Лекция 14

Булевы схемы и формулы

14.1 Булевы схемы

Булевой схемой от переменных $x_1, \ldots, x_n$ мы будем называть последовательность булевых функций $g_1, \ldots, g_s$, в которой всякая g_i или равна одной из переменных, или получается из предыдущих применением одной из логических операций отрицание, конъюнкция и дизъюнкция. Другими словами, для всякого i или $g_i = x_j$, $1 \leqslant j \leqslant n$, или существуют такие $j, k < i$, что $g_i = g_j \wedge g_k$, $g_i = g_j \vee g_k$ или $g_i = \neg g_j$. Также в булевой схеме задано некоторое число $m \geqslant 1$ и члены последовательности $g_{s-m+1}, \ldots, g_s$ называются выходами схемы (их как раз m). Число m называют числом выходов схемы. Мы говорим, что схема вычисляет булево отображение $f\colon \{0,1\}^n \to \{0,1\}^m$, если для всякого $x \in \{0,1\}^n$ верно $f(x) = (g_{s-m+1}(x), \ldots, g_s(x))$. Размером схемы называют число s.

Пример 14.1. Схема

$$x_1, x_2, x_1 \wedge x_2$$

с одним выходом вычисляет конъюнкцию переменных x_1 и x_2. Размер этой схемы равен 3.

Схема

$$x_1, x_2, \neg x_1, \neg x_2, x_1 \wedge \neg x_2, \neg x_1 \wedge x_2, (x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$$

с одним выходом вычисляет, как нетрудно проверить, функцию $x_1 \oplus x_2$. Размер этой схемы равен 7.

Схема

$$x_1, x_2, x_3, x_1 \vee x_2, (x_1 \vee x_2) \wedge x_3, (x_1 \wedge x_2), ((x_1 \vee x_2) \wedge x_3) \vee (x_1 \wedge x_2)$$

с одним выходом вычисляет функцию $MAJ_3(x_1, x_2, x_3)$. Эта функция равна 1 тогда и только тогда, когда хотя бы две из ее переменных равны 1. Размер этой схемы равен 7.

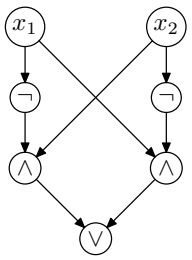

Рис. 14.1: Схема для функции $x_1 \oplus x_2$

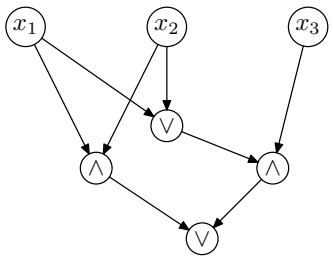

Рис. 14.2: Схема для функции MAJ_3

На практике удобно изображать схемы не в виде последовательности функций, а в виде ориентированного графа с s вершинами $v_1, \ldots, v_s$ (см. примеры на Рис. 14.1, 14.2). Вершины этого графа соответствуют функциям схемы: вершина v_i соответствует функции g_i. Вершины $v_1, \ldots, v_n$ при этом помечены переменными $x_1, \ldots, x_n$ соответственно. Всякая вершина v_i для $i > n$ помечена логической связкой, с помощью которой функция g_i получена из предыдущих. При этом, если функция g_i была получена из функций g_j и g_k то мы проводим ребра из вершин v_j и v_k в вершину v_i. Если функция g_i получена как отрицание g_j, то мы проводим только ребро из v_j в v_i. Вершины $v_{s-m+1}, \ldots, v_s$ помечены как выходные вершины схемы.

Рассмотрим более содержательные примеры.

Пример 14.2. Пусть нам даны две n-битовых двоичных записи чисел x и y и мы хотим вычислить двоичную запись их суммы $z = x + y$. Для удобства обозначим $x = x_{n-1} \ldots x_1 x_0$, где x_0 – младший разряд двоичной записи. Аналогично, $y = y_{n-1} \ldots y_1 y_0$. Во-первых, заметим, что в двоичной записи z будет не более $n+1$ разрядов. Так что мы хотим построить схему с $2n$ входами и $n+1$ выходом.

Идея конструкции схемы будет та же, что и в обычном школьном сложении в столбик. Мы будем складывать числа x и y поразрядно, попутно вычисляя биты переноса в следующий разряд.

Для удобства будем обозначать через b_i бит, который переносится в i-ый разряд из предыдущих.

