Есипов Александр Сергеевич

ЛОГИЧЕСКИЕ ОСНОВЫ ПОСТРОЕНИЯ И РАБОТЫ КОМПЬЮТЕРОВ

Журнал продолжает знакомить читателей с готовящимися к изданию учебниками по информатике.

В этом номере мы представляем учебник А.С. Есипова «Информатика и вычислительная техника». Учебник прошел многолетнюю апробацию в школах Ленинградской области и отражает, с нашей точки зрения, «классический» подход к преподаванию информатики. Учебник соответствует рекомендованной Министерством Образования Российской Федерации программе курса информатики для X-XI классов объемом в 136 часов.

Более детальное представление о стиле учебника и его особенностях читатель сможет составить по одной из глав, которая публикуется с небольшими сокращениями.

1. ОБЩИЕ СВЕДЕНИЯ О ДВОИЧНЫХ АЛГЕБРАХ.

Алгебра, геометрия, изучаемые в школе, не единственные. Математики изучают геометрии с произвольным и даже бесконечным числом измерений. Алгебр также множество.

Чтобы задать алгебру, нужно задать некоторое множество элементов (переменных и констант) и определяемое на них некоторое множество операций. Кроме этого требуется задать алфавит алгебры, правила записи и преобразования формул.

В школьной алгебре в качестве множества констант и переменных выступает множество вещественных чисел, а множество операций включает операции: сложение, вычитание, умножение, деление, возведение в степень и извлечение корня. Множество входящих в алгебру операций называют *базисом*. Результаты операций определяют функции: сложение – сумму, вычитание – разность, умножение – произведение и т.д.

Заметим, что набор операций школьной алгебры избыточен. Например, возведение в степень можно заменить умножением, умножение можно заменить сложением. Вычитание также можно заменить сложением, как это делается в компьютерах, имеющих только схему сумматора. Уменьшение набора операций привело бы к громоздким, трудно обозримым формулам. Увеличение набора добавлением других операций также нежелательно.

Особое место среди алгебр занимают двоичные алгебры. Из множества двоичных алгебр познакомимся только с двумя: алгеброй логики и булевой алгеброй. Это алгебры двоичных переменных, констант и функций, принимающих только два значения: «истина» и «ложь» в алгебре логики и, соответственно, «единицу» и «нуль» в булевой алгебре. Начнем с булевой алгебры, имеющей более простой базис, состоящий всего из трех операций.

Важность знакомства с двоичными алгебрами заключается в следующем. Во-первых, они являются математической основой построения всех логических схем компьютеров, обрабатывающих информа-

цию в двоичной системе счисления. Во-вторых, они служат математической основой решения сложных логических задач.

2. ДВОИЧНЫЕ ПЕРЕМЕННЫЕ И ФУНКЦИИ.

Двоичными называют переменные, способные принимать только два значения: 0 и 1. Двоичные функции – это функции двоичных переменных. Они также принимают только два значения – 0 и 1. Двоичные переменные и функции называют булевыми переменными и булевыми функциями (БФ).

Область определения БФ n аргументов составляют:
при n = 1 – значения 0 и 1 этого аргумента ($2^1 = 2$);
при n = 2 – наборы 00, 01, 10 и 11 – 4 набора значений аргументов ($2^2 = 4$);
при n = 3 – наборы 000, 001, 010, 011, 100, 101, 110 и 111 – 8 наборов ($2^3 = 8$) и т.д.

Можно сделать вывод, что область определения БФ n аргументов состоит из 2^n наборов. Область значений БФ – это множество всего из двух значений 0 и 1.

Конечность области определения и области значений БФ позволяют задавать эти функции не только формулами, но и таблично. В виде таблицы БФ задают значениями 0 или 1 на каждом из наборов значений ее аргументов. Такие таблицы называют *таблицами истинности* (в алгебре логики в таких таблицах, вместо 0, пишут «Л» – ложь, вместо 1, пишут «И» – истина).

Таблица 1 представляет собой пример таблицы истинности функций $f_m(x_1,x_2)$ и $f_{13}(x_1,x_2)$ двух аргументов (x_1 и x_2).

Величина α обозначает номер набора значений аргументов, m – номер функции.

Таблица 1. *Таблица истинности.*

α	x_1x_2	$f_m(x_1,x_2)$	$f_{13}(x_1,x_2)$
0	00	f(0,0)	1
1	01	f(0,1)	1
2	10	f(1,0)	0
3	11	f(1,1)	1

Функция $f_m(x_1,x_2)$ – это задание в общем виде любой из 16-ти функций двух аргументов, $f_m(0,0)$ – значение функции при подстановке в ее формулу значений $x_1 = 0$ и $x_2 = 0$, то есть на нулевом наборе аргументов, $f_m(0,1)$ – значение функции на первом наборе аргументов и так далее.

Значения булевой функции на всех наборах своих аргументов образуют двоичное число, называемое *кодом функции*. Перевод кода функции в десятичную систему дает *номер функции*. В таблице 1 записана функция $f_{13}(x_1,x_2)$, с кодом 1101 и номером 13.

