Профессор
Игорь Н. Бекман

КОМПЬЮТЕРНЫЕ НАУКИ

Курс лекций

Лекция 6. ЛОГИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРОВ

Содержание

1. АЛГЕБРА ЛОГИКИ 1
1.1 Историческая справка 1
1.2 Понятия формальной логики 2
1.3 Логические операции и таблицы истинности 4
1.3.1 Логическое выражение 5
1.3.2 Элементарные булевы функции 6
1.3.3 Логическое отрицание (инверсия) 7
1.3.4 Логическое умножение (конъюнкция) 8
1.3.5 Сложение по модулю "2" 9
1.3.6 Логическое сложение (дизъюнкция) 9
1.3.7 Стрелка Пирса 10
1.3.8 Логическое следование (импликация) 10
1.3.9 Логическое тождество (эквиваленция). 12
1.3.10 Штрих Шеффера 13
1.3.11 Таблицы истинности 13
1. 4 Логические формулы. Законы алгебры логики 15
1.4.1 Основные законы булевой алгебры 17
1.4.2 Преобразование выражений, состоящих из булевых функций. 18
1.4.3 Алгебры булевых функций 19
1.4.4 Функция сложения по модулю 2 (xor) 20
2. АЛГЕБРА ЛОГИКИ В ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКЕ 20

Алгебра логики (алгебра высказываний) – раздел математической логики, изучающий строение (форму, структуру) сложных логических высказываний и способы установления их истинности с помощью алгебраических методов. При этом под высказыванием (суждением) понимают повествовательное предложение, относительно которого можно сказать, истинно или ложно. Логика в информатике - это направления исследований и отрасли знания, где логика применяется в информатике и искусственном интеллекте.

В данной лекции мы рассмотрим основные аспекты алгебры логики, включая историю её развития, понятия, виды логических операций и таблиц истинности, логические формулы, а также законы алгебры логики. Заключительная часть лекции посвящена использованию алгебры логики в компьютерных науках.

1. АЛГЕБРА ЛОГИКИ

1.1 Историческая справка

Алгебра логики возникла в середине XIX века в трудах английского математика **Джорджа Буля**. Её создание представляло собой попытку решать традиционные логические задачи алгебраическими методами.

Отцом алгебры логики по праву считается английский математик 19-го столетия Джорж Буль (1815-1864). Именно он построил один из разделов формальной логики в виде некоторой «алгебры», аналогичной алгебре чисел, но не сводящейся к ней. Алгебра в широком смысле слова – наука об общих операциях, аналогичных сложению и умножению, которые могут выполняться не только с числами, но и над другими математическими объектами. Существуют алгебры натуральных чисел, многочленов, векторов, матриц, множеств и т.д.

Большой вклад в становление и развитие алгебры логики внесли Августус де Морган (1806-1871), Уильям Стенли Джевонс (1835-1882), П.С. Порецкий (1846 – 1907), Чарлз Сандерс Пирс (1839-1914), А.А. Марков (1903-1979), А.Н. Колмогоров (1903-1987) и др.

Долгое время алгебра логики была известна достаточно узкому классу специалистов. Прошло почти 100 лет со времени создания алгебры логики Дж. Булем, прежде чем в 1938 Клод Шеннон (1916-2001) показал, что алгебра логики применима для описания самых разнообразных процессов, в том числе функционирования релейно-контактных и электронно-ламповых схем.

Алгебра логики явилась математической основой теории электрических и электронных переключателей схем, используемых в ЭВМ. В компьютерных науках её предпочитают называть не алгеброй логики, а Булевой алгеброй - по имени её создателя.

Алгебра логики изучает свойства функций, у которых и аргументы, и значения принадлежат заданному двухэлементному множеству (например, {0,1}). Иногда вместо термина «алгебра логики» употребляют термин «двузначная логика».

Алгебра логики - предельно важная для цифровых компьютеров тема. И с точки зрения их устройства, схемотехники, и с точки зрения их функционирования и программирования поведения. Действительно, мало-мальски сложное действие невозможно без обратной связи, без анализа условий выполнения. Например, «ЕСЛИ нам хочется пить, ТО мы пьём, ИНАЧЕ мы даже не думаем об этом». «ЕСЛИ компьютер не работает И питание включено, ТО компьютер сгорел». «ЕСЛИ точка левее левой стороны квадрата ИЛИ правее правой, ТО точка расположена не в квадрате». «*Ревёт ли зверь в лесу глухом, трубит ли рог, гремит ли гром...*». «*Кошелёк или жизнь*». Помимо манипуляций константами «да» и «нет» логические переменные могут являться результатом применения к числам операторов отношения (меньше, больше, равно и т.п.).

В компьютерах булевы переменные представляются (кодируются) битами (разрядами двоичной системы счисления), где 1 означает истину, а 0 - ложь. Манипуляции высказываниями и их комбинациями используются для получения некоего единственного результата, который можно использовать, например, для выбора той или иной последовательности действий. Поскольку логические переменные кодируются по тем же принципам, что и числа, символы и прочая информация, то можно комбинировать операции логики с операциями арифметики для реализации различных алгоритмов.

Таким образом, алгебра логики (другое название - Булева алгебра) - это область математики. Она оперирует величинами, которые могут принимать два значения (булевых значения). Эти два значения могут быть обозначены как угодно, лишь бы по-разному. Самые распространенные варианты:

0, 1
F, T
false, true
ложь, истина
Л, И

При применении булевой алгебры в вычислительной технике, булевы значения - это 0 и 1. Они представляют собой состояние ячейки памяти объёмом в 1 бит или наличие/отсутствие напряжения в электрической схеме. Алгебра логики позволяет строить сложные электронные узлы, элементы которых работают согласно этой математической теории. При применении булевой алгебры в логических построениях в математике, булевы значения - это «ложь» и «истина». Они определяют истинность или ложность некоторого высказывания. Под высказываниями понимаются математические формулы. При применении булевой алгебры в повседневных рассуждениях, булевы значения - это также «ложь» и «истина». Они представляют собой оценку истинности или ложности некоторого высказывания. Под высказываниями понимаются фразы, которые удовлетворяют строго определенному списку свойств.

Алгебра логики применяется: 1) для упрощения сложных логических формул и доказательств тождеств; 2) при решении логических задач; 3) в контактных схемах; 4) при доказательствах теорем; 5) в базах данных при составлении запросов.

1.2 Понятия формальной логики

Логика – наука о законах и формах мышления

Логика - наука, изучающая способы обоснования суждений, доказательства, мышления и логического вывода. В математической логике используются для этого методы алгебры или теории алгоритмов.

Алгебра логики (булева алгебра) - раздел математики, изучающий методы оперирования логическими (булевыми) переменными, принимающими только два значения - истина и ложь.

Алгебра логики - раздел математической логики, в котором изучаются логические операции над высказываниями. Высказывания могут быть истинными, ложными или содержащими истину и ложь в разных соотношениях.
Математическая логика (теоретическая логика, символическая логика) - раздел математики, изучающий доказательства и вопросы оснований математики.
Логическое высказывание - утверждение, которому всегда можно поставить в соответствие одно из двух логических значений ложь (0, ложно, *false*) или истина (1, истинно, *true*). Логическое высказывание принято обозначать заглавными латинскими буквами. **Высказывательной формой** называется логическое высказывание, в котором один из объектов заменён переменной. При подстановке вместо переменной какого-либо значения высказывательная форма превращается в высказывание.
Пример: *A(x)* = «В городе *x* идет дождь» *A* - высказывательная форма, *x* - объект.
Отрицание логического высказывания - логическое высказывание, принимающее значение «истинно», если исходное высказывание ложно, и наоборот.
Конъюнкция двух логических высказываний - логическое высказывание, истинное только тогда, когда они одновременно истинны.
Дизъюнкция двух логических высказываний - логическое высказывание, истинное только тогда, когда хотя бы одно из них истинно.
Импликация двух логических высказываний *A* и *B* - логическое высказывание, ложное только тогда, когда *B* ложно, а *A* истинно.
Равносильность (эквивалентность) двух логических высказываний - логическое высказывание, истинное только тогда, когда они одновременно истинны или ложны.

Кванторное логическое высказывание с квантором **всеобщности** ($\forall x A(x)$) - логическое высказывание, истинное только тогда, когда для каждого объекта *x* из заданной совокупности высказывание *A(x)* истинно.

Кванторное логическое высказывание с квантором **существования** ($\exists x A(x)$) - логическое высказывание, истинное только тогда, когда в заданной совокупности существует объект *x*, такой, что высказывание *A(x)* истинно.
Высказывание (суждение) – некоторое предложение, которое может быть истинно (верно) или ложно
Утверждение – суждение, которое требуется доказать или опровергнуть
Рассуждение – цепочка высказываний или утверждений, определенным образом связанных друг с другом
Умозаключение – логическая операция, в результате которой из одного или нескольких данных суждений получается (выводится) новое суждение
Логическое выражение – запись или устное утверждение, в которое, наряду с постоянными, обязательно входят переменные величины (объекты). В зависимости от значений этих переменных логическое выражение может принимать одно из двух возможных значений: ИСТИНА (логическая 1) или ЛОЖЬ (логический 0)
Сложное логическое выражение – логическое выражение, составленное из одного или нескольких простых (или сложных) логических выражений, связанных с помощью логических операций.

Слово **логика** означает совокупность правил, которым подчиняется процесс мышления. Сам термин «логика» происходит от древнегреческого *logos*, означающего «слово, мысль, понятие, рассуждение, закон». *Формальная логика* - наука о формах и законах мышления. Законы логики отражают в сознании человека свойства, связи и отношения объектов окружающего мира. Логика как наука позволяет строить формальные модели окружающего мира, отвлекаясь от содержательной стороны. Основными формами мышления являются *понятия, суждения* и *умозаключения*.
Понятие - форма мышления, которая выделяет существенные признаки предмета или класса предметов, отличающие его от других. Например, компьютер, человек, ученики.
Суждения - это форма мышления, в которой утверждается или отрицается связь между предметом и его признаком, отношения между предметами или факт существования предмета и которая может быть либо истинной, либо ложной. Языковой формой выражения суждения является повествовательное предложение. Вопросительные и побудительные предложения суждениями не являются. Суждения рассматриваются не с точки зрения их смысла и содержания, а только с точки зрения их истинности или ложности. Истинным будет суждение, в котором связь понятий правильно отражает свойства и отношения реальных объектов. «Дважды два равно четырём» - истинное суждение, а вот «Процессор предназначен для печати» - ложное. Суждения могут быть простыми и сложными. «Весна наступила, и грачи прилетели» - сложное суждение, состоящее из двух простых. Простые суждения (высказывания) выражают связь двух понятий. Сложные - состоят из нескольких простых суждений.
Умозаключение - приём мышления, позволяющий на основе одного или нескольких суждений-посылок получить новое суждение (знание или вывод). Примерами умозаключений являются доказательства теорем в геометрии. Посылками умозаключения по правилам формальной логики могут быть только истинные суждения. Тогда и умозаключение будет истинным. Иначе можно прийти к ложному умозаключению.

Исследования в алгебре логики тесно связаны с изучением высказывания (хотя высказывание – предмет изучения формальной логики). С помощью высказывания мы устанавливаем свойства, взаимосвязи между объектами. Высказывание истинно, если оно адекватно отображает эту связь, в противном случае оно ложно.

Математическая логика изучает вопросы применения математических методов для решения логических задач и построения логических схем, которые лежат в основе работы любого компьютера. Суждения в математической логике называют *высказываниями* или *логическими выражениями*. Подобно тому, как для описания действий над переменными был разработан раздел математики алгебра, так и для обработки логических выражений в математической логике была создана *алгебра высказываний*, или *алгебра логики*.

Таким образом, алгебра логики - раздел математической логики, в котором изучаются логические операции над высказываниями. Высказывания могут быть истинными и ложными.

Логика высказываний послужила основным математическим инструментом при создании компьютеров. Она легко преобразуется в битовую логику: истинность высказывания обозначается одним битом (0 - ЛОЖЬ, 1 - ИСТИНА); тогда операция $\neg$ приобретает смысл вычитания из единицы; $\vee$ - немодульного сложения; & (или $\wedge$) - умножения; $\leftrightarrow$ - равенства; $\oplus$ - в буквальном смысле сложения по модулю 2 (исключающее Или - *XOR*); $|$ - непревосходства суммы над 1 (то есть $A|B = (A + B) <= 1$).