Заметим, что мы уже готовы вычислить первый разряд ответа $z_0 = x_0 \oplus y_0$. Конечно, мы не можем сразу применить операцию $\oplus$, но выше мы показали, как ее можно вычислить небольшой схемой. Добавим эту маленькую схему в нашу, как подсхему. Далее, заметим, что $b_1 = x_0 \wedge y_0$, добавим соответствующий элемент в схему. Перейдем к следующему разряду. Здесь $z_1 = x_1 \oplus y_1 \oplus b_1$ и $b_2 = MAJ_3(x_1, y_1, b_1)$. Для вычисления первого добавим сначала подсхему, вычисляющую промежуточную величину $c_1 = x_1 \oplus y_1$, а затем подсхему, вычисляющую $z_1 = c_1 \oplus b_1$. Для вычисления b_2

просто добавим подсхему, вычисляющую функцию MAJ_3. Такая схема также приведена выше. Дальше, случай произвольных z_i и b_i полностью аналогичен случаю z_1 и b_1 и мы можем последовательно вычислить все эти значения.

Оценим теперь размер описанной схемы. Для каждого разряда ответа нам нужно не больше двух раз применить подсхему для вычисления функции $\oplus$ и не более одного раза подсхему для вычисления MAJ_3. Все эти схемы имеют постоянный размер, так что для вычисления каждого разряда z мы используем постоянное число элементов. Всего нам нужно $O(n)$ элементов.

Пример 14.3. Построим теперь схему для умножения n-битовых чисел. Пусть на вход снова подаются два числа $x = x_{n-1} \dots x_1 x_0$ и $y = y_{n-1} \dots y_1 y_0$. На этот раз мы хотим вычислить $z = x \cdot y$. Заметим, что z имеет не больше $2n$ разрядов. Действительно, $x, y < 2^n$, так что $z = x \cdot y < 2^{2n}$, а значит для его записи достаточно $2n$ разрядов.

Для вычисления z снова воспользуемся школьным методом. В нем умножение двух чисел сводится к сложению n чисел. Действительно, чтобы умножить x на y достаточно для всякого $i = 0, \dots, n-1$ умножить x на y_i, приписать в конце числа i нулей и затем сложить все полученные числа. Умножение x на y_i легко реализуется с помощью n конъюнкций. После этого остается сложить n чисел длины не более $2n$. Для этого мы можем $n-1$ раз применить схему для сложения, описанную выше. Размер каждой схемы для сложения линейный, так что суммарная сложность схемы для умножения получается $O(n^2)$.

Пример 14.4. Мы разобрали, как в модели булевых схем выполнять базовые арифметические операции. Теперь обсудим одну из простейших комбинаторных задач. Нам дан неориентированный граф G на n вершинах, нужно проверить, является ли он связным. Во-первых, стоит обсудить, как задавать граф в пригодном для схем виде. Для этого удобно воспользоваться так называемой *матрицей смежности* графа. Перенумеруем вершины графа $v_1, v_2, \dots, v_n$. Матрицей смежности графа G называется матрица $A \in \{0,1\}^{n \times n}$, в которой на пересечении строки i со столбцом j стоит 1 тогда и только тогда, когда $v_i, v_j \in E$. Матрица смежности полностью описывает, какие пары вершин соединены ребрами, так что булевой схеме достаточно подать на вход матрицу смежности графа. Более того, заметим, что матрица смежности симметрична и на диагонали у нее обязательно стоят нули (мы запрещали петли – ребра, ведущие из вершины в нее же саму). Так что на вход схеме можно подать, скажем, только верхнюю половину матрицы смежности.

Оказывается, матрица смежности также удобна для проверки связности графа. Можно матрицу A интерпретировать следующим образом: на пересечении строки i и столбца j написано количество путей длины 1 из вершины v_i в вершину v_j. Теперь возведем матрицу A в квадрат (над действительными числами). Если посмотреть на формулу для произведения матриц можно заметить, что на пересечении строки i и столбца j матрицы A^2 записано количество путей длины 2 из вершины v_i в вершину v_j. По индукции можно доказать, что на пересечении строки i и столбца j матрицы A^k записано число путей длины k из вершины v_i в вершину v_j. Заметим теперь, что

если между какими-то двумя вершинами в графе есть путь, то обязательно есть путь длины не больше $n-1$ (из пути всегда можно выкинуть циклы, если они там есть, после этого все вершины в пути разные). Так что для проверки связности у нас появляется такой план: вычислим все матрицы $A, A^2, \ldots, A^{n-1}$ и для каждой пары вершин v_i и v_j проверим, есть ли между ними путь длины не больше $n-1$. Если это так, то граф связен, иначе не связен.