С ростом числа аргументов n число различных функций резко возрастает. Если число наборов определяется формулой $K_н = 2^n$, то число функций вычисляется по формуле $K_ф = 2^{Kн} = 2^{2^n}$:
при n = 1 $K_н = 2$, $K_ф = 4$;
при n = 2 $K_н = 4$, $K_ф = 16$;
при n = 4 $K_н = 16$, $K_ф > 65000$
...
при n = 10 $K_ф$ сравнимо с числом элементарных частиц во Вселенной.

Все функции изучить невозможно. Поэтому, как и в обычной школьной алгебре, выбирают некоторое количество функций малого числа аргументов (одного, двух) и применяют их в качестве операций для построения формул функций любой сложности.

Рассмотрим булевы функции одного и двух аргументов и подробно остановимся на функциях, входящих в базис булевой алгебры, базис алгебры логики, базисы других двоичных алгебр.

3. БУЛЕВЫ ФУНКЦИИ ОДНОГО АРГУМЕНТА. ИНВЕРСИЯ. ИНВЕРТОР.

Все четыре функции на двух наборах одного аргумента показаны в таблице 2.

Таблица 2.

x	$f_0(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$
0	0	0	1	1
1	0	1	0	1

Функция $f_0(x)$ называется *константа нуль*. Она независимо от значений аргумента равна нулю.

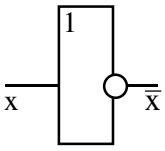


Рисунок 1. *Условное обозначение инвертора.*

Функция $f_3(x)$ называется *константа единица*. Она независимо от значений аргумента равна единице.

Функция $f_1(x)$ – это *переменная x*. Она повторяет значения переменной.

Функция $f_2(x)$ называется *инверсия* (*отрицание x*). Она входит в базис булевой алгебры и алгебры логики. Рассмотрим ее подробнее.

Инверсия – функция одного аргумента. Логическая операция над аргументом – *отрицание*. Часто отождествляют функцию с операцией и говорят: *функция отрицание* или *операция инверсия*.

Знак операции – черта над аргументом ($\bar{x}$) или значок ¬ перед аргументом (¬x). Такая запись читается: «НЕ x» или «отрицание x». В языках программирования также широко используются логические операции, реализующие булевы функции. Функция *инверсия аргумента x* записывается так: NOT x.

Функцию *инверсия* в схемах компьютера реализует логический элемент *инвертор* (элемент «НЕ»). Схема инвертора показана на рисунке 1. Работает инвертор так: если на входе 0, то на выходе 1, если на входе 1, то на выходе 0.

4. БУЛЕВЫ ФУНКЦИИ ДВУХ АРГУМЕНТОВ.

Двум аргументам соответствуют четыре набора их значений, что приводит к шестнадцати различным кодам функций ($n = 2$, $K_н = 2^n = 4$, $K_ф = 2^{K_н} = 16$). Все 16 функций двух аргументов записаны в таблице 3.

Среди функций двух аргументов имеются уже знакомые по функциям одного аргумента:

f_0 и f_{15}- соответственно функция *константа 0* и ее инверсия – *константа 1*;

f_3 и f_{12}- *переменная* x_1 и ее *инверсия* $\bar{x}_1$;

f_5 и f_{10} – *переменная* x_2 и ее *инверсия* $\bar{x}_2$.

Функции f_0 и f_{15} фиктивно зависят от обоих аргументов – изменения значений аргументов не влияют на значения этих функций. Функции f_3, f_5, f_{10} и f_{12} фиктивно зависят от одного аргумента и существенно зависят от другого аргумента.

Из оставшихся десяти функций также можно выделить пять пар – пять функций и их инверсий: f_1 и f_{14}, f_2 и f_{13}, f_4 и f_{11}, f_6 и f_9, f_7 и f_8. Рассмотрим их.

Конъюнкция. Булева функция $f_1(x_1,x_2)$ с кодом 0001 называется *конъюнкцией*. *Конъюнкция* – это такая булева функция, которая равна единице тогда и только тогда, когда все аргументы функции равны единице. Другое определение – это такая функция, которая равна нулю, если хотя бы один аргумент функции равен нулю.

Таблица истинности функции *конъюнкция* – таблица 4.

Таблица 4. *Таблица истинности функции конъюнкция.*

α	x_1 x_2	$f_1(x_1x_2)$
0	0 0	0
1	0 1	0
2	1 0	0
3	1 1	1

Таблица 3. *Булевы функции двух аргументов.*

α	x_1x_2	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	00	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
1	01	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
2	10	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
3	11	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Функции *конъюнкция* соответствует операция *логическое умножение*.

Знак операции: &, · или ∧ (в алгебре логики). В формулах, как и в обычной алгебре, знак чаще всего опускается:

$f(x_1,x_2) = x_1 \,\&\, x_2 = x_1 \cdot x_2 = x_1x_2$.

Читается формула так: «x_1 и x_2».

Запись в языках программирования: x_1 AND x_2.

Функцию *конъюнкция* реализует логический элемент конъюнктор (элемент «И»). Условное обозначение конъюнктора в логических схемах показано на рисунке 2.