Впоследствии булева алгебра была обобщена от логики высказываний путём введения характерных для логики высказываний аксиом. Это позволило рассматривать, например, логику кубитов, тройственную логику (когда есть три варианта истинности высказывания: «истина», «ложь» и «не определено») и др.

1.3 Логические операции и таблицы истинности

Булевы величины (или булевы константы) - два заранее выбранных разных символа.

По традиции применяются символы 0 и 1. Так будем поступать и мы. Но надо понимать, что формулы булевой алгебры будут работать независимо от того, как обозначить булевы величины и какой смысл им придать. Например, в электронике это может быть наличие или отсутствие потенциала в +5 вольт в определенной точке схемы, при доказательстве математической теоремы - суждения «истинно» и «ложно», а в экспертной системе - ответы «да» и «нет».

Булевы переменные - переменные, которые могут принимать булевы значения.

Для того, чтобы некоторую величину можно было обозначать булевой переменной, должны выполняться следующие ограничения: 1) Величина должна принимать два возможных состояния, но не более того. 2) В любой момент времени величина не может принимать оба состояния одновременно. 3) В любой момент времени величина не может принимать ни одного состояния. 4) Если рассматриваются несколько таких величин, то допускается, чтобы каждая из них принимала одно из двух состояний независимо. 5) Не допускается применять одну пару состояний для одной величины, а для другой - другую.

Эти 5 правил определяют ситуации, в которых алгебра логики может быть применена.

Пример 1. Пусть речь идет о цехе автомобильного завода, где сушатся только что покрашенные автомобили и мы хотим применить алгебру логики, рассуждая об автомобилях в этом цеху. Мы можем применить алгебру логики к цвету автомобилей, если все они либо зелёные, либо красные. По правилу 1, если в цехе есть помимо красных и зелёных еще и жёлтые автомобили, то мы не можем применить алгебру логики, деля их на красные и зелёные. Потому, что кроме двух значений красный и зелёный" появляется третье: жёлтый. Можно выйти из затруднения, рассмотрев автомобили зелёные и незелёные (т. е. всех остальных цветов). По правилу 2, если в цехе есть красные автомобили в зелёную полоску или зелёные в красный горошек, то мы не можем применить алгебру логики, деля их на красные и зелёные. Потому, что о некоторых автомобилях можно будет сказать, что он и красный, и зелёный одновременно. Можно выйти из затруднения, если договориться считать красными автомобили, которые сначала покрывают красной краской. По правилу 3, если в цехе есть помимо красных или зеленых вовсе некрашеные автомобили, то мы не можем применить алгебру логики деля их на красные и зелёные. Потому, что о некрашеных автомобилях еще нельзя сказать, что они красные или зеленые. Можно выйти из этого затруднения, если считать красными те автомобили, которые запланировано покрасить в красный цвет, а зелёными те, которые запланировано покрасить в зелёный. По правилу 4, если в цехе не все автомобили одновременно зелёные или красные, это не помешает применению алгебры логики к цвету автомобилей. С другой стороны, если в цехе всегда только красные автомобили или только зелёные, то нет смысла заводить столько переменных, сколько автомобилей. Достаточно одной переменной для всех автомобилей сразу. По правилу 5, если в цехе есть только зелёные и красные автомобили, и все они - либо сломаны, либо исправны, то мы все равно не можем смешивать в одних и тех же формулах переменные, обозначающие цвет автомобилей, и их исправность. Из этого затруднения можно выйти, если рассматривать не цвет и исправность самих автомобилей, а истинность или ложность

правильно составленных фраз насчет цвета и исправности. Итак, каждому автомобилю будет соответствовать две булевы переменные: автомобиль зелёный и автомобиль исправный. Каждая переменная может принимать два значения истина или ложь.

Правила 1, 2, 3 и 5 должны обязательно выполняться все. Если не выполняется хотя бы одно из них, алгебра логики не применима. Правило 4 поясняет одну ситуацию, когда могут возникнуть сомнения насчет применения алгебры логики. Примеры демонстрируют следующие практические факты: 1) алгебра логики может быть применена не всегда, а лишь с соблюдением определенных ограничений. 2) если алгебра логики не может быть применена одним способом, то часто можно обнаружить другой способ - совсем рядом. Достаточно немного изменить условия. В этом нет ничего необычного или сомнительного - так работает вся математика. Нужно просто проявлять с одной стороны внимательность, а с другой - изобретательность. Например, в арифметике нельзя суммировать гайки и яблоки по количеству, но зато их можно суммировать по весу. Эта ситуация очень похожа на ту, когда нельзя применить булеву алгебру к цвету и исправности, но можно применить к фразам о цвете и исправности.

Булева алгебра весьма распространена. Принцип работы большинства компьютеров основан на ней. Большинство формул математики могут быть только истинными или ложными, так что булева алгебра применима и почти ко всей математике. Оказывается, что и в обыденной речи алгебра логики вполне применима, хотя не везде и не всегда. Таким образом, булева алгебра полезна, но не претендует на сверхуниверсальность. Это - инструмент, который может оказаться удобен для решения определенных задач, для других - неудобен, а для третьих - вообще неприменим.

1.3.1 Логическое выражение

Логическое выражение - это символическая запись, состоящая из логических величин (констант или переменных), объединенных логическими операциями (связками). В булевой алгебре простым высказываниям ставятся в соответствие **логические переменные**, значение которых равно 1, если высказывание истинно, и 0, если высказывание ложно. Обозначаются логические переменные буквами латинского алфавита. Существуют разные варианты обозначения истинности и ложности переменных:

Истина	И	*True*	*T*	1
Ложь	Л	*False*	*F*	0

Связки «НЕ», «И», «ИЛИ» заменяются логическими операциями *инверсия, конъюнкция, дизъюнкция*. Это основные логические операции, при помощи которых можно записать любое логическое выражение.

Как уже упоминалось, **алгебра логики** - раздел математики, изучающий высказывания, рассматриваемые со стороны их логических значений (истинности или ложности) и логических операций над ними. Логическое высказывание - любое повествовательное предложение, в отношении которого можно однозначно сказать, истинно оно или ложно.

Алгебра логики изучает строение (форму, структуру) сложных логических высказываний и способы установления их истинности с помощью алгебраических методов.

Так, например, предложение «*6 - четное число*» следует считать высказыванием, так как оно истинное. Предложение «*Рим - столица Франции*» тоже высказывание, так как оно ложное.

Разумеется, **не всякое предложение является логическим высказыванием**. Высказываниями не являются, например, предложения «*ученик десятого класса*» и «*информатика - интересный предмет*». Первое предложение ничего не утверждает об ученике, а второе использует слишком неопределённое понятие «*интересный предмет*». Вопросительные и восклицательные предложения также не являются высказываниями, поскольку говорить об их истинности или ложности не имеет смысла. Предложения типа «*в городе А более миллиона жителей*», «*у него голубые глаза*» не являются высказываниями, так как для выяснения их истинности или ложности нужны дополнительные сведения: о каком конкретно городе или человеке идет речь. Такие предложения называются *высказывательными формами*.

Высказывательная форма - повествовательное предложение, которое прямо или косвенно содержит хотя бы одну переменную и становится высказыванием, когда все переменные замещаются своими значениями.

Алгебра логики рассматривает любое высказывание только с одной точки зрения - является ли оно истинным или ложным. Заметим, что **зачастую трудно установить истинность высказывания**. Так, например, высказывание «*площадь поверхности Индийского океана равна 75 млн кв. км*» в одной ситуации можно посчитать ложным, а в другой - истинным. Ложным - так как указанное значение неточное и вообще не является постоянным. Истинным - если рассматривать его как некоторое приближение, приемлемое на практике.

Употребляемые в обычной речи слова и словосочетания «**не», «и», «или», «если... , то», «тогда и только тогда»** и другие позволяют из уже заданных высказываний строить новые высказывания. Такие слова и словосочетания называются **логическими связками.** Высказывания, образованные из других высказываний с помощью логических связок, называются **составными.** Высказывания, не являющиеся составными, называются **элементарными.** Так, например, из элементарных высказываний «*Петров – врач*», «*Петров - шахматист»* при помощи связки «*и»* можно получить составное высказывание «*Петров - врач и шахматист»*, понимаемое как «*Петров - врач, хорошо играющий в шахматы»*. При помощи связки «*или»* из этих же высказываний можно получить составное высказывание «*Петров — врач или шахматист»*, понимаемое в алгебре логики как «*Петров или врач, или шахматист, или и врач и шахматист одновременно»*.

Истинность или ложность получаемых таким образом составных высказываний зависит от истинности или ложности элементарных высказываний.

Чтобы обращаться к логическим высказываниям, им назначают имена. Пусть через ***A*** обозначено высказывание «*Тимур поедет летом на море»,* а через ***B*** - высказывание «*Тимур летом отправится в горы».* Тогда составное высказывание «*Тимур летом побывает и на море, и в горах»* можно кратко записать как ***A и B***. Здесь «**и»** - логическая связка, ***A, B*** - логические переменные, которые могут принимать только два значения – «истина» или «ложь», обозначаемые, соответственно, «1» и «0».

Каждая логическая связка рассматривается как операция над логическими высказываниями и имеет свое название и обозначение.

1.3.2 Элементарные булевы функции

Двоичной, булевой функцией от набора двоичных переменных называется функция, результатом которой могут быть только значения 0 и 1. Любую булеву функцию можно задать с помощью таблицы, в которой всем возможным наборам значений двоичных переменных сопоставлены соответствующие им значения функции. Такая таблица называется таблицей истинности, поскольку она определяет истинность или ложность сложного высказывания в зависимости от истинности или ложности составляющих высказываний.

Для функций одной переменной может существовать всего четыре различные булевы функции g_1, g_2, g_3 и g_4, представленные в следующей таблице:

x	g_1	g_2	g_3	g_4
0	0	0	1	1
1	0	1	0	1

Из таблицы следует, что функции g_1 и g_4 не зависят от аргумента и являются соответственно константами 0 и 1, а функция g_2 повторяет значение аргумента, т.е. $g_2=x$. Функция g_3 называется отрицанием или инверсией переменной x и обозначается как *not(x)*.

Для функций двух переменных может существовать 16 (и только 16) различных функций. Таблица истинности этих функций следующая:

x_1	x_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	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

В число этих функций входят 6 вырожденных функций (константы: $F_0=0$ и $F_{15}=1$; переменные: $F_3=x_1$ и $F_5=x_2$; инверсии: $F_{12}=not\ x_1$ и $F_{10}=not\ x_2$). Остальные функции с их обозначениями приведены ниже:

Функция	Название	Выражение через конъюнкцию, дизъюнкцию и отрицание	Читается как
F_1	конъюнкция	$x_1\ and\ x_2$	x_1 и x_2
F_7	дизъюнкция	$x_1\ or\ x_2$	x_1 или x_2
F_6	сложение по модулю 2	$(x_1\ and\ not\ x_2)\ or\ (not\ x_1\ and\ x_2)$	x_1 неравнозначно x_2
F_8	функция Пирса	$not\ x_1\ and\ not\ x_2$	ни x_1, ни x_2
F_9	эквивалентность	$(not\ x_1\ and\ not\ x_2)\ or\ (x_1\ and\ x_2)$	x_1 равнозначно x_2
F_{11}	импликация	$x_1\ or\ not\ x_2$	если x_2, то x_1
F_{14}	штрих Шеффера	$not\ x_1\ or\ not\ x_2$	неверно, что x_1 и x_2

F_2	запрет по x_2	x_1 *and not* x_2	неверно, что если x_1, то x_2
F_4	запрет по x_1	*not* x_1 *and* x_2	неверно, что если x_2, то x_1
F_{13}	импликация	*not* x_1 *or* x_2	если x_1, то x_2 (x_1 -> x_2)

Обратите внимание! Несмотря на экзотические названия, это просто набор всех возможных функций (функция - это однозначное отображение, преобразование набора аргументов из области определения в значение из области значений). Натуральный ряд бесконечен и для натуральных переменных возможно бесконечное множество функций, включающих функции сложения, вычитания, умножения и т.п. В логике переменные имеют всего два возможных значения, поэтому количество различных функций ограничено и здесь они перечислены все.

