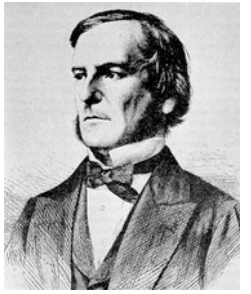
Основные понятия алгебры логики

Алгебра двоичной логики – раздел математики, изучающий логические операции над двоичными переменными.



Джордж Буль (1815–1864)

Логическая (булева) переменная – такая переменная, значения которой могут быть лишь "1" или "0".

В естественном языке «булева переменная» = «высказывание».

Высказывание – утверждение, про которое можно однозначно сказать, истинно оно или ложно.

Обозначения: "истина" и "ложь", "true" и "false" или "1" и "0".

Логические функции и операции

Логическая (булева) функция f(x, y, z, ...) – функция, использующая входные логические переменные x, y, z, выполняющая логические операции над ними и генерирующая значения 0 или 1.

Основные логические операции:

$\overline{X}$ – отрицание или инверсия ($\neg X$)

$X \wedge Y$ – логическое умножение или конъюнкция

$X \vee Y$ – логическое сложение или дизъюнкция

Таблица истинности функции – таблица всех значений этой логической функции.

Таблицы истинности

Существует только четыре разных логических функций одной переменной *F(X)*. Любая другая самая сложная функция будет иметь одну из четырёх таблиц истинности (ТИ):

X	$F_0(X)=0$	$F_1(X)=X$	$F_2(X)=\overline{X}$	$F_3(X)=1$
0	0	0	1	1
1	0	1	0	1

$$F(X)=\overline{X}\wedge X\vee X$$

Логическая функция двух переменных *F(X, Y)*. Сколько существует разных $F_i(X, Y)$?

X	Y	$F_1(X,Y)=X\vee Y$	$F_2(X,Y)=X\wedge Y$	$F_3(X,Y)=\ldots$
0	0	0	0	…
0	1	1	0	…
1	0	1	0	…
1	1	1	1	…

Таблицы истинности общего вида

Значения K элементов булевых операндов					Значение булевых функций вида $F(A_1, A_2, ..., A_{K-1}, A_K)$			
A_1	A_2	...	A_{K-1}	A_K	*FK,0*	*FK,1*	...	*FK,N*
0	0	...	0	0	0	1	...	1
0	0	...	0	1	0	0	...	1
0	0	...	1	0	0	0	...	1
...	...	...	...	...	...	...	...	...
1	1	...	1	1	0	0	...	1

L строк (от 000...0 до 111...1 сверху вниз)

N колонок (от 000...0 до 111...1 слева направо)

$$L = 2^K, N = 2^L \Rightarrow N = 2^{(2^K)}$$

Пример 1. Число булевых функций одной переменной равно 4.

Пример 2. Число булевых функций двух переменной равно 16.

Обобщённая таблица истинности

X	1	1	0	0	Обозначение	Название оператора
Y	1	0	1	0		
F	0	0	0	0	F2,0 = FALSE	Противоречие, логический нуль
F	0	0	0	1	F2,1 = X ↓ Y = X NOR Y = NOR (X, Y) = = X НЕ-ИЛИ Y = НЕ-ИЛИ (X, Y)	Стрелка Пирса, функция Вебба
F	0	0	1	0	F2,2 = X ←/ Y	Отрицание обратной импликации
F	0	0	1	1	F2,3 = ¬X	Отрицание
F	0	1	0	0	F2,4 = X →/ Y	Материальная обратная импликация
F	0	1	0	1	F2,5 = ¬Y	Отрицание
F	0	1	1	0	F2,6 = X ⊕ Y = X XOR Y = XOR (X, Y) = X >< Y = = X <> Y = X NE Y	Сложение по модулю 2, исключающее «ИЛИ», сумма Жегалкина, не равно
F	0	1	1	1	F2,7 = X | Y = X NAND Y = NAND (X, Y) = = X НЕ-И Y = НЕ-И (X, Y)	Штрих Шеффера, пунктир Чулкова

Обобщённая таблица истинности (2)