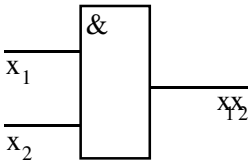


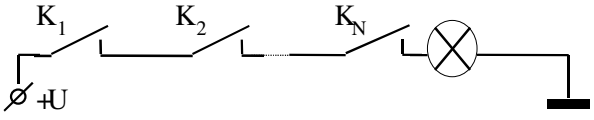
Рисунок 2. *Условное обозначение логического элемента конъюнктор.*

Используя принцип суперпозиции (подстановку функции в качестве аргументов в другую функцию), функция *конъюнкция* обобщается на n аргументов:

$f(x_1,x_2,...,x_n) = x_1 \,\&\, x_2 \,\&\, ... \,\&\, x_n$.

В качестве содержательного примера реализации функции *конъюнкция* рассмотрим схему голосования «только все!». На рисунке 3 показана цепь с N кнопками, позволяющими включать индикаторную лампочку. На электрических выключателях принято отмечать: «0» – выключено и «1» – включено. Лампочка засветится только в случае, если будут замкнуты все ключи, то есть на все N входов будут «поданы» единицы. Такая схема реализует функцию *конъюнкция*.

Рисунок 3. *Реализация функции конъюнкции в схеме голосования «только все».*



Дизъюнкция. Булева функция $f_7(x_1,x_2)$ с кодом функции 0111 называется *дизъюнкцией*. *Дизъюнкция* – это такая двоичная функция, которая равна нулю тогда и только тогда, когда все аргументы функции равны нулю. Другое определение: *дизъюнкция* – это такая функция, которая равна единице, если хотя бы один аргумент равен единице.

Таблица истинности функции дизъюнкция – таблица 5.

Таблица 5. *Таблица истинности функции дизъюнкция.*

a	$x_1\ x_2$	$f_1(x_1x_2)$
0	0 0	0
1	0 1	1
2	1 0	1
3	1 1	1

Функции *дизъюнкция* соответствует операция *логическое сложение*.

Знак операции: ∨. Пример записи формулы функции *дизъюнкция*:

$f(x_1,x_2) = x_1 \vee x_2$.

Читается формула так: «x_1 или x_2».

Запись на языках программирования: x_1 OR x_2.

Функцию *дизъюнкция* реализует логический элемент дизъюнктор (элемент «ИЛИ»). Условное обозначение дизъюнктора показано на рисунке 4.

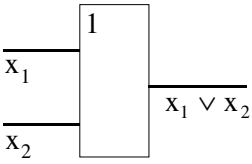


Рисунок 4. *Условное обозначение логического элемента конъюнктор.*

Функция *дизъюнкция* обобщается на n аргументов:

$f(x_1,x_2,...,x_n) = x_1 \vee x_2 \vee ... \vee x_n$.

В качестве примера реализации функции *дизъюнкция* рассмотрим схему голосования «хотя бы один!». На рисунке 5

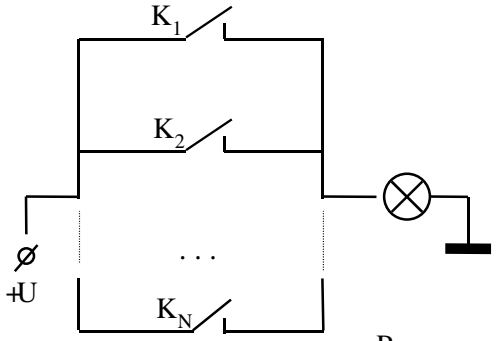


Рисунок 5. *Реализация функции дизъюнкция в схеме голосования «хотя бы один».*

показана цепь с N кнопками, позволяющими включать индикаторную лампочку. Лампочка засветится в случае, если будет замкнут хотя бы один ключ, то есть схема реализует функцию *дизъюнкция*.

Инверсия конъюнкции. Функция Шеффера. В таблице истинности функций двух аргументов (таблица 3) эта функция $f_{14}(x_1,x_2)$. Она имеет код функции 1110.

Запись формулы функции следующая:

$$f(x_1,x_2) = \overline{x_1 \& x_2} = \overline{x_1 \cdot x_2} = \overline{x_1 x_2}$$

Инверсия конъюнкции, как видно из ее формулы, образована из двух более простых функций: *конъюнкции* и *инверсии*. Эта функция замечательна тем, что она в единственном числе образует базис алгебры – алгебры Шеффера, позволяющей записать в виде формулы любую двоичную логическую функцию.

В алгебре Шеффера единственная функция – функция Шеффера. Знак операции: «|»- штрих Шеффера. В алгебре Шеффера функция *инверсия конъюнкции* записывается так: $f(x_1,x_2) = x_1 | x_2$.

Логический элемент, реализующий функцию Шеффера, называется элементом Шеффера или элементом «И-НЕ». Условное обозначение элемента И-НЕ показано на рисунке 6.

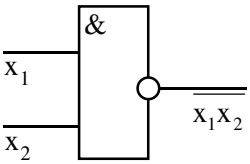


Рисунок 6. *Условное обозначение логического элемента И-НЕ.*

Функция Шеффера также обобщается на n аргументов.

Инверсия дизъюнкции. Функция Пирса. В таблице 2 она под номером 8 и имеет код функции 1000. Это функция

$$f_8(x_1,x_2) = \overline{x_1 \vee x_2}$$

Функцию *инверсия дизъюнкции* еще называют функцией Пирса. Она, как и функция Шеффера, универсальна в том смысле, что в единственном числе образует алгебру Пирса и позволяет записать формулой любую двоичную функцию. Знак операции: «↓» – стрелка Пирса. Запись функции Пирса: $f(x_1,x_2) = x_1 \downarrow x_2$.