Для записи штриха Шеффера «|» в выражениях обычно используется апостроф ($x_1'x_2$), а для сложения по модулю 2 - слово *xor* (от *eXslusive OR*, "исключающее или", читается «ксор»). В ассемблерах конъюнкция, дизъюнкция и отрицание обычно записываются соответствующими английскими словами (*and, or, not*). В языках же высокого уровня эти функции могут обозначаться как словами (Паскаль), так и специальными знаками (в С использованы знаки «&», «|» и «~»). Иногда по аналогии с теорией множеств конъюнкция называется пересечением, а по аналогии с арифметикой в формулах используется знак умножения «*». Для дизъюнкции же используется знак сложения «+». И действительно - если вспомнить, как ведёт себя нуль в операциях умножения и сложения, а потом посмотреть на таблицы истинности конъюнкции и дизъюнкции, то можно найти много общего.

В языках программирования приоритет конъюнкции обычно выше, чем приоритет дизъюнкции, и в выражении (x_1 *or* x_2 *and* x_3) подразумевается, что (x_2 *and* x_3) будет выполнено первым.

1.3.3 Логическое отрицание (инверсия)

В обыденной речи мы часто пользуемся словом «НЕ», или словами «НЕВЕРНО, ЧТО», когда хотим что-то отрицать. Пусть, например, кто-то сказал: «*Тоска зелёная*» (Обозначим это высказывание ***A***). Если Вы не согласны, Вы скажете: «*Тоска НЕ зелёная*». Или: «Неверно, что тоска зеленая». (Ваше высказывание обозначим ***B***). Нетрудно заметить, что значения истинности высказываний ***A*** и ***B*** находятся в определенной связи: если ***A*** истинно, то ***B*** ложно, и наоборот. Операция, с помощью которой из высказывания ***A*** получается высказывание ***B***, называется логическим отрицанием и само высказывание ***B*** называется отрицанием высказывания **A** и обозначается ¬ ***A***. Таким образом, отрицанием ¬ A некоторого высказывания ***A*** называется такое высказывание, которое истинно, когда ***A*** ложно, и ложно, когда ***A*** истинно. Отрицание высказывания ***A*** обозначим ¬***A***. Определение отрицания может быть записано с помощью так называемой таблицы истинности:

A	¬ ***A***
И	Л
Л	И

В ней указано, какие значения истинности (Истина, Ложь) принимает отрицание ¬***A*** в зависимости от значений истинности исходного высказывания A.

Функция НЕ

A	ncA
0	1
1	0

Операция, выражаемая словом «*не*», называется **отрицанием** и обозначается чертой над высказыванием (или знаком ¬). Высказывание $\overline{A}$ истинно, когда ***A*** ложно, и ложно, когда ***A*** истинно. Пример. «Луна - спутник Земли» (***A***); «Луна - не спутник Земли» ($\overline{A}$).

Логическое отрицание: ИНВЕРСИЯ - если исходное выражение истинно, то результат отрицания будет ложным, и наоборот, если исходное выражение ложно, то результат отрицания будет истинным. Данная операция означает, что к исходному логическому выражению добавляется частица **НЕ** или слова **НЕВЕРНО, ЧТО**

Отрицание – логическая операция, которая каждому элементарному высказыванию ставит в соответствие новое высказывание, значение которого противоположно исходному.

Рассмотренная операция - одноместна (унитарна), далее мы рассмотрим двуместные (бинарные) операции.

В русском языке для построения отрицания используется связка «неверно, что». Хотя связка «неверно, что» и не связывает двух каких-либо высказываний в одно, она трактуется логиками как логическая связка, поскольку, поставленная перед произвольным высказыванием, образует нечто новое.

Пример 2. Отрицанием высказывания *У меня дома есть компьютер* будет высказывание Неверно, что у меня есть компьютер или, что в русском языке то же самое, *У меня дома нет компьютера.*

Пример 3. Отрицанием высказывания *Я не знаю корейского языка* будет высказывание *Неверно, что я не знаю корейского языка* или *Я знаю корейский язык*.
Пример 4. Отрицанием высказывания *Все юноши 11-х классов – отличники* является высказывание *Неверно, что все юноши 11-х классов – отличники* или *Не все юноши 11-х классов – отличники* или другими словами, *Некоторые юноши 11-х классов не отличники.*

На первый взгляд кажется, что построить отрицание к заданному высказыванию достаточно просто. Однако это не так.
Пример 5. Высказывание *Все юноши 11-х классов – не отличники* не является отрицанием высказывания *Все юноши 11-х классов – отличники*. Объясняется это следующим образом. Высказывание *Все юноши 11-х классов* – не отличники ложно. Отрицанием к ложному высказыванию должно быть высказывание, являющееся истинным. Но высказывание *Все юноши 11-х классов – не отличники* не является истинным, т.к. среди одиннадцатиклассников есть как так и не отличники.
Пример 6. Для высказывания На стоянке стоят красные Жигули следующие предложения отрицаниями не будут: 1) На стоянке стоят не красные Жигули; 2) На стоянке стоит белый Мерседес; 3) Красные Жигули стоят не на стоянке.

Проанализировав приведённые примеры, можно вывести полезное правило построения отрицания к простому высказыванию.
При построении отрицания к простому высказыванию используется простой речевой оборот «неверно, что», либо отрицание строится к сказуемому, тогда к сказуемому добавляется частица «не», при этом слово «все» заменяется на «некоторые» и наоборот.

1.3.4 Логическое умножение (конъюнкция)

Название логическое умножение (конъюнкция) происходит от латинского *conjunctio* - союз, связь.

Высказывание, составленное из двух высказываний путём объединения их связкой «и», называется **конъюнкцией** или **логическим умножением**. Высказывая конъюнкцию, мы утверждаем, что выполняются оба события, о которых идёт речь в составляющих высказываниях. Например, сообщая: *Ивановы привезли на зиму уголь и закупили дрова на растопку камина*, мы выражаем в своём высказывании своё убеждение в том, что произошли оба этих события.

Если два высказывания соединены союзом «И», то полученное сложное высказывание обычно считается истинным тогда и только тогда, когда истинны оба составляющие его высказывания. Если хотя бы одно из составляющих высказываний ложно, то и полученное из них с помощью союза «И» сложное высказывание также считается ложным. Например, возьмем два высказывания: «*У кота есть хвост*» (***A***), «*У зайца есть хвост*» (***B***). Сложное высказывание «*У кота есть хвост и у зайца есть хвост*» истинно, т.к. истинны оба высказывания ***A*** и ***B***. Но если взять другие высказывания: «*У кота длинный хвост*» (***C***), «*У зайца длинный хвост*» (***D***), то сложное высказывание «*У кота длинный хвост и у зайца длинный хвост*» будет ложным, т.к. ложно высказывание (***D***). Таким образом, исходя из обычного смысла союза «И», приходим к определению соответствующей логической операции - конъюнкции.

Таким образом, конъюнкцией двух высказываний ***A*** и ***B*** называется такое высказывание, которое истинно тогда и только тогда, когда истинны оба высказывания ***A*** и ***B***. Конъюнкцию высказываний ***A*** и ***B*** мы обозначим: ***A&B***. Знак & - амперсент - читается как английское «*and*» (помните *Procter & Gamble* или *Wash & Go*?). Часто встречается обозначение ***A∧B***. Иногда, для краткости, пишут просто ***AB***. Определение конъюнкции может быть записано в виде таблицы истинности, в которой для каждого из четырех возможных наборов значений исходных высказываний ***A*** и ***B*** задаётся соответствующее значение конъюнкции ***A & B***:

A	B	A&B
и	и	и
и	л	л
л	и	л
л	л	л

Определение конъюнкции двух высказываний естественным образом распространяется на любое конечное число составляющих: конъюнкция A_1 & A_2 & A_3 &...& A_N истинна тогда и только тогда, когда истинны все высказывания A_1, A_2, A_3, ...A_N (а, следовательно, ложна, когда ложно хотя бы одно из этих высказываний).
Функция И
Другие названия этой функции: «И», «логическое И», «логическое умножение», «булево умножение». Функция «И» дает 1 только когда оба операнда равны 1. Эта функция называется иногда логическим умножением, поскольку ее результаты совпадают с аналогичной операцией в арифметике: 0*0=0, 0*1=0, 1*0=0, 1*1=1.

Операция, выражаемая связкой «*и*», называется **конъюнкцией** (*conjunctio* - соединение) или логическим **умножением** и обозначается точкой «·» (может также обозначаться знаками ∧, **&, and,** ∩**, И**). Высказывание **A**·**B** истинно тогда и только тогда, когда оба высказывания **A** и **B** истинны. Например, высказывание «*10 делится на 2 и 5 больше 3*» истинно, а высказывания «*10 делится на 2 и 5 не больше 3», «10 не делится на 2 и 5 больше 3», «10 не делится на 2 и 5 не больше 3*» - ложны.

Пример 7. Рассмотрим два высказывания *A = Завтра будет мороз* и *B= Завтра будет идти снег*. Очевидно, новое высказывание ***A&B*** *= Завтра будет мороз, и завтра будет идти снег* истинно только в том случае, когда одновременно истинны высказывания ***A*** и ***B***, а именно, когда истинно, что завтра будет мороз и снег. Высказывание ***A&B*** будет ложно во всех остальных случаях: будет идти снег, но будет оттепель (т.е. не будет мороза); мороз будет, а снег не будет идти; не будет ни мороза, ни снега.

A	B	F
1	1	1
1	0	0
0	1	0
0	0	0

F = A&B.

Логическое умножение КОНЪЮНКЦИЯ - это новое сложное выражение будет истинным только тогда, когда истинны оба исходных простых выражения. Конъюнкция определяет соединение двух логических выражений с помощью союза **И**.

В русском языке конъюнкции соответствует не только союз «и», но и другие речевые обороты, например, связки «а» или «но».

1.3.5 Сложение по модулю «2»

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Другие названия этой функции: «исключающее ИЛИ», «логическое ЛИБО», «неравносильность», «неэквивалентность», «логическое сложение», «булево сложение». Обозначения этой функции: ⊕., $+_2$, +, xor, ЛИБО.

Функция ⊕ даёт 1 только когда первый операнд не равен второму операнду. Она похожа на арифметическое сложение: 0 ⊕0 = 0, 0 ⊕1 = 1, 1⊕0 = 1, но не всегда: в булевой алгебре 1 ⊕1 = 0, а не 2, как в арифметике - булевы величины 1 и 0 не являются числами и для них арифметика неприменима.

1.3.6 Логическое сложение (дизъюнкция)

Логическое сложение (дизъюнкция) происходит от латинского *disjunctio* - разобщение, различие.

Высказывание, состоящее из двух высказываний, объединённых связкой «или», называется **дизъюнкцией** или **логическим сложением**, **нестрогой дизъюнкцией, логическим или**. В высказываниях, содержащих связку «или», указывается на существование двух возможных событий, из которых хотя бы одно должно быть осуществлено.

Пример 8. Сообщая: *Петя читает книгу, или пьёт чай*, мы имеем ввиду, что хотя бы одно Петя делает. При этом Петя может одновременно читать книгу и пить чай. И в этом случае дизъюнкция будет истинна.

Если два высказывания соединены союзом «ИЛИ», то полученное сложное высказывание обычно считается истинным, когда истинно хотя бы одно из составляющих высказываний. **Пример 9.** Возьмем два высказывания: «*Мел чёрный*» (***A***), «*Доска чёрная*» (***B***). Высказывание «*Мел чёрный или доска чёрная*» будет истинным, т.к. одно из исходных высказываний (***B***) истинно.

Таким образом, дизъюнкцией двух высказываний называется такое новое высказывание, которое истинно тогда и только тогда, когда истинно хотя бы одно из этих высказываний.

Дизъюнкцию высказываний **A** и **B** мы обозначим символом **A**∨**B** и будем читать: **A** или **B**. Определение дизъюнкции может быть записано в виде таблицы истинности:

A	*B*	*A*∨*B*
И	И	И
И	Л	И
Л	И	И
Л	Л	Л

Определение дизъюнкции двух высказываний естественным образом распространяется на любое конечное число составляющих: дизъюнкция $A_1 \vee A_2 \vee A_3 \vee ... \vee A_N$ истинна тогда и только тогда, когда истинно хотя бы одно из высказываний A_1, A_2, A_3, ..., A_N (а следовательно, ложна, когда ложны все эти высказывания).

Функция ИЛИ

Операция, выражаемая связкой «*или*» (в неисключающем смысле этого слова), называется **дизъюнкцией** (*disjunctio* - разделение) или логическим **сложением** и обозначается знаком ∨, ∪, OR, ИЛИ, ||, |).

С обозначением этой функции очень много путаницы. Во-первых, в программировании для неё используется обозначение x|y, но в математике это обозначение уже занято другой логической функцией - штрих Шеффера. Во-

вторых, сплошь и рядом для этой функции применяется обозначение «+», но так же часто обозначение «+» применяется для сложения по модулю «2».