Этот план можно несколько упростить. Во-первых, можно немного модифицировать матрицу смежности. Рассмотрим матрицу A', которая отличается от матрицы A тем, что у нее на главной диагонали стоят единицы, а не нули (в остальном матрицы совпадают). В терминах графов это означает, что к каждой вершине мы добавляем петлю. В модели простых неориентированных графов мы этого не допускали, но ничего не мешает нам рассмотреть графы с петлями. Идея состоит в том, что теперь, если между двумя вершинами есть путь длины меньше $n-1$, то есть и путь длины ровно $n-1$ (достаточно добавить к пути нужное количество петель). Так что теперь не обязательно смотреть на все степени матрицы смежности, достаточно взглянуть на $(A')^{n-1}$. Если в ячейках этой матрицы нет нулей, то граф связен, иначе не связен.

Второе упрощение связано со способом возведения матрицы в степень. В данный момент мы это делаем над действительными числами, что заставляет нас складывать и умножать целые числа. Чтобы делать это с помощью схем, нам придется использовать описанные выше схемы для сложения и умножения, а чтобы оценить размер получившейся схемы придется оценивать величину возникающих в процессе вычислений целых чисел. Все это не очень хочется делать. Решение состоит в том, чтобы вместо умножения матриц над целыми числами воспользоваться так называемым *булевым умножением матриц*. В нем формулы для умножения матриц такие же, как и в обычном умножении, только вместо операции умножения используется конъюнкция, а вместо сложения – дизъюнкция. Тогда можно по индукции доказать, что в (булевой) матрице $(A')^k$ на пересечении строки i и столбца j стоит 1 тогда и только тогда, когда в графе есть путь из v_i в v_j длины не больше k.

Теперь мы готовы описать схему для проверки графа на связность. На вход схема (по существу) получает матрицу смежности A'. Схема последовательно вычисляет булевы степени этой матрицы $(A')^2, \ldots, (A')^{n-1}$. Затем схема вычисляет конъюнкцию всех ячеек матрицы $(A')^{n-1}$ и подает ее на выход.

Оценим размер получившийся схемы. Для булева умножения двух булевых матриц $n \times n$ достаточно $n^2 \cdot O(n) = O(n^3)$ операций (каждая ячейка произведения матриц вычисляется за линейное число операций, всего ячеек n^2). Всего нам нужно $(n-1)$ умножение матриц, так что для вычисления матрицы $(A')^{n-1}$ достаточно $O(n^4)$ операций. На последний этап (конъюнкция ячеек $(A')^{n-1}$) нужно $O(n^2)$ операций, итого получается $O(n^4) + O(n^2) = O(n^4)$ операций.

Задача 14.5. При построении схемы, реализующей описанный нами алгоритм, и при подсчете числа ее элементов мы действовали довольно грубо. Так, то же самое можно было сделать за $O(n^3 \log n)$ операций. Как это сделать?

Мы увидели, что конкретные функции можно вычислять схемами, причем раз-

мер схем получается не слишком большим (полиномиальным). Далее заметим, что всякую булеву функцию можно вычислить булевой схемой. По существу мы уже обсуждали этот вопрос, когда обсуждали дизъюнктивную нормальную форму функции. Для полноты изложения повторим кратко это рассуждение.

Лемма 14.1. *Всякую функцию* $f\colon \{0,1\}^n \to \{0,1\}$ *можно вычислить схемой размера не больше* $O(n2^n)$.

Доказательство. Для всякого $a \in \{0,1\}^n$ рассмотрим функцию $f_a\colon \{0,1\}^n \to \{0,1\}$, такую что $f(x) = 1$ тогда и только тогда, когда $x = a$. Будет удобно ввести обозначение $x^1 = x$ и $x^0 = \neg x$. Тогда функцию f_a можно записать формулой

$$f_a(x) = \bigwedge_{i=1}^{n} x_i^{a_i},$$

где $x = (x_1, \ldots, x_n)$ и $a = (a_1, \ldots, a_n)$.

Для произвольной функции f уже не сложно записать формулу через функции f_a:

$$f(x) = \bigvee_{a \in f^{-1}(1)} f_a(x).$$

Теперь эти формулы можно переделать в схему. Наша схема сначала будет вычислять отрицания всех переменных, на это нужно n элементов. После этого можно вычислить все функции f_a. Для вычисления каждого нужно $n-1$ раз применить конъюнкцию. Всего получается $2^n(n-1)$ элемент. Наконец, для вычисления f нужно взять дизъюнкцию нужных функций f_a, на это уйдет не более 2^n элементов. Суммарно в нашей схеме получается $O(n2^n)$ элементов. □

Из этого несложно получить схему для произвольной функцию с многими выходами $f\colon \{0,1\}^n \to \{0,1\}^m$.