Логический элемент, реализующий функцию Пирса, называют элементом Пирса или элементом ИЛИ-НЕ. Условное обозначение элемента на рисунке 7.

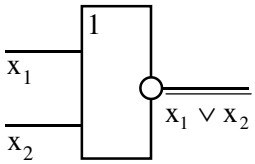


Рисунок 7. *Условное обозначение логического элемента ИЛИ-НЕ.*

Импликация. Это логическая функция двух двоичных логических переменных x_1 и x_2. Различают *импликацию* от x_1 к x_2 (функция f_{13} с кодом 1101) и *импликацию* от x_2 к x_1 (функция f_{11} с кодом 1011). Обе эти функции приведены в таблице 6. Операцию, соответствующую функции *импликация*, также называют *импликацией* и обозначают стрелкой от одной логической переменной к другой.

Таблица 6. *Функция импликация.*

x_1 x_2	$x_1 \rightarrow x_2$	$x_1 \rightarrow x_2$
0 0	1	1
0 1	1	0
1 0	0	1
1 1	1	1

Операция *импликация* входит в базис алгебры логики, где оперирует с истинными и ложными высказываниями и обозначает логическое следствие.

Запись формул функций *импликация* в булевой алгебре и в алгебре логики следующая:

$$f_{13}(x_1,x_2) = \overline{x_1} \vee x_2 = x_1 \rightarrow x_2;$$

$$f_{11}(x_1,x_2) = x_1 \vee \overline{x_2} = x_2 \rightarrow x_1.$$

Логическая операция IMP (импликация) используется в языках программирования. Запись следующая: x_1 IMP x_2.

Неравнозначность. Эту логическую функцию двух аргументов еще называют «исключительным ИЛИ», а также «суммой по модулю 2». Код функции – 0110, номер – 6 в таблице 3. Функция принимает единичное значение только на тех наборах значений двух аргументов, на ко-

торых эти значения различны (01 и 10). Функции *неравнозначность* соответствует логическая операция *сумма по модулю 2*. Операция входит в базис алгебры Жегалкина и имеет знак операции: ⊕. Записи формулы функции *неравнозначность* в булевой алгебре, алгебре Жегалкина и в языках программирования следующие:

$f_9(x_1, x_2) = \overline{x_1}x_2 \vee x_1\overline{x_2}$,
$f_6(x_1,x_2) = x_1 \oplus x_2$,
x_1 XOR x_2.

Равнозначность. Является отрицанием *неравнозначности*. Функция *равнозначность* (код функции 1001, номер функции 9) принимает значение 1 только на наборах с одинаковыми значениями аргументов (00 и 11). В качестве операции *двойной импликации* включается в базис алгебры логики (знак операции: ↔). Функцию *равнозначность* также называют *эквивалентностью*. Запись формулы функции в булевой алгебре, алгебре логики и в языках программирования следующая:

$f_9(x_1,x_2) = x_1 \leftrightarrow x_2$,
$f_9(x_1, x_2) = \overline{x_1} \leftrightarrow \overline{x_2}$,
x_1 EQV x_2.

Функции запрета. Это функции f_2 и f_4. Они являются отрицанием к функциям импликации. Широко использовались при проектировании и описании работы логических элементов ЭВМ 1-го и 2-го поколений.

5. АЛГЕБРЫ. СРАВНЕНИЕ ПО НАБОРУ ОПЕРАЦИЙ.

Из перечисленных в таблице 7 двоичных алгебр широко используются на практике только две – алгебра Буля и алгебра логики. Алгебры Шеффера и Пирса активно изучались на предмет возможности построения компьютера из однотипных элементов. К тому же, ввиду технологических особенностей, наиболее просто реализуются именно логические элементы Шеффера (И-НЕ) и Пирса (ИЛИ-НЕ).

6. БУЛЕВА АЛГЕБРА. ОСНОВНЫЕ ЗАКОНЫ И ТОЖДЕСТВА.

Многие законы и правила преобразования формул булевой алгебры совпадают с законами в школьной алгебре. Но среди них есть тождества, присущие только булевой алгебре. Перечислим все законы булевой алгебры. Это тождества, используемые для преобразования формул логических функций. Их называют правилами или законами преобразования формул.

1. $XY = YX$ переместительный закон
   $X \vee Y = Y \vee X$ (коммутативность)
2. $X(YZ) = (XY)Z$ сочетательные законы
   $X \vee (Y \vee Z) = (X \vee Y) \vee Z$ (ассоциативность)
3. $X(Y \vee Z) = XY \vee XZ$ распределительный
   $X \vee YZ = (X \vee Y)(X \vee Z)$ закон (дистрибутивность)
4. $XX = X$ тавтология
   $X \vee X = X$ (идемпотентность)
5. $X \cdot 1 = X$ правила выполнения
   $X \vee 0 = X$ операций над константами
   $X \cdot 0 = 0$
   $X \vee 1 = 1$
6. $X\overline{X} = 0$ законы инверсии для
   $X \vee \overline{X} = 1$ конъюнкции и дизъюнкции
7. $\overline{X \cdot Y} = \overline{X} \vee \overline{Y}$ правила де Моргана
   $\overline{X \vee Y} = \overline{X} \cdot \overline{Y}$