Высказывание **A∨B** ложно тогда и только тогда, когда оба высказывания A и B ложны.

Функция «ИЛИ» даёт 0 только когда оба операнда равны 0.

Дизъюнкция истинна, когда хотя бы одно из двух образующих её высказываний истинно.

Пример 10. Высказывание «*10 не делится на 2 или 5 не больше 3*» ложно, а высказывания «*10 делится на 2 или 5 больше 3*», «*10 делится на 2 или 5 не больше 3*», «*"10 не делится на 2 или 5 больше 3*» - истинны.

A	B	F
1	1	1
1	0	1
0	1	1
0	0	0

F = A+ B

Логическое сложение – ДИЗЪЮНКЦИЯ - это новое сложное выражение будет истинным тогда и только тогда, когда истинно хотя бы одно из исходных (простых) выражений. Дизъюнкция определяет соединение двух логических выражений с помощью союза **ИЛИ**

Пример . Рассмотрим два высказывания ***A***= *Колумб был в Индии* и ***B*** = *Колумб был в Египте*. Новое высказывание ***A∨B***= *Колумб был в Индии или в Египте* истинно как в случае, если он не был в Индии, но не был в Египте, так и в случае, если он не был в Индии, но был в Египте, а также в случае, если он был и в Индии, и в Египте. Но это высказывание будет ложным, если Колумб не был ни в Индии, ни в Египте.

1.3.7 Стрелка Пирса

A	B	A↓B
1	1	0
1	0	0
0	1	0
0	0	1

Другие названия этой функции: "ИЛИ-НЕ". Эта функция даёт 1 только когда оба операнда равны 0.

1.3.8 Логическое следование (импликация)

Логическое следование (импликация) происходит от латинского *implico* - тесно связываю.

ЕСЛИ-ТО Операция, выражаемая связками «*если ..., то*», «*из ... следует*», «*... влечёт ...*», называется **импликацией** (*implico* - тесно связаны) и обозначается знаком → (⊃, ⇒). Высказывание ***A→B*** ложно тогда и только тогда, когда ***A*** истинно, а ***B*** ложно. Эта функция дает 0 только когда первый операнд равен 1, а второй равен 0.

A	B	F
1	1	1
1	0	0
0	1	1
0	0	1

В наших рассуждениях, особенно в математических доказательствах, мы часто пользуемся сложными высказываниями, образованными с помощью слов «если..., то...». Здесь высказывание, расположенное после слова «если», называется основанием или посылкой, а высказывание, расположенное после слова «то», называется следствием или заключением.

Пример 11. Утверждение «если каждое слагаемое делится на 3, то и сумма делится на 3» истинно, т.е. из высказывания «каждое слагаемое делится на 3» следует высказывание «сумма делится на 3». Посмотрим, какие наборы значений истинности посылки и заключения возможны, когда истинно все утверждение. Возьмем, например, в качестве слагаемых числа 6 и 9. В этом случае истинны и посылка, и заключение, и все утверждение. Если же взять числа 4 и 5, то посылка будет ложной, а заключение истинным. Для чисел 4 и 7 и посылка и заключение ложны. (Если Вы сомневаетесь в истинности высказывания для последнего случая попробуйте произнести его в сослагательном наклонении: если бы числа 4 и 7 делились бы на 3, то и их сумма делилась бы на 3). Очевидно, что только один случай невозможен: мы не найдем таких двух слагаемых, чтобы каждое из них делилось на 3, а их сумма не делилась на 3, т.е. чтобы посылка была истинной, а заключение ложным. Из истины не может следовать ложь, иначе логика теряет смысл.

Предложение, образованное из двух предложений, объединённых связкой «если…, то …», в грамматике называется условным предложением, а в логике такое высказывание называется импликацией.

Импликация – логическая операция, ставящая в соответствие двум элементарным высказываниям новое высказывание, являющееся ложным тогда и только тогда, когда условие (посылка) – истинно, а следствие (заключение) ложно.

Высказывание «Если ***A***, то ***B***» с логической точки зрения имеет тот же смысл, что и высказывание «неверно, что ***A*** истинно и B ложно». Это означает, что функцию импликации можно заменить комбинацией двух функций (отрицания и конъюнкции). Обычно, когда мы хотим установить ложность высказывания «Если ***A***, то ***B***», мы стараемся показать, что возможен случай, когда ***A*** истинно, а ***B*** ложно (доказательство «от противного»). Обозначим импликацию символом => и запись «***A*** => ***B***» будем читать: «Из ***A*** следует ***B***».

Импликацией ***A*** => ***B*** называется высказывание, которое ложно тогда и только тогда, когда ***A*** истинно и ***B*** ложно. Запишем это определение в виде таблицы истинности:

A	*B*	*A=>B*
И	И	И
И	Л	Л
Л	И	И
Л	Л	И

Логическое следование: ИМПЛИКАЦИЯ - связывает два простых логических выражения, из которых первое является условием (***A***), а второе (***B***)– следствием из этого условия. Результатом ИМПЛИКАЦИИ является ЛОЖЬ только тогда, когда условие ***A*** истинно, а следствие ***B*** ложно. Обозначается символом «следовательно» и выражается словами **ЕСЛИ … , ТО …**

Пример 12. Как импликация связывает два элементарных высказывания покажем на примере высказываний: «*данный четырёхугольник – квадрат*» (***A***) и «*около данного четырёхугольника можно описать окружность*» (***B***). Рассмотрим составное высказывание ***A→B***, понимаемое как *если данный четырёхугольник квадрат, то около него можно описать окружность.* Есть **три варианта,** когда высказывание ***A→B*** истинно: 1) ***A*** истинно и ***B*** истинно, то есть данный четырёхугольник квадрат, и около него можно описать окружность; 2) ***A*** ложно и ***B*** истинно, то есть данный четырёхугольник не является квадратом, но около него можно описать окружность (разумеется, это справедливо не для всякого четырёхугольника); 3) ***A*** ложно и ***B*** ложно, то есть данный четырёхугольник не является квадратом, и около него нельзя описать окружность. 4) Ложен только один вариант, когда ***A*** истинно, а ***B*** ложно, то есть данный четырёхугольник является квадратом, но около него нельзя описать окружность.

Импликацию мы используем тогда, когда хотим показать, что некоторое событие зависит от другого события.

Пример 13. Пусть человек сказал: Если завтра будет хорошая погода, то я пойду гулять. Здесь ***A***= *Завтра будет хорошая погода* и ***B***= *Я пойду гулять*. Ясно, что человек окажется лжецом лишь в том случае, если погода действительно окажется хорошей, а гулять он не пойдёт. Если же погода будет плохой, то независимо от того, пойдёт он гулять ли нет, во лжи его нельзя обвинить: обещание пойти гулять он давал лишь при условии, что погода будет хорошей.

В обычной речи связка «*если …, то*» описывает причинно-следственную связь между высказываниями. Но в логических операциях смысл высказываний не учитывается. Рассматривается только их истинность или ложность. Поэтому не надо смущаться «бессмысленностью» импликаций, образованных высказываниями, совершенно не связанными по содержанию. Например, такими: «*если президент США - демократ, то в Африке водятся жирафы», «если арбуз - ягода, то в бензоколонке есть бензин».*

Импликация заведомо истинна, если условие А ложно. Другими словами, из неверного условия может следовать всё, то угодно. Например, высказывание Если 2>3, то крокодилы летают является истинным.

Союз «или» может применяться в речи и в другом, «исключающем» смысле. Тогда ое соответствует другому высказыванию – разделительной, или строгой дизъюнкции.

Высказывание, образованное из двух высказываний, объединённых связкой «либо» (точнее: «либо только…, либо только…»), называется **разделительной (строгой) дизъюнкцией**, **исключающим ИЛИ, сложением по модулю 2**.

В отличие от обычной дизъюнкции (связка «или»), в высказывании, являющемся разделительной дизъюнкцией, мы утверждаем, что произойдёт только одно событие из двух. **Пример 14.** Сообщая: *Петя сидит на трибуне А либо на трибуне Б*, мы утверждаем, что Петя сидит только на трибуне А, либо только на трибуне Б.

Строгая, или разделительная дизъюнкция – логическая операция, ставящая в соответствие двум элементарным высказываниям новое высказывание, являющееся истинным тогда и только тогда, когда ровно одно из двух высказываний является истинным. Логическая операция разделительная дизъюнкция определяется следующей таблицей истинности:

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Пример 15. Рассмотрим два высказывания *A= Кошка охотится за мышами* и *B = Кошка спит на диване.* Очевидно, что новое высказывание *A⊕B* истинно только в двух случаях: когда кошка охотится за мышами или когда кошка мирно спит. Это высказывание будет ложным, если кошка не делает ни того, ни другого, т.е. когда оба события не происходят. Но это высказывание будет ложным и тогда, предполагается, что оба события будут происходить одновременно.

В логике связкам «либо» и «или» придаётся разное значение, однако в русском языке связку «или» иногда употребляют вместо связки «либо». Чтобы определить значение связки «или», нужно проанализировать содержание высказывания по смыслу. Например, анализ высказывания *Петя сидит на трибуне А или на трибуне Б* однозначно укажет на логическую операцию разделительная дизъюнкция, так как человек не может находиться в двух местах одновременно.

Подавляющее число зависимостей между событиями можно описать с помощью импликации.

Импликация, образованная из высказываний ***A*** и ***B***, может быть записана на естественном языке при помощи следующих предложений: «Если ***A***, то ***B***», «Из ***A*** следует ***B***», «***A*** влечёт ***B***».

Может показаться странным, что высказывание «Если А, то В» всегда истинно, если посылка (высказывание А) ложна. Но для математика это вполне естественно. В самом деле, исходя из ложной посылки, можно путём верных рассуждений получить как истинное, так и ложное утверждение.

Пример 16. Допустим, что 1=2, тогда и 2=1. Складывая эти равенства, получим 3=3, т.е. из ложной посылки путём тождественных преобразований мы получили истинное высказывание.

Большинство математических теорем являются импликациями. Однако те импликации, в которых посылки (условия) и заключения (следствия) являются предложениями без взаимной (по существу) связи, не могут играть в науке важной роли. Они являются бесплодными предложениями, т. к. не ведут к выводам более глубокого содержания. В математике ни одна теорема не является импликацией, в которой условие и заключение не были бы связаны по содержанию. Достаточно часто в математических теоремах импликации формулируются в виде *только необходимого* и *только достаточного* условия.

Обратная импликация

A	B	A<=B
1	1	1
1	0	1
0	1	0
0	0	1

Другие обозначения этой функции ←, ⊂: Эта функция дает 0 только когда первый операнд равен 0, а второй равен 1.

1.3.9 Логическое тождество (эквиваленция).

Высказывание, образованное из двух высказываний при помощи связки «только и только тогда, когда» в логике называется эквивалентностью. Эквивалентность используется в тех случаях, когда необходимо выразить взаимную обусловленность. Например, сообщая: *Я получу паспорт тогда и только тогда, когда мне исполнится 14 лет*, человек утверждает не только то, что после того, как ему исполнится 14 лет, он получит паспорт, но и то, что паспорт он сможет получить только после того, как ему исполнится 14 лет.

Интуитивно можно догадаться, что высказывания эквивалентны (равносильными), когда их значения истинности одинаковы. Например, эквивалентны высказывания: «железо тяжелое» и «пух легкий», так же как и высказывания: «железо легкое» и «пух тяжёлый». Обозначим эквиваленцию символом <=> и запись «***A***<=>***B***» будем читать «***A*** эквивалентно ***B***», или «***A*** равносильно ***B***», или «***A***, если и только если ***B***». Таким образом, эквиваленцией двух высказываний ***A*** и ***B*** называется такое высказывание, которое истинно тогда и только тогда, когда оба эти высказывания ***A*** и *B* истинны или оба ложны. Отметим, что высказывание типа «***A***, если и только если ***B***» можно заменить высказыванием «Если ***A***, то ***B*** и, если ***B***, то ***A***»). Следовательно, функцию эквиваленции можно заменить комбинацией функций импликации и конъюнкции. Запишем таблицу истинности для эквиваленции:

A	***B***	***A***<=>***B***
И	И	И
И	Л	Л
Л	И	Л
Л	Л	И

Приведем примеры записи сложных высказываний с помощью обозначения логических связок:

«*Быть иль не быть - вот в чем вопрос*» (В. Шекспир) ***A***V ¬ ***A*** <=> ***B***

«*Если хочешь быть красивым, поступи в гусары*» (К. Прутков) ***A*** => ***B***

РАВНОСИЛЬНО Операция, выражаемая связками «*тогда и только тогда*», «*необходимо и достаточно*», «... *равносильно* ...», называется **эквиваленцией** или двойной импликацией и обозначается знаком ↔, **~, ≡.** Высказывание истинно тогда и только тогда, когда значения ***A*** и ***B*** совпадают. Эта функция дает 1 только когда оба операнда равны между собой.