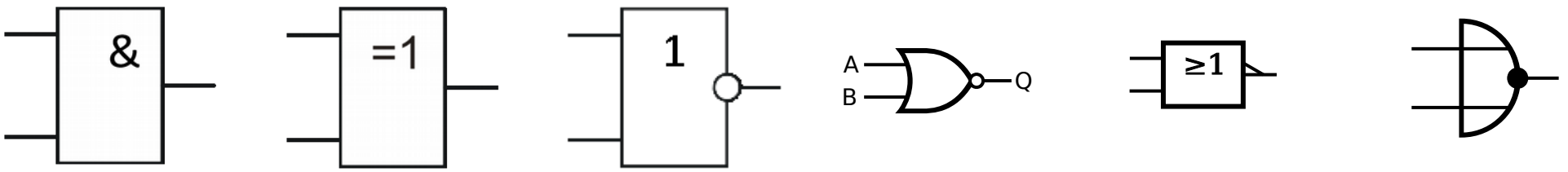
X	1	1	0	0	Обозначение	Название оператора
Y	1	0	1	0		
F	1	0	0	0	F2,8 = X ∧ Y = X * Y = XY = X AND Y = = min (X, Y)	Конъюнкция
F	1	0	0	1	F2,9 = X ≡ Y = X ~ Y = X ↔ Y = EQV (X, Y)	Эквивалентность, эквиваленция, равнозначность
F	1	0	1	0	F2,10 = Y	Проекция, повторение
F	1	0	1	1	F2,11 = X ⟶ Y = X ⊃ Y = X ≤ Y = X LE Y	Импликация, следование
F	1	1	0	0	F2,12 = X	Проекция, повторение
F	1	1	0	1	F2,13 = X ⟵ Y	Обратная импликация
F	1	1	1	0	F2,14 = X ∨ Y = X OR Y = OR (X, Y) = X ИЛИ Y = = max (X, Y)	Дизъюнкция
F	1	1	1	1	F2,15 = TRUE	Тавтология, логическая единица

Обозначение булевых функций на электрич. схеме

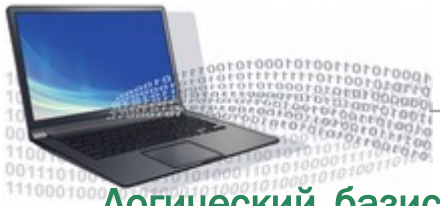
Логический элемент – простейшее устройство ЭВМ, выполняющее одну определённую логическую операцию над входными сигналами согласно правилам алгебры логики.

Современные стандарты для условных графических обозначений (УГО) логических элементов:

- ANSI (*англ*. American National Standards Institute – американский национальный институт стандартов)
- MIL/IEC (*англ*. MILitary – военный, International Electrotechnical Commission – международная электротехническая комиссия)
- DIN (*нем*. Deutsches Institut für Normung e.V. – немецкий институт по стандартизации)
- ГОСТ 2.743-91 Единая система конструкторской документации. Обозначения условные графические в схемах. Элементы цифровой техники

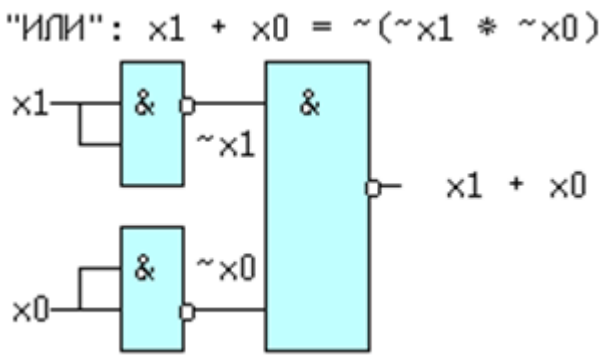


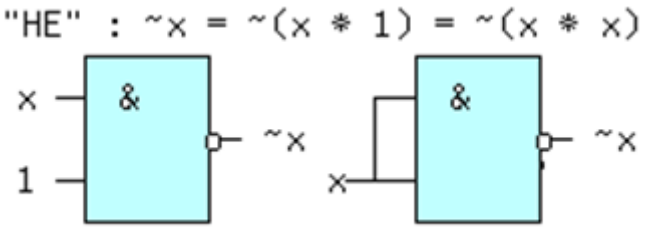
И Исключающее ИЛИ ИЛИ-НЕ *ANSI* *IEC* *DIN*

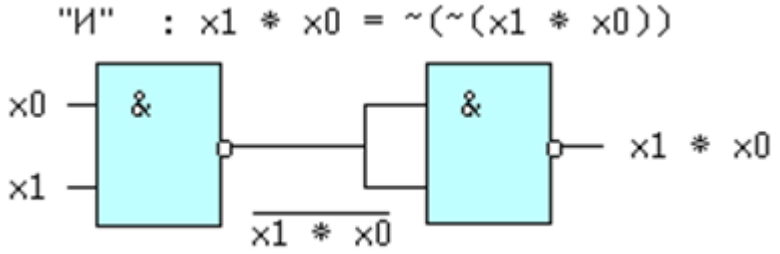


Логический базис (ЛБ) – набор булевых функций, позволяющих реализовать любую другую булеву функцию. Три наиболее востребованных логических базиса: {**И, ИЛИ, НЕ**}, {**И-НЕ**}, {**ИЛИ-НЕ**}.

Пример. Реализация функций И, ИЛИ, НЕ в базисе И-НЕ (картинки взяты с сайта *http://phys.bspu.by/static/lib/teh/electron/cif-obu.htm*)







Цена логической схемы

Уиллард Квайн (1908–2000)

Цена схемы по Квайну – суммарное число входов во всех логических элементах.