Таблица 7. *«Школьная» и двоичные алгебры.*

Алгебра	Множество переменных и констант	Базис алгебры (набор операций)
Школьная	вещественные числа	+, -, *, /, $\sqrt{}$, возведен. в степень
Булева	наборы из нулей и единиц	&, ∨, ¬
Шеффера	– « -	| – штрих Шеффера
Пирса	– « -	↓ – стрелка Пирса
Жегалкина	– « -	& , ⊕, 1
Алгебра логики	– « -	&, ∨, ¬, →, ↔

8. $XY \vee X\overline{Y} = X$ правила склеивания $(X \vee Y)(X \vee \overline{Y}) = X$
9. $X \vee XY = X$ правила поглощения $X(X \vee Y) = X$
10. $\overline{\overline{X}} = X$ правило двойной инверсии

Обратим внимание на то, что большинство законов и правил записаны в две строки. При этом вторая строка образуется из первой заменой *конъюнкции* на *дизъюнкцию*, *дизъюнкции* на *конъюнкцию* и заменой констант 1 на 0 и 0 на 1. В этом выражается двойственный характер функций *конъюнкция* и *дизъюнкция*. Это замечательное свойство двойственности позволяет, например, сделать следующее. Если в логической схеме заменить все конъюнкторы на дизъюнкторы, дизъюнкторы на конъюнкторы, единичные сигналы на нулевые и нулевые на единичные, то схема будет реализовывать ту же исходную функцию.

Докажем справедливость некоторых тождеств. Из способов доказательства выберем два. Первый способ заключается в приведении путем преобразования одной части равенства к другой части. Если это удается сделать, то тождество можно считать доказанным. Второй способ состоит в определении кодов функций левой и правой частей равенства. Если коды равны, то и функции равны, что доказывает справедливость тождества.

Первый способ. Первая строка распределительного закона не вызывает сомнений. Докажем справедливость второй строки:

$X \vee YZ = (X \vee Y)(X \vee Z)$

Правую часть равенства преобразуем к виду левой части. Справа будем записывать используемые при преобразовании законы и правила:

$(X \vee Y)(X \vee Z) =$ (раскрываем скобки
$XX \vee XY \vee XZ \vee YZ =$ $X \cdot X = X$)
$X \vee XY \vee XZ \vee YZ =$ (выносим X за скобки)
$X(1 \vee Y \vee Z) \vee YZ =$ (используем $X \vee 1 = 1$)
$X \vee YZ$

Справедливость тождества доказана.

Второй способ. Докажем справедливость правила де Моргана. Словами его можно сформулировать так: 1) *инверсия конъюнкции равна дизъюнкции инверсий* и 2) *инверсия дизъюнкции равна конъюнкции инверсий*. Докажем первую часть правила: $\overline{X \vee Y} = \overline{X} \cdot \overline{Y}$. Для доказательства используем таблицу 8, строки которой соответствуют наборам значений аргументов X и Y.

Таблица 8.

XY	$\overline{XY}$	$\overline{X} \vee \overline{Y}$
0 0	$\overline{0 \cdot 0} = \overline{0} = 1$	$\overline{0} \vee \overline{0} = 1 \vee 1 = 1$
0 1	$\overline{0 \cdot 1} = \overline{0} = 1$	$\overline{0} \vee \overline{1} = 1 \vee 0 = 1$
1 0	$\overline{1 \cdot 0} = \overline{0} = 1$	$\overline{1} \vee \overline{0} = 0 \vee 1 = 1$
1 1	$\overline{1 \cdot 1} = \overline{1} = 0$	$\overline{1} \vee \overline{1} = 0 \vee 0 = 0$

Подставляя в формулы левой и правой части равенства значения аргументов, вычисляем значения функций. В итоге получили, что коды обеих функций совпали. Тождество доказано.

7. КАНОНИЧЕСКИЕ ФОРМЫ БУЛЕВЫХ ФУНКЦИЙ И ПЕРЕХОДЫ ОТ ОДНОГО СПОСОБА ЗАДАНИЯ К ДРУГОМУ.

7.1. Совершенная дизъюнктивная нормальная форма.

Одна и та же функция может быть задана множеством различных формул. Поэтому функции трудно сравнивать. Канонические формы – это запись функций по единым правилам. Такие формы еще называют совершенными. Различают дизъюнктивную и конъюнктивную совершенные нормальные формы. Определим дизъюнктивную форму (пока не совершенную).

Дизъюнктивной нормальной формой (ДНФ) булевой функции называют *дизъюнкцию конъюнкций* (логическую сумму логических произведений). Формула функции в ДНФ не имеет скобок и общих для нескольких аргументов отрицаний.

Пример ДНФ функции:

$F(X, Y, Z) = X \vee \overline{Y}Z \vee \overline{X}Z \vee X\overline{Y}Z$.

Для определения совершенной ДНФ (СДНФ) рассмотрим полную (все аргументы) *конъюнкцию* отрицаемых или неотрицаемых аргументов. Обозначают – $K(\alpha)$,

где α – номер того единственного набора, на котором $K(\alpha) = 1$.