Например, высказывания «*24 делится на 6 тогда и только тогда, когда 24 делится на 3*», «*23 делится на 6 тогда и только тогда, когда 23 делится на 3*» истинны, а высказывания «*24 делится на 6 тогда и только тогда, когда 24 делится на 5*», «*21 делится на 6 тогда и только тогда, когда 21 делится на 3*» ложны.

A	B	F
1	1	1
1	0	0
0	1	0
0	0	1

Логическая равнозначность: ЭКВИВАЛЕНТНОСТЬ - определяет результат сравнения двух простых логических выражений A и B. Результатом ЭКВИВАЛЕНТНОСТИ является новое логическое выражение, которое будет истинным тогда и только тогда, когда оба исходных выражения одновременно истинны или одновременно ложны. Обозначается символом «эквивалентности»

Высказывания ***A*** и ***B*,** образующие составное высказывание ***A↔B***, могут быть совершенно не связаны по содержанию, например: «*три больше двух*» (***A***), «*пингвины живут в Антарктиде*» (***B***). Отрицаниями этих высказываний являются высказывания «*три не больше двух*» ($\overline{A}$), «*пингвины не живут в Антарктиде*» ($\overline{B}$). Образованные из высказываний ***A*** и ***B*** составные высказывания ***A↔B*** и $\overline{A} \leftrightarrow \overline{B}$ истинны, а высказывания ***A*↔** $\overline{B}$ и $\overline{A}$ **↔*B* -** ложны.

Итак, нами рассмотрены пять логических операций: отрицание, конъюнкция, дизъюнкция, импликация и эквиваленция.

Импликацию можно выразить через дизъюнкцию и отрицание:

$$A \rightarrow B = \overline{A} \vee B.$$

Эквиваленцию можно выразить через *отрицание*, *дизъюнкцию* и *конъюнкцию*:

$$A \leftrightarrow B = (\overline{A} \vee B) \cdot (\overline{B} \vee A).$$

Таким образом, операций отрицания, дизъюнкции и конъюнкции достаточно, чтобы описывать и обрабатывать логические высказывания.

Утверждение ***A*∨*B*** считается истинным тогда и только тогда, когда истинно хотя бы одно из исходных утверждений; утверждение ***A*&*B*** – когда истинны оба утверждения.

Пример 17. Рассмотрим возможные значении сложного высказывания, являющегося эквивалентностью: Учитель утверждает, что 5 в четверти ученику он поставит тогда и только тогда, когда ученик получит 5 на зачёте. 1) Учение получил 5 на зачёте и 5 в четверти, т.е. учитель выполнил своё обещание, следовательно, высказывание является истинным. 2) Ученик не получил на зачёте 5, и учитель не поставил ему 5 в четверти, т.е. учитель своё обещание сдержал, высказывание является истинным. 3) Ученик не получил на зачёте 5, но учитель поставил ему 5 в четверти, т.е. учитель своё обещание не сдержал, высказывание является ложным. 4) Ученик получил на зачёте 5, но учитель не поставил ему 5 в четверти, т.е. учитель своё обещание не сдержал, высказывание является ложным.

В математических теоремах эквивалентность выражается связкой «необходимо и достаточно».

1.3.10 Штрих Шеффера

A	B	A|B
0	0	1
0	1	1
1	0	1
1	1	0

Другие названия этой функции: **И-НЕ**. Другие обозначения этой функции: ' (апостроф).
Эта функция даёт 0 только когда оба операнда равны 1

1.3.11 Таблицы истинности

Приведём сводную таблицу истинности для бинарных логических операций.

A	B	0	A↓B	A<B	~A	A>B	~B	A⊕B	A|B	A&B	A⇔B	B	A⇒B	A	A⇐B	A∨B	1
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Итак, общий формат для обозначения функции от двух аргументов через знак бинарной операции выглядит как (*F*)#(*G*), где *F* и *G* - некоторые формулы, а # - знак бинарной операции. Для вычисления значения формулы (*F*)#(*G*) надо вычислить значение формул *F* и *G*, и подставить их в таблицу истинности как первый и второй аргументы функции.

Порядок выполнения логических операций в сложном логическом выражении:

1. инверсия
2. конъюнкция
3. дизъюнкция
4. импликация
5. эквивалентность

Для изменения указанного порядка выполнения операций используются скобки.

Порядок выполнения логических операций задается круглыми скобками. Но для уменьшения числа скобок договорились считать, что сначала выполняется операция отрицания («не»), затем конъюнкция («и»), после конъюнкции - дизъюнкция («или») и в последнюю очередь - импликация.

Дизъюнкцию и конъюнкцию можно рассматривать как особые операции, определённые не на числах, а на логических значениях ИСТИНА и ЛОЖЬ. Для этих операций существуют таблицы, подобные таблице умножения.

A	B	AvB
ИСТИНА	ИСТИНА	ИСТИНА
ИСТИНА	ЛОЖЬ	ИСТИНА
ЛОЖЬ	ИСТИНА	ИСТИНА
ЛОЖЬ	ЛОЖЬ	ЛОЖЬ

A	B	A & B
ИСТИНА	ИСТИНА	ИСТИНА
ИСТИНА	ЛОЖЬ	ЛОЖЬ
ЛОЖЬ	ИСТИНА	ЛОЖЬ
ЛОЖЬ	ЛОЖЬ	ЛОЖЬ

Логические значения ИСТИНА и ЛОЖЬ называют также булевыми значениями – в честь английского математика Джорджа Буля, который в XIX в. заложил основы современной математической логики. Функции с булевыми аргументами называют булевыми функциями. Всего булевых функций от 2 переменных – 16. Для всех булевых функций от двух переменных имеются соответствующие конструкции на русском языке. В информатике в основном используются следующие булевы функции:

- логическое **ИЛИ** (дизъюнкция)
- логическое **И** (конъюнкция)
- логическое отрицание («**НЕ**», обозначается ~ и противоположно своему аргументу)
- исключающее **ИЛИ**

Из этих основных складываются комбинированные функции: **ИЛИ-НЕ**, **И-НЕ**. Именно они получили наибольшее распространение в логической электронике, в компьютерах.

Замечание. Программисты обычно используют следующие обозначения:

- «Истина» - *true*
- «Лож» - *false*
- Логическое «и» - *and*
- Логическое «или» - *or*
- Логическое отрицание - *not*

Порядок операций обозначают скобками и предполагают, что отрицание имеет наибольший приоритет. То есть выражение *A and not B* следует понимать как *A and (not B)*

Правила выполнения операций и построение таблиц истинности:

Операция отрицания: Логическое <<и>>: Логическое <<или>>:

---

not true = false *false and false = false* *false or false = false*
not false = true *true and false = false* *true or false = true*
fasle and true = false *fasle or true = true*
true and true = true *true or true = true*

Употребляемые в обычной речи связки «и», «или», «не», «если…, то…», «тогда и только тогда, когда…» и т.п. позволяют из уже заданных высказываний строить новые сложные высказывания. Истинность или ложность получаемых таким образом высказываний зависит от истинности и ложности исходных высказываний и соответствующей трактовки связок как логических операций над высказываниями. Логическая операция может быть полностью описана таблицей истинности, указывающей, какие значения принимает сложное высказывание при всех возможных значениях простых высказываний.

В алгебре логики логические связки и соответствующие им логические операции обозначаются следующим образом:

Логическая связка	Названия логической операции	Обозначения
не	Отрицание, инверсия	$\bar{}$, ⌉, ¬
и, а, но, хотя	Конъюнкция, логическое умножение	&, · , ∧
или	Дизъюнкция, нестрогая дизъюнкция, логическое сложение	∨, +
либо	Разделительная (строгая) дизъюнкция, исключающее ИЛИ, сложение по модулю 2	⊕, Δ
если ..., то	Импликация, следование	⇒, →
тогда и только тогда, когда	Эквивалентность, эквиваленция, равнозначность	⇔, ~, ≡, ↔

Рассмотрим теперь способы построения таблиц истинности для сложных выражений.

Количество строк = 2^n + две строки для заголовка (n - количество простых высказываний)

Количество столбцов = количество переменных + количество логических операций

При построении таблицы надо учитывать все возможные сочетания логических значений 0 и 1 исходных выражений. Затем – определить порядок действий и составить таблицу с учетом таблиц истинности основных логических операций.

Пример 18. Составить таблицу истинности сложного логического выражения D = не A & (B+C)

A, B, C - три простых высказывания, поэтому : количество строк = 23 +2 = 10 (n=3, т.к. на входе три элеманта A, B, C) количество столбцов : 1) A 2) B 3) C 4) не A это инверсия A (обозначим E) 5) B+ C это операция дизъюнкции (обозначим F) 6) D = неA & (B+C), т.е. $D = E\&F$ это операция конъюнкции

1	2	3	4	5	6
A	**B**	**C**	**E = не A** (не 1)	**F = B+C** (2+3)	**D = E&F** (4*5)
1	**1**	**1**	0	1	0
1	**1**	**0**	0	1	0
1	**o**	**1**	0	1	0
1	**o**	**0**	0	0	0
0	**1**	**1**	1	1	1
0	**1**	**0**	1	1	1
0	**0**	**1**	1	1	1
0	**0**	**0**	1	0	0

1. 4 Логические формулы. Законы алгебры логики

Математики под словом «алгебра» подразумевают науку, которая изучает некие объекты и операции над ними. Например, школьная алгебра (алгебра действительных чисел) изучает действительные числа и операции над ними. Предметом алгебры логики являются высказывания, операции над ними, а также логические функции. При этом для обозначения высказываний используются буквы.

Логическая переменная - переменная, значением которой может быть любое высказывание. Логические переменные обозначаются латинскими буквами, иногда снабжёнными индексами, как обычные алгебраические переменные x, y, x_1, y_1, x_k, y_n и т.п.

Понятие логической формулы является формализацией понятия сложного высказывания. Введём его индуктивно.

Логической формулой является: 1) любая логическая переменная, а также каждая из двух логических констант – 0 (ложь) и 1 (истина). 2) Если A и B – формулы, то $\overline{B}$ и $A*B$ – тоже формулы, где знак «*» означает любую из логических бинарных операций. Формулой является, например, выражение: $(x\&y)\rightarrow z$. Каждой формуле при заданных значениях входящих в неё переменных приписывается одно из двух значений – 0 или 1. Формулы A и B, зависящие от одного и того же набора переменных $x_1, x_2, x_3,\ldots,x_n$, называют равносильными или эквивалентными, если на любом наборе значений переменных $x_1, x_2, x_3,\ldots,x_n$ они имеют одинаковые значения. Для обозначения равносильности формул используется знак равенства, например $A=B$.

Любую формулу можно преобразовать к равносильной ей, в которой используются только аксиоматически введённые операции &, ∨ и отрицание.

С помощью логических переменных и символов логических операций любое высказывание можно формализовать, то есть заменить логической формулой.

Определение *<u>логической формулы</u>*:

Всякая логическая переменная и символы «истина» (1) и «ложь (0) - формулы.

Если A и B - формулы, то $\overline{A}$, $A \cdot B$, $A \vee B$, $A \to B$, $A \leftrightarrow B$ - формулы.

Никаких других формул в алгебре логики нет.

Здесь в п. 1 определены **элементарные формулы**; в п. 2 даны **правила образования из любых данных формул новых формул.**

В качестве примера рассмотрим высказывание «*если я куплю яблоки или абрикосы, то приготовлю фруктовый пирог».* Это высказывание формализуется в виде $(A \vee B) \to C$. Такая же формула соответствует высказыванию «*если Игорь знает английский или японский язык, то он получит место переводчика».*

Как показывает анализ формулы $(A \vee B) \to C$, при определённых сочетаниях значений переменных A, B и C она принимает значение «истина», а при некоторых других сочетаниях - значение «ложь» (разберите самостоятельно эти случаи). Такие формулы называются **выполнимыми**.