Следствие 14.2. *Всякую функцию* $f\colon \{0,1\}^n \to \{0,1\}^m$ *можно вычислить схемой размера не больше* $O(nm2^n)$.

Доказательство. Для данной функции f рассмотрим функции $f_i\colon \{0,1\}^n \to \{0,1\}$, для $i = 1, \ldots, m$, такие что для всякого $x \in \{0,1\}^n$ $f(x) = (f_1(x), \ldots, f_m(x))$. Иными словами f_i выдает i-ую координату f. Мы можем вычислить каждую функцию f_i схемой размера $O(n2^n)$. Объединив эти схемы в одну, мы получим схему для f размера $O(mn2^n)$. □

Мы показали, что всякую функцию можно вычислить схемой, но размер схемы при этом получился большим. Оказывается, это неизбежно.

Теорема 14.3. *Для всякого* $n \geqslant 10$ *существует функция* $f\colon \{0,1\}^n \to \{0,1\}$, *которую нельзя вычислить схемой размера меньше* $2^n/10n$.

Доказательство. Для доказательства применим мощностной метод: докажем, что функций больше, чем маленьких схем. Тогда маленьких схем не хватит, чтобы вычислить все функции.

Всего булевых функций от n переменных 2^{2^n}.

Далее, заметим, что всякую схему размера не больше S с n переменными можно описать с помощью не больше чем $S \cdot (2 + \max(2 + 2\log S, 1 + \log n))$ битов. Действительно, для каждого из элементов схемы, а их не более S, нужно указать его тип (конъюнкция, дизъюнкция, отрицание или переменная), на что достаточно потратить два бита, а также указать, к каким из предыдущих элементов применяется операция, или указать номер переменной, которой равен этот элемент. На это требуется не более $\max(2 + 2\log S, 1 + \log n)$ битов (можно просто выписать номера элементов или номер переменной).

Мы применим эту оценку на длину описания схемы при $S = \lfloor 2^n/10n \rfloor > n$ при $n \geqslant 10$. В этом случае она упрощается:

$$S \cdot (2 + \max(2 + 2\log S, 1 + \log n)) = S \cdot (2 + 2 + 2\log_2 S) \leqslant 4S \log S.$$

Таким образом, при $n \geqslant 10$ и $S = \lfloor 2^n/10n \rfloor$ всякую схему размера не больше S можно описать строкой из не более $4\frac{2^n}{10n}(n - \log_2(10n)) \leqslant 2 \cdot 2^n/5$ битов. Значит всего таких схем не больше, чем количество таких строк, то есть не больше $2^{2 \cdot 2^n/5}$. Видно, что это меньше 2^{2^n}, а значит не всякую функцию можно вычислить схемой размера $\leqslant S$. □

Мы доказали верхнюю и нижнюю оценку сложности функций n переменных, причем наши оценки достаточно близки: и та, и та оценка экспоненциальная. Если интересоваться более точным значением максимальной сложности функции, то оказывается, что наша нижняя оценка точнее, то есть можно доказать, что всякую функцию от n переменных можно вычислить схемой размера не больше $O(2^n/n)$. Однако, доказать этот факт не так просто, и мы не будем приводить здесь его доказательство.

Заметим, что наше доказательство существования трудной функции не конструктивное, “явной” функции мы не предъявляем. Оказывается, это сделать не так просто: предъявить достаточно простую функцию со сверхлинейной схемной сложностью – это открытый вопрос.

Для иллюстрации покажем, как можно доказывать хотя бы линейные нижние оценки сложности конкретных булевых функций.

Для этого рассмотрим функцию XOR_n от n переменных. Эта функция задается формулой

$$XOR_n(x_1, \ldots, x_n) = x_1 \oplus x_2 \oplus \ldots \oplus x_n.$$

Ясно, что ее можно вычислить схемой линейного размера. Действительно, можно сначала вычислить $x_1 \oplus x_2$, пользуясь схемой описанной выше, затем вычислить $(x_1 \oplus x_2) \oplus x_3$, пользуясь той же схемой, и так далее. На добавление каждой переменной требуется 5 элементов), так что общий размер схемы получается $n + 5(n-1) = 6n - 5$.

Теперь мы докажем нижнюю линейную оценку.

Лемма 14.4. *Сложность булевых схем для функции XOR_n не меньше $3n-2$.*

Доказательство. Нам будет удобнее доказать более общий факт. Давайте рассмотрим булевы схемы, в которых используются не только операции конъюнкции, дизъюнкции и отрицания, но и произвольные операции вида $(x^a \wedge x^b)^c$. Иначе говоря, мы можем использовать конъюнкции и дизъюнкции и брать при этом отрицания бесплатно, то есть не учитывать их при подсчете размера схемы. Ясно, что теперь мы рассматриваем более общие схемы, а значит доказывать нижние оценки будет только труднее.

Мы будем вести доказательство индукцией по n. И на самом деле, удобно будет усилить доказываемое утверждение: мы будем доказывать нижнюю оценку $3n-2$ одновременно для двух функций XOR_n и $\neg XOR_n$.

Итак, для $n=1$ нам нужно доказать, что размер схемы не меньше 1, что очевидно. Пусть мы доказали утверждение для n, докажем его для $n+1$. Пусть, от противного есть схема размера $S \leqslant 3(n+1)-3 = 3n$ для функции XOR_{n+1} или для функции $\neg XOR_{n+1}$. Будем также считать, что это минимальная схема, то есть схемы размера меньше S для этих функций нет. Рассмотрим первый элемент g_{n+2} этой схемы, после переменных. Ясно, что ему на вход подаются две переменные. Пусть, для определенности, это переменные x_1 и x_2. Заметим, что можно так зафиксировать переменную x_1, что элемент g_{n+2} всегда оказывается равен константе. Действительно, если g_{n+2} – конъюнкция, то нужно сделать то, к чему конъюнкция применяется (x_1 или ее отрицание), равным 0. Если же g_{n+2} – дизъюнкция, то нужно сделать то, к чему применяется дизъюнкция, равным 1. Далее, заметим, что g_{n+2} подается на вход какому-то еще элементу (иначе g_{n+2} можно было бы просто удалить из схемы, что противоречит минимальности S). Пусть это элемент g_k и второй его вход – элемент g_j. Заметим, что после того, как g_{n+2} обратился в константу, элемент g_k равен либо g_j, либо его отрицанию.

Зафиксируем переменную x_1 так, чтобы g_{n+2} стал константой. Удалим из схемы элементы g_1, g_{n+2} и g_k. Все элементы, в которые подставлялись g_1 и g_{n+2} попрежнему можно вычислить – они вычисляют либо какие-то предыдущие элементы, либо их отрицания. Если в какой-то элемент подставлялся g_k, то вместо него можно подставить g_j, либо его отрицание. Так что после удаления этих трех элементов мы снова получили некоторую булеву схему. Эта схема вычисляет функцию XOR_n или функцию $\neg XOR_n$ от оставшихся n переменных. При этом размер схемы равен $S-3 \leqslant 3n-3$. Мы получили противоречие с предположением индукции. □

До сих пор мы обсуждали размер схем. Вторым важным параметром схемы является ее глубина. Рассмотрим схему как ориентированный граф. Глубиной вершины в схеме называется длина наибольшего пути из какой-нибудь переменной в эту вершину. Глубиной схемы называется максимальная глубина вершины в этой схеме. Например, глубина схемы для функции $x_1 \oplus x_2$ на рисунке 14.1 равна 3. Глубина схемы для MAJ_3 на рисунке 14.2 также равна 3.

Неформально, глубина схемы характеризует время, необходимое для вычисления значения схемы на данном входе, если вычисление элементов схемы можно произ-

водить параллельно: на первом шаге можно параллельно вычислить все элементы глубины 1, на втором – все элементы глубины 2, и так далее, на некотором k-м шаге можно параллельно вычислить все элементы глубины k. Это можно сделать, так как на вход элементам глубины k подаются только элементы меньшей глубины, а их значения уже вычислены.

Пример 14.6. Рассмотрим функцию $x_1 \wedge x_2 \wedge \ldots \wedge x_n$. Ее можно вычислить схемой размера $n-1$, просто вычислив последовательно $x_1 \wedge x_2$, $x_1 \wedge x_2 \wedge x_3$ и так далее. Однако глубина такой схемы также равна $n-1$, поскольку каждый следующий элемент получает на вход предыдущий. Ту же функцию можно вычислить схемой глубины $\lceil \log_2 n \rceil$, если организовать схему как двоичное дерево. Разобьем переменные на пары x_1 и x_2, x_3 и x_4 и так далее. Вычислим конъюнкцию каждой пары. Все эти элементы имеют глубину 1. Повторим рассуждение: разобьем полученные элементы на пары и вычислим конъюнкцию парных элементов. Если количество переменных n есть степень двойки, то мы получим полное двоичное дерево глубины $\log_2 n$. Если количество элементов не равно степени двойки, то на некоторых шагах у некоторых элементов не будет парных и мы будем брать их конъюнкцию с самими собой (по существу, мы будем переносить их на следующий шаг). В результате получится поддерево полного двоичного дерева глубины $\lceil \log_2 n \rceil$.

То же самое рассуждение применимо к функции $x_1 \vee x_2 \vee \ldots \vee x_n$ и к функции XOR_n. Их обе можно вычислить схемой глубины $O(\log n)$.

Из приведенного выше примера и конструкции схемы для произвольной функции из леммы 14.1 видно, что всякую функцию можно вычислить схемой глубины $O(n)$. Действительно, в конструкции с дизъюнктивной нормальной формой возникают конъюнкции n элементов и дизъюнкции 2^n элементов. Если вычислять их так, как описано выше, то глубина схемы будет $O(n)$.

Обсудим подробнее глубину схем для функции “связность”.

Лемма 14.5. *Проверку графа на n вершинах на связность можно вычислить схемой глубины не больше $O(\log^2 n)$.*

Доказательство. Идея состоит в том, чтобы использовать уже построенную выше схему для этой задачи с некоторыми модификациями.

Мы будем вычислять степени матрицы смежности A' с единицами на диагонали. Во-первых, оценим за какую глубину можно по данной матрице $B \in \{0,1\}^{n \times n}$ вычислить матрицу B^2, где умножение матриц – булево. Ячейка (i,j) матрицы B^2 вычисляется по формуле

$$\bigvee_{k=1}^{n} (b_{ik} \wedge b_{kj}).$$

На вычисление всех конъюнкций нам потребуется глубина 1, а на вычисление дизъюнкции, как мы обсудили выше, достаточно глубины порядка $\log_2 n$. Поскольку все ячейки матрицы B^2 можно вычислять параллельно, мы получаем, что для возведения матрицы в квадрат достаточно глубины $O(\log n)$, если быть точными, то $\lceil \log_2 n \rceil + 1$.

Далее, как мы уже обсуждали, чтобы проверить, есть ли между двумя данными вершинами путь в графе на n вершинах, достаточно проверить, есть ли путь длины не больше n. Вместо этого нам будет удобнее проверить, есть ли в графе пути длины не больше 2^k, где $k = \lceil \log_2 n \rceil$. Ясно, что $2^k \geqslant n$, так что этого для нас будет достаточно. С другой стороны, для такой проверки нам достаточно вычислить матрицу $(A')^{2^k}$. Для этого мы можем просто последовательно возвести матрицу A' в квадрат k раз. Матрица A' нам по существу подается на вход, на каждое возведение в квадрат мы тратим глубину $O(\log n)$, так что, поскольку мы возводим в квадрат k раз, суммарная глубина есть $kO(\log n) = O(\log^2 n)$.

После этого нужно лишь взять конъюнкцию всех ячеек матрицы $(A')^{2^k}$, что можно сделать за глубину $O(\log n^2) = O(\log n)$. Так что общая глубина всей схемы есть $O(\log^2 n) + O(\log n) = O(\log^2 n)$. □

Задача 14.7. В доказательстве, чтобы вписаться в глубину $O(\log^2 n)$, мы вычислили матрицу $(A')^{2^k}$ вместо матрицы $(A')^n$. Но на самом деле, можно было обойтись и без этого трюка. Докажите, что матрицу $(A')^n$ также можно вычислить схемой глубины $O(\log^2 n)$.

Задача 14.8. В предыдущей лемме мы рассматривали задачу проверки на связность неориентированного графа. Можно рассмотреть аналогичную задачу для ориентированных графов. Докажите, что и для этой задачи есть схема глубины $O(\log^2 n)$.

Отметим, что вопрос о том, можно ли проверить граф (ориентированный или неориентированный) на связность схемой глубины $O(\log n)$, является важным открытым вопросом сложности вычислений. Можно, однако, сказать, что ориентированный случай не проще неориентированного.

Задача 14.9. Докажите, что если для задачи проверки ориентированного графа на связность есть схема глубины $O(\log n)$, то такая схема есть и для проверки неориентированного графа на связность.

В оставшейся части лекции мы сосредоточимся на схемах с одним выходом. Такие схемы от n переменных вычисляют функции вида $f\colon \{0,1\}^n \to \{0,1\}$.

14.2 Формулы

Рассмотрим булевы схемы специального вида, которые мы будем называть *формулами*. В них из каждой вершины, кроме выхода схемы, выходит ровно одно ребро. Для единообразия удобнее будет считать, что допускается несколько вершин, помеченных одной и той же переменной, и из каждой такой вершины также выходит ровно одно ребро. На рисунке 14.3 приведен пример формулы для функции $x_1 \oplus x_2$. Таким образом, формулами называются схемы, графы которых являются деревьями. В терминах последовательности функций, схема является формулой, если в ней разрешается выписывать переменные по несколько раз, и каждый элемент, кроме последнего, используется для построения одного из следующих ровно один раз.

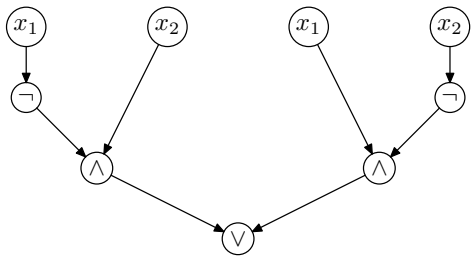

Рис. 14.3: Формула для функции $x_1 \oplus x_2$

Формулами такие схемы называются не случайно. Несложно заметить, что они соответствуют формулам булевой логики, которые мы обсуждали в начале курса. Действительно, рассмотрим формулу булевой логики ϕ, то есть формулу от переменных $x_1, \ldots, x_n$ со связками конъюнкция, дизъюнкция и отрицание. Построим по этой формуле схему. Для каждого вхождения переменной в формулу добавим в схему вершину, помеченную этой переменной (например, если переменная x_1 встречается в формуле 7 раз, то мы добавляем в схему 7 соответствующих вершин). Посмотрим на связку, которая применяется в формуле первой и добавим соответствующий элемент в схему. Посмотрим на связку, применяющуюся следующей, и добавим элемент, с такой же связкой, примененный к тем же подформулам, и так далее. Тогда элементы схемы будут вычислять подформулы формулы ϕ, а последний элемент будет равен самой ϕ. Видно, что при этом и из каждого элемента выходит ровно одна стрелка (ко всякой подформуле связка применяется только один раз).

С другой стороны, всякую схему-формулу (то есть, схему с одним выходным ребром из каждой вершины) можно обратно переделать в формулу булевой логики: можно просто последовательно во всех элементах выписать формулу, которая в этом элементе вычисляется. Для переменных соответствующая формула – просто переменная. Для каждого следующего элемента формула получается применением соответствующей связки к формулам, соответствующим предыдущим элементам.

Нетрудно видеть, что эти два сведения обратны друг к другу, то есть построив по формуле булевой логики схему, а затем по ней обратно формулу булевой логики, мы придем к той же формуле с которой начали.

Таким образом, формулы булевой логики – это в точности схемы, в которых из каждой вершины выходит не более одного ребра. Размер такой схемы при этом соответствует числу связок плюс числу вхождений переменных в формулу булевой логики. На рисунке 14.3 приведен пример схемы, соответствующей формуле $(\neg x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2)$.

Формулы – это частный случай схем. При этом всякую схему можно передe-

лать в формулу. Глубина при этом не увеличится, однако размер может вырасти экспоненциально.

Лемма 14.6. *По всякой схеме C от переменных $x_1, \ldots, x_n$ глубины d и размера s можно построить формулу, вычисляющую ту же функцию, глубины не больше d и размера не больше $2^s - 1$.*

Доказательство. Доказательство можно вести индукцией по s и d. По существу, мы доказываем утверждения для глубины и размера отдельно, но рассуждение одинаково, так что мы сделаем это параллельно. В качестве базы рассмотрим формулу, вычисляющую одну из переменных. Ее размер n и глубина равна нулю. Собственно, эта схема уже является формулой и утверждение леммы очевидно ($2^n - 1 \geqslant n$ для всякого n).

Предположим, что для схем размера меньше s (глубины меньше d) утверждение уже доказано. Рассмотрим схему размера s (глубины d). Посмотрим на ее последний элемент g_s. Он получается применением какой-то логической связки к предыдущим элементам g_i и g_j. Рассмотрим подсхемы нашей схемы, вычисляющие g_i и g_j. Их размер меньше s (глубина меньше d), так что по предположению индукции их можно вычислить формулами размера не больше $2^{s-1} - 1$ (глубины меньше d).

Добавим к этой паре формул новую вершину, соответствующую элементу g_s и проведем в нее ребра из элементов, вычисляющих g_i и g_j. Полученная схема имеет размер не больше $2 \cdot (2^{s-1} - 1) + 1 = 2^s - 1$ (глубину не больше d).

Этот же процесс можно описать и напрямую. Для этого отсортируем вершины по слоям по их глубине, то есть рассмотрим отдельно вершины глубины 1, вершины глубины 2, и так далее. Будем для всех вершин последовательно добиваться того, чтобы из них выходило не более одного ребра, при этом будем рассматривать вершины по убыванию их глубины. Из вершин глубины d ребер не выходит, так что для них условие выполняется изначально. Пусть мы уже добились выполнения условия для вершин глубины больше k. Рассмотрим вершины глубины k. Если из каких-то из них выходит больше одного ребра, то сделаем несколько копий этой вершины, по числу выходящих ребер, и из каждой копии вершины выпустим одно ребро.

Зачем мы делаем этот процесс по убыванию глубины: "размножение" вершин глубины больше k может увеличить выходную степень вершин глубины k, так что мы сначала размножаем все вершины глубины больше k, и только потом смотрим на вершины глубины k, их выходная степень уже не вырастет. □

Рассмотрим теперь вопрос о соотношении глубины и размера схем. Между этими величинами несложно доказать неравенство в одну сторону.

Лемма 14.7. *Всякая схема глубины d эквивалентна (то есть, вычисляет ту же функцию) некоторой схеме глубины d и размера не больше 2^{d+1}.*

Доказательство. На самом деле, строить эквивалентную схему по существу не нужно. Достаточно просто удалить из текущей схемы все неиспользующиеся элементы. Более точно, если в схеме есть вершина, не являющаяся выходом, из которой не

выходит ребер, то такую вершину можно безболезненно удалить из схемы. Будем удалять подобные вершины до тех пор, пока их не останется.

После этого можно утверждать, что размер схемы не больше 2^{d+1}. Действительно, можно переделать эту схему в формулу по предыдущей лемме. При этом процессе размер схемы не уменьшается. Но получившаяся в конце формула является поддеревом двоичного дерева глубины не больше d, так что в ней не больше 2^{d+1} вершин.

□

Таким образом, мы доказали, что размер схемы не больше экспоненты от глубины схемы. Можно ли доказать обратное соотношение, является открытым вопросом.

Задача 14.10. Докажите, что если для всякой схемы от n переменных размера s существует эквивалентная схема глубины не больше $O(\log s)$, то достижимость в графе можно проверить схемой глубины $O(\log n)$.

Однако, оказывается, что для частного случая формул обратное утверждение также верно.

Теорема 14.8. *Для всякой формулы размера s есть эквивалентная формула глубины $3 \log s$.*

Вспомнив, что формула – это дерево, можно заметить, что результат этой теоремы означает, что для всякой формулы есть эквивалентная формула, которая не сильно больше изначальной, и в которой ветви имеют примерно одинаковую длину.

Доказательство. В этой теореме нам уже придется всерьёз перестраивать формулу.

Для этого нам потребуется вспомогательное утверждение: нам потребуется "средняя" вершина в дереве.

Утверждение 14.9. *Пусть в формуле ϕ имеет размер s. Тогда в ней есть подформула ψ размера не меньше $s/2$, такая что все подформулы формулы ψ имеют размер меньше $s/2$.*

Доказательство этого утверждения совсем простое. Достаточно начать из выходного элемента и спускаться по ребрам формулы в подформулы так, чтобы каждый раз оставаться в подформуле размера не меньше $s/2$. В выходном элементе размер подформулы равен s, а в переменных размер подформулы равен 1, так что в какой-то момент мы не сможем спуститься из очередной вершины в подформулу. Это как раз и будет означать, что у текущей подформулы размер не меньше $s/2$, а во всех ее подформулах уже меньше.

Теперь сделаем следующее. Посмотрим на вершину, соответствующую подформуле ψ и удалим из дерева поддерево, соответствующее этой подформуле. Теперь у дерева образовался новый лист, в той вершине, где раньше была подформула ψ. Пометим эту вершину константой 0. Мы получили новую формулу, обозначим результат ее вычисления через ϕ_0. Аналогично сделаем для константы 1: пометим новый лист единицей и обозначим результат через ϕ_1.

Теперь рассмотрим следующую формулу: $(\phi_0 \wedge \psi) \vee (\phi_0 \wedge \neg\psi)$. Нетрудно видеть, что она вычисляет ту же функцию, что и изначальная формула. Теперь мы можем рассуждать индукцией по размеру формулы s. Поскольку размер формулы ψ не меньше $s/2$, то размер формул ϕ_0 и ϕ_1 не больше $s/2$. Так что по предположению индукции эти функции можно вычислить и формулами глубины $3\log_2 s - 3$. Функцию ψ можно вычислить подформулой глубины $(\log_2 s - 3) + 1$ ($\log_2 s$ достаточно для самых больших подформул ψ и еще единица нужна на вычисление последней операции в ψ). Суммарно, глубина полученной формулы не превышает $3\log_2 s$.

□

Таким образом, мы получаем, что для всякой функции $f\colon \{0,1\}^n \to \{0,1\}$ глубина минимальной схемы и логарифм размера минимальной формулы полиномиально связаны. То есть, изучение размера формул – это по существу то же самое, что изучение глубины схем.