Пример: $K(5) = X\overline{Y}Z$. K(5) равна 1 только на 5-м наборе значений аргументов (101).

Совершенной дизъюнктивной нормальной формой (СДНФ) функции называют *дизъюнкцию полных конъюнкций*, равных единице на тех же наборах, что и функция.

7.2. Переход от таблицы истинности функции к СДНФ.

Пусть заданы две булевы функции 3-х аргументов таблицей истинности (таблица 9). Функцию F(X,Y,Z) запишем в СДНФ. Функцию W(X,Y,Z) используем для следующих построений.

Таблица 9.

α	XYZ	F(X,Y,Z)	W(X,Y,Z)
0	0 0 0	0	0
1	0 0 1	1	1
2	0 1 0	1	0
3	0 1 1	0	1
4	1 0 0	1	0
5	1 0 1	0	1
6	1 1 0	0	1
7	1 1 1	1	0

По определению СДНФ выпишем из таблицы *полные конъюнкции* для наборов, на которых функция F равна 1:

$$F = K(1,2,4,7) = K(1) \vee K(2) \vee K(4) \vee K(7) =$$

$$\left\{ \begin{array}{cccccccc} XYZ & \vee & XYZ & \vee & XYZ & \vee & XYZ \\ 0\,0\,1 & & 0\,1\,0 & & 1\,0\,0 & & 1\,1\,1 \end{array} \right\}$$

$$= \overline{X} \cdot \overline{Y} \cdot Z \vee \overline{X} \cdot Y \cdot \overline{Z} \vee X \cdot \overline{Y} \cdot \overline{Z} \vee X \cdot Y \cdot Z$$

Правило перехода от таблицы к формуле функции в СДНФ следующее.

Для перехода нужно записать дизъюнкцию полных конъюнкций по числу единиц в коде функции (в примере – 4 единицы), подписать под ними наборы, на которых функция равна единице, и поставить отрицания аргументов, соответствующих нулям в этих наборах.

7.3. Переход от схемы к формуле функции.

Переход от логической схемы (ЛС), построенной из логических элементов к формуле булевой функции требует знания булевых функций, реализуемых логическими элементами. Начиная от входов ЛС, используя принцип суперпозиции (подстановки функций в качестве аргументов в другие функции), получаем на выходе функцию, реализуемую всей ЛС. Рассмотрим этот переход на примере ЛС, показанной на рисунке 8.

7.4. Переход от формулы к СДНФ и таблице истинности.

Такой переход можно сделать, выполнив следующие действия:

1. При наличии общих для нескольких переменных отрицаний, используя правило де Моргана, «опускаем» их на переменные. При наличии двойных отрицаний убираем их:

$$W = \overline{X\overline{Y}Z} \vee \overline{\overline{XY} \cdot \overline{Z}} = (\overline{X} \vee \overline{Y})Z \vee \overline{\overline{XY}}\,\overline{Z} =$$
$$= (\overline{X} \vee \overline{Y})Z \vee XY\overline{Z}$$

2. Раскрываем все скобки и добавляем недостающие переменные умножением членов полученной ДНФ на скобки, равные единице:

$$W = \overline{X}Z \vee \overline{Y}Z \vee XY\overline{Z} =$$
$$= \overline{X}(Y \vee \overline{Y})Z \vee (X \vee \overline{X})\overline{Y}Z \vee XY\overline{Z}$$

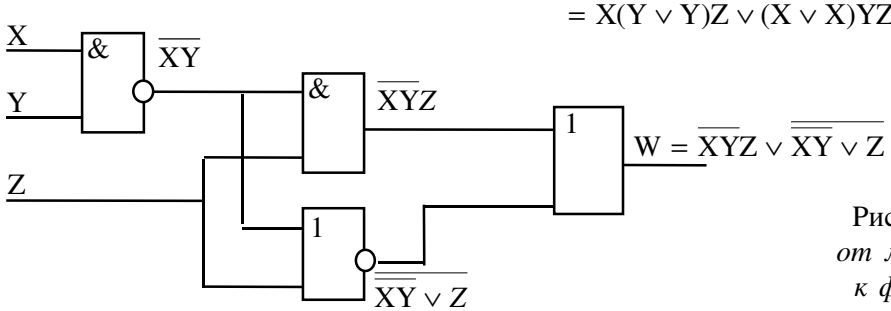


Рисунок 8. *Переход от логической схемы к формуле функции.*

3. Раскрываем скобки и оставляем в формуле только по одной из повторяющихся конституент (используем правило X∨X = X). В результате получим СДНФ функции:

$$W = \overline{X}YZ \vee \underline{\overline{X} \cdot \overline{Y}Z} \vee X\overline{Y}Z \vee \underline{\overline{X} \cdot \overline{Y}Z} \vee XY\overline{Z} =$$

$$= \overline{X}YZ \vee \overline{X}\overline{Y}Z \vee X\overline{Y}Z \vee XY\overline{Z}$$

4. Осталось определить номера наборов, на которых полученные конституенты единицы равны единице. Это соответственно наборы: 011, 001, 101 и 110 (наборы 3, 1, 5 и 6). Записываем в таблице истинности на этих наборах 1, на остальных наборах 0. Функция W(X,Y,Z) записана в таблице 9.