Некоторые формулы принимают значение «истина» при любых значениях истинности входящих в них переменных. Таковой будет, например, формула $A \vee \overline{A}$, соответствующая высказыванию *Этот треугольник прямоугольный или косоугольный.* Эта формула истинна и тогда, когда треугольник прямоугольный, и тогда, когда треугольник не прямоугольный. Такие формулы называются **тождественно истинными формулами** или **тавтологиями**. Высказывания, которые формализуются тавтологиями, называются **логически истинными высказываниями.**

В качестве другого примера рассмотрим формулу $A \cdot \overline{A}$ которой соответствует, например, высказывание *Катя самая высокая девочка в классе, и в классе есть девочки выше Кати.* Очевидно, что эта формула ложна, так как либо A, либо $\overline{A}$ обязательно ложно. Такие формулы называются **тождественно ложными формулами** или **противоречиями**. Высказывания, которые формализуются противоречиями, называются **логически ложными высказываниями.**

Если две формулы A и B одновременно, то есть при одинаковых наборах значений входящих в них переменных, принимают одинаковые значения, то они называются **равносильными**. Равносильность двух формул алгебры логики обозначается символом «=» или символом «≡» Замена формулы другой, ей равносильной, называется **равносильным преобразованием** данной формулы.

Для преобразования формул в равносильные важную роль играют следующие равенства, отражающие свойства логических операций, которые по аналогии с алгеброй вещественных чисел будем называть законами:

1) Законы коммутативности

$x\&y=y\&x$, $x \vee y=y \vee x$;

2) Законы ассоциативности

$(x\&y)\&z=x\&(y\&z)$, $(x \vee y) \vee z=x \vee (y \vee z)$;

3) Законы поглощения (нуля и единицы)

$x \vee 0=x$, $x\&1=x$;

4) Законы дистрибутивности

$x\&(y \vee z)=(x\&y) \vee (x\&z)$, $x \vee (y\&z)=(x \vee y)\&(x \vee z)$;

5) Закон противоречия

$x \vee \overline{x} =0$;

6) Закон исключённого третьего

$x \vee \bar{x} = 1$;

7) Закон идемпотентности

$x\&x=x$, $x \vee x = x$;

8) Закон двойного отрицания

$\bar{\bar{x}} = \bar{x}$;

9) Законы де Моргана

$\overline{x \& y} = \bar{x} \vee \bar{y}$, $\overline{x \vee y} = \bar{x} \& \bar{y}$;

9) Законы поглощения

$x \vee (x\&y) = x$, $x\&(x \vee y) = x$.

Любой из этих законов может быть доказан с помощью таблиц истинности.

Аксиомы

1. ¬(¬x)=x, x=¬(¬x);
2. x*(¬x)=0;
3. x+1=1;
4. x+x=x, x=x+x+x;
5. x+0=x;
6. x*x=x, x=x*x*x;
7. x*0=0;
8. x*1=x;
9. x+(¬x)=1.

Табл. 1. Основные законы алгебры логики

Закон	Для ИЛИ	Для И
Переместительный	$x \vee y = y \vee x$	$x \cdot y = y \cdot x$
Сочетательный	$x \vee (y \vee z) = (x \vee y) \vee z$	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$
Распределительный	$x \cdot (y \vee z) = x \cdot y \vee x \cdot z$	$x \vee (y \cdot z) = (x \vee y) \cdot (x \vee z)$
Правила де Моргана	$\overline{x \vee y} = \bar{y} \cdot \bar{x}$	$\overline{x \cdot y} = \bar{y} \vee \bar{x}$
Идемпотенции	$x \vee x = x$	$x \cdot x = x$
Поглощения	$x \vee (x \cdot y) = x$	$x \cdot (x \vee y) = x$
Склеивания	$(x \cdot y) \vee (\bar{x} \cdot y) = y$	$(x \vee y) \cdot (\bar{x} \vee y) = y$
Операция переменной с её инверсией	$x \vee \bar{x} = 1$	$x \cdot \bar{x} = 0$
Операция с константами	$x \vee 0 = x, x \vee 1 = 1$	$x \cdot 1 = x, x \cdot 0 = 0$
Двойного отрицания	$\bar{\bar{x}} = x$	

1.4.1 Основные законы булевой алгебры

Две формулы булевой алгебры равносильны (равны, эквивалентны), если равны сопоставляемые им функции (т.е. они принимают одинаковые значения на всех наборах значений аргументов). Ниже даны основные законы булевой алгебры, позволяющие проводить тождественные преобразования формул булевой алгебры (обратите внимание, насколько они похожи на законы классической арифметики):

1. закон двойного отрицания: *not not x = x*
2. закон коммутативности (от перестановки аргументов результат не меняется): x_1 *or* $x_2 = x_2$ *or* x_1
3. x_1 *and* $x_2 = x_2$ *and* x_1
4. закон ассоциативности (порядка вычислений): x_1 *or* $(x_2$ *or* $x_3) = (x_1$ *or* $x_2)$ *or* x_3 x_1 *and* $(x_2$ *and* $x_3) = (x_1$ *and* $x_2)$ *and* x_3

5. закон дистрибутивности (раскрытия скобок): x_1 *or* (x_2 *and* x_3) = (x_1 *or* x_2) *and* (x_1 *or* x_3)
x_1 *and* (x_2 *or* x_3) = (x_1 *and* x_2) *or* (x_1 *and* x_3)
6. правила де Моргана: *not* (x_1 *or* x_2) = *not* x_1 *and not* x_2 *not* (x_1 *and* x_2) = *not* x_1 *or not* x_2
7. правила операций с константами 0 и 1: *not 0 = 1, not 1 = 0, x or 0 = x, x or 1 = 1, x and 1 = x, x and 0 = 0*
8. правила операций с переменной и её инверсией: *x or not x = 1 x and not x = 0*

Из основных законов следуют важные соотношения:

1. закон поглощения: x_1 *or* (x_1 *and* x_2) = x_1 x_1 *and* (x_1 *or* x_2) = x_1
2. закон идемпотентности (повторное применение не даёт ничего нового): *x or x or ... or x = x x and x and ... and x = x*
3. на основании закона дистрибутивности, а также 7-го и 6-го законов: x_1 *or* (*not* x_1 *and* x_2) = x_1 *or* x_2

1.4.2 Преобразование выражений, состоящих из булевых функций.

В математической логике преобразование выше указанных выражений проводится для различных целей – от упрощения исходного до доказательства утверждений. В информатике же оно используется в основном для упрощения, ведь при производстве цифровой электроники, как и любого другого товара, требуются наименьшие затраты. Для упрощения булевых выражений используются те же методы, что и при упрощении алгебраических. Для начала была проведена аналогия между алгебраическими операторами от двух аргументов (сложение, вычитание, умножение и т.д.) и булевыми. Было выяснено, что умножение и логическое «И» обладают сходными свойствами:

- от перестановки мест аргументов результат не изменяется

$A\&B = B\&A$

- существует следующий закон

$A\&(B\&C) = (A\&B)\&C$

Также существуют некоторые тождества, опирающиеся на особые свойства функции, например:

1) $A\&(\sim A)$ = ЛОЖЬ
2) $(\sim A)\&(\sim B) = \sim(A \vee B)$

Аналогично, сложение и логическое «ИЛИ»:

- от перестановки мест аргументов результат не изменяется

$A \vee B = B \vee A$

- существует следующий закон

$(A \vee B) \vee C = A \vee (B \vee C)$

- можно выносить общий множитель за скобки

$(A\&B) \vee (C\&B) = B\&(A \vee C)$

И также некоторые собственные законы:

1) $A \vee (\sim A)$ = ИСТИНА
2) $(\sim A) \vee (\sim B) = \sim (A\&B)$

Когда вычисляется значение булевого выражения, то выполняется определённая очерёдность действий: на очерёдность влияют скобки, сначала считаются «И», затем «ИЛИ». Благодаря этой очерёдности возможно создание электронных цифровых схем.

Исходное выражение можно найти по его значениям.

В отличие от алгебраических выражений, булевы можно восстановить, зная их аргументы и соответственные им значения. Пусть нам дана булева функция от 3 переменных:

$X1$	$X2$	$X3$	F
0	0	0	0
1	0	0	0
0	1	0	0
1	1	0	1
0	0	1	0
1	0	1	1
0	1	1	0
1	1	1	1

Составим для неё таблицу и условимся обозначать ИСТИНУ - 1, а ЛОЖЬ – 0. Для начала выпишем все аргументы функции, при которых функция равна 1. Это:

F (1, 1, 0) = 1

$F(1, 0, 1) = 1$
$F(1, 1, 1) = 1$
Теперь запишем 3 таких выражения (функция принимает значение 1 три раза), что они принимают значение 1 только при вышеуказанных значениях.
$X1 \& X2 \& (\sim X3)$
$X1 \& (\sim X2) \& X3$
$X1 \& X2 \& X3$
И запишем их логическую сумму:
$(X1\&X2\&(\sim X3))v(X1 \& (\sim X2)\&X3)v(X1\&X2\&X3)$ – это выражение принимает значение 1 при тех же значениях, что и исходная функция. Полученное выражение можно упростить.
$(X1\&X2\&(\sim X3))v(X1\&(\sim X2)\&X3)v(X1\&X2\&X3)=X1\&((X2\&(\sim X3))v((\sim X2)\&X3)v(X2\&X3))=X1\&$
$((X2\&(\sim X3))vX3\&((\sim X2)vX2))= X1\&((X2\&(\sim X3))vX3)$ – эта формула несколько длиннее исходной, но намного проще полученной в первый раз. Дальнейшие пути упрощения более сложны и представляют большой интерес для проектировщиков интегральных микросхем, т.к. меньшее число операций требует меньшее число элементов, их которых состоит ИС.

Конъюнкция	Дизъюнкция	Инверсия
$A \wedge \bar{A}=0$ $A \wedge A=A$ $A \wedge 1=A$ $A \wedge 0=0$	$A \vee \bar{A}=1$ $A \vee A=A$ $A \vee 1=1$ $A \vee 0=A$	$\bar{\bar{A}}=A$
Переместительный закон	**Сочетательный закон**	**Распределительный закон**
$A \vee B=B \vee A$ $A \wedge B=B \wedge A$	$(A \vee B) \vee C=A \vee (B \vee C)$ $(A \wedge B) \wedge C=A \wedge (B \wedge C)$	$(A \vee B) \wedge C=(A \wedge C) \vee (B \wedge C)$ $(A \wedge B) \vee C=(A \vee C) \wedge (B \vee C)$
Закон инверсии	**Формулы склеивания**	**Формулы поглощения**
$\overline{A \vee B}=\bar{A} \wedge \bar{B}$ $\overline{A \wedge B}=\bar{A} \vee \bar{B}$	$(A \wedge B) \vee (A \wedge \bar{B})=A$ $(A \vee B) \wedge (A \vee \bar{B})=A$	$A \vee (A \wedge B)=A$ $A \wedge (A \vee B)=A$ $A \vee (\bar{A} \wedge B)= A \vee B$ $A \wedge (\bar{A} \vee B)= A \wedge B$
$A \to B = \bar{A} \vee B$ $A \Leftrightarrow B\ (A \to B) \wedge (B \to A)$	$(A \to B) \Leftrightarrow (\bar{B} \to \bar{A})$ $(A \to B) \wedge (B \to C) \to (A \to C)$	$(A \Leftrightarrow B) \wedge (B \Leftrightarrow C) \to (A \Leftrightarrow C)$

1.4.3 Алгебры булевых функций

Любую булеву функцию с любым количеством аргументов можно построить через суперпозицию (подстановку функций вместо аргументов других функций) элементарных логических функций (функций одного и двух переменных). С другой стороны, для представления любой функции алгебры логики достаточно ограниченного числа функций, составляющих функционально полную систему - базис. Конъюнкция, дизъюнкция и отрицание - один из таких базисов, и прочие функции могут выражаться через эти функции. Другие базисы:

- *not x, x_1 and x_2*
- *not x, x_1 or x_2*
- $x_1 | x_2$ (штрих Шеффера)
- *not x_1 and not x_2* (функция Пирса)
- *x_1 xor x_2* (сложение по модулю 2), *x_1 and x_2, 1*
- *x_1 and not x_2* (запрет по x_2), *1*

Не правда ли, замечательные функции - штрих Шеффера и функция Пирса? Их одних достаточно для построения всех прочих функций алгебры логики. Действительно:

$x_1 | x_1 = not\ x_1\ or\ not\ x_1 = not\ x_1$
$(x_1 | x_1) | (x_2 | x_2) = not\ (not\ x_1\ or\ not\ x_1)\ or\ not\ (not\ x_2\ or\ not\ x_2)$
$= not\ not\ x_1\ or\ not\ not\ x_2 = x_1\ or\ x_2$
$(x_1 | x_2) | (x_1 | x_2) = not\ (not\ x_1\ or\ not\ x_2)\ or\ not\ (not\ x_1\ or\ not\ x_2)$
$= (x_1\ and\ x_2)\ or\ (x_1\ and\ x_2) = x_1\ and\ x_2$

В электронике это означает, что для реализации всего многообразия схем преобразования сигналов, представляющих логические значения, достаточно нескольких типовых элементов. С другой стороны, использование функций, не входящих в некоторый базис (функциональная избыточность), позволяет существенно сократить сложность реализующих выражения схем и тем самым повысить их надёжность.

