.

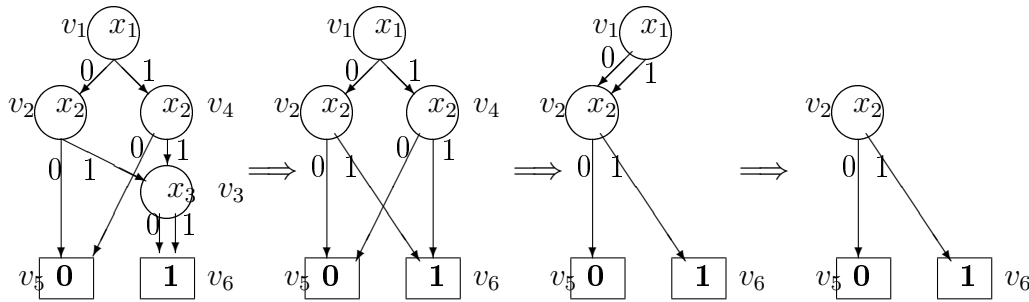


Рис. 10: Применение алгоритма СОКРАЩЕНИЕ-УБДР

На исходной УБДР слева все вершины уже занумерованы. Стрелки разделяют УБДР, получаемые после очередных итераций основного цикла в строках 2 - 19. При первом исполнении цикла $i = 3$, $V(3) = \{v_3\}$. Для вершины v_3 условие в строке 6 выполнено ($w = v_6$), поэтому применяется правило сокращения и эта вершина удаляется, а ее входы направляются в v_6 . При следующем исполнении цикла $i = 2$, $V(2) = \{v_2, v_4\}$. После цикла в строках 5 - 10 $key(v_2) = \{5, 6\}$ и $key(v_4) = \{5, 6\}$. После сортировки $u_1 = v_2$, $u_2 = v_4$. В цикле в строках 13-18 для u_2 выполнено условие в строке 14. Поэтому применяется правило слияния и вершина v_4 удаляется, а ее вход передается вершине v_2 . При третьем исполнении цикла $i = 1$, $V(1) = \{v_1\}$. Для вершины v_1 условие в строке 6 выполнено ($w = v_2$), поэтому применяется правило

сокращения и эта вершина удаляется.

Оказывается, что построенная алгоритмом УБДР является единственной и минимальной для заданного порядка.

Теорема 3.2.

- 1) Алгоритм СОКРАЩЕНИЕ-УБДР строит сокращенную УБДР, эквивалентную исходной УБДР D .
- 2) Эта сокращенная УБДР является при данном порядке переменных единственной (с точностью до изоморфизма) и минимальной.

Доказательство первого пункта непосредственно следует из выполнения критерия теоремы 3.1, так как к результирующей диаграмме никакое правило сокращения или слияния неприменимо.

Доказательство второго пункта основано на следующем индуктивном утверждении:

Лемма 3.2. После выполнения i -ой итерации алгоритма в полученной диаграмме для каждой подфункции $f(\sigma_1, \sigma_2, \dots, \sigma_{i-1}, x_i, \dots, x_n)$ ($\sigma_k \in \{0, 1\}$ при $k = 1, 2, \dots, i-1$), существенно зависящей от x_i , имеется ровно одна вершина – корень поддиаграммы, реализующей эту подфункцию.

Напомним, что функция $f(x_1, x_2, \dots, x_i, \dots, x_n)$ существенно зависит от переменной x_i , если существуют такие два набора значений аргументов $(\sigma_1, \dots, \sigma_{i-1}, 0, \sigma_{i+1}, \dots, \sigma_n)$ и $(\sigma_1, \dots, \sigma_{i-1}, 1, \sigma_{i+1}, \dots, \sigma_n)$, различающиеся только значением x_i , на которых f принимает разные значения: $f(\sigma_1, \dots, \sigma_{i-1}, 0, \sigma_{i+1}, \dots, \sigma_n) \neq f(\sigma_1, \dots, \sigma_{i-1}, 1, \sigma_{i+1}, \dots, \sigma_n)$.

Доказательство этой леммы и вывод из нее утверждения 2 теоремы 3.2 оставляем в качестве задач 3.2 и 3.3 на стр. 18.

3.3 Построение сокращенных УБДР по формулам

Алгоритм СОКРАЩЕНИЕ-УБДР позволяет построить сокращенную УБДР для функции f , по любой другой ее УБДР. Но как построить УБДР, если f задана, например, с помощью формулы? Можно, конечно, попытаться построить полное бинарное дерево решений, объединить в нем все листья с меткой 0 в один сток, а листья с меткой 1 — в другой. Затем применить к получившейся УБДР алгоритм СОКРАЩЕНИЕ-УБДР. Но этот подход годится только для функций от небольшого числа переменных, так как полное БДР для $f(x_1, \dots, x_n)$ будет содержать 2^n листьев.

Другой подход связан с построением УБДР “сверху-вниз”. Объясним его для естественного порядка переменных: $x_1 < x_2 < \dots < x_n$.

Начнем построение с корня, помеченного x_1 . Рассмотрим две остаточные функции: $f_0(x_1, \dots, x_n) = f(0, x_2, \dots, x_n)$ и $f_1(x_2, \dots, x_n) = f(1, x_2, \dots, x_n)$. Если они одинаковы, то f не зависит от x_1 и тогда изменим метку у корня на x_2 . Если обе функции f_0 и f_1 существенно зависят от x_2 , то для каждой из них добавляем вершину, помеченную x_2 , и далее реализуем по индукции. Если f_k ($k \in \{0, 1\}$) не зависит от переменных x_2, \dots, x_j , но зависит существенно от x_{j+1} , то добавляем вершину, помеченную x_{j+1} , и проводим в нее ребро с меткой k из вершины, соответствующей f . Пусть для некоторого i уже построены вершины для всех различных остаточных функций вида $f_{\sigma_1 \dots \sigma_i}(x_{i+1}, \dots, x_n) = f(\sigma_1 \dots \sigma_i, x_{i+1}, \dots, x_n)$, существенно зависящих от x_i . Для каждой из них получим две остаточные функции $f_{\sigma_1 \dots \sigma_i 0}(x_{i+2}, \dots, x_n) = f(\sigma_1 \dots \sigma_i, 0, x_{i+2}, \dots, x_n)$ и $f_{\sigma_1 \dots \sigma_i 1}(x_{i+2}, \dots, x_n) = f(\sigma_1 \dots \sigma_i, 1, x_{i+2}, \dots, x_n)$. Затем выберем из множества этих функций разные, для каждой из них добавим в диаграмму вершину, помеченную x_{i+1} , и проведем в них соответствующие ребра из вершин, помеченных x_i . Продолжая построение, дойдем до функций от 1-ой переменной x_n и до констант, для которых минимальные реализации очевидны.

Пример 3.4. Рассмотрим, например, функцию

$$f(x_1, x_2, x_3, x_4) = (x_1 \wedge x_2 \wedge x_4) \vee (\neg x_1 \wedge x_2 \wedge \neg x_4) \vee (\neg x_2 \wedge x_3) \vee (\neg x_2 \wedge x_4).$$

и построим для нее УБДР относительно порядка $x_1 < x_2 < x_3 < x_4$, используя описанную выше процедуру.

Вначале создадим корень, помеченный x_1 и рассмотрим остаточные функции, получающиеся при $x_1 = 0$ и $x_1 = 1$. Имеем

$$f_0(x_2, x_3, x_4) = (x_2 \wedge \neg x_4) \vee (\neg x_2 \wedge x_3) \vee (\neg x_2 \wedge x_4),$$

$$f_1(x_2, x_3, x_4) = (x_2 \wedge x_4) \vee (\neg x_2 \wedge x_3) \vee (\neg x_2 \wedge x_4).$$

Они разные и обе существенно зависят от x_2 . Поэтому добавим для каждой из них вершину, помеченную x_2 . Затем для каждой из них определим остаточные функции, получающиеся при $x_2 = 0$ и $x_2 = 1$. Получим

$$f_{00}(x_3, x_4) = (x_3 \vee x_4), \quad f_{01}(x_3, x_4) = \neg x_4, \quad f_{10}(x_3, x_4) = (x_3 \vee x_4), \quad f_{11}(x_3, x_4) = x_4,$$

Так как $f_{00} = f_{10}$, а f_{01} и f_{11} от x_3 не зависят, то нам потребуется только одна вершина, помеченная x_3 . Она будет представлять функцию $f_{00} = f_{10} = (x_3 \vee x_4)$. При $x_3 = 0$ она превращается в x_4 , а при $x_3 = 1$ равна константе 1. В результате получается УБДР D_f , показанная на рис. 11.

3.4 Задачи

Задача 3.1. 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 .

Задача 3.2. Докажите лемму 3.2 на стр. 16 возвратной индукцией по i .

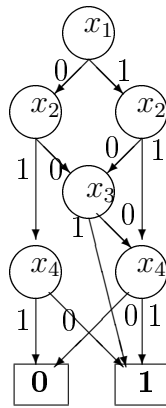


Рис. 11: УБДР D_f для функции f из примера 3.4 на стр. 16

Задача 3.3. Используя лемму 3.2 на стр. 16, докажите утверждение 2 теоремы 3.2 на стр. 15.

Задача 3.4. Постройте минимальные УБДР для двуместных функций:
 $x \wedge y, x \vee y, x + y, x \rightarrow y, x|y$.

Задача 3.5. Постройте минимальные УБДР для функции
 $f(x_1, x_2, x_3, x_4, x_5, x_6) = (x_1 \wedge x_2) + (x_3 \wedge x_4) + (x_5 \wedge x_6)$
относительно двух упорядочений переменных:

- $x_1 < x_2 < x_3 < x_4 < x_5 < x_6$ и
- $x_1 < x_3 < x_5 < x_2 < x_4 < x_6$.

Задача 3.6. Пороговая функция T_n^k от n переменных с порогом k выдает 1, если во входном наборе имеется не менее k единиц; $T_n^k(x_1, x_2, \dots, x_n) = 1 \Leftrightarrow x_1 + x_2 + \dots + x_n \geq k$.

- Постройте минимальные УБДР для пороговых функций T_3^2, T_4^2, T_5^3 .
- Зависит ли сложность минимальной УБДР для пороговых функций от порядка переменных?
- Оцените сложность минимальной УБДР для пороговой функции T_n^k .

Задача 3.7. Выберите подходящий порядок переменных и построьте для него минимальные УБДР, реализующие функции из задач 2.5 и 2.6 на стр. 9.

Задача 3.8. Как мы видели, логические схемы естественным образом реализуются в виде неветвящихся программ. Наоборот, для деревьев решений и УБДР естественным программным представлением являются ветвящиеся программы,

включающие лишь условные операторы вида **if ... then ... else ...** с тестами вида “ $x = 0?$ ” и “ $x = 1?$ ” (они соответствуют внутренним вершинам диаграмм) и операторы присвоения значения 0 или 1 результату (они соответствуют вершинам-стокам).

Напишите ветвящиеся программы, вычисляющие функции, представляемые УБДР D_2 на рис. 8 на стр. 12 и D_f на рис. 11 на стр. 17.