Глава 3 Булевы функции

Занимаясь исследованием законов мышления, английский математик Джордж Буль[1] применил в логике систему формальных обозначений и правил, близкую к математической. Так возникла алгебра логики, или булева алгебра. Значение логической алгебры долгое время игнорировалось, поскольку она не находила применения в науке и технике того времени. С появлением электронной техники операции, введенные Булем, оказались весьма полезны. Они изначально ориентированы на работу только с двумя сущностями — истина и ложь, что хорошо согласуется с двоичным кодом, который в современных компьютерах также представляется всего двумя сигналами — ноль и единица.

Данная глава имеет двойное назначение. С одной стороны, помимо основных фактов из теории булевых функций здесь затрагиваются весьма общие понятия, такие как *реализация функций формулами*, *нормальные формы*, *двойственность*, *полнота*. Эти понятия затруднительно описать исчерпывающим образом на выбранном элементарном уровне изложения, но знакомство с ними необходимо. Поэтому рассматриваются частные случаи указанных понятий на простейшем примере булевых функций. С другой стороны, материал этой главы служит для «накопления фактов» и овладения базисной логической техникой, что должно создать у читателя необходимый эффект «узнавания знакомого» при изучении довольно абстрактного и формального материала следующей главы.

3.1. Элементарные булевы функции

Подобно тому как в классической математике знакомство с основами анализа начинают с изучения *элементарных функций* (рациональные функции от вещественной переменной x, e^x, $\ln x$ и их суперпозиции), изложение теории булевых функций естественно начать с выделения, идентификации и изучения *элементарных булевых функций* (дизъюнкция, конъюнкция и т. д.).

3.1.1. Функции алгебры логики

Функции $f\colon E_2^n \to E_2$, где $E_2 \stackrel{\text{Def}}{=} \{0,1\}$, называются *функциями алгебры логики*, или *булевыми функциями* от n переменных. Элементы множества $E_2 = \{0,1\}$ называются *истинностными значениями*. Множество булевых функций от n переменных обозначим P_n, $P_n \stackrel{\text{Def}}{=} \{f \mid f\colon E_2^n \to E_2\}$.

Булеву функцию от n переменных можно задать *таблицей истинности*:

x_1	...	x_{n-1}	x_n	$f(x_1,\ldots,x_n)$
0	...	0	0	$f(0,\ldots,0,0)$
0	...	0	1	$f(0,\ldots,0,1)$
0	...	1	0	$f(0,\ldots,1,0)$
...	...	...	...	...
1	...	1	1	$f(1,\ldots,1,1)$

[1] Джордж Буль (1815–1864).

Если число переменных равно n, то в таблице истинности имеется 2^n строк, соответствующих всем различным комбинациям значений переменных. Следовательно, существует 2^{2^n} различных столбцов, каждый из которых определяет булеву функцию от n переменных. Таким образом, $|P_n| = 2^{2^n}$.

Если нужно задать несколько (например, k) булевых функций от n переменных, то это удобно сделать с помощью одной таблицы, использовав несколько столбцов:

x_1	...	x_n	$f_1(x_1, \ldots, x_n)$	...	$f_k(x_1, \ldots, x_n)$
0	...	0	$f_1(0, \ldots, 0)$	...	$f_k(0, \ldots, 0)$
...	...	...	...	...	...
1	...	1	$f_1(1, \ldots, 1)$	...	$f_k(1, \ldots, 1)$

Такая таблица будет иметь 2^n строк, n столбцов для значений переменных и ещё k столбцов для значений функций.

Вообще говоря, значения переменных можно не хранить, если принять соглашение о перечислении наборов переменных в определённом порядке. Во всех таблицах истинности в этом учебнике переменные всегда перечисляются в лексикографическом порядке, а кортежи булевых значений — в порядке возрастания целых чисел, задаваемых кортежами, как двоичными шкалами, что совпадает с лексикографическим порядком на кортежах. Такой порядок мы называем *установленным*.

Если $k > 2^n$ (что, как показывает предыдущая формула, не редкость), то таблицу истинности можно «транспонировать», выписывая наборы значений в столбцах, а значения функций — в строках:

x_1	0	...	1
...	...	...	...
x_n	0	...	1
f_1	$f_1(0, \ldots, 0)$	...	$f_1(1, \ldots, 1)$
...	...	...	...
f_k	$f_k(0, \ldots, 0)$	...	$f_k(1, \ldots, 1)$

Именно такой способ записи таблицы истинности использован в пп. 3.1.3 и 3.1.4.

3.1.2. Существенные и несущественные переменные

Булева функция $f \in P_n$ *существенно зависит* от переменной x_i, если существует такой набор значений $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n$, что

$$f(a_1, \ldots, a_{i-1}, 0, a_{i+1}, \ldots, a_n) \neq f(a_1, \ldots, a_{i-1}, 1, a_{i+1}, \ldots, a_n).$$

В этом случае x_i называют *существенной переменной*, в противном случае x_i называют *несущественной* (*фиктивной*) переменной.

Пример. Пусть булевы функции $f_1(x_1, x_2)$ и $f_2(x_1, x_2)$ заданы таблицей истинности, приведённой ниже. Для этих функций переменная x_1 — существенная, а переменная x_2 — несущественная.

x_1	x_2	f_1	f_2
0	0	0	1
0	1	0	1
1	0	1	0
1	1	1	0

Пусть заданы две булевы функции, $f_1(x_1, \ldots, x_{n-1})$ и $f_2(x_1, \ldots, x_{n-1}, x_n)$, и пусть переменная x_n — несущественная для функции f_2, а при одинаковых значениях остальных переменных значения функций совпадают:

$$\forall\, a_1, \ldots, a_{n-1}, a_n \;\; (f_1(a_1, \ldots, a_{n-1}) = f_2(a_1, \ldots, a_{n-1}, a_n)) .$$

В таком случае говорят, что f_2 получается из f_1 *введением* несущественной переменной x_n, а f_1 получается из f_2 *удалением* несущественной переменной x_n.

ЗАМЕЧАНИЕ

Процедурно введение и удаление несущественных переменных выполняются достаточно просто. Чтобы ввести несущественную переменную, нужно продублировать каждую строку таблицы истинности, добавить новый столбец и заполнить этот столбец чередующимися значениями 0 и 1 (или 1 и 0, что несущественно). Удаление несущественной переменной выполняется аналогично: нужно отсортировать таблицу истинности так, чтобы она состояла из пар строк, различающихся только в разряде несущественной переменной, после чего удалить столбец несущественной переменной и удалить каждую вторую строку таблицы.

Всюду в дальнейшем булевы функции рассматриваются *с точностью до несущественных переменных*. Это позволяет считать, что все булевы функции (в данной системе функций) зависят от одних и тех же переменных. Для этого в данной системе булевых функций можно сначала удалить все несущественные переменные, а потом добавить несущественные переменные так, чтобы уравнять количество переменных у всех функций.

3.1.3. Булевы функции одной переменной

В следующей таблице собраны все булевы функции одной переменной. Две из них фактически являются константами, поскольку их значения не зависят от значения аргумента.

	Переменная x	0	1	
Название	Обозначение			Несущественные
Нуль	0	0	0	x
Тождественная	x	0	1	
Отрицание	$\neg x$, $\bar{x}$, x', $\sim x$	1	0	
Единица	1	1	1	x

3.1.4. Булевы функции двух переменных

В следующей таблице собраны все булевы функции двух переменных. Из них две являются константами, четыре зависят от одной переменной и только десять существенно зависят от обеих переменных.

	Переменная x	0	0	1	1	
	Переменная y	0	1	0	1	
Название	Обозначение					Несущественные
Нуль	0	0	0	0	0	x, y
Конъюнкция	$\cdot$, $\&$, $\wedge$	0	0	0	1	
		0	0	1	0	
		0	0	1	1	y
		0	1	0	0	
		0	1	0	1	x
Сложение по модулю 2	$+$, $+_2$, $\not\equiv$, $\oplus$, $\triangle$	0	1	1	0	
Дизъюнкция	$\vee$	0	1	1	1	
Стрелка Пирса[1]	$\downarrow$	1	0	0	0	
Эквивалентность	$\equiv$	1	0	0	1	
		1	0	1	0	x
		1	0	1	1	
		1	1	0	0	y
логическое следование						
Импликация	$\rightarrow$, $\Rightarrow$, $\supset$	1	1	0	1	
Штрих Шеффера[2]	$\mid$	1	1	1	0	
Единица	1	1	1	1	1	x, y

ЗАМЕЧАНИЕ

Пустоты в столбцах «Название» и «Обозначение» в предыдущей таблице означают, что булева функция редко используется, а потому не имеет специального названия и обозначения.

3.2. Функции k-значной логики

Рассмотрим множество $E_k = \{0, 1, \ldots, k-1\}$ и множество функций $P_{k,n}$, имеющих n аргументов типа E_k и возвращающих значение типа E_k. Такие функции называются *функциями k-значной логики* и являются обобщением функций алгебры логики: $P_{2,n} = P_n$. Функции k-значной логики можно задавать таблицами, подобными таблицам истинности для обычной двузначной логики, причем в таблицах используются элементы из множества E_k, а не из множества E_2. Нетрудно видеть, что $|P_{k,n}| = k^{k^n}$. Число функций k-значной логики растет ещё быстрее, чем число булевых функций. Так, уже для трехзначной логики аналог таблицы п. 3.1.4 для всех функций двух переменных содержит $3^9 = 19\,683$ строк и не может быть помещён в книгу.

Функцию k-значной логики можно представить как систему функций обычной двузначной логики следующим образом. Функция $f(x_1, \ldots, x_n) \in P_{k,n}$ представляется системой функций

$$\{g_1(y_{11}, \ldots, y_{1m} \ldots, y_{n1}, \ldots, y_{nm}), \ldots, g_m(y_{11}, \ldots, y_{1m}, \ldots, y_{n1}, \ldots, y_{nm})\},$$

где $m = \lceil \log_2 k \rceil$. Значения функции f и переменных x_i принадлежат множеству E_k, а значения функций g_j и переменных y_{ij} принадлежат множеству E_2. При этом

[1] Чарльз Сандерс Пирс (1839–1914).
[2] Генри М. Шеффер (1882–1964).

если значение переменной x_i имеет двоичный код $a_{i1}\dots a_{im}$, то значение функции $f(x_1,\dots,x_n)$ имеет двоичный код

$$g_1(a_{11},\dots,a_{1m},\dots,a_{n1},\dots,a_{nm})\dots g_m(a_{11},\dots,a_{1m},\dots,a_{n1},\dots,a_{nm}).$$

Другими словами, функция g_j вычисляет j-ю цифру в двоичном представлении числа $f(x_1,\dots,x_n)$.

Таким образом, хотя k-значная логика может быть сведена к двузначной логике и многие полезные утверждения двузначной логики распространяются на случай k-значной логики, но вычислительные процедуры во всех случаях являются существенно более трудоёмкими для функций k-значной логики.

Пример. В трёхзначной логике $E_3=\{0,1,2\}$, $k=3$ и $m=\lceil\log_2 3\rceil=2$. Рассмотрим функцию «трёхзначная конъюнкция», которую можно определить, например, как минимум истинностных значений аргументов: $x_1\ \&\ x_2:=\min(x_1,x_2)$. В таком случае таблица истинности имеет следующий вид.

x_1	$y_{11}y_{12}$	x_2	$y_{21}y_{22}$	$x_1\ \&\ x_2$	g_1g_2
0	00	0	00	0	00
0	00	1	01	0	00
0	00	2	10	0	00
1	01	0	00	0	00
1	01	1	01	1	01
1	01	2	10	1	01
2	10	0	00	0	00
2	10	1	01	1	01
2	10	2	10	2	10

Таким образом, конъюнкцию представляет следующая система функций:
$g_1(y_{11},y_{12},y_{21},y_{22})=y_{11}\ \&\ y_{21}$,
$g_2(y_{11},y_{12},y_{21},y_{22})=y_{12}\ \&\ y_{22}\ \vee\ y_{12}\ \&\ y_{21}\ \vee\ y_{11}\ \&\ y_{22}$.

Функции k-значной логики далее в этом учебнике не рассматриваются.

3.2.1. Формулы

В этом разделе обсуждается целый ряд важных понятий, которые часто считаются самоочевидными, а потому их объяснение опускается. Такими понятиями являются, в частности, само понятие *формулы*, *реализация* функций формулами, *интерпретация* формул для вычисления значений функций, *равносильность* формул, *подстановка* и *замена* в формулах. Между тем программная реализация работы с формулами требует учёта некоторых тонкостей, связанных с данными понятиями, которые целесообразно явно указать и обсудить.

3.2.2. Реализация функций формулами

Начнём обсуждение с привычного понятия формулы.

Вообще говоря, *формула* — это цепочка символов, имеющих заранее оговорённый смысл. Обычно формулы строятся по определённым правилам из обозначений объектов и вспомогательных символов — разделителей. Применительно к рассматриваемому случаю объектами являются переменные и булевы функции, перечисленные в таблицах пп. 3.1.3 и 3.1.4, а разделителями — скобки «(», «)» и запятая «,». Мы

считаем, что обозначения всех объектов заранее определены и синтаксически отличимы друг от друга и от разделителей, а правила составления формул общеизвестны.

ЗАМЕЧАНИЕ

Для обозначения объектов используются как отдельные символы, так и группы символов, в том числе со специальным начертанием: верхние и нижние индексы, наклон шрифта и т. п. Допущение о синтаксической различимости остаётся в силе для всех таких особенностей.

Пример. Операцию сложения вещественных чисел принято обозначать знаком «+», $+\colon \mathbb{R}\times\mathbb{R}\to\mathbb{R}$, а функцию «синус» — словом «sin», которое записывается прямым шрифтом и считается одним символом, $\sin\colon \mathbb{R}\to\mathbb{R}$.

ОТСТУПЛЕНИЕ

В математических текстах часто используется *нелинейная* форма записи формул, при которой формула *не* является цепочкой символов. Примерами могут служить дроби, радикалы, суммы, интегралы. Подобные нелинейные формы записи формул привычны и удобны. Использование нелинейной записи формул не является принципиальным расширением языка формул. Можно ограничиться только линейными цепочками символов, что блестяще подтверждает система TEX, с помощью которой подготовлена эта книга.

Пусть $F=\{f_1,\ldots,f_m\}$ — некоторое множество булевых функций от n переменных. *Формулой* $\mathcal{F}$ над F называется выражение (цепочка символов) вида

$$\mathcal{F}[F]=f(t_1,\ldots,t_n),$$

где $f\in F$ и t_i — либо переменная, либо формула над F. Множество F называется *базисом*, функция f называется *главной* (*внешней*) операцией (функцией), а t_i называются *подформулами*. Если базис F ясен из контекста, то его обозначение опускают. Множество всех формул над базисом F здесь обозначается $\mathfrak{F}[F]$.

ЗАМЕЧАНИЕ

Обычно при записи формул, составленных из элементарных булевых функций, обозначения бинарных булевых функций записываются как знаки операций в инфиксной форме, отрицание записывается как знак унарной операции в префиксной форме, тождественные функции записываются как константы 0 и 1. Кроме того, устанавливается приоритет операций ($\neg$, $\&$, $\vee$, $\to$) и лишние скобки опускаются.

Всякой формуле $\mathcal{F}$ однозначно соответствует некоторая функция f. Это соответствие задается *алгоритмом интерпретации*, который позволяет вычислить значение формулы $\mathcal{F}$ при заданных значениях переменных.

ОТСТУПЛЕНИЕ

Некоторые программистские замечания по поводу процедуры Eval.

1. При программной реализации алгоритма интерпретации формул важно учитывать, что в общем случае результат вычисления значения формулы может быть не определён (**fail**). Это имеет место, если формула построена синтаксически неправильно или если в ней используются функции (операции), способ вычисления которых не задан, то есть они не входят в базис. Таким образом, необходимо либо проверять правильность формулы до начала работы алгоритма интерпретации, либо предусматривать невозможность вычисления значения в самом алгоритме.

Алгоритм 3.1. Интерпретация формул — рекурсивная функция Eval

Алгоритм *интерпретации* формулы.

Вход: формула $\mathcal{F}$, множество F функций базиса, значения переменных $x_1, \ldots, x_n$.

Выход: значение формулы $\mathcal{F}$ на значениях $x_1, \ldots, x_n$ или значение **fail**, если значение формулы не может быть определено.

```
if F = 'x_i' then
  return x_i  // значение переменной задано
end if
if F = 'f(t_1, ..., t_n)' then
  if f ∉ F then
    return fail  // функция не входит в базис
  end if
  for t ∈ {t_1, ..., t_n} do
    y_i := Eval (t_i, F, x_1, ..., x_n)  // значение i-го аргумента
    if y_i = fail then
      return fail  // аргумент не может быть вычислен
    end if
  end for
  return f(y_1, ..., y_n)  // вычисленное значение главной операции
end if
return fail  // это не формула
```

2. Это не единственный возможный алгоритм вычисления значения формулы, более того, он не самый лучший. Для конкретных классов формул, например для некоторых классов формул, реализующих булевы функции, известны более эффективные алгоритмы. (см., например, п. 3.5.3).
3. Сама идея этого алгоритма: «сначала вычисляются значения аргументов, а потом значение функции» — не является догмой.

   Например, можно построить такой алгоритм интерпретации, который вычисляет значения только *некоторых* аргументов, а потом вычисляет значение формулы, в результате чего получается новая *формула*, реализующая функцию, которая зависит от меньшего числа аргументов. Такой алгоритм интерпретации называется *смешанными вычислениями*.
4. Порядок вычисления аргументов *считается* неопределённым, то есть предполагается, что базисные функции не имеют *побочных эффектов*. Если же базисные функции имеют побочные эффекты, то результат вычисления значения функции может зависеть от порядка вычисления значений аргументов. Побочные эффекты могут иметь различные причины. Наиболее частая: использование глобальных переменных. Например, если $f(x) \stackrel{\text{Def}}{=} a := a + 1; \textbf{return } x + a$, $g(x) \stackrel{\text{Def}}{=} a := a * 2; \textbf{return } x * a$, причём переменная a глобальна, то (если $a \neq 6$) $f(2) + g(3) \neq g(3) + f(2)$.

Если формула $\mathcal{F}$ и базис F заданы (причём $\mathcal{F}$ является правильно построенной формулой над базисом F), то процедура Eval$(\mathcal{F}, F, x_1, \ldots, x_n)$ является некоторой булевой функцией f переменных $x_1, \ldots, x_n$. В этом случае говорят, что формула $\mathcal{F}$ *реализует* функцию f и записывают это так: $\operatorname{func}\mathcal{F} = f$.

ЗАМЕЧАНИЕ

Для обозначения реализуемости применяют и другие приёмы. Выбранное обозначение обладает тем достоинством, что согласовано с другими обозначениями в книге.

Зная таблицы истинности для функций базиса, можно вычислить таблицу истинности той функции, которую реализует данная формула.

Примеры

1. $F_1 := (x_1 \wedge x_2) \vee ((x_1 \wedge \overline{x}_2) \vee (\overline{x}_1 \wedge x_2))$

x_1	x_2	$x_1 \wedge \overline{x}_2$	$\overline{x}_1 \wedge x_2$	$(x_1 \wedge \overline{x}_2) \vee (\overline{x}_1 \wedge x_2)$	$x_1 \wedge x_2$	F_1
0	0	0	0	0	0	0
0	1	0	1	1	0	1
1	0	1	0	1	0	1
1	1	0	0	0	1	1

Таким образом, формула F_1 реализует дизъюнкцию.

2. $F_2 := (x_1 \wedge x_2) \to x_1$

x_1	x_2	$x_1 \wedge x_2$	F_2
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	1

Таким образом, формула F_2 реализует константу 1.

3. $F_3 := ((x_1 \wedge x_2) + x_1) + x_2$

x_1	x_2	$x_1 \wedge x_2$	$(x_1 \wedge x_2) + x_1$	$((x_1 \wedge x_2) + x_1) + x_2$
0	0	0	0	0
0	1	0	0	1
1	0	0	1	1
1	1	1	0	1

Таким образом, формула F_3 также реализует дизъюнкцию.

3.2.3. Равносильные формулы

Одна функция может иметь множество реализаций (над данным базисом). Формулы, реализующие одну и ту же функцию, называются *равносильными*:

$$\mathcal{F}_1 = \mathcal{F}_2 \stackrel{\text{Def}}{=} \exists\, f \;\; (\operatorname{func} \mathcal{F}_1 = f \;\&\; \operatorname{func} \mathcal{F}_2 = f)\,.$$

Отношение равносильности формул является эквивалентностью. Имеют место, в частности, следующие равносильности.

1) $a \vee a = a, \quad a \wedge a = a$;
2) $a \vee b = b \vee a, \quad a \wedge b = b \wedge a$;
3) $a \vee (b \vee c) = (a \vee b) \vee c, \quad a \wedge (b \wedge c) = (a \wedge b) \wedge c$;
4) $(a \wedge b) \vee a = a, \quad (a \vee b) \wedge a = a$;
5) $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c), \quad a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$;
6) $a \vee 1 = 1, \quad a \wedge 0 = 0$;
7) $a \vee 0 = a, \quad a \wedge 1 = a$;
8) $\neg\neg a = a$;
9) $\neg(a \wedge b) = \neg a \vee \neg b, \quad \neg(a \vee b) = \neg a \wedge \neg b$;
10) $a \vee \neg a = 1, \quad a \wedge \neg a = 0$.

Все указанные равносильности могут быть легко проверены построением соответствующих таблиц истинности. Таким образом, $\langle E_2;\, \vee, \wedge, \neg \rangle$ — булева алгебра (п. 2.6.6).

ЗАМЕЧАНИЕ

Считаем необходимым повторить здесь с уточнениями замечание из п. 1.2.6. Если некоторая бинарная операция ассоциативна, то в формуле, построенной с применением такой операции, порядок выполнения операций не важен и скобки можно опустить. Другими словами, ассоциативная бинарная операция определяет *групповую операцию*, применимую не только к паре, но и к любому набору аргументов. В этом случае употребляют также термины *кратная операция* и *вариаргументная операция*. Если же операция вдобавок коммутативна, то не только скобки можно опустить, но и аргументы групповой операции можно записывать в любом порядке. Ввиду выполнения равносильностей 2 и 3 можно определить кратные операции дизъюнкции и конъюнкции, для которых используются следующие обозначения:

$$\bigvee_{i=1}^{n} x_i \stackrel{\text{Def}}{=} x_1 \vee \ldots \vee x_n, \qquad \bigwedge_{i=1}^{n} x_i \stackrel{\text{Def}}{=} x_1 \wedge \cdots \wedge x_n \stackrel{\text{Def}}{=} x_1 \ldots x_n.$$

Аналогичными свойствами обладают сложение и умножение чисел, объединение и пересечение множеств.

3.2.4. Подстановка и замена

Если в формулу $\mathcal{F}$ входит переменная x, то это обстоятельство обозначается так: $\mathcal{F}(\ldots x \ldots)$. Соответственно, запись $\mathcal{F}(\ldots \mathcal{G} \ldots)$ обозначает, что в формулу $\mathcal{F}$ входит подформула $\mathcal{G}$.

Вместо подформулы (в частности, вместо переменной) в формулу можно подставить другую формулу (в частности, переменную), в результате чего получится новая правильно построенная формула.

Если подстановка формулы $\mathcal{G}$ производится вместо *всех* вхождений заменяемой переменной x (или подформулы), то результат подстановки обозначается следующим образом: $\mathcal{F}(\ldots x \ldots)[\mathcal{G}/x]$. Если же подстановка производится вместо *некоторых* вхождений (в том числе вместо одного), то результат подстановки не имеет общепринятого обозначения. В этом учебнике принято дублировать знак подстановки и писать между знаками список номеров вхождений, подлежащих замене.

Примеры

1. Замена всех вхождений переменной: $x \vee \neg x[y \wedge z/x] = (y \wedge z) \vee \neg(y \wedge z)$.
2. Замена всех вхождений подформулы: $x \vee y \vee z[\neg x/y \vee z] = x \vee \neg x$.
3. Замена первого вхождения переменной: $x \vee \neg x[y/1/x] = y \vee \neg x$.
4. Замена первого вхождения подформулы: $x \vee y \vee z[\neg x/1/y \vee z] = x \vee \neg x$.

Известны два правила: *правило подстановки* и *правило замены*, которые позволяют преобразовывать формулы с сохранением равносильности.

ТЕОРЕМА 3 (Правило подстановки). *Если в двух равносильных формулах вместо всех вхождений некоторой переменной x подставить одну и ту же формулу, то получатся равносильные формулы:*

$$\forall\, \mathcal{G}\ (\mathcal{F}_1(\ldots x \ldots) = \mathcal{F}_2(\ldots x \ldots) \Longrightarrow \mathcal{F}_1(\ldots x \ldots)[\mathcal{G}/x] = \mathcal{F}_2(\ldots x \ldots)[\mathcal{G}/x])\,.$$

Доказательство. Чтобы доказать равносильность двух формул, нужно показать, что они реализуют одну и ту же функцию. А это можно сделать, если взять произвольный набор значений переменных и убедиться, что значения, полученные при

вычислении формул, совпадают. Рассмотрим произвольный набор значений $a_1, \ldots, a_x, \ldots, a_n$ переменных $x_1, \ldots, x, \ldots, x_n$. Обозначим
$a := \mathrm{Eval}(\mathcal{G}, F, a_1, \ldots, a_x, \ldots, a_n)$. По определению алгоритма интерпретации

$$\mathrm{Eval}(\mathcal{F}_1[\mathcal{G}/x], F, a_1, \ldots, a_x, \ldots, a_n) = \mathrm{Eval}(\mathcal{F}_1, F, a_1, \ldots, a, \ldots, a_n)$$

и, аналогично,

$$\mathrm{Eval}(\mathcal{F}_2[\mathcal{G}/x], F, a_1, \ldots, a_x, \ldots, a_n) = \mathrm{Eval}(\mathcal{F}_2, F, a_1, \ldots, a, \ldots, a_n).$$

Но $\mathcal{F}_1 = \mathcal{F}_2$ и, значит,

$$\mathrm{Eval}(\mathcal{F}_1, F, a_1, \ldots, a, \ldots, a_n) = \mathrm{Eval}(\mathcal{F}_2, F, a_1, \ldots, a, \ldots, a_n),$$

откуда $\mathrm{Eval}(\mathcal{F}_1[\mathcal{G}/x], F, a_1, \ldots, a_x, \ldots, a_n) = \mathrm{Eval}(\mathcal{F}_2[\mathcal{G}/x], F, a_1, \ldots, a_x, \ldots, a_n)$. □

ЗАМЕЧАНИЕ

В правиле подстановки условие замены *всех* вхождений существенно:
например, $x \vee \neg x = 1$ и $x \vee \neg x[y/x] = y \vee \neg y = 1$, но $x \vee \neg x[y/1/x] = y \vee \neg x \neq 1$.

ТЕОРЕМА 4 (Правило замены)**.** *Если в формуле заменить некоторую подформулу равносильной формулой, то получится формула, равносильная исходной:*

$$\forall \mathcal{F}(\ldots \mathcal{G}_1 \ldots)\ (\mathcal{G}_1 = \mathcal{G}_2 \Longrightarrow \mathcal{F}(\ldots \mathcal{G}_1 \ldots) = \mathcal{F}(\ldots \mathcal{G}_1 \ldots)[\mathcal{G}_2//\mathcal{G}_1])\,.$$

Доказательство. Рассмотрим произвольный набор значений $a_1, \ldots, a_n$ переменных $x_1, \ldots, x_n$. Имеем $\mathcal{G}_1 = \mathcal{G}_2$ и, значит,

$$\mathrm{Eval}(\mathcal{G}_1, F, a_1, \ldots, a_n) = \mathrm{Eval}(\mathcal{G}_2, F, a_1, \ldots, a_n),$$

откуда $\mathrm{Eval}(\mathcal{F}[\mathcal{G}_1//\mathcal{G}_1], F, a_1, \ldots, a_n) = \mathrm{Eval}(\mathcal{F}[\mathcal{G}_2//\mathcal{G}_1], F, a_1, \ldots, a_n)$. □

Пусть $F = \{f_1, \ldots, f_m\}$ и $G = \{g_1, \ldots, g_m\}$. Тогда говорят, что формулы $\mathcal{F}[F]$ и $\mathcal{G}[G]$ имеют *одинаковое строение*, если $\mathcal{F}$ совпадает с результатами подстановки в формулу $\mathcal{G}$ функций f_i вместо функций g_i: $\mathcal{F}[F] = \mathcal{G}[G][f_i/g_i]_{i=1}^m$.

3.2.5. Алгебра булевых функций

Булевы функции $\vee$, $\wedge$, $\neg$ (и любые другие) могут рассматриваться как операции на множестве булевых функций, $\vee, \wedge\colon P_n \times P_n \to P_n$, $\neg\colon P_n \to P_n$. Действительно, пусть формулы $\mathcal{F}_1$ и $\mathcal{F}_2$ равносильны и реализуют функцию f, а формулы $\mathcal{G}_1$ и $\mathcal{G}_2$ равносильны и реализуют функцию g: $\mathrm{func}\,\mathcal{F}_1 = f$, $\mathrm{func}\,\mathcal{F}_2 = f$, $\mathrm{func}\,\mathcal{G}_1 = g$, $\mathrm{func}\,\mathcal{G}_2 = g$. Тогда, применяя правило замены нужное число раз, имеем
$\mathcal{F}_1 \vee \mathcal{G}_1 = \mathcal{F}_2 \vee \mathcal{G}_2$, $\mathcal{F}_1 \wedge \mathcal{G}_1 = \mathcal{F}_2 \wedge \mathcal{G}_2$, $\neg\mathcal{F}_1 = \neg\mathcal{F}_2$. Таким образом, если взять любые формулы $\mathcal{F}$ и $\mathcal{G}$, реализующие функции f и g соответственно, то каждая из формул $\mathcal{F} \wedge \mathcal{G}$, $\mathcal{F} \vee \mathcal{G}$ и $\neg\mathcal{F}$ реализует одну и ту же функцию, независимо от выбора реализующих формул $\mathcal{F}$ и $\mathcal{G}$. Следовательно, функции, которые реализуются соответствующими формулами, можно по определению считать результатами применения соответствующих операций. Другими словами, если $\mathrm{func}\,\mathcal{F} = f$, $\mathrm{func}\,\mathcal{G} = g$, то $f \wedge g \stackrel{\mathrm{Def}}{=} \mathrm{func}(\mathcal{F} \wedge \mathcal{G})$, $f \vee g \stackrel{\mathrm{Def}}{=} \mathrm{func}(\mathcal{F} \vee \mathcal{G})$, $\neg f \stackrel{\mathrm{Def}}{=} \mathrm{func}(\neg\mathcal{F})$. Алгебраическая структура $\langle P_n; \vee, \wedge, \neg\rangle$ называется *алгеброй булевых функций*.

ТЕОРЕМА. *Алгебра булевых функций является булевой алгеброй.*

Доказательство. Действительно, равносильности, перечисленные в п. 3.2.2, могут быть проверены путем построения таблиц истинности. Ясно, что эти таблицы не зависят от того, откуда взялись значения a, b, c. Таким образом, вместо a, b, c можно подставить любые функции, а значит, любые реализующие их формулы, если только выполнено правило подстановки. Следовательно, аксиомы булевой алгебры выполнены в алгебре $\langle P_n; \vee, \wedge, \neg\rangle$. □

Пусть $[\mathcal{F}]$ — множество формул, равносильных $\mathcal{F}$ (то есть класс эквивалентности по отношению равносильности). Рассмотрим множество $\mathcal{K}$ классов эквивалентности по отношению равносильности. Пусть операции $\vee, \wedge\colon \mathcal{K}\times\mathcal{K}\to\mathcal{K}$, $\neg\colon \mathcal{K}\to\mathcal{K}$ определены следующим образом:

$$[\mathcal{F}_1]\ \vee\ [\mathcal{F}_2] \stackrel{\text{Def}}{=} [\mathcal{F}_1\ \vee\ \mathcal{F}_2],\ [\mathcal{F}_1]\wedge[\mathcal{F}_2] \stackrel{\text{Def}}{=} [\mathcal{F}_1\wedge\mathcal{F}_2],\ \neg[\mathcal{F}_1] \stackrel{\text{Def}}{=} [\neg\mathcal{F}_1].$$

Тогда функция func, сопоставляющая формуле булевскую функцию, которую она реализует, является гомоморфизмом относительно операций $\wedge, \vee, \neg$, и к ней можно применить теорему о гомоморфизме (п. 1.7.6). В этом случае $\operatorname{Im}\text{func}\subset P_n$ и $\operatorname{Dom}\text{func} = \mathfrak{F}[\vee, \wedge, \neg]$.

ЗАМЕЧАНИЕ

В пп. 3.4.3 и 3.6.4 показано, что $\text{func}(\mathfrak{F}[\vee, \wedge, \neg]) = P_n$, то есть достаточно рассматривать только формулы, построенные над базисом $\{\vee, \wedge, \neg\}$.

Применение теоремы о гомоморфизме гласит $\mathfrak{F}[\vee, \wedge, \neg]/\ker\text{func} \sim \langle P_n; \vee, \wedge, \neg\rangle$.
Другими словами, алгебра классов равносильных формул (*алгебра Линденбаума–Тарского*) изоморфна алгебре булевых функций и, следовательно, является булевой алгеброй. Носитель этой алгебры — множество классов формул.

ОТСТУПЛЕНИЕ

На практике мы говорим о функциях, а пишем формулы, хотя формулы и функции — разные вещи. Например, формул бесконечно много, а функций (булевых функций n переменных) — только конечное число, и свободная алгебра формул *не изоморфна* алгебре функций. Но алгебра функций *изоморфна* алгебре классов равносильных формул, что позволяет манипулировать формулами, имея в виду функции.

3.3. Двойственность и симметрия

Понятие двойственности и симметрии с успехом применяется в самых разных областях математики. Мы рассматриваем двойственность и симметрию на простейшем примере булевых функций.

3.3.1. Двойственные и самодвойственные функции

Пусть $f(x_1, \ldots, x_n) \in P_n$ — булева функция. Тогда функция $f^*(x_1, \ldots, x_n)$, определённая следующим образом: $f^*(x_1, \ldots, x_n) \stackrel{\text{Def}}{=} \overline{f(\overline{x}_1, \ldots, \overline{x}_n)}$, называется *двойственной* к функции f.

Из определения легко видно, что двойственность инволютивна: $f^{**} = f$, и по этой причине отношение «быть двойственной к» на множестве булевых функций симметрично, то есть если $f^* = g$, то $g^* = f$.

Если в таблице истинности булевой функции f инвертировать *все* значения, то получим таблицу истинности двойственной функции f^*.