7.5. Переходы «Задача – алгоритм – таблица истинности – формулы функций – логическая схема».

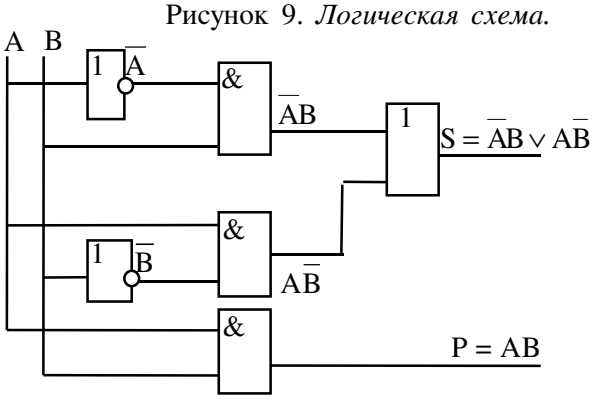
Постановка задачи. Построить логическую схему, суммирующую два одноразрядных двоичных числа A и B и вырабатывающую их сумму S и перенос P.

Алгоритм. Алгоритм сложения можно задать, например, описательно, в виде блок-схемы, другими способами. Ввиду простоты алгоритма, запишем его непосредственно таблицей истинности – таблицей работы логической схемы:

Таблица истинности.

AB	S	P
0 0	0	0
0 1	1	0
1 0	1	0
1 1	0	1

Рисунок 9. *Логическая схема.*



Формулы функций. По рассмотренным выше правилам выписываем формулы суммы и переноса:

$$S = \overline{A}B \vee A\overline{B}; \quad P = AB.$$

Простейший анализ формул показывает, что для построения схемы требуется два инвертора (элемента НЕ), три конъюнктора (элемента И), и один дизъюнктор (элемент ИЛИ). Логическая схема показана на рисунке 9.

Обратим внимание на важный момент построения логической схемы. В качестве булевых переменных выступают цифровые разряды двоичного числа. Пример показывает применение булевой алгебры для построения всех логических схем компьютера, преобразующего информацию, представленную в двоичной системе счисления.

8. ПРИМЕНЕНИЕ ЛОГИЧЕСКИХ ОПЕРАЦИЙ ПРИ ПРОГРАММИРОВАНИИ.

Результаты операций отношения (знаки операций: =, <>, >, <, > =, < =) принимают значение «истина» (TRUE), если условие выполняется и «ложь» (FALSE), если условие не выполняется. Отношения можно интерпретировать как простые высказывания, которые могут быть истинными или ложными.

С помощью логических операций AND, OR, NOT, XOR, EQV и IMP из простых отношений можно строить сложные, составные конструкции и использовать их в качестве сложных условий, например, в операторах IF-THEN-ELSE.

Программа 1 печатает результаты операций отношения величин чисел a, b и c.

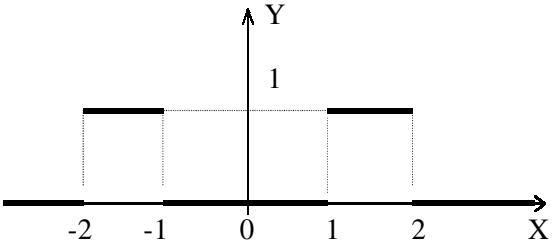
Программа 1.

```
REM Значения результатов отношений (истина – «-1», ложь – «0»)
a = 5: b = 3: c = 1
f1 = a > b                          (f1 = -1)
f2 = a < b                          (f2 = 0)
f3 = a < b OR b > c                 (f3 = -1)
f4 = a > b AND NOT (b > c)          (f4 = 0)
PRINT «f1=»; f1, «f2=»; f2, «f3=»; f3, «f4=»; f4
END
```

В качестве компонент отношений могут выступать арифметические выражения. В таких смешанных выражениях соблюдается следующая очередность выполнения операций: арифметические, отношения, логические. При этом очередность выполнения логических операций такая: NOT, AND, OR, XOR и EQV, IMP.

Рассмотрим пример использования сложного логического условия. Пусть требуется табулировать функцию, заданную графиком на рисунке 10.

Рисунок 10. *График функции.*



Это можно сделать с помощью следующей программы:

Программа 2.

```
REM Табуляция функции, заданной графиком
INPUT «x =», x
IF x<-2 OR x>-1 AND x<1 OR x>2 THEN y = 0 ELSE y = 1
PRINT «При x =»; x , «y =»; y
END
```

Эту же функцию можно задать так:

```
IF x>= -2 AND x<= -1 OR x>= 1 AND x<= 2 THEN y = 1 ELSE y = 0.
```

9. ПРИМЕНЕНИЕ КОМПЬЮТЕРА ДЛЯ МОДЕЛИРОВАНИЯ ЛОГИЧЕСКИХ ФУНКЦИЙ.

Некоторые из рассмотренных выше задач связаны с преобразованиями формул булевых выражений, вычислением их значений на различных наборах значений аргументов. К таким задачам можно отнести доказательства справедливости основных законов и тождеств булевой алгебры, переход от произвольной формулы функции к таблице ее истинности, другие задачи. Заметим, что подобные тождественные преобразования даже несложных функций вызывают затруднения, требуют внимательности, связаны с возможными ошибками. В таких случаях следует привлекать компьютер.