Минимизация функции – сокращение цены функции с помощью преобразования её к более простому эквивалентному выражению.

$F(X, Y, Z) = \neg X \vee Y \vee XY\neg Z \vee Z$

$\neg X = T_1$, $\neg Z = T_2$ $\quad S_Q = 2$

$F(X, Y, Z) = T_1 \vee Y \vee XYT_2 \vee Z$

$XYT_2 = T_3$ $\quad S_Q = 2 + 3$

$F(X, Y, Z) = T_1 \vee Y \vee T_3 \vee Z$

$S_Q = 2 + 3 + 4 = 9$

Основные тождества булевой алгебры

1) Законы коммуникативности (переместительности):

$$x \vee y = y \vee x, x \wedge y = y \wedge x$$

2) Законы ассоциативности (сочетания):

$$(x \wedge y) \wedge z = x \wedge (y \wedge z), (x \vee y) \vee z = x \vee (y \vee z)$$

3) Законы дистрибутивности (распределения):

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z), x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$$

4) Законы тавтологии (идемпотентности):

$$x \vee x = x, x \wedge x = x$$

5) Закон двойственности (де Моргана):

$$\overline{x \wedge y} = \overline{x} \vee \overline{y}, \overline{x \vee y} = \overline{x} \wedge \overline{y}$$

Основные тождества булевой алгебры (2)

6) Законы поглощения:

$$x \wedge (x \vee y) = x, x \vee (x \wedge y) = x$$

7) Закон двойного отрицания (инвалютивности):

$$\overline{\overline{x}} = x$$

8) Закон противоречия (существования 0):

$$\overline{x} \wedge x = 0$$

9) Закон «третьего не дано» (исключённого третьего, существования 1):

$$\overline{x} \vee x = 1$$

10) Закон универсального множества (нейтральности):

$$x \wedge (y \vee \overline{y}) = x, x \vee (y \vee \overline{y}) = 1$$

11) Закон пустого множества:

$$x \vee 0 = x, x \wedge 0 = 0$$

Формы записи математических выражений

«Арность операции» = «количество операндов» = «вместимость»

- √A (унарная)
- A*B (бинарная)
- A?B:C (тернарная)

Виды нотаций:

- 1489: инфиксная: A+B
- 1920: префиксная (*aka* польская): +AB
- 1957: постфиксная (*aka* обратная польская): AB+

Префиксная (польская) нотация

Пример. Инфиксная нотация: **(A + B + C) – $E^{D * F * G}$**

1. – (A + B + C) ($E^{D * F * G}$).
2. – (+ (+AB) C) ($E^{D * F * G}$).
3. – (+ (+AB) C) (^ E (D * F * G)).
4. – (+ (+AB) C) (^ E (* (*DF) G)).

Префиксная нотация: **–++ABC^E**DFG**

Популярная Lisp-разновидность префиксной нотации: (–(+ABC)(^E(*DFG)))

Особенности:

1. Не требуется скобок, если арность фиксирована.
2. Запись выражения получается короче, чем инфиксная.
3. Не требуется знать приоритет операций.
4. Легко декодировать выражение с помощью стека.
5. Малоприменима на практике (кроме Lisp).

Постфиксная (обратная польская) нотация

Пример. Инфиксная нотация: **$(A + B + C) - E^{D * F * G}$**

1. $(A + B + C)\ (E^{D * F * G})\ -$.
2. $((AB+)\ C +)\ (E^{D * F * G})\ -$.
3. $((AB+)\ C +)\ (E(D * F * G)^{\wedge})\ -$.
4. $((AB+)\ C +)\ (E((DF *)\ G *)^{\wedge})\ -$.

Постфиксная нотация: **AB+C+EDF*G*^–**

Особенности:

1. Не требуется скобок, если арность фиксирована.
2. Запись выражения получается короче, чем инфиксная.
3. Не требуется знать приоритет операций.
4. Легко декодировать выражение с помощью стека.
5. Успешно применяется в компиляторах, в небольшом количестве языков программирования (Forth) и некоторых ЭВМ (калькуляторы «Электроника» и HP).
6. Используется для хранения байт-кода в программном комплексе 1С:Предприятие.

Алгоритм вычисления выражения, записанного в постфиксной нотации

1. Обработка входного символа:

- Если на вход подан операнд, он помещается на вершину стека.
- Если на вход подан знак операции, то соответствующая операция выполняется над требуемым количеством значений, извлечённых из стека, взятых в порядке добавления. Результат выполненной операции кладётся на вершину стека.

2. Если входной набор символов обработан не полностью, перейти к шагу 1.

3. После полной обработки входного набора символов результат вычисления выражения лежит на вершине стека.

Декодирование постфиксной нотации

Пример. a + b + a*c = ab+ac*+

push a	push b	add	push a
a	b	a+b	a
	a		a+b

push c	mul	add	pop a
c	a*c	a+b+a*c	
a	a+b		
a+b			