Да-нет в среде Mathcad
или
Оригами Буля

В среде Mathcad есть функции (реализованные в виде префиксного и инфиксного операторов – см. правую часть рис. 1), которые возвращают только два значения: 1 или 0: Да–Нет, True (Истина) – False (Ложь). Эти функции собраны на панели инструментов Boolean(английская версия Mathcad 15 –см. левую частьрис. 1) иСравнение (русская версия MathcadPrime – см. рис. 2).

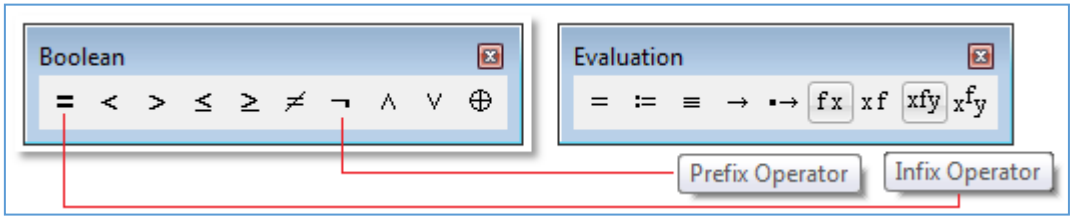


Рис.1. Панель булевых операторов в среде Mathcad 15 (слева)

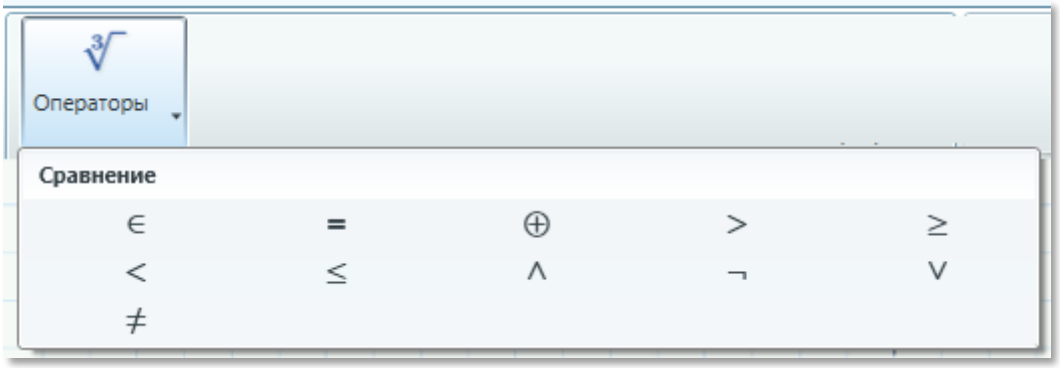


Рис.2. Панель булевых операторов в среде Mathcad Prime

Примечание.

Слово «булевый» происходит от имени Джорджа Буля — английского математика и логика, профессора математики Королевского колледжа Корка (ныне Университетский колледж Корк). Один из предтеч математической логики.

Иногда булевы функции называют *двоичными* или *логическими*. Аргументами этих функций (операндами операторов) в среде Mathcad являются вещественные числа, но при этом число, отличное от нуля, воспринимается как истина, а ноль как ложь.

В некоторых языках программирования (Pascal, например) начать работу с булевыми и другими типами переменных можно только после того, как они будут перечислены с указанием типа переменной. Это сделано для экономии памяти компьютера: под хранение булевой

переменной выделяется намного меньше места, чем под хранение целочисленной, а тем более вещественной. Сейчас память компьютера, как правило, не является лимитирующим фактором при решении задач и от такого предварительного объявления типов переменных отказались. Так, по крайней мере, делается в среде Mathcad.

Поговорим об этих специфических функциях, отталкиваясь от проблем, возникающих при работе в среде Mathcad или в иных математических программах и языках программирования, да и вообще, при использовании цифровой вычислительной техники, в основе которой лежит двоичный «атом» – элементарный элемент памяти, находящийся в одном из двух состояниях (заряжено–разряжено, намагничено–размагничено и т.д.). Из «атомов» (биты) составляются «молекулы» (байты), которые, в свою очередь, формируют новые «соединения» – переменные, массивы переменных и т.д. – все то, чем оперируют программисты. Арифметические и прочие действия над числами в переменных и массивах переменных – это по своей сути не что иное, как разнообразные булевые (побитовые) операции.

1. Азы булевой математики

1.1. Булевы функции одного аргумента

Булевых функций одного аргумента *четыре*: см. таблицу 1, но на практике работают только с одной – с f_1, которую называют *отрицанием* (*инверсией*). Остальные три функции возвращают либо свой аргумент (функция f_2), либо константы 1 (f_3) и 0 (f_4).

Таблица 1. Булевы функции одного аргумента

a	f_1	f_2	f_3	f_4
0	1	0	1	0
1	0	1	1	0
Обозначение	¬a	a	1	0
	Not(a) !a ā			

В первой строке графы Обозначение таблицы 1 (и таблицы 2 ниже) показаны операторы Mathcad, которыми булевы функции реализуется. Символ ¬ (префиксный оператор отрицания, или инверсии) расположен на панелях, показанных на рис. 1 и 2.

Примечание.

Определение *префиксный* означает, что символ оператора расположен *до* операнда. В среде Mathcad есть и *постфиксные* встроенные операторы – оператор факториала, например, (5!), где символ оператора (!) стоит *после* операнда (5). У *инфиксного* оператора два аргумента, а его символ стоит между ними. Есть в среде Mathcadтакже и *древовидный* оператор – см. рис. 8 и 11 ниже.

1.2. Булевы функции двух аргументов

Булевых функций двух аргументов шестнадцать: см. таблицу 2.

Таблица 2. Булевы функции двух аргументов

a b	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}	f_{16}
0 0	0	0	1	0	1	1	1	1	0	0	1	1	0	0	1	0
0 1	0	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0
1 0	0	1	0	1	1	0	0	1	1	0	0	1	1	0	1	0
1 1	1	1	1	0	1	1	0	0	0	0	0	0	1	1	1	0
Обозначение	∧	∨	=	⊕ ≠	≥	≤			>	<	¬a	¬b	a	b	1	0
	* × • И And & && min	+ ИЛИ Or || max	↔ ≡ ⇔ Eqv ==	<> >< Xor !=	→ ⊇ ⇒ Imp	→ ⊇ ⇒ Imp	↓ ¬And	| ¬Or								

Таблица 2 условно делится на две половины – на «именную» (f_1 – f_8) и безымянную (f_9 – f_{16}). Вот имена первых восьми функций. Вернее, семи («великолепная семерка»): две функции (f_5 и f_6) называются одинаково – импликация или логическое следование.

- f_1 – конъюнкция (логическое умножение);
- f_2 – дизъюнкция (логическое сложение);
- f_3 – равнозначность (эквивалентность, тождественность);
- f_4 – неравнозначность (неэквивалентность, разделительная дизъюнкция, сумма по модулю 2) ;
- f_5 и f_6 – импликация: f_5 – импликация (логическое следование) от a к b, f_6 – импликация (логическое следование) от b к a;
- f_7 – функция (стрелка) Пирса (функция Вебба, функция Даггера, антидизъюнкция);
- f_8 – функция (штрих) Шеффера (антиконъюнкция);

Остальные восемь функций таблицы 2 (f_9 – f_{16}, как и три последние функции таблицы 1) не имеют ни имен, ни практического применения. Это либо константы (f_{15} и f_{16}), либо функции только одного аргумента (f_{11} – f_{14}). Имя, да и то условно, можно дать только функциям f_9 и f_{10}: инверсия (отрицание) импликации. Но эти две функции, как правило, применяются только к вещественным аргументам: пример $\pi > e = 1$ (да, численное значение отношения длины окружности к ее диаметру больше, чем численное значение основания натурального логарифма).

Примечание.

В среде MathcadPrime появился еще один оператор, возвращающий нуль или единицу: ∈ – *оператор принадлежности*: сравните рис. 1 (10 операторов) и рис. 2 (11 операторов). Этот инфиксный оператор определяет, принадлежит ли его первый операнд к одному из возможных числовых множеств (второй операнд): **C** – комплексные числа, **Q** – рациональные числа (этот оператор относится к символьной математике), **R** – вещественные числа и **Z** – целые числа. На рисунке 3 показаны примеры вызова этого оператора.

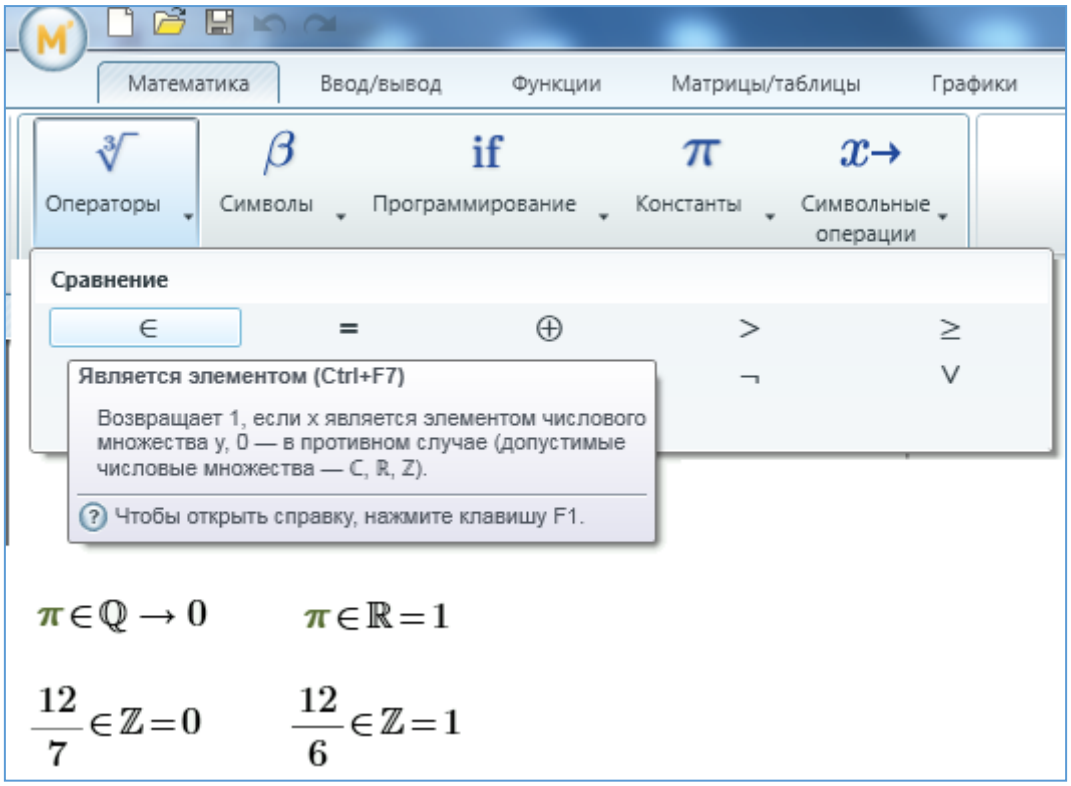


Рис. 3. Проверка принадлежности чисел определенным множествам

Из рисунка 3 видно, что π не является рациональным числом, а является иррациональным. Полезно также при программировании проверять, хранит ли та или иная переменная целое или нецелое число. А вот проверка числа на то, является ли оно комплексным, не имеет особой практической ценности, т.к. все числа в среде Mathcad относятся к множеству комплексных чисел, хотя многие инженеры считают, что если у числа нет мнимой части, то это уже не комплексное число.

На рисунке 4 показаны операторы вызова в среде Mathcad 15 некоторых булевых функций двух аргументов из таблицы 2. Эти операторы реализованы в виде инфиксных операторов – операторов, символ которого, как уже отмечено, находится между двух операндов. Стрелка над операндами-векторами на рис. 4 – это оператор векторизации, т.е. поэлементного выполнения действия над векторами.

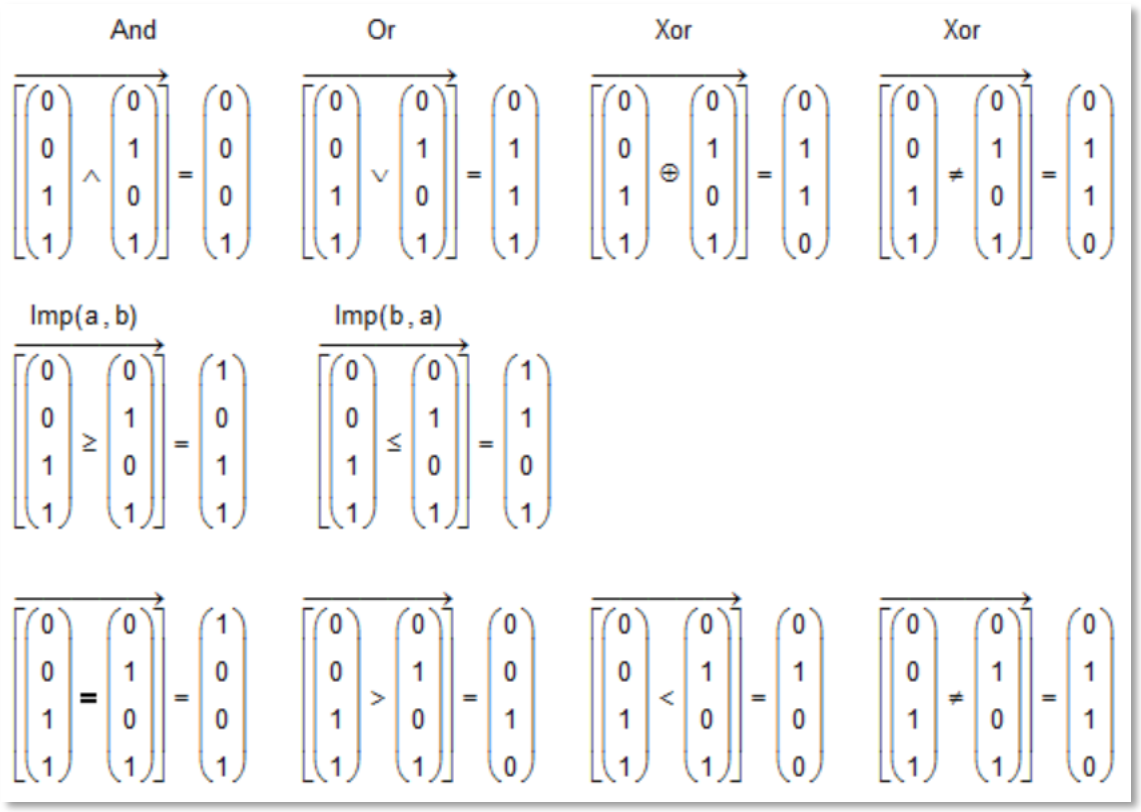


Рис. 4. Операторы вызова булевых функций в Mathcad 15

У некоторых операторов, показанных на рис. 4, по идее могут быть только булевы операнды (0 или 1), а у других – и вещественные: 0.1, 0.9, 2.5 и т.д. На рисунке 5 показана работа операторов Исключающее Или (Xor) и Не равно с вещественными операндами: не с 0 и 1, а с 0.1 и 0.9. Эти операторы при булевых операндах (рис. 4) дали одинаковые ответы, а при вещественных операндах (рис. 5) разные.

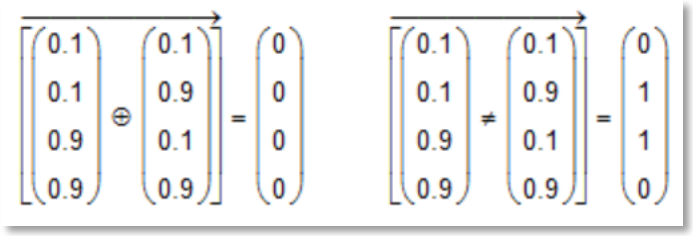


Рис. 5. Оператор "Исключающее или" и "Не равно" с вещественными аргументами

Ответ первого оператора на рис. 5 (одни нули) объясняется тем, что числа, отличные от нуля, булевыми функциями пакета Mathcad воспринимаются как единицы (Истина). Это связано с тем, что в ранних версиях Mathcad не было операторов And и Or, и данные булевы операции реализовывались через арифметические операторы умножения и сложения – см. рис. 6. При сложении же двух единиц (оператор Or) получается двойка (истина) – число, отличное от нуля.

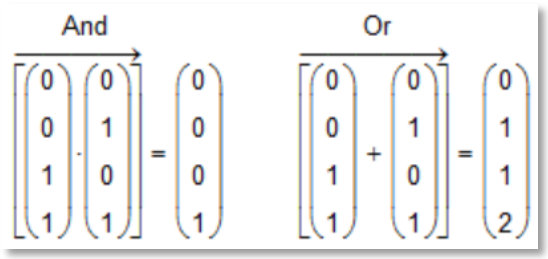


Рис. 6. Логическое умножение и логическое сложение в Mathcad ранних версий

Булевы функции And и Or часто иллюстрируют электрической цепью: последовательное соединение выключателей (устройств, находящихся в двух возможных положениях 0 и 1) – это конъюнкция (логическое умножение), а параллельное – дизъюнкция (логическое сложение). На рис. 7 показан также менее тривиальный пример – электрический аналог эквиваленции (Eqv): схема соединения двух выключателей, так чтобы свет независимо зажигался и тушился из двух разных мест. Такие выключатели будут полезны в длинном коридоре или на лестнице многоэтажного дома.

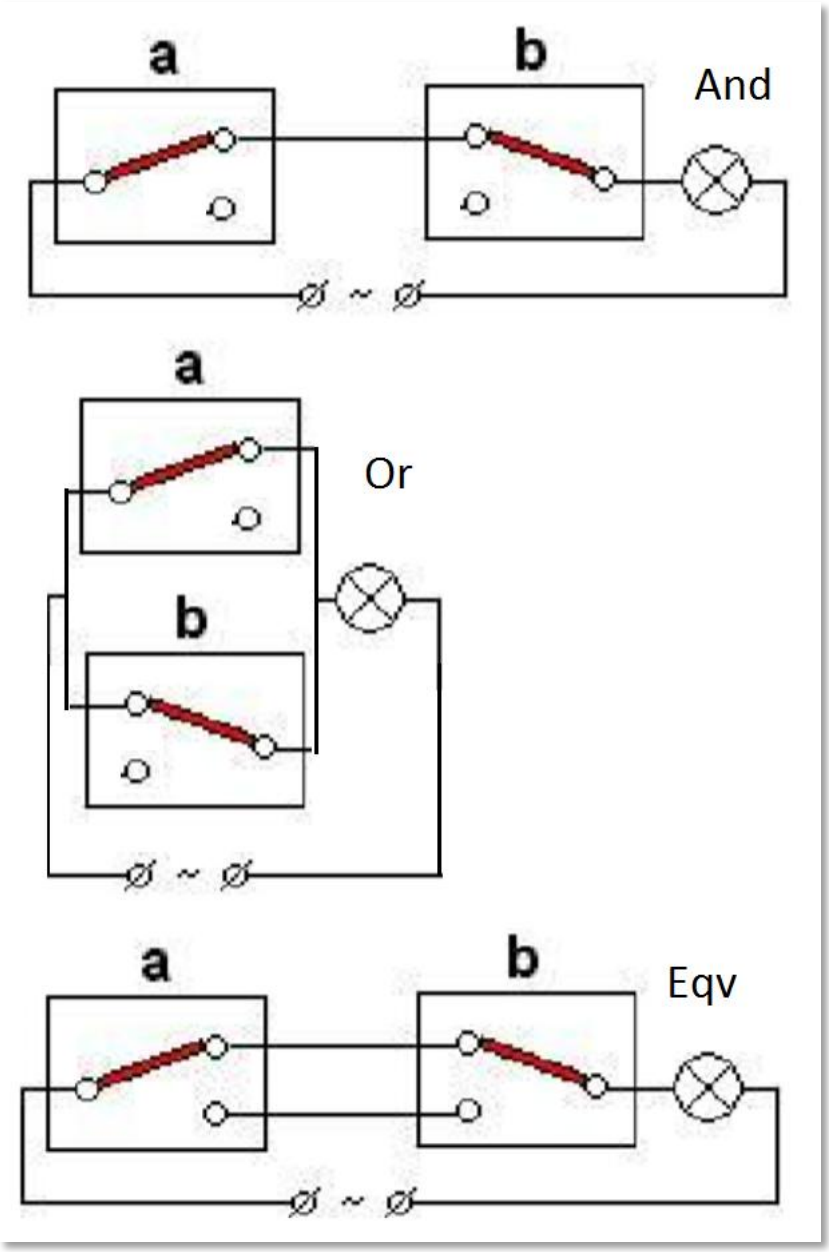


Рис. 7. Электрические аналоги трех булевых функцийAnd, Orи Eqv

В таблицах 1 и 2 сделана попытка собрать все имена функций и символы операторов, использующихся для реализации булевой алгебры. Список, конечно, неполный. Можно расширить его примерами из других языков программирования (Pascal, Fortran и др.) и математических программ (Maple, MatLab, Mathematica и др.).

Можно отметить *избыточность* функции, собранных в таблицах 1 и 2. В конкретных языках программирования есть некий ограниченный набор встроенных булевых функций и операторов. Вот перечень таких функций и операторов, встроенных в популярные программные среды:

- язык программирования BASIC: Not, And, Or, Xor и Imp
- язык программирования C: !,& (логическое умножение), && (побитовая конъюнкция), !=, ‖и = =
- математическая программа Mathcad: ¬, ^, ˅ и ⊕ (см. рис. 1 и 2)

Недостающие двоичные функции (операторы) программист может ввести в программу сам через механизм пользовательских функций, и мы это покажем ниже.

Но разделение двоичных функций и операторов на основные (базисные) и вспомогательные появилось задолго до компьютеров и узаконилось в виде двоичных алгебр (в скобках отмечен их базис):

- алгебра логики (¬, &, ˅, → и ↔);
- булева алгебра (¬, & и ˅);
- алгебра Жегалкина (&, ˅ и ⊕);
- алгебра Пирса (↓);
- алгебра Шеффера (|) .

Две последние двоичные алгебры примечательны тем, что в их базисе всего лишь одна двоичная функция↓ или |, опираясь на которую можно построить все остальные.

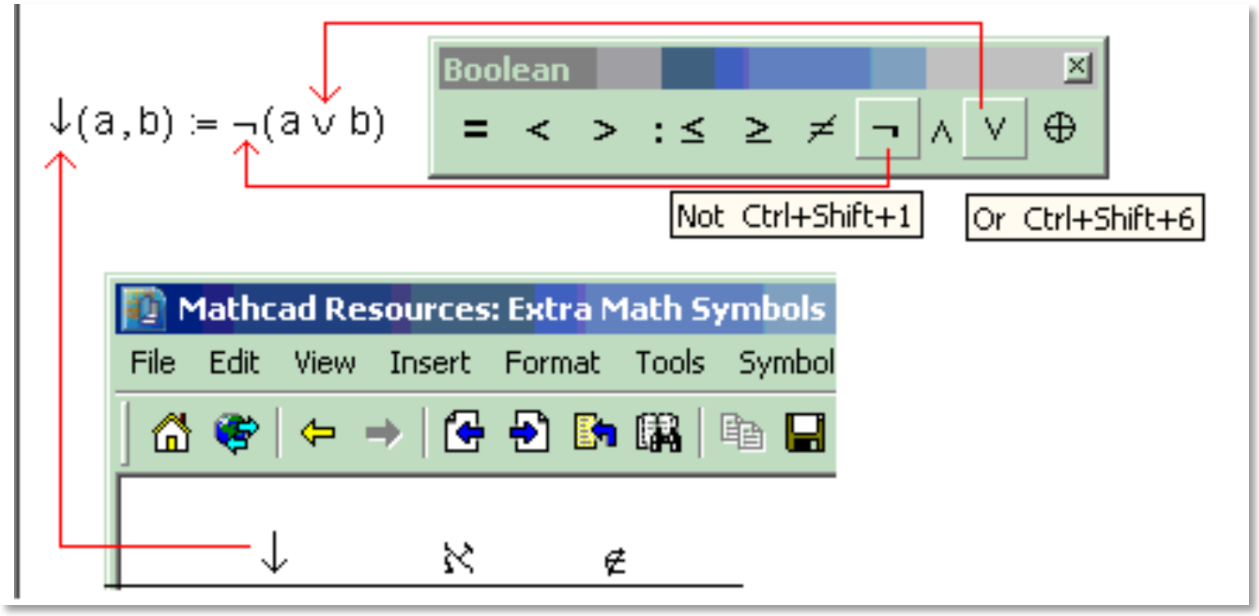


Рис. 8. Создание в среде Mathcad булевой функции Пирса с опорой на отрицание и логическое сложение