Заметьте: помимо отрицания в базис булевой алгебры входят и конъюнкция, и дизъюнкция, хотя достаточно только одной из них. Вероятно, здесь есть связь с близостью этих функций человеческому

мышлению, выражаемому через естественные языки. Однако, несомненно, прочие функции также имеют свои достоинства.

Минимальность и избыточность - важные аспекты теории информации. Как факт: измерения избыточности русского языка дали около 80%. В сленгах (например, языке авиадиспетчеров) избыточность ещё выше.

1.4.4 Функция сложения по модулю 2 (xor)

Завершая тему базисов следует отметить, что набор функций языков программирования обычно включает конъюнкцию, дизъюнкцию, отрицание и «исключающее или». Ниже дана таблица истинности этих функций (вырезка из полной таблицы для всех функций):

x_1	x_2	*and*	*or*	*xor*
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Почему *xor* называется «сложение по модулю 2»? Потому что так оно и есть: в двоичной системе *0+0=0, 0+1=1+0=1, 1+1=10*, а по модулю 2 (остаток от деления на 2) последняя сумма как раз и даёт 0. Функция *xor* обладает замечательными свойствами: 1) при инвертировании одного из аргументов эта функция также инвертируется; 2)эта функция показывает, когда аргументы не равны (а при инвертировании одного из аргументов - когда равны); 3)она позволяет проводить управляемое инвертирование: при нулевом аргументе другой аргумент не меняется, при единичном же значении второй аргумент инвертируется; 4)эта функция инволютивна (её повторное применение возвращает к исходному аргументу): если ($x_3=x_1$ *xor* x_2), то (x_3 *xor* $x_1=x_2$) и (x_3 *xor* $x_2=x_1$).

На последнем свойстве построен трюк с обменом значений двух переменных без использования третьей переменной (в *C* это записывается как *a^=b^=a^=b*). В графике эта функция применяется при выводе спрайтов на картинку - повторное её применение убирает спрайт с картинки. Также эта функция используется в криптографии - одна из схем заключается в наложении некоего кода на поток данных через функцию xor. Зашифрованный таким образом поток на исходный поток не похож, но может быть легко восстановлен повторным применением шифрующего кода.

На самом деле, функции «исключающее или», «неравнозначность» и «сложение по модулю 2» - разные функции, идентичные только в случае двух аргументов. Вот таблица истинности этих функций для трёх аргументов:

x_1	x_2	x_1	исключающее или	неравнозначность(несовпадение)	сложение по модулю 2	дизъюнкция (ИЛИ)
0	0	0	0	0	0	0
0	0	1	1	1	1	1
0	1	0	1	1	1	1
0	1	1	0	1	0	1
1	0	0	1	1	1	1
1	0	1	0	1	0	1
1	1	0	0	1	0	1
1	1	1	0	0	1	1

Несложно заметить, что инволютивность (при равенстве двух аргументов результат равен третьему) получается только для сложения по модулю 2. Поэтому, когда говорится о функции *xor*, обычно следует понимать именно сложение по модулю 2.

Неоднозначность технических и даже логических терминов идёт от неоднозначности разговорного языка. Например, в грамматике русского языка (да и английского тоже) не различаются исключающее ИЛИ и просто ИЛИ. Так, во фразе *ревёт ли зверь* союз «ли» (краткая форма «или») имеет значение дизъюнкции, а в приветствии *кошелёк или жизнь* (как и в напутствии «со щитом или на щите»), тот же союз выражает уже исключающее ИЛИ.

2. АЛГЕБРА ЛОГИКИ В ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКЕ

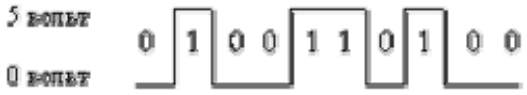
После изготовления первого компьютера стало ясно, что при его производстве возможно использование только цифровых технологий – ограничение сигналов связи единицей и нулём для большей

надёжности и простоты архитектуры компьютера. Благодаря своей бинарной природе, математическая логика получила широкое распространение в вычислительной технике и информатике. Были созданы электронные эквиваленты логических функций, что позволило применять методы упрощения булевых выражений к упрощению электрической схемы. Кроме того, благодаря возможности нахождения исходной функции по таблице позволило сократить время поиска необходимой логической схемы.

В программировании логика незаменима как строгий язык и служит для описания сложных утверждений, значение которых может определить компьютер. Математический аппарат алгебры логики очень удобен для описания того, как функционируют аппаратные средства компьютера, поскольку основной системой счисления в компьютере является двоичная, в которой используются цифры 1 и 0, а значений логических переменных тоже два: «1» и «0».

Из этого следует два вывода: 1) одни и те же устройства компьютера могут применяться для обработки и хранения как числовой информации, представленной в двоичной системе счисления, так и логических переменных; 2)на этапе конструирования аппаратных средств алгебра логики позволяет значительно упростить логические функции, описывающие функционирование схем компьютера, и, следовательно, уменьшить число элементарных логических элементов, из десятков тысяч которых состоят основные узлы компьютера. Данные и команды представляются в виде двоичных последовательностей различной структуры и длины. Существуют различные физические способы кодирования двоичной информации.

В электронных устройствах компьютера двоичные единицы чаще всего кодируются более высоким уровнем напряжения, чем двоичные нули (или наоборот), например:



Логическими элементами компьютеров являются электронные схемы **И, ИЛИ, НЕ, И—НЕ, ИЛИ—НЕ** и другие (называемые также **вентилями**), а также **триггер.** С помощью этих схем можно реализовать любую логическую функцию, описывающую работу устройств компьютера. Обычно у вентилей бывает от двух до восьми входов и один или два выхода. Чтобы представить два логических состояния – «1» и «0» в вентилях, соответствующие им входные и выходные сигналы имеют один из двух установленных уровней напряжения. Например, +5 вольт и 0 вольт. Высокий уровень обычно соответствует значению «истина» (1), а низкий — значению «ложь» (0).

Каждый логический элемент имеет свое условное обозначение, которое выражает его логическую функцию, но не указывает на то, какая именно электронная схема в нём реализована. Это упрощает запись и понимание сложных логических схем.

Работу логических элементов описывают с помощью таблиц истинности.

Таблица истинности - табличное представление логической схемы (операции), в котором перечислены все возможные сочетания значений истинности входных сигналов (операндов) вместе со значением истинности выходного сигнала (результата операции) для каждого из этих сочетаний.

Рис. 1. Базовые логические элементы компьютера.

Схема И

Схема И реализует конъюнкцию двух или более логических значений. Условное обозначение на структурных схемах схемы **И** с двумя входами представлено на **Рис. 2**.

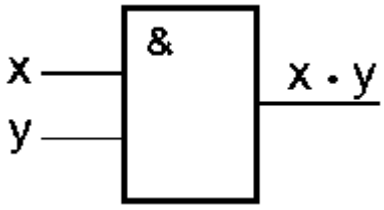


Рис. 2

Таблица истинности схемы **И**

x	*y*	*x*·*y*
0	0	0
0	1	0
1	0	0
1	1	1

Единица на выходе схемы И будет тогда и только тогда, когда на всех входах будут единицы. Когда хотя бы на одном входе будет ноль, на выходе также будет ноль. Связь между выходом *z* этой схемы и входами *x* и *y* описывается соотношением: *z*=*x*·*y* (читается как «*x* и *y*»). Операция конъюнкции на структурных схемах обозначается знаком «&» (читается как «амперсэнд»), являющимся сокращенной записью английского слова *and*.

Схема ИЛИ

Схема **ИЛИ** реализует дизъюнкцию двух или более логических значений**.** Когда хотя бы на одном входе схемы **ИЛИ** будет единица, на её выходе также будет единица.

Условное обозначение на структурных схемах схемы **ИЛИ** с двумя входами представлено на **Рис. 3**. Знак «**1**» на схеме — от устаревшего обозначения дизъюнкции как «**>=1**» (т.е. значение дизъюнкции равно единице, если сумма значений операндов больше или равна 1). Связь между выходом *z* этой схемы и входами ***x*** и ***y*** описывается соотношением: ***z* = *x*∨*y*** (читается как «***x* или *y***»).

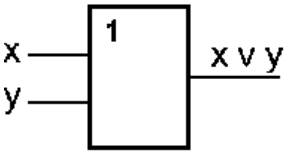


Рис. 3

Таблица истинности схемы **ИЛИ**

x	*y*	*x*∨*y*
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**

Схема НЕ

Схема **НЕ** (инвертор) реализует операцию отрицания. Связь между входом ***x*** этой схемы и выходом **z** можно записать соотношением $z = \overline{x}$, где $\overline{x}$ читается как «**не *x***» или «**инверсия *x*».**

Если на входе схемы **0,** то на выходе **1.** Когда на входе **1,** на выходе **0.** Условное обозначение на структурных схемах инвертора - на **Рис.4**

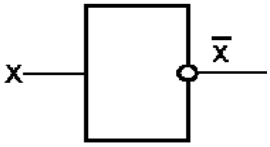


Рис. 4

Таблица истинности схемы **НЕ**

x	$\overline{x}$
0	1
1	0

Схема И—НЕ

Схема **И—НЕ** состоит из элемента **И** и инвертора и осуществляет отрицание результата схемы **И.** Связь между выходом **z** и входами **x** и **y** схемы записывают следующим образом: $z = \overline{x \cdot y}$, где $\overline{x \cdot y}$ читается как «**инверсия x и y».** Условное обозначение на структурных схемах схемы **И—НЕ** с двумя входами представлено на **Рис.5**.

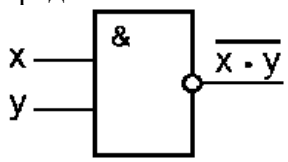


Рис. 5.

Таблица истинности схемы **И—НЕ**

x	*y*	$\overline{x \cdot y}$
0	0	1
0	1	1
1	0	1
1	1	0

Схема ИЛИ—НЕ

Схема **ИЛИ—НЕ** состоит из элемента **ИЛИ** и инвертора и осуществляет отрицание результата схемы **ИЛИ.** Связь между выходом *z* и входами *x* и *y* схемы записывают следующим образом: $z = \overline{x \vee y}$, где $\overline{x \vee y}$, читается как «**инверсия *x* или *y*».** Условное обозначение на структурных схемах схемы **ИЛИ—НЕ** с двумя входами представлено на **Рис. 6**.

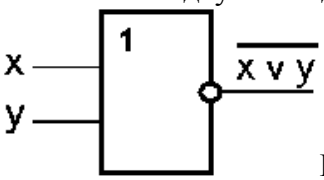


Рис. 6

Таблица истинности схемы **ИЛИ—НЕ**

x	*y*	$\overline{x \vee y}$
0	0	1
0	1	0
1	0	0
1	1	0

Триггер - электронная схема, широко применяемая в регистрах компьютера для надёжного запоминания одного разряда двоичного кода. Триггер имеет два устойчивых состояния, одно из которых соответствует двоичной единице, а другое — двоичному нулю.

Термин **триггер** происходит от английского слова *trigger* - защёлка, спусковой крючок. Для обозначения этой схемы в английском языке чаще употребляется термин *flip-flop*, что в переводе означает «хлопанье». Это звукоподражательное название электронной схемы указывает на её способность почти мгновенно переходить («перебрасываться») из одного электрического состояния в другое и наоборот.

Самый распространённый тип триггера - так называемый *RS*-триггер (*S* и *R*, соответственно, от английских *set* - установка, и *reset* - сброс). Условное обозначение триггера - на **Рис. 7**.

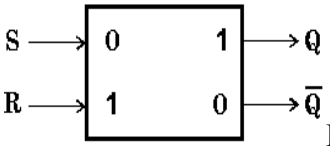


Рис. 8

Он имеет два симметричных входа *S* и *R* и два симметричных выхода *Q* и $\overline{Q}$, причем выходной сигнал *Q* является логическим отрицанием сигнала $\overline{Q}$. На каждый из двух входов *S* и *R* могут подаваться входные сигналы в виде кратковременных импульсов (

). Наличие импульса на входе будем считать единицей, а его отсутствие - нулём. На **Рис. 8** показана реализация триггера с помощью вентилей **ИЛИ—НЕ** и соответствующая таблица истинности.

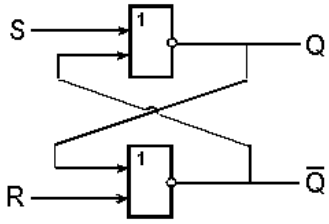


Рис. 8

S	R	Q	$\overline{Q}$
0	0	запрещено	
0	1	1	0
1	0	0	1
1	1	хранение бита	

Проанализируем возможные комбинации значений входов R и S триггера, используя его схему и таблицу истинности схемы **ИЛИ—НЕ**.

1. Если на входы триггера подать *S*=1, *R*=0, то (независимо от состояния) на выходе *Q* верхнего вентиля появится 0. После этого на входах нижнего вентиля окажется *R=0*, *Q*=0 и выход $\overline{Q}$ станет равным 1.
2. Точно так же при подаче 0 на вход *S* и 1 на вход *R* на выходе $\overline{Q}$ появится 0, а на *Q*- 1.
3. Если на входы *R* и *S* подана логическая 1, то состояние Q и $\overline{Q}$ не меняется.
4. Подача на оба входа *R* и *S* логического 0 может привести к неоднозначному результату, поэтому эта комбинация входных сигналов запрещена.

Поскольку один триггер может запомнить только один разряд двоичного кода, то для запоминания байта нужно 8 триггеров, для запоминания килобайта, соответственно, $8x2^{10}$ = 8192 триггеров. Современные микросхемы памяти содержат миллионы триггеров.

Сумматор - электронная логическая схема, выполняющая суммирование двоичных чисел

Сумматор служит, прежде всего, центральным узлом арифметико-логического устройства компьютера, однако он находит применение также и в других устройствах машины.

Многоразрядный двоичный сумматор, предназначенный для сложения многоразрядных двоичных чисел, **представляет собой комбинацию одноразрядных сумматоров,** с рассмотрения которых мы и начнём.

Условное обозначение одноразрядного сумматора на **Рис.9**.

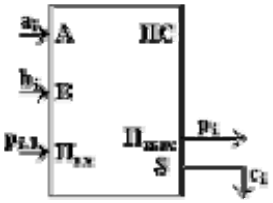


Рис. 9

При сложении чисел A и B в одном *i*-ом разряде приходится иметь дело с тремя цифрами: **1.** цифра a_i первого слагаемого; **2.** цифра b_i второго слагаемого; **3.** перенос p_{i-1} из младшего разряда. В результате сложения получаются две цифры: **1.** цифра c_i для суммы; **2.** перенос p_i из данного разряда в старший.

Таким образом, **одноразрядный двоичный сумматор есть устройство с тремя входами и двумя выходами**, работа которого может быть описана следующей таблицей истинности:

Входы			Выходы	
Первое слагаемое	Второе слагаемое	Перенос	Сумма	Перенос
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Если требуется складывать двоичные слова длиной два и более бит, то можно использовать последовательное соединение таких сумматоров, причём для двух соседних сумматоров выход переноса одного сумматора является входом для другого. Например, схема вычисления суммы $C = (c_3\ c_2\ c_1\ c_0)$ двух двоичных трехразрядных чисел $A = (a_2\ a_1\ a_0)$ и $B = (b_2\ b_1\ b_0)$ может иметь вид:

В алгебре логики выполняются законы, позволяющие производить *тождественные преобразования логических выражений*:

В компьютерах и других автоматических устройствах широко применяются электрические схемы, содержащие сотни и тысячи переключательных элементов: реле, выключателей и т.п. Разработка таких схем весьма трудоёмкое дело. Оказалось, что здесь с успехом может быть использован аппарат алгебры логики.

Переключательная схема - схематическое изображение некоторого устройства, состоящего из переключателей и соединяющих их проводников, а также из входов и выходов, на которые подаётся и с которых снимается электрический сигнал.

Каждый переключатель имеет только два состояния: замкнутое и разомкнутое. Переключателю ***X*** поставим в соответствие логическую переменную ***x***, которая принимает значение 1 в том и только в том случае, когда переключатель ***X*** замкнут и схема проводит ток; если же переключатель разомкнут, то ***x*** равен нулю.

Будем считать, что два переключателя ***X*** и $\overline{X}$ связаны таким образом, что когда ***X*** замкнут, то $\overline{X}$ разомкнут, и наоборот. Следовательно, если переключателю ***X*** поставлена в соответствие логическая переменная ***x***, то переключателю $\overline{X}$ должна соответствовать переменная $\overline{x}$.

Всей переключательной схеме также можно поставить в соответствие логическую переменную, равную единице, если схема проводит ток, и равную нулю - если не проводит. Эта переменная является функцией от переменных, соответствующих всем переключателям схемы, и называется **функцией проводимости**.

Найдем функции проводимости ***F*** некоторых переключательных схем:

а) 

Схема не содержит переключателей и проводит ток всегда, следовательно ***F*=1**;

б) 

Схема содержит один постоянно разомкнутый контакт, следовательно ***F*=0**;

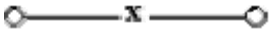
в) 

Схема проводит ток, когда переключатель ***x*** замкнут, и не проводит, когда ***x*** разомкнут, следовательно, ***F(x) = x***;

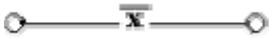
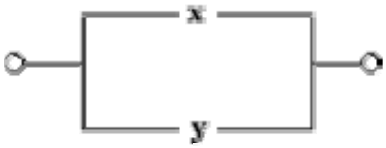
г) 

Схема проводит ток, когда переключатель ***x*** разомкнут, и не проводит, когда x замкнут, следовательно, ***F(x) =*** $\bar{x}$;

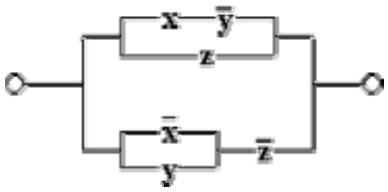
д) 

Схема проводит ток, когда оба переключателя замкнуты, следовательно, ***F(x) = x · y***;



е)

Схема проводит ток, когда хотя бы один из переключателей замкнут, следовательно, ***F(x)=xvy*;**



ж)

Схема состоит из двух параллельных ветвей и описывается функцией

$$F(x, y, z) = (x \cdot \bar{y}) \vee z \vee (\bar{x} \vee y) \cdot \bar{z}\,.$$

Две схемы называются *равносильными*, если через одну из них проходит ток тогда и только тогда, когда он проходит через другую (при одном и том же входном сигнале).

Из двух равносильных схем *более простой* считается та схема, функция проводимости которой содержит меньшее число логических операций или переключателей. Задача нахождения среди равносильных схем наиболее простых является очень важной. Две схемы называются *равносильными*, если через одну из них проходит ток тогда и только тогда, когда он проходит через другую (при одном и том же входном сигнале).

Из двух равносильных схем *более простой* считается та схема, функция проводимости которой содержит меньшее число логических операций или переключателей..

При рассмотрении переключательных схем возникают две основные задачи: **синтез и анализ схемы.**

СИНТЕЗ СХЕМЫ по заданным условиям ее работы сводится к следующим трём этапам:

1. составлению функции проводимости по таблице истинности, отражающей эти условия;
2. упрощению этой функции;
3. построению соответствующей схемы.

АНАЛИЗ СХЕМЫ сводится к

1. определению значений её функции проводимости при всех возможных наборах входящих в эту функцию переменных.
2. получению упрощённой формулы.

Примеры.

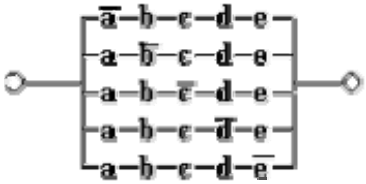
19. Построим схему, содержащую 4 переключателя *x*, *y*, *z* и *t*, такую, чтобы она проводила ток тогда и только тогда, когда замкнут контакт переключателя *t* и какой-нибудь из остальных трёх контактов.

Решение. В этом случае можно обойтись без построения таблицы истинности. Очевидно, что функция проводимости имеет вид ***F(x, y, z, t) = t · (xvyvz)***, а схема выглядит так:

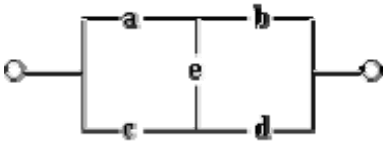
20. Построим схему с пятью переключателями, которая проводит ток в том и только в том случае, когда замкнуты ровно четыре из этих переключателей.

Решение : $F(a,b,c,d,e)=\bar{a}\cdot b\cdot c\cdot d\cdot e\vee a\cdot\bar{b}\cdot c\cdot d\cdot e\vee a\cdot b\cdot\bar{c}\cdot d\cdot e\vee a\cdot b\cdot c\cdot\bar{d}\cdot e\vee a\cdot b\cdot c\cdot d\cdot\bar{e}$

Схема имеет вид:



21. Найдем функцию проводимости схемы:



Решение. Имеется четыре возможных пути прохождения тока при замкнутых переключателях *a*, *b, c, d, e* : через переключатели *a, b*; через переключатели *a, e, d*; через переключатели *c, d* и через переключатели *c, e, b*. Функция проводимости ***F(a, b, c, d, e) = a· bva·e·dvc·dvc·e·b.***

22. Упростим переключательные схемы:

а)

Решение: $F(x,y)=x\cdot y\vee\bar{y}\cdot x=x\cdot(y\vee\bar{y})=x\cdot 1=x$

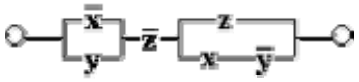
Упрощенная схема:

б)

Решение : $F(x,y,z)=\bar{z}\cdot(\bar{x}\vee y)\vee(z\vee x\cdot\bar{y})=1$.

Здесь первое логическое слагаемое $\bar{z}\cdot(\bar{x}\vee y)$ является отрицанием второго логического слагаемого $(z\vee x\cdot\bar{y})$, а дизъюнкция переменной с ее инверсией равна 1.

Упрощенная схема : 



в)

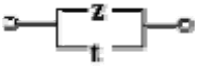
Решение : $F(x,y,z)=(\bar{x}\vee y)\cdot\bar{z}\cdot(z\vee x\cdot\bar{y})=(\bar{x}\cdot\bar{z}\vee y\cdot\bar{z})\cdot(z\vee x\cdot\bar{y})=$

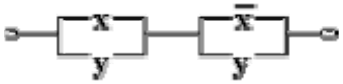
$=\bar{x}\cdot\bar{z}\cdot z\vee\bar{x}\cdot\bar{z}\cdot x\cdot\bar{y}\vee y\cdot\bar{z}\cdot z\vee y\cdot\bar{z}\cdot x\cdot\bar{y}=0\vee 0\vee 0\vee 0=0$

Упрощенная схема:

г)

Решение : $F(x,y,z,t)=(x\vee\bar{y}\vee y)\cdot(z\vee t)=1\cdot(z\vee t)=z\vee t$

Упрощенная схема: 

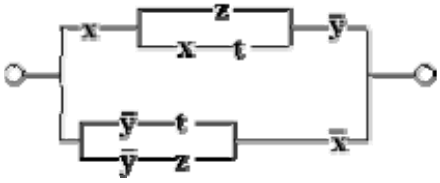


д)

Решение : $F(x,y)=(x\vee y)\cdot(\bar{x}\vee y)=y$ (по закону склеивания)

Упрощенная схема:

e)



Решение:

$$F(x,y,z,t) = x\cdot(z\vee x\cdot t)\cdot\bar{y}\vee(\bar{y}\cdot t\vee\bar{y}\cdot z)\cdot\bar{x} = x\cdot\bar{y}\cdot z\vee x\cdot\bar{y}\cdot x\cdot t\vee\bar{x}\cdot\bar{y}\cdot(t\vee z) =$$

$$= x\cdot\bar{y}\cdot(z\vee t)\vee\bar{x}\cdot\bar{y}\cdot(t\vee z) = (t\vee z)\cdot(x\cdot\bar{y}\vee\bar{x}\cdot\bar{y}) = (t\vee z)\cdot\bar{y}\cdot(x\vee\bar{x}) = \bar{y}\cdot(t\vee z)$$

Упрощенная схема: