# Логические формулы и схемы

Н. В. Верещагин А. Шень

Эта статья — сокращенный вариант главы из книги «Вычислимые функции», издание которой планируется в ближайшее время.

## 1. Высказывания и операции

«Если число $\pi$ рационально, то $\pi$ — алгебраическое число. Но оно не алгебраическое. Значит, $\pi$ не рационально». Мы не обязаны знать, что такое число $\pi$, какие числа называют рациональными и какие алгебраическими, чтобы признать, что это рассуждение правильно — в том смысле, что из двух сформулированных посылок действительно вытекает заключение. Такого рода ситуации — когда некоторое утверждение верно независимо от смысла входящий в него высказываний — составляют предмет *логики высказываний*.

Такое начало (особенно если учесть, что курс логики входит в программу философского факультета, где в свое время изучалась и «диалектическая логика») настораживает, но на самом деле наши рассмотрения будут иметь вполне точный математический характер, хотя мы начнём с неформальных мотивировок.

*Высказывания* могут быть *истинными* и *ложными*. Например, «$2^{16}+1$ — простое число» — истинное высказывание, а «$2^{32}+1$ — простое число» — ложное (это число делится на 641). Про высказывание «существует бесконечно много простых $p$, для которых $p+2$ — также простое» никто не берётся сказать наверняка, истинно оно или ложно. А фраза «$x$ делится на 2» в этом смысле не является высказыванием, пока не сказано, чему равно $x$; при разных $x$ получаются разные высказывания, одни истинные (при чётном $x$), другие — ложные (при нечётном $x$).

Высказывания можно соединять друг с другом с помощью *логических связок*. Эти связки имеют довольно странные, но традиционные названия и обозначения (табл. 1). Отметим также, что в $A \Rightarrow B$ высказывание $A$ называют *посылкой*, или *антецедентом импликации*, а $B$ — *заключением*, или *консеквентом*.

Говорят также, что высказывание имеет *истинностное значение* **И** (истина), если оно истинно, или **Л** (ложь), если оно ложно. Иногда вместо **И** употребляется буква **Т** (true) или число 1, а вместо **Л** — буква **F**

| связка | обозначение | название |
|---|---|---|
| $A$ и $B$ | $A\&B$ $A \wedge B$ $A$ `and` $B$ | конъюнкция |
| $A$ или $B$ | $A \vee B$ $A$ `or` $B$ | дизъюнкция |
| не $A$ $A$ неверно | $\neg A$ $\sim A$ $\overline{A}$ `not` $A$ | отрицание |
| из $A$ следует $B$ если $A$, то $B$ $A$ влечёт $B$ $B$ — следствие $A$ | $A \to B$ $A \Rightarrow B$ $A \supset B$ `if` $A$ `then` $B$ | импликация следование |

***Табл. 1.*** *Логические связки, обозначения и названия*

(false) или число 0. (На первый взгляд идея выбрать числа 0 и 1 произвольным образом кажется дикой — какая польза могла бы быть, скажем, от сложения истинностных значений? Удивительным образом в последние годы обнаружилось, что такая польза есть, и если оперировать с истиной и ложью как элементами конечного поля, можно получить много неожиданных результатов.)

Логические связки позволяют составлять сложные высказывания из простых. При этом истинность составного высказывания определяется истинностью его частей в соответствии с таблицей 2.

| $A$ | $B$ | $A \wedge B$ | $A \vee B$ | $A \to B$ |
|---|---|---|---|---|
| **Л** | **Л** | **Л** | **Л** | **И** |
| **Л** | **И** | **Л** | **И** | **И** |
| **И** | **Л** | **Л** | **И** | **Л** |
| **И** | **И** | **И** | **И** | **И** |

| $A$ | $\neg A$ |
|---|---|
| **Л** | **И** |
| **И** | **Л** |

***Табл. 2.*** *Таблицы истинности для логических связок*

Те же правила можно изложить словесно. Высказывание $A \wedge B$ истинно, если оба высказывания $A$ и $B$ истинны. Высказывание $A \vee B$ истинно, если хотя бы одно из высказываний $A$ и $B$ истинно. Высказывание $A \to B$ ложно в единственном случае: если $A$ истинно, а $B$ ложно. Наконец, $\neg A$ истинно в том и только том случае, когда $A$ ложно.

Из всех связок больше всего вопросов вызывает импликация. В самом деле, не очень понятно, почему надо считать, скажем, высказывания «если $2 \times 2 = 5$, то $2 \times 2 = 4$» и «если $2 \times 2 = 5$, то $3 \times 3 = 1$» истинными. (Именно

так говорят наши таблицы: **Л** $\to$ **И** = **Л** $\to$ **Л** = **И**.) Следующий пример показывает, что в таком определении есть смысл.

Общепризнанно, что если число $x$ делится на 4, то оно делится на 2. Это означает, что высказывание

$$(x \text{ делится на } 4) \ \to \ (x \text{ делится на } 2)$$

истинно при всех $x$. Подставим сюда $x = 5$: обе части ложны, а импликация истинна. При $x = 6$ посылка импликации ложна, а заключение истинно, и вся импликация истинна. Наконец, при $x = 8$ посылка и заключение истинны и вся импликация истинна. С другой стороны, обратное утверждение (если $x$ делится на 2, то $x$ делится на 4) неверно, и число 2 является контрпримером. При этом посылка импликации истинна, заключение ложно, и сама импликация ложна. Таким образом, если считать, что истинность импликации определяется истинностью её частей (а не наличием между ними каких-то причинно-следственных связей), то все строки таблицы истинности обоснованы. Чтобы подчеркнуть такое узко-формальное понимание импликации, философски настроенные логики называют её «материальной импликацией».

Теперь от неформальных разговоров перейдём к определениям. Элементарные высказывания (из которых составляются более сложные) мы будем обозначать маленькими латинскими буквами (с индексами, если понадобится) и называть *пропозициональными переменными*. Из них строятся *пропозициональные формулы* (слово «пропозициональный» для краткости будем опускать) по таким правилам:

- ▷ Всякая пропозициональная переменная есть формула.
- ▷ Если $A$ — пропозициональная формула, то $\neg A$ — пропозициональная формула.
- ▷ Если $A$ и $B$ — пропозициональные формулы, то $(A \wedge B)$, $(A \vee B)$ и $(A \to B)$ — пропозициональные формулы.

Можно ещё сказать так: формулы — это минимальное множество, обладающее указанными свойствами (слово «минимальное» здесь существенно: ведь если бы мы объявили любую последовательность переменных, скобок и связок формулой, то эти три свойства были бы тоже выполнены).

Пусть формула $\varphi$ содержит $n$ переменных $p_1, p_2, \ldots, p_n$. Если подставить вместо этих переменных истинностные значения (**И** или **Л**), то по таблицам можно вычислить истинностное значение формулы в целом. Таким образом, формула задаёт некоторую функцию от $n$ аргументов, каждый из которых может принимать значения **Л** и **И**. Значения функции также лежат в множестве $\{\mathbf{Л}, \mathbf{И}\}$, которое мы будем обозначать $\mathbb{B}$. Как

уже говорилось, мы будем следовать традиции и отождествлять **И** с единицей, а **Л** — с нулём, тем самым $\mathbb{B}$ есть $\{0,1\}$. Формула $\varphi$ задаёт отображение $\mathbb{B}^n \to \mathbb{B}$. Такие отображения называют также *булевыми функциями от $n$ аргументов*.

Пример. Рассмотрим формулу $(p \wedge (q \wedge \neg r))$. Она истинна в единственном случае — когда $p$ и $q$ истинны, а $r$ ложно (см. табл. 3).

| $p$ | $q$ | $r$ | $\neg r$ | $(q \wedge \neg r)$ | $(p \wedge (q \wedge \neg r))$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 |

***Табл. 3.*** *Таблица истинности конъюнкции* $(p \wedge (q \wedge \neg r))$

Некоторые формулы выражают логические законы — составные высказывания, истинные независимо от смысла их частей. Такие формулы (истинные при всех значениях входящих в них переменных) называют *тавтологиями*.

Пример. Формула $((p \wedge q) \to p)$ является тавтологией (это можно проверить, например, составив таблицу). Она выражает такой логический закон: из конъюнкции утверждений следует первое из них.

Задача 1. Как выглядит симметричное утверждение для дизъюнкции?

Две формулы называют *эквивалентными*, если они истинны при одних и тех же значениях переменных (другими словами, если они задают одну и ту же булеву функцию). Например, формула $(p \wedge (p \to q))$ истинна лишь при $p = q =$ **И** и потому эквивалентна формуле $(p \wedge q)$.