Программа 2 строит и заполняет таблицу истинности. Она последовательно вырабатывает все наборы значений аргументов и вычисляет значения функции на этих наборах. Программа легко дополняется для генерации наборов 4-х, 5-и и более аргументов. В рассмотренном примере программа работает с формулой функции из 7.4:

$$W = \overline{XYZ} \vee \overline{\overline{XY}} \vee Z$$

Программа 3.

```
REM Переход от формулы к таблице истинности
CLS : PRINT « x»; « y»; « z», « w»: PRINT
FOR x = 0 TO 1
FOR y = 0 TO 1
FOR z = 0 TO 1
w = NOT (x AND y) AND z OR NOT (NOT (x AND y) OR z)
PRINT x; y; z, w
NEXT z, y, x
END
```

В результате работы программы будет распечатана таблица истинности функции W(x,y,z) аналогичная таблице 9.

При моделировании логических функций возможно получение в коде функций значений «-1» вместо «1» и «-2» вместо «0». Это легко изменить перед печатью результатов.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ.

1) В таблице истинности заданы четыре булевы функции трех переменных: f_1, f_2, f_3 и f_4 (таблица 10). Для каждой из этих функций выполнить следующее:

а) перейти от таблицы истинности к записи функции в СДНФ (совершенной дизъюнктивной нормальной форме);

б) используя законы и правила преобразования формул булевых функций, в частности операции склеивания и поглощения, упростить запись функции (уменьшить в формуле число вхождений переменных

Таблица 10.

α	$x_1x_2x_3$	f_1	f_2	f_3	f_4
0	0 0 0	1	0	1	1
1	0 0 1	0	1	0	1
2	0 1 0	0	0	0	0
3	0 1 1	1	1	0	0
4	1 0 0	1	1	1	1
5	1 0 1	0	0	1	1
6	1 1 0	0	1	1	0
7	1 1 1	1	0	1	1

и их отрицаний);

в) реализовать полученную запись функции логической схемой на элементах И, ИЛИ и НЕ.

2) Для заданных на рисунке 11 логических схем, реализующих булевы функции 3-х аргументов, выполнить следующие действия:

а) записать формулы реализуемых логическими схемами функций;

б) перейти от произвольной формы записи формул функций к записи их в СДНФ;

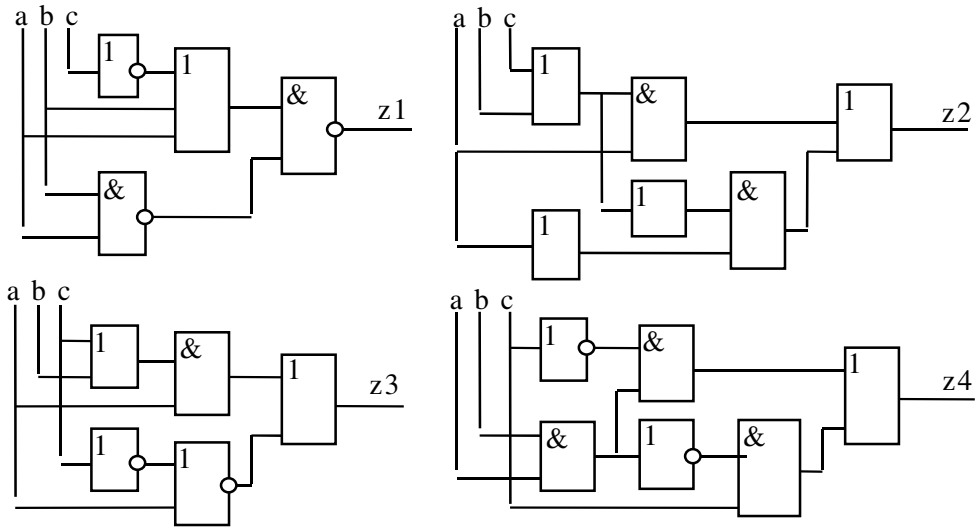
в) перейти от СДНФ функций к записи их в таблице истинности;

г) подать на входы схемы набор значений двоичных переменных, например, 101 и проследить значения сигналов на выходах логических элементов и на выходах логических схем; сравните полученные значения со значениями функций на этом наборе в таблице истинности.

3) Как было показано, любую булеву функцию можно записать в СДНФ. Это говорит о функциональной полноте булевого базиса (конъюнкция, дизъюнкция и инверсия). Чтобы убедиться в функциональной полноте базисов алгебр Шеффера, Пирса и Жегалкина, достаточно с помощью функций, входящих в их базисы, реализовать функции булева базиса.

Например, инверсия в базисах алгебр Шеффера, Пирса и Жегалкина реализуется так: $\bar{x} = x|x = x \downarrow x = 1 \oplus x$. Реализуйте в этих базисах функции конъюнкцию и дизъюнкцию.

Рисунок 11. *Логические схемы к заданию 2 самостоятельной работы.*



Есипов Александр Сергеевич, кандидат техн. наук, доцент, учитель информатики Сосновской средней школы Приозерского р-на Лен. области.

НАШИ АВТОРЫ