Рассмотрим формулу $((p \vee q) \wedge q)$. Она истинна, если и только если переменная $q$ истинна. Хотелось бы сказать, что она эквивалентна формуле $q$, но тут есть формальная трудность: она содержит две переменные и потому задаёт функцию от двух аргументов (типа $\mathbb{B} \times \mathbb{B} \to \mathbb{B}$), в то время как $q$ задаёт функцию от одного аргумента. Мы не будем обращать на это внимания и будем считать эти формулы эквивалентными. Вообще, если есть список переменных $p_1, \ldots, p_n$, содержащий все переменные некоторой формулы $\varphi$ (и, возможно, ещё какие-то переменные), можно

считать, что формула $\varphi$ задаёт функцию от $n$ аргументов, возможно, на деле зависящую не от всех аргументов (постоянную по некоторым аргументам).

После таких оговорок легко проверить следующий факт: формулы $\varphi$ и $\psi$ эквивалентны тогда и только тогда, когда формула $((\varphi \to \psi) \wedge (\psi \to \varphi))$ является тавтологией. Используя сокращение $(p \leftrightarrow q)$ для $((p \to q) \wedge (q \to p))$, можно записывать утверждения об эквивалентности формул в виде тавтологий. Вот несколько таких эквивалентностей:

Теорема 1. *Следующие формулы являются тавтологиями:*

$$(p \wedge q) \leftrightarrow (q \wedge p);$$
$$((p \wedge q) \wedge r) \leftrightarrow (p \wedge (q \wedge r));$$
$$(p \vee q) \leftrightarrow (q \vee p);$$
$$((p \vee q) \vee r) \leftrightarrow (p \vee (q \vee r));$$
$$(p \wedge (q \vee r)) \leftrightarrow ((p \wedge q) \vee (p \wedge r));$$
$$(p \vee (q \wedge r)) \leftrightarrow ((p \vee q) \wedge (p \vee r));$$
$$\neg(p \wedge q) \leftrightarrow (\neg p \vee \neg q);$$
$$\neg(p \vee q) \leftrightarrow (\neg p \wedge \neg q);$$
$$(p \vee (p \wedge q)) \leftrightarrow p;$$
$$(p \wedge (p \vee q)) \leftrightarrow p;$$
$$(p \to q) \leftrightarrow (\neg q \to \neg p);$$
$$p \leftrightarrow \neg\neg p.$$

Доказательство. Первые четыре эквивалентности выражают коммутативность и ассоциативность конъюнкции и дизъюнкции. Проверим, например, вторую: левая и правая части истинны в единственном случае (когда все переменные истинны), и потому эквивалентны. (Для дизъюнкции удобнее смотреть, когда она ложна.)

Две следующие эквивалентности утверждают дистрибутивность — заметим, что в отличие от сложения и умножения в кольцах здесь верны оба свойства дистрибутивности. Проверить эквивалентность легко, если отдельно рассмотреть случаи истинного и ложного $p$.

Следующие два свойства называются *законами де Моргана*. Их легко проверить, вспомнив, что конъюнкция истинна, а дизъюнкция ложна ровно в одном случае. Законы де Моргана иногда выражают словами: «конъюнкция двойственна дизъюнкции».

Далее следуют два очевидных *закона поглощения* (один из них мы уже упоминали).

За ними идёт правило *контрапозиции*, которое говорит, в частности, что утверждения «если $x$ совершенно, то $x$ нечётно» и «если $x$ нечётно,

то $x$ несовершенно» равносильны. Хотя оно и очевидно проверяется с помощью таблиц истинности, с ним связан любопытный парадокс. Вот он. Биолог А выдвинул гипотезу: все вороны чёрные. Проверяя её, он вышел во двор и обнаружил на дереве ворону. Она оказалось чёрной. Вроде бы у него есть основания радоваться — гипотеза подтверждается. Биолог Б переформулировал гипотезу так: все не-чёрные предметы — не вороны (применив наше правило контрапозиции) и не стал выходить во двор, а открыл холодильник и нашёл там оранжевый предмет. Он оказался апельсином, а не вороной. Биолог Б обрадовался — гипотеза подтверждается — и позвонил биологу А. Тот удивляется — у него тоже есть апельсин в холодильнике, но с его точки зрения никакого отношения к его гипотезе апельсин не имеет...

Последнее (и очевидное) правило называется *снятием двойного отрицания.*

Задача 2. Перечисленные эквивалентности соответствуют равенствам для множеств: например, первая гарантирует, что $P \cap Q = Q \cap P$ для любых множеств $P$ и $Q$. Какие утверждения соответствуют остальным эквивалентностям?

Задача 3. Две формулы, содержащие только переменные и связки $\wedge$, $\vee$ и $\neg$, эквивалентны. Докажите, что они останутся эквивалентными, если всюду заменить $\wedge$ на $\vee$ и наоборот.

Заметим, что далеко не все тавтологии имеют ясный интуитивный смысл. Например, формула $(p \to q) \vee (q \to p)$ является тавтологией (если одно из утверждений $p$ и $q$ ложно, то из него следует всё, что угодно; если оба истинны, то тем более формула истинна), хотя и отчасти противоречит нашей интуиции — почему, собственно, из двух никак не связанных утверждений одно влечёт другое? Ещё более загадочна тавтология

$$((p \to q) \to p) \to p$$

(хотя её ничего не стоит проверить с помощью таблиц истинности).

Отступление о пользе скобок. На самом деле наши рассуждения содержат серьёзный пробел. Чтобы обнаружить его, зададим себе вопрос: зачем нужны скобки в формулах? Представим себе, что мы изменим определение формулы, и будем говорить, что $P \wedge Q$ и $P \vee Q$ являются формулами для любых $P$ и $Q$. Останутся ли наши рассуждения в силе?

Легко понять, что мы столкнёмся с трудностью при определении булевой функции, соответствующей формуле. В этом определении мы подставляли нули и единицы на место переменных и затем вычисляли значение формулы с помощью таблиц истинности для связок. Но теперь, когда мы изменили определение формулы, формула $p \wedge q \vee r$ может быть получена

двумя способами — из формул $p \wedge q$ и $r$ с помощью операции $\vee$ и из формул $p$ и $q \vee r$ с помощью операции $\wedge$. Эти два толкования дадут разный результат при попытке вычислить значение $0 \wedge 0 \vee 1$.

Из сказанного ясно, что скобки нужны, чтобы гарантировать однозначность синтаксического разбора формулы. Точнее говоря, верно такое утверждение:

Теорема 2 (однозначность разбора). *Пропозициональная формула, не являющаяся переменной, может быть представлена ровно в одном из трёх видов $(A \wedge B)$, $(A \vee B)$ или $\neg A$, где $A$ и $B$ — некоторые формулы, причём $A$ и $B$ (в первых двух случаях) восстанавливаются однозначно.*

Доказательство. Формальное доказательство можно провести так: определим понятие *скобочного итога* как разницы между числом открывающихся и закрывающихся скобок. Индукцией по построению формулы легко доказать такую лемму:

> Скобочный итог любой формулы равен нулю. Скобочный итог любого начала формулы неотрицателен и равен нулю, лишь если это начало совпадает со всей формулой, пусто или состоит из одних символов отрицания.

Слова «индукцией по построению» значат вот что: мы проверяем интересующее нас утверждение для переменных, а также доказываем, что если оно верно для формул $A$ и $B$, то оно верно и для формул $(A \wedge B)$, $(A \vee B)$ и $\neg A$.

После того как лемма доказана, разбор формулы проводится так: если она начинается с отрицания, то может быть образована лишь по третьему правилу. Если же она начинается со скобки, то надо скобку удалить, а потом искать начало, имеющее нулевой скобочный итог. Такое начало единственно. (Это легко проверить, используя лемму.) Тем самым формула разбирается однозначно.

В дальнейшем мы будем опускать скобки, если они либо не играют роли (например, можно написать конъюнкцию трёх членов, не указывая порядок действий в силу ассоциативности), либо ясны из контекста.

Задача 4. Польский логик Лукасевич предлагал обходиться без скобок, записывая в формулах сначала знак операции, а потом операнды (без пробелов и разделителей). Например, $(a + b) \times (c + (d \times e))$ в его обозначениях запишется как $\times + ab + c \times de$. Эту запись ещё называют *польской* записью. Обратная польская запись отличается от неё тем, что знак операции идёт после операндов. Покажите, что в обоих случаях порядок действий восстанавливается однозначно.

## 2. Полные системы связок

Рассматриваемая нами система пропозициональных связок $(\wedge, \vee, \neg)$ *полна* в следующем смысле:

Теорема 3 (Полнота системы $\vee, \wedge, \neg$). *Любая булева функция от $n$ аргументов может быть записана в виде пропозициональной формулы.*

Доказательство. Проще всего пояснить это на примере. Пусть, например, булева функция $\varphi(p, q, r)$ задана таблицей 4.

| $p$ | $q$ | $r$ | $\varphi(p,q,r)$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

$$(\neg p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge q \wedge r) \vee (p \wedge q \wedge r)$$

***Табл. 4.*** *Таблица значений булевой функции и задающая её формула*

В таблице есть три строки с единицами в правой колонке — три случая, когда булева функция истинна (равна 1). Напишем три конъюнкции, каждая из которых покрывает один случай (а в остальных строках ложна), и соединим их дизъюнкцией. Нужная формула построена.

Ясно, что аналогичная конструкция применима и для любой таблицы (и с любым числом переменных).

Для формул подобного вида есть специальное название: формулы в *дизъюнктивной нормальной форме*. Более подробно: *литералом* называется переменная или отрицание переменной, *конъюнктом* называется произвольная конъюнкция литералов, а дизъюнктивной нормальной формой называется дизъюнкция конъюнктов. В нашем случае в каждый конъюнкт входит $n$ литералов (где $n$ — число переменных), а число конъюнктов равно число строк с единицами и может меняться от нуля (тогда, правда, получается не совсем формула, а «пустая дизъюнкция», и её можно заменить какой-нибудь всегда ложной формулой типа $p \wedge \neg p$) до $2^n$ (если булева функция всегда истинна).

Задача 5. Длина построенной в доказательстве теоремы 3 формулы зависит от числа единиц: формула будет короткой, если единиц в таблице

мало. А как написать (сравнительно) короткую формулу, если в таблице мало нулей, а в основном единицы?

Иногда используется *конъюнктивная нормальная форма*, которая представляет собой конъюнкцию *дизъюнктов*, каждый из которых состоит из литералов, связанных дизъюнкциями. Теорему 3 можно теперь усилить так:

Теорема 4. *Всякая булева функция может быть выражена формулой, находящейся в дизъюнктивной нормальной форме, а также формулой, находящейся в конъюнктивной нормальной форме.*

Доказательство. Первая часть утверждения уже доказана. Вторую часть можно доказать аналогично первой, надо только для каждой строки с нулём написать подходящий дизъюнкт.

Можно также представить функцию $\neg\varphi$ в дизъюнктивной нормальной форме, а затем воспользоваться правилами де Моргана, чтобы внести отрицание внутрь.

Задача 6. Проведите второй вариант рассуждения подробно.

Вообще говоря, определение дизъюнктивной нормальной формы не требует, чтобы в каждом конъюнкте (или дизъюнкте) встречались все переменные. (Повторять переменную больше одного раза смысла нет; если, например, формула и её отрицание входят в одну конъюнкцию, то эта конъюнкция всегда ложна и её можно выбросить.)

Задача 7. Приведите пример булевой функции от $n$ аргументов, у которой любая дизъюнктивная или конъюнктивная нормальная форма содержит лишь члены длины $n$. (Указание: рассмотрите функцию, которая меняет своё значение при изменении значения любой переменной.)

Заметим, что при доказательстве теоремы 3 мы обошлись без импликации. Это и не удивительно, так как она выражается через дизъюнкцию и отрицание:

$$(p \to q) \;\leftrightarrow\; (\neg p \vee q)$$

(проверьте!). Вообще-то мы могли бы обойтись только конъюнкцией и отрицанием, так как

$$(p \vee q) \;\leftrightarrow\; \neg(\neg p \wedge \neg q),$$

или только дизъюнкцией и отрицанием, так как

$$(p \wedge q) \;\leftrightarrow\; \neg(\neg p \vee \neg q),$$

(обе эквивалентности вытекают из законов де Моргана; их легко проверить и непосредственно). Можно сказать, что система связок $\wedge, \neg$, а также система связок $\vee, \neg$ являются *полными*. (По определению это означает, что с их помощью можно записать любую булеву функцию).

Задача 8. Докажите, что система связок $\neg, \to$ полна. (Указание: как записать через них дизъюнкцию?)

А вот без отрицания обойтись нельзя. Система связок $\wedge, \vee, \to$ неполна — и по очень простой причине: если все переменные истинны, то любая их комбинация, содержащая только указанные связки, истинна. (Как говорят, все эти связки «сохраняют единицу».)

Задача 9. Легко понять, что любая формула, составленная только с помощью связок $\wedge$ и $\vee$, задаёт монотонную булеву функцию (в том смысле, что от увеличения значения любого из аргументов значение функции может только возрасти — или остаться прежним). Покажите, что любая монотонная булева функция может быть выражена формулой, содержащей только $\wedge$ и $\vee$.

Задача 10. Пусть $\varphi \to \psi$ — тавтология. Покажите, что найдётся формула $\tau$, которая включает в себя только общие для $\varphi$ и $\psi$ переменные, для которой формулы $\varphi \to \tau$ и $\tau \to \psi$ являются тавтологиями. (Более общий вариант этого утверждения, в котором формулы берутся в языке первого порядка, называется леммой Крейга.)

В принципе мы не обязаны ограничиваться четырьмя рассмотренными связками. Любая булева функция может играть роль связки. Например, можно рассмотреть связку ($p$ `notand` $q$), задаваемую эквивалентностью

$$(p \texttt{ notand } q) \leftrightarrow \neg(p \wedge q)$$

(словами: ($p$ `notand` $q$) ложно, лишь если $p$ и $q$ истинны). Через неё выражается отрицание ($p$ `notand` $p$), после чего можно выразить конъюнкцию, а затем, как мы знаем, и вообще любую функцию. (Знакомые с цифровыми логическими схемами малого уровня интеграции хорошо знакомы с этим утверждением: достаточно большой запас схем И-НЕ позволяет реализовать любую требуемую зависимость выхода от входов.)

Другая интересная полная система связок — это сложение по модулю 2, конъюнкция и константа 1 (которую можно считать 0-арной связкой, задающей функцию от нуля аргументов).

Назовём *мономом* конъюнкцию любого набора переменных или константу 1 (которую естественно рассматривать как конъюнкцию нуля переменных). Название это естественно, так как при наших соглашениях (1 — истина, 0 — ложь) конъюнкция соответствует умножению.

Назовём *полиномом* сумму таких мономов по модулю 2 (это значит, что $0+0=0$, $0+1=1+0=1$ и $1+1=0$). Ясно, что два повторяющихся монома можно сократить (ведь сложение по модулю 2), так что будем рассматривать только полиномы без повторяющихся мономов. При этом,

естественно, порядок членов в мономе и порядок мономов в полиноме несущественен, их можно переставлять.

Теорема 5 (полиномы Жегалкина). *Всякая булева функция однозначно представляется полиномом указанного вида.*

Доказательство. Чтобы доказать существование искомого полинома, можно сослаться на известное из алгебры утверждение, что всякая функция с аргументами из конечного поля (в данном случае это двухэлементное поле вычетов по модулю 2) задаётся полиномом. Правда, в алгебре понятие полинома более общее, разрешены степени. Но это не важно, так как переменные принимают лишь значения 0 и 1 и потому степени роли не играют.

Далее можно заметить, что полиномов столько же, сколько булевых функций, а именно $2^{2^n}$. В самом деле, булева функция может принимать любое из двух значений в каждой из $2^n$ точек булева куба $\mathbb{B}^n$, а многочлен может включать или не включать любой из $2^n$ мономов. (Мономов столько, потому что каждой моном включает или не включает любую из $n$ переменных.) Поэтому избытка полиномов нет, и если любая функция представима полиномом, то единственным образом.

Можно и не ссылаться на сведения из алгебры, а дать явную конструкцию. Это удобно сделать индукцией по $n$. Пусть мы уже умеем представлять любую булеву функцию от $n-1$ аргументов с помощью полинома. Тогда $\varphi(p_1,\ldots,p_n)$ можно представить как

$$\begin{aligned}\varphi(p_1,\ldots,p_n) =& \varphi(0,p_2,\ldots,p_n) + \\ &+ [\varphi(0,p_2,\ldots,p_n) + \varphi(1,p_2,\ldots,p_n)]p_1\end{aligned}$$

(проверьте, что всё сходится). Остаётся заметить, что правую часть можно представить полиномом по предположению индукции.

Для единственности также есть другое доказательство: пусть два многочлена равны. Тогда их сумма (или разность — вычисления происходят по модулю 2) является ненулевым многочленом (содержит какие-то мономы), но тождественно равна нулю. Так не бывает, и это легко доказать по индукции. В самом деле, любой многочлен $A(p_1,\ldots,p_n)$ можно представить в виде

$$A(p_1,\ldots,p_n) = B(p_2,\ldots,p_n) + p_1 C(p_2,\ldots,p_n),$$

где $B$ и $C$ — многочлены от меньшего числа переменных. Подставляя сначала $p_1 = 0$, а затем $p_1 = 1$, убеждаемся, что многочлены $B$ и $C$ равны нулю во всех точках, и потому (согласно предположению индукции) равны нулю как многочлены (не содержат мономов).

Задача 11. Назовём *мультилинейной* функцией полином от $n$ переменных, в котором все показатели степеней равны либо 0, либо 1. (Таким

образом, каждый моном в ней есть произведение коэффициента и некоторого набора переменных без повторений). Пусть $F$ — произвольное поле. Будем рассматривать $\mathbb{B} = \{0, 1\}$ как подмножество $F$. Докажите, что всякая булева функция $\mathbb{B}^n \to \mathbb{B}$ однозначно продолжается до мультилинейной функции $F^n \to F$, и коэффициенты в мультилинейной функции будут целыми.

Если рассматривать произвольные булевы функции в качестве связок, возникает вопрос: в каком случае они образуют полный базис? Ответ дает следующая теорема.

Теорема 6 (критерий Поста). *Набор булевых функций тогда и только тогда является полным (это значит, что любая булева функция представляется в виде их композиции, т. е. записывается в виде формулы, где связками служат функции набора), когда он не содержится ни в одном из пяти «предполных классов»*:

- ▷ *монотонные функции*;
- ▷ *функции, сохраняющие нуль*;
- ▷ *функции, сохраняющие единицу*;
- ▷ *линейные функции*;
- ▷ *самодвойственные функции.*

(Функция $f$ *монотонна*, если она монотонно неубывает по каждому из своих аргументов. Функция $f$ *сохраняет нуль/единицу*, если $f(0, \ldots, 0) = 0$ (соответственно $f(1, \ldots, 1) = 1$). Функция $f$ *линейна*, если она представима многочленом, в котором все мономы содержат не более одной переменной. Наконец, функция $f$ называется *самодвойственной*, если $f(1 - p_1, \ldots, 1 - p_n) = 1 - f(p_1, \ldots, p_n)$.)

Если набор содержится в одном из классов, то и все композиции также не выходят за пределы этого класса (легко проверить для каждого из классов в отдельности) и поэтому набор не является полным. Доказательство обратного утверждения опустим (читатель может попробовать доказать его самостоятельно).

## 3. Схемы из функциональных элементов

Формулы представляют собой способ записи композиции функций. Например, если мы сначала применяем функцию $f$, а потом функцию $g$, это можно записать формулой $g(f(x))$. Но есть и другой способ: можно изобразить каждую функцию в виде прямоугольника с «входом» и «выходом» и соединить выход функции $f$ со входом функции $g$ (рис. 1).

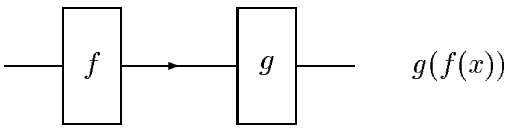


***Рис. 1.*** *Два способа изобразить композицию*

Такое представление отнюдь не является чисто теоретическим. В течение нескольких десятков лет электронная промышленность выпускает микросхемы, которые выполняют логические операции. Такая микросхема имеет электрические контакты, напряжение на которых кодирует логические значения **И** и **Л**. Конкретное напряжение зависит от типа схемы, но обычно это единицы вольт, и высокий потенциал (относительно заземления) считается единицей, а низкий — нулём.

Одной из типичных схем является схема И-НЕ, она имеет два входа и один выход. Сигнал на выходе является отрицанием конъюнкции сигналов на входе. Другими словами, на выходе появляется высокий потенциал (сигнал 1) тогда и только тогда, когда на одном из входов потенциал низкий (0). Из такой схемы легко получить схему НЕ (изменяющую уровень сигнала на противоположный), соединив проводом два входа. При этом на оба входа поступает один и тот же сигнал, и операция И его не меняет ($p \wedge p = p$), а НЕ меняет на противоположный. Взяв два элемента и используя второй из них в качестве элемента НЕ, меняющего сигнал с выхода первого элемента, получаем схему, которая реализует функцию И. А если поставить два элемента НЕ перед каждым из входов элемента И-НЕ, получим схему, реализующую функцию ИЛИ: $\neg(\neg p \wedge \neg q) \leftrightarrow (p \vee q)$.

Теорема 3 о полноте системы связок теперь гарантирует, что любую булеву функцию можно реализовать в виде схемы. Надо иметь в виду, однако, что предлагаемая в её доказательстве конструкция (дизъюнктивная нормальная форма) имеет скорее теоретический интерес, поскольку приводит к схемам очень большого размера даже для простых функций (если число аргументов велико). Например, схема, сравнивающая два 16-битовых числа, должна иметь 32 входа и поэтому в её реализации с помощью дизъюнктивной нормальной формы будет порядка $2^{32}$ элементов — что мало реально. (Между тем такую схему можно построить гораздо проще, из нескольких сотен элементов.)

Поэтому вопрос о том, сколько элементов нужно для реализации той или иной функции, представляет большой интерес — как практический, так и философский. (Одна из центральных проблем математики и информатики, так называемая «проблема перебора», может быть сформулирована в этих терминах.)

Мы сейчас дадим более формальное определение схемы и реализуемой ею булевой функции. Но прежде всего ответим на такой вопрос — почему мы вообще говорим о схемах? Ведь можно записать композицию булевых функций в виде формулы, не будет ли это то же самое?

Оказывается, не совсем, и разницу легко видеть на примере (рис. 2).

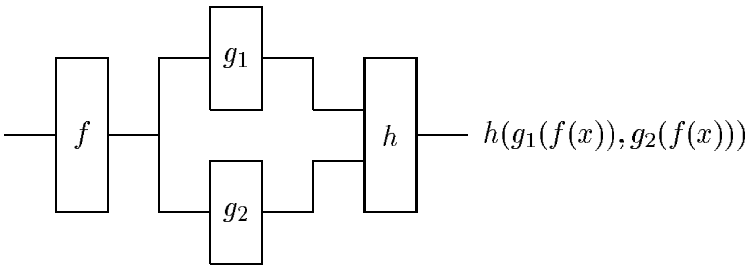


***Рис. 2.*** *Элемент входит в формулу дважды*

Здесь один и тот же элемент схемы ($f$) приходится указывать в формуле дважды, поскольку его выход используется в качестве входа двух других элементов. Схемы, в которых такого ветвления нет (на практике оно вполне возможно, хотя и ограничено «нагрузочной способностью выхода», как говорят инженеры), как раз и соответствуют формулам. Но в общем случае формула может быть длинной, даже если схема содержит небольшое число элементов, поскольку число копий может расти экспоненциально с ростом глубины схемы.

Хотя идея образования схемы из функциональных элементов, реализующих булевы функции, достаточно наглядна, дадим более формальное определение. Пусть имеется $n$ булевых переменных $x_1, \ldots, x_n$, называемых *входами*. Пусть также имеется некоторое число переменных $y_1, \ldots, y_m$, называемых *проводниками*. Пусть для каждого проводника схемы задана булева функция из некоторого множества $B$, выражающая его значение через другие проводники и входы. При этом требуется, чтобы не было циклов (когда $y_i$ зависит от $y_j$, которое зависит от $y_k$, ..., которое зависит от $y_i$). Пусть, кроме того, среди проводников выделен один, называемый *выходом*. В таком случае говорят, что задана *схема размера $m$ из функциональных элементов в базисе $B$* с $n$ входами. (С точки зрения инженера размер — это число использованных элементов, а базис $B$ — это ассортимент доступных ему элементов.)

Отсутствие циклов гарантирует, что есть проводник, зависящий только от входов (иначе можно было бы прийти к циклу: возьмём какой-то проводник, затем возьмём тот проводник, от которого он зависит, и т. д.). Его значение, таким образом, однозначно определяется сигнала-

ми на входах. Среди оставшихся проводников также нет цикла, поэтому можно найти один из них, зависящий только от уже известных, и определить его значение. Перенумеровав проводники в таком порядке, мы можем записать последовательность присваиваний (программу)

$$\begin{aligned} y_1 &:= f_1(\ldots);\\ y_2 &:= f_2(\ldots);\\ &\ldots\\ y_m &:= f_m(\ldots); \end{aligned}$$

в правых частях которых стоят функции из $B$, применённые ко входам и уже найденным значениям. При этом можно считать, что результат схемы есть $y_m$ (все последующие присваивания уже не нужны). Такая программа определяет $y_m$ при известных значениях входов и тем самым *вычисляет* некоторую булеву функцию.

Набор булевых функций $B$ называется *полным*, если любая булева функция может быть задана схемой из $B$-элементов (существует программа, её вычисляющая, при этом в правых частях присваиваний стоят функции из $B$). Ясно, что это равносильно возможности записать булеву функцию в виде формулы со связками из $B$ (как мы говорили, разница только в том, что один и тот же элемент будет фигурировать в формуле многократно).

*Сложностью* булевой функции $f$ относительно $B$ называется минимальный размер схемы из $B$-элементов, вычисляющей функцию $f$. Этот размер будем обозначать $\mathrm{size}_B(f)$.

Теорема 7. *Пусть $B_1$ и $B_2$ — два полных набора булевых функций. Тогда соответствующие им сложности отличаются не более чем на постоянный множитель: найдётся такое число $C$, что $\mathrm{size}_{B_1}(f) \leqslant C\,\mathrm{size}_{B_2}(f)$ и $\mathrm{size}_{B_2}(f) \leqslant C\,\mathrm{size}_{B_1}(f)$ для любой функции $f$.*

Доказательство. Утверждение почти очевидно: поскольку наборы $B_1$ и $B_2$ полны, то каждая функция одного из наборов может быть вычислена какой-то программой, составленной из функций другого набора. Теперь можно взять в качестве $C$ наибольшую длину таких программ, и неравенства будут выполняться: каждую строку программы можно заменить на $C$ (или меньше) строк с использованием функций другого набора.

Что можно сказать о сложности произвольной булевой функции от $n$ аргументов? Следующая теорема показывает, что она экспоненциально зависит от $n$ (для «наугад взятой» функции).

Теорема 8. *а) Сложность любой булевой функции от $n$ аргументов не превосходит $C^n$ для некоторого константы $C$. б) Сложность*

*большинства булевых функций от $n$ аргументов не меньше $c^n$ для некоторой константы $c$.*

Доказательство. Первое утверждение очевидно: размер схемы, реализующей дизъюнктивную нормальную форму, есть $O(n2^n)$ (имеется не более $2^n$ конъюнктов размера $O(n)$).

Чтобы доказать второе утверждение, оценим число различных схем с $n$ аргументами размера $N$. Каждая такая схема может быть описана последовательностью из $N$ присваиваний, выражающих одну из переменных через предыдущие. Для каждой формулы есть не более $3(N+n)^2$ вариантов (три типа операций — конъюнкция, дизъюнкция, отрицание, и каждый из не более чем двух аргументов выбирается среди не более чем $N+n$ вариантов). Отсюда легко получить оценку $2^{O(N\log N)}$ на число всех функций сложности не более $N$ (считая $N \geqslant n$).

Всего булевых функций с $n$ аргументами имеется $2^{2^n}$. Из сравнения этих формул видно, что что при $c < 2$ и при достаточно больших $n$ булевы функции сложности меньше $c^n$ составляют меньшинство, так как $2^{O(c^n \log c^n)}$ много меньше $2^{2^n}$. (Уменьшая константу $c$, можно добиться, чтобы они составляли меньшинство при всех $n$, а не только достаточно больших.)

Эта теорема, однако, ничего не говорит о сложности конкретных булевых функций. Ситуация здесь такова. Есть разнообразные методы и приёмы получения верхних оценок. Но про нижние оценки неизвестно практически ничего. Про многие функции мы подозреваем, что их сложность велика (экспоненциальна), но доказать это пока не удаётся. Весьма нетривиальные идеи позволяют доказывать экспоненциальные нижние оценки для некоторых специальных классов схем, например, схем из монотонных элементов или схем ограниченной глубины (использующих элементы И и ИЛИ с произвольным числом входов). Получение экспоненциальных оценок для более общих схем — один из возможных подходов к знаменитой *проблеме перебора*, центральной проблеме теории сложности вычислений.

Мы не будем углубляться в эту теорию, а приведём лишь несколько верхних оценок для конкретных задач. При этом мы не претендуем на полноту, а хотим лишь показать несколько интересных идей и приемов.

Рассмотрим уже упоминавшуюся функцию сравнения двух $n$-битовых чисел. Она имеет $2n$ аргументов ($n$ для одного числа и $n$ для другого). Обозначим эту функцию $\mathrm{Comp}_n$.

Теорема 9. *Пусть $B$ — полный набор функций. Существует такая константа $C$, что* $\mathrm{size}_B(\mathrm{Comp}_n) \leqslant Cn$.

Доказательство. Заметим, что поскольку в формулировке теоремы оценка размера проводится с точностью до константы, то выбор конкретного базиса не имеет значения. Другими словами, мы можем предполагать, что любое конечное число необходимых нам функций в этом базисе есть.

Схема сравнения чисел будет рекурсивной (чтобы сравнить два числа, мы отдельно сравниваем их левые и правые половины, а затем объединяем результаты). При этом, как часто бывает, надо усилить утверждение, чтобы индукция прошла. А именно, мы будем строить схему с $2n$ входами $x_1, \ldots, x_n, y_1, \ldots, y_n$ и двумя выходами, которая указывает, какой из трёх случаев $x < y$, $x = y$ или $x > y$ имеет место. (Здесь $x$ — число, записываемое в двоичной системе как $x_1 \ldots x_n$). Два выходных бита кодируют четыре возможности, а нужно только три, так что есть некоторый запас. Для определённости можно считать, что первый выходной бит истинен, если числа равны, а второй — если $x < y$. Тогда возможны три варианта сигналов на выходе: 10 (равенство), 01 (при $x < y$) и 00 (при $x > y$).

Объясним теперь, как собрать, скажем, схему сравнения двух 16-битовых чисел. Соберём отдельно схему сравнения старших 8 битов и младших 8 битов. Каждая из них даст ответ в форме двух битов. Теперь из этих четырёх битов надо собрать два. (Если в старших разрядах неравенство, то оно определяет результат сравнения; если старшие разряды равны, то результат сравнения определяется младшими разрядами.) Написанная в скобках фраза определяет булеву функцию с четырьмя битами на входе и двумя битами на выходе, и может быть реализована некоторой схемой фиксированного размера. Таким образом, если через $T(n)$ обозначить размер схемы, сравнивающей $n$-битовые числа, то получаем оценку

$$T(2n) \leqslant 2T(n) + c,$$

где $c$ — некоторая константа, зависящая от выбора базиса. Отсюда следует, что $T(2^k) \leqslant c'2^k$ при некотором $c'$. В самом деле, для достаточно большого $c'$ можно доказать по индукции, что

$$T(m) \leqslant c'm - c$$

(мы должны усилить неравенство, вычтя из правой части $c$, чтобы индуктивный шаг прошёл; база индукции остается верной, если $c'$ достаточно велико).

Ту же самую оценку можно объяснить и наглядно. Наша схема имеет вид иерархического дерева. На каждом уровне из двух двухбитовых сигналов получается один. Остаётся вспомнить, что в полном двоичном дереве число внутренних вершин (которое определяет размер схемы) на единицу меньше числа листьев. (В турнире по олимпийской системе число

игр на единицу меньше числа команд, так как после каждой игры одна команда выбывает.)

Каждая внутренняя вершина и каждый лист (где сравниваются два бита) представляют собой схемы ограниченного размера, откуда и вытекает оценка $T(2^k) \leqslant c'2^k$.

Осталось лишь сказать, что делать, если размер чисел (который мы обозначали через $n$) не есть точная степень двойки. В этом случае можно увеличить размер до ближайшей сверху степени двойки (не более чем в два раза) и подать на старшие разряды входов нули. Оба действия приводят к увеличению размера схемы не более чем в константу раз.

Теперь рассмотрим задачу о сложении двух $n$-битовых чисел. (Строго говоря, тут возникает не булева функция, а функция $\mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}^{n+1}$, но все наши определения очевидно переносятся на этот случай.)

Теорема 10. *Существует схема сложения двух $n$-битовых чисел размера $O(n)$.*

Доказательство. Напомним смысл обозначения $O(n)$: нам надо построить схему сложения $n$-битовых чисел, имеющую размер не более $cn$ для некоторого $c$ и для всех $n$.

Это совсем просто. Посмотрим на сложение в столбик:

$$\begin{array}{ccccc} & 0 & 1 & 1 & \\ & 1 & 0 & 0 & 1 \\ & 1 & 0 & 1 & 1 \\ \hline 1 & 0 & 1 & 0 & 0 \end{array}$$

Верхняя строка — биты переноса, нижняя — результат. Заметим, что каждый из битов переноса или результата определяется тремя другими битами (бит результата равен сумме двух битов слагаемых и бита переноса по модулю 2, а бит переноса равен 1, если хотя бы два из этих трёх битов равны 1). Поэтому можно составить схему, которая вычисляет эти биты справа налево и имеет размер $n$.

Заметим, что предыдущее утверждение (теорема 9) является следствием этого: чтобы сравнить числа $x$ и $y$, сложим число $(2^n - 1) - x$ (то есть число $x$, в котором все единицы заменены нулями и наоборот) и число $y$. Если в старшем разряде появится единица, значит, $y > x$, а если нет, то $y \leqslant x$. Остаётся заметить, что и сложение, и обращение битов в числе $x$ реализуются схемами линейного размера.

Тем не менее конструкция, использованная при доказательстве теоремы 9, имеет некоторые преимущества. Назовём *глубиной* схемы максимальное число элементов на пути от входа к выходу. Если представить себе, что сигнал на выходе элемента появляется не сразу после подачи

сигналов на входы, а с некоторой задержкой, то глубина схемы определяет суммарную задержку. Легко понять, что рекурсивная схема сравнения имела глубину $O(\log n)$ (число уровней пропорционально логарифму размера входа), в то время как построенная только что схема сложения имеет глубину, пропорциональную $n$ (биты переноса вычисляются последовательно, справа налево). Но можно соединить эти два результата:

Теорема 11. *Существует схема сложения двух $n$-битовых чисел размера $O(n)$ и глубины $O(\log n)$.*

Доказательство. Как мы видели, проблема в том, что биты переноса вычисляются последовательно, а не параллельно. Если удастся их все вычислить схемой размера $O(n)$ и глубины $O(\log n)$, то дальнейшее очевидно.

Как мы уже говорили, вычисление битов переноса равносильно сравнению, так что достаточно научиться сравнить параллельно все «суффиксы» чисел, т. е. для каждого $i$ сравнить числа $x_i x_{i+1} \ldots x_n$ и $y_i y_{i+1} \ldots y_n$.

Вспомним, что мы делали при сравнении чисел (скажем, длины 8). На нижнем уровне мы сравнивали биты:

$$\begin{array}{cccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8 \end{array}$$

На следующем уровне мы сравнивали двузначные числа

$$\begin{array}{cccc} x_1x_2 & x_3x_4 & x_5x_6 & x_7x_8 \\ y_1y_2 & y_3y_4 & y_5y_6 & y_7y_8 \end{array}$$

затем четырёхзначные

$$\begin{array}{cc} x_1x_2x_3x_4 & x_5x_6x_7x_8 \\ y_1y_2y_3y_4 & y_5y_6y_7y_8 \end{array}$$

и, наконец, восьмизначные:

$$\begin{array}{c} x_1x_2x_3x_4x_5x_6x_7x_8 \\ y_1y_2y_3y_4y_5y_6y_7y_8 \end{array}$$

Таким образом, для суффиксов длины 8, 4, 2 и 1 результаты сравнения уже есть. Для суффикса длины 6 результат можно получить, комбинируя результат сравнения $x_3x_4?y_3y_4$ и $x_5x_6x_7x_8?y_5y_6y_7y_8$. После этого у нас есть информация о суффиксах всех чётных длин, и соединяя её с информацией с первого этапа, получаем сведения про все суффиксы. Например, для сравнения суффиксов длины 7, то есть $x_2 \ldots x_8$ и $y_2 \ldots y_8$, мы соединяем результаты сравнения $x_2$ и $y_2$ с результатами сравнения суффиксов длины 6, то есть $x_3 \ldots x_8$ и $y_3 \ldots y_8$.

В общем случае картина такая: после «сужающегося дерева» мы строим «расширяющееся»; за $k$ шагов до конца мы знаем результаты

сравнения всех суффиксов, длины которых кратны $2^k$. Это дерево имеет размер $O(n)$ и глубину $O(\log n)$, что завершает доказательство.

Задача 12. Показать, что вычитание двух $n$-битовых чисел по модулю $2^n$ выполняется схемой размера $O(n)$ и глубины $O(\log n)$. (Указание: вычитание легко сводится к сложению, если заменить нули на единицы и наоборот.)

Теперь займёмся умножением. Схема умножения двух $n$-битовых чисел имеет $2n$ входов (по $n$ для каждого множителя) и $2n$ выходов для произведения.

Посмотрим, какие оценки даёт обычный способ умножения чисел столбиком. В нём умножение двух $n$-разрядных чисел сводится к сложению $n$ копий первого числа (частично заменённых на нули в зависимости от цифр второго числа) со сдвигами.

Получение этих копий требует схемы размера $O(n^2)$ (общее число цифр в копиях) и глубины $O(1)$. Сложение двух $n$-разрядных чисел мы можем выполнить с помошью схемы размера $O(n)$ и глубины $O(\log n)$, так что необходимые $n-1$ сложений можно выполнить схемой размера $O(n^2)$ и глубины $O(\log^2 n)$ (если складывать сначала попарно, потом результаты снова попарно и т. д.). Оказывается, этот результат можно улучшить. Наиболее экономные способы основаны на преобразовании Фурье и далеко выходят за рамки нашего обсуждения, но два улучшения мы приведём.

Теорема 12. *Существует схема умножения двух $n$-битовых чисел размера $O(n^2)$ и глубины $O(\log n)$.*

Доказательство. Как мы уже говорили, умножение двух $n$-битовых чисел сводится к сложению $n$ таких чисел, и остаётся выполнить такое сложение схемой размера $O(n^2)$ и глубины $O(\log n)$. Ключевым моментом здесь является сведение сложения трёх чисел к сложению двух с помощью простой схемы размера $O(n)$ и глубины $O(1)$. В самом деле, пусть есть три числа $x$, $y$ и $z$. Если мы будем складывать отдельно в каждом разряде, то в разряде может накопиться любая сумма от 0 до 3, то есть в двоичной записи от 00 до 11. Сформируем из младших битов этих двухбитовых сумм число $u$, а из старших (сдвинутых влево) — число $v$. Тогда, очевидно, $x+y+z=u+v$. Получение цифр числа $u$ и $v$ происходит параллельно во всех разрядах и требует размера $O(n)$ и глубины $O(1)$.

Теперь, если надо сложить $n$ чисел, можно разбить их на тройки и из каждых трёх чисел получить по два. В следуюший круг, таким образом, выйдут $(2/3)n$ чисел (примерно — граничные эффекты большой роли не играют). Их снова можно сгруппировать по тройкам и т. д. Получается дерево, в котором размеры уровней образуют геометрическую прогрес-

сию со знаменателем 3/2, поэтому глубина этого дерева логарифмическая. Число вершин в нём (считая за вершину преобразование трёх чисел в два) будет примерно $n$, так как при каждом преобразовании число слагаемых уменьшается на единицу. Итак, эта конструкция имеет общий размер $O(n^2)$ и глубину $O(\log n)$. Надо только отметить, что в конце у нас получается не одно число, а два, и их напоследок надо сложить — что мы умеем делать с глубиной $O(\log n)$ и размером $O(n)$.

Задача 13. Доказать, что схема, вычисляющая булеву функцию $f$ от $n$ аргументов, у которой ни один аргумент не является фиктивным, имеет размер не менее $cn$ и глубину не менее $c\log n$ (где $c$ — некоторая константа, зависящая от выбранного набора элементов). (Аргумент функции называют фиктивным, если при его изменении значение функции не меняется.)

Эта задача показывает, что если в процессе умножения двух $n$-битовых чисел мы суммируем $n$ слагаемых, то оценки $O(n^2)$ для размера и $O(\log n)$ для глубины, полученные при доказательстве теоремы 12, существенно улучшить нельзя.

Однако никто не обязывает нас следовать традиционному способу умножения столбиком — отказавшись от него, мы можем уменьшить размер схемы.

Теорема 13. *Существует схема умножения двух $n$-битовых чисел размера $O(n^{\log_2 3})$ и глубины $O(\log^2 n)$.*

Доказательство. Начнём с такого замечания. Вычисляя произведение двух комплексных чисел

$$(a+bi)(c+di) = (ac-bd)+(ad+bc)i$$

обычным способом, мы делаем четыре умножения. Но можно обойтись и тремя с помощью такого трюка: вычислить $ac$, $bd$ и $(a+b)(c+d)$, а потом найти $ad+bc$ как разность $(a+b)(c+d)-ac-bd$.

Аналогичный фокус можно проделать и для целых чисел. Разобьём $2n$-битовое число на две $n$-битовые части, то есть представим его в виде $a2^n+b$. Теперь запишем произведение двух таких чисел:

$$(a2^n+b)(c2^n+d) = ac2^{2n}+(ad+bc)2^n+bd.$$

Отсюда видно, что для определения всех трех слагаемых в правой части равенства достаточно найти три произведения $ac$, $bd$ и $(a+b)(c+d)$. Получается, что умножение двух $2n$-битовых чисел сводится к трём умножениям $n$-битовых и к нескольким сложениям и вычитаниям. (На самом деле при умножении $(a+b)$ на $(c+d)$ сомножители могут быть $(n+1)$-битовыми, но это не страшно, так как обработка лишнего разряда соответствует нескольким сложениям.)

Для размера схемы, таким образом, получается рекурсивная оценка

$$S(2n) \leqslant 3S(n) + O(n),$$

из которой следует, что $S(n) = O(n^{\log_2 3})$. (В самом деле, для умножения $n$-битовых чисел получается дерево рекурсивных вызовов глубины $\log_2 n$ и степени ветвления 3, так что число его листьев и общее число вершин есть $O(3^{\log_2 n}) = O(n^{\log_2 3})$. Остаётся понять только, что средний размер схемы в каждой вершине есть $O(1)$. В самом деле, он пропорционален числу складываемых битов. При переходе от одного уровня к следующему (более близкому к корню) число битов растёт вдвое, а число вершин уменьшается втрое, поэтому общее число элементов на этом уровне уменьшается в полтора раза. Таким образом, при движении по уровням от листьев к корню получается убывающая геометрическая прогрессия со знаменателем 2/3, сумма которой всего лишь втрое превосходит её первый член.

Оценка глубины также очевидна: на каждом уровне мы имеем схему сложения глубины $O(\log n)$, а число уровней есть $O(\log n)$.

Рассмотрим теперь функцию *голосования* (majority). Она имеет нечётное число аргументов, и значение её равно 0 или 1 в зависимости от того, какое из двух значений чаще встречается среди входов.

Теорема 14. *Существует вычисляющая функцию голосования схема размера $O(n)$ и глубины $O(\log n \log\log n)$.*

Доказательство. На самом деле можно даже вычислить общее число единиц среди входов. Это делается рекурсивно: считаем отдельно для каждой половины, потом складываем. Получается логарифмическое число уровней. На верхнем уровне надо складывать числа размера $\log n$, на следующем — размера $\log n - 1$ и так до самого низа, где складываются однобитовые числа (то есть биты входа). Какой средний размер складываемых чисел? Половина вершин в дереве приходится на нижний уровень (числа длины 1), четверть — на следующий (числа длины 2) и т. д. Вспоминая, что ряд $\sum(k/2^k)$ сходится, видим, что средний размер складываемых чисел есть $O(1)$ и обший размер схемы есть $O(n)$. А обшая глубина есть $O(\log n \log\log n)$, так как на каждом из $\log n$ уровней стоит схема глубины $O(\log\log n)$.

Заметим, что построенная схема вычисления функции голосования содержит в себе немонотонные элементы (поскольку операция сложения не монотонна). Мы уже говорили, что всякую монотонную функцию можно составить из конъюнкций и дизъюнкций. Для функции голосования есть очевидный способ это сделать: написать дизъюнкцию всех конъюнкций размера $(n+1)/2$ (напомним, что число входов $n$ предполагается

нечётным). Однако при этом получится схема экспоненциального по $n$ размера.

Теорема 15. *Существует схема размера $O(n^c)$ и глубины $O(\log n)$, составленная только из элементов И и ИЛИ (с двумя входами), вычисляющая функцию голосования.*

Доказательство. Для начала заметим, что ограничение на размер является следствием ограничения на глубину, так как элементы И и ИЛИ имеют только два входа и число входов схемы глубины $d$ есть $O(2^d)$.

Схема будет строиться из элементов большинства с тремя входами (3-большинства). Каждый из них можно собрать из конъюнкций и дизъюнкций по формуле $(a\wedge b)\vee(a\wedge c)\vee(b\wedge c)$. Выход схемы будет большинством из трёх значений, каждое из которых есть большинство из трёх значений и т. д. (рис. 3).

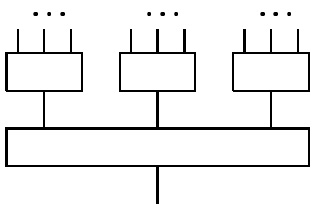

***Рис. 3.*** *Дерево из элементов 3-большинства*

Продолжая эту конструкцию на $k$ уровнях, мы получим схему с $3^k$ входами. (Отметим, что эта схема не будет вычислять большинство среди своих входов — по той же причине, по которой результат непрямого голосования может отличаться от мнения большинства.) Но мы сделаем вот какую странную вещь: возьмём $k$ равным $c\log n$ при достаточно большом коэффициенте пропорциональности $c$ (при этом число входов такой схемы будет полиномиально зависеть от $n$) и напишем на входах случайно выбранные переменные из данного нам набора $x_1,\ldots,x_n$. (Переменные, записываемые на разных входах, выбираются независимо.) Оказывается, что с ненулевой вероятностью эта схема будет вычислять функцию большинства среди $x_1,\ldots,x_n$, если константа $c$ достаточно велика. Следовательно, искомая схема существует.

Обратите внимание: нам удастся доказать существование интересующей нас схемы, не предъявив её явно. (Такое использование вероятностных методов в комбинаторных рассуждениях часто бывает полезно.)

Итак, почему же схема с положительной вероятностью вычисляет функцию большинства? Это доказывается так: рассмотрим какой-то один набор значений на входах и докажем, что на этом конкретном

наборе случайная схема выдаёт правильный ответ с вероятностью, очень близкой к единице (равной $1-\varepsilon$ при очень малом $\varepsilon$).

Если число $\varepsilon$ настолько мало, что остаётся меньшим единицы даже после умножения на число возможных входов ($2^n$), то получаем требуемое (каждое из $2^n$ событий имеет вероятность не меньше $1-\varepsilon$, значит их пересечение имеет вероятность не меньше $1-2^n\varepsilon > 0$).

Итак, осталось оценить вероятность того, что случайная схема даст правильный ответ на данном входе. Пусть доля единиц среди всех входов равна $p$. Тогда на каждый входной провод схемы подаётся единица с вероятностью $p$ и нуль с вероятностью $1-p$ (выбор случайной переменной даёт единицу с вероятностью $p$), причём сигналы на всех входах независимы.

Если на трёх входах элемента 3-большинства сигналы независимы, и вероятность появления единицы на входе есть $p$, то вероятность появления единицы на выходе есть $\varphi(p) = 3p^2(1-p) + p^3 = 3p^2 - 2p^3$. На следующих уровнях вероятность появления единицы будет равна

$$\varphi(\varphi(p)), \varphi(\varphi(\varphi(p))), \ldots$$

График функции $\varphi(x)$ на отрезке $[0,1]$ (рис. 4) показывает, что при итерациях функции $\varphi$ дисбаланс (отклонение от середины) нарастает. Поэтому последовательность итераций стремится к краю отрезка. Надо только оценить число шагов.

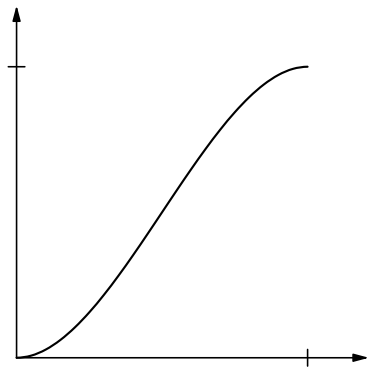

***Рис. 4.*** *Итерируемая функция $\varphi$*

Если вначале единицы составляют большинство из $n$ аргументов (напомним, $n$ нечётно), то их как минимум $(n+1)/2$, так что $p \geqslant (n+1)/2n = 1/2 + 1/(2n)$. Таким образом, начальный дисбаланс составляет как минимум $1/2n$. А в конце нам нужно приблизиться к краю отрезка на расстояние $2^{-n}$.

Итак, нам осталось доказать такую лемму (относящуюся скорее к математическому анализу):

Лемма. *Пусть функция* $\varphi : [0,1] \to [0,1]$ *задана формулой*

$$\varphi(x) = 3x^2 - 2x^3.$$

*Пусть последовательность* $x_k$ *определена рекуррентной формулой* $x_{k+1} = \varphi(x_k)$. *Пусть* $x_0 \geqslant 1/2 + 1/(2n)$. *Тогда последовательность* $x_k$ *монотонно возрастает и приближается к* 1 *на расстояние* $2^{-n}$ *за* $O(\log n)$ *шагов.*

Набросок доказательства: посмотрим на поведение функции в неподвижных точках. В окрестности точки 1/2 функция близка к линейной и производная больше 1, поэтому удаление от 1/2 растёт как геометрическая прогрессия, и точка перейдёт какую-то фиксированную границу (например, 0,51) не позднее чем за $O(\log n)$ шагов. Затем потребуется $O(1)$ шагов, чтобы дойти, скажем, до 0,99. В окрестности единицы первая производная функции равна нулю, поэтому расстояние до единицы каждый раз примерно возводится в квадрат, и потому для достижения погрешности $2^{-n}$ потребуется $O(\log n)$ шагов (каждый раз число совпадающих цифр увеличивается примерно вдвое, как в методе Ньютона отыскания корня). Всего получается $O(\log n) + O(1) + O(\log n)$ шагов, что и требовалось.

На самом деле справедливо гораздо более сильное утверждение: существует схема размера $O(n \log n)$ и глубины $O(\log n)$, состоящая только из элементов И и ИЛИ, которая имеет $n$ входов и $n$ выходов и осуществляет сортировку (это означает, что на выходе столько же единиц, сколько на входе, причём выходная последовательность всегда невозрастающая). Ясно, что средний бит выхода в такой ситуации реализует функцию большинства.

При кажущейся простоте формулировки единственная известная конструкция такой схемы (сортирующая сеть AKS, придуманная Айтаи, Комлошом и Спемереди сравнительно недавно, в 1983 году[1]) весьма сложна, и появление какой-то более простой конструкции было бы замечательным достижением.

Вообще многие нетривиальные результаты можно переформулировать в терминах сложности каких-то булевых функций. Например, есть вероятностный алгоритм проверки простоты большого числа (с помощью которого в криптографии проверяются числа из нескольких тысяч цифр). Используя этот алгоритм, можно доказать такое утверждение:

[1] Ajtai M., Komlós J., Szemerédi E. An $O(n \log n)$ sorting network // Proc. of the 15th Annual ACM Symposium on Theory of Computing. Boston, MA, 1983. P. 1–9.

существует схема проверки простоты $n$-битового числа (на вход подаются $n$ битов, на выходе появляется единица, если число простое), размер которой ограничен полиномом от $n$.

Вернёмся к общим утверждениям о схемах и формулах. Мы уже говорили, что с точки зрения измерения размера схемы и формулы — это разные вещи (схемы экономичнее, так как в них одинаковые подформулы учитываются только один раз). Оказывается, что размер формулы можно связать с глубиной схемы.

Будем называть *размером* формулы число логических связок в ней. Мы предполагаем, что формула использует конъюнкции, дизъюнкции и отрицания, и в схемах будем использовать такие же элементы. Напомним, что размером схемы мы называли число элементов, а сложностью булевой функции — минимальный размер схемы, её вычисляющей. Сложность функции $h$ обозначалась $\mathrm{size}(h)$ (точнее $\mathrm{size}_B(h)$, где $B$ — набор разрешённых функциональных элементов, но сейчас мы договорились использовать конъюнкции, дизъюнкции и отрицания и опускаем индекс $B$).

Минимальный размер формулы, выражающей функцию $h$, будем обозначать $\mathrm{fsize}(h)$. Очевидно, $\mathrm{size}(h) \leqslant \mathrm{fsize}(h)$. Более интересно, однако, следующее утверждение, связывающее размер схемы с глубиной формулы. Обозначим через $\mathrm{depth}(h)$ минимальную глубину схемы, вычисляющей функцию $h$.

Теорема 16. *Имеют место оценки* $\mathrm{fsize}(h) \leqslant c_1^{\mathrm{depth}(h)}$ *и* $\mathrm{depth}(h) \leqslant c_2 \log \mathrm{fsize}(h)$ (*для некоторых констант* $c_1$ *и* $c_2$ *и для всех* $h$). *Другими словами, меры сложности* depth *и* log fsize *отличаются не более чем в константу раз.*

Доказательство. Первая оценка очевидна: если мы скопируем повторяющиеся фрагменты схемы, чтобы развернуть её в дерево, то глубина не изменится. Если она равна $k$, то в полученном дереве будет не больше $2^k - 1$ элементов (напомним, что элементами являются конъюнкции, дизъюнкции и отрицания, и потому ветвление не больше 2). То же самое можно сказать индуктивно. Пусть глубина схемы равна $k$. Выход схемы является выходом некоторого элемента. Тогда на его входы подаются булевы функции глубины не больше $k-1$. По предположению индукции их можно записать формулами размера $2^{k-1} - 1$. Таких формул максимум две, так что общий размер не превосходит $2(2^{k-1} - 1) + 1 = 2^k - 1$.

Вторая оценка более сложна. Если мы будем преобразовывать формулу в схему естественным образом (введя по переменной для каждой подформулы), то глубина получившейся схемы может быть близка к размеру формулы, а не к его логарифму. Например, если формула имеет вид $(\ldots((p_1 \wedge p_2) \wedge p_3) \wedge \ldots p_n)$, то у нас получится цепочка элементов И, у

которых каждый следующий подвешен к левому входу предыдущего, и глубина есть $n-1$. Конечно, если использовать ассоциативность конъюнкции, скобки можно переставить и получить более сбалансированное дерево глубины примерно $\log n$, как и требуется. Но как выполнить такое преобразование в случае произвольной формулы?

Обозначим данную нам формулу через $F$. Выберем у неё некоторую подформулу $G$ (как именно, мы объясним позже). Рассмотрим формулу $F_0$, которая получится, если вместо $G$ подставить 0 (ложь), а также формулу $F_1$, которая получится, если подставить 1. Легко понять, что $F$ равносильна формуле

$$((F_0 \wedge \neg G) \vee (F_1 \wedge G)).$$

Если теперь удастся заменить формулы $F_0, F_1, G$ схемами глубины не больше $k$, то для $F$ получится схема глубины не больше $k+3$.

Такое преобразование полезно, если все три формулы $F_1, F_0, G$ имеют заметно меньший размер, чем исходная формула $F$.

Лемма. *У любой формулы размера $n$ (при достаточно больших $n$) есть подформула размера от $n/4$ до $3n/4$.*

Доказательство. Каждая формула есть конъюнкция двух подформул, дизъюнкция двух подформул или отрицание подформулы. Начав со всей формулы, будем переходить к её подформулам, на каждом шаге выбирая из двух подформул наибольшую. Тогда на каждом шаге размер убывает не более чем в два раза, и потому мы не можем миновать промежуток $[n/4, 3n/4]$, концы которого отличаются втрое[2)]. Лемма доказана.

Выбирая подформулу $G$ с помощью этой леммы, мы гарантируем, что размер всех трёх формул $F_0, F_1, G$ не превосходит $3/4$ размера исходной формулы (подстановка нуля или единицы может только уменьшить размер формулы — некоторые части можно будет выбросить).

Применим ко всем трём формулам $F_0$, $F_1$ и $G$ тот же приём, выделим в них подформулы среднего размера и так далее, пока мы не спустимся до формул малого размера, которые можно записать в виде схем как угодно. В итоге получится дерево с логарифмическим числом уровней, на каждом из которых стоят схемы глубины 3, а в листьях находятся схемы глубины $O(1)$.

Другими словами, индукцией по размеру формулы, выражающей некоторую функцию $h$, легко получить оценку $\operatorname{depth}(h) = O(\log \operatorname{fsize}(h))$.

[2)] Тут есть небольшая неточность: размер формулы может убывать чуть быстрее, чем вдвое, так как размер формулы на единицу больше суммы размеров частей, но у нас есть запас, поскольку концы промежутка отличаются втрое, а не вдвое.

Задача 14. Определим глубину формулы как максимальное число вложенных пар скобок; для единообразия будем окружать отрицание скобками и писать $(\neg A)$ вместо $\neg A$. Покажите, что при этом не получится ничего нового: минимальная глубина формулы, записывающей некоторую функцию $f$, совпадает с минимальной глубиной схемы, вычисляющей $f$.

Определение формульной сложности $\mathrm{fsize}(h)$ зависит от выбора базиса. Оказывается, что здесь (в отличие от схемной сложности) выбор базиса может изменить $\mathrm{fsize}(h)$ более чем в константу раз.

Задача 15. Объясните, почему доказательство теоремы 7 не переносится на случай формул.

Пример такого рода доставляет функция $p_1 \oplus p_2 \oplus \ldots \oplus p_n$ (знак $\oplus$ обозначает сложение по модулю 2). Эта функция имеет формульную сложность $O(n)$, если сложение по модулю 2 входит в базис. Однако в базисе И, ИЛИ, НЕ она имеет большую сложность, как доказала Б. А. Субботовская. Идея доказательства такова: если заменить случайно выбранную переменную в формуле с конъюнкциями и дизъюнкциями на случайно выбранное значение 0 или 1, то формула упростится (не только эта переменная пропадёт, но с некоторой вероятностью пропадут и другие). Если делать так многократно, то от формулы останется небольшая часть — с другой стороны, эта часть всё равно должна реализовывать сложение оставшихся аргументов по модулю 2.

Задача 16. Доказать, что функция большинства может быть вычислена не только схемой, но и формулой полиномиального размера, содержащей только связки И и ИЛИ.

Задача 17. Доказать, что значения $\mathrm{fsize}_1(h)$ и $\mathrm{fsize}_2(h)$ для одной булевой функции $h$ и различных полных базисов полиномиально связаны: существует полином $P$ (зависящий от выбора базисов), для которого $\mathrm{fsize}_2(h) \leqslant P(\mathrm{fsize}_1(h))$ при всех $h$. (Указание: использовать теорему 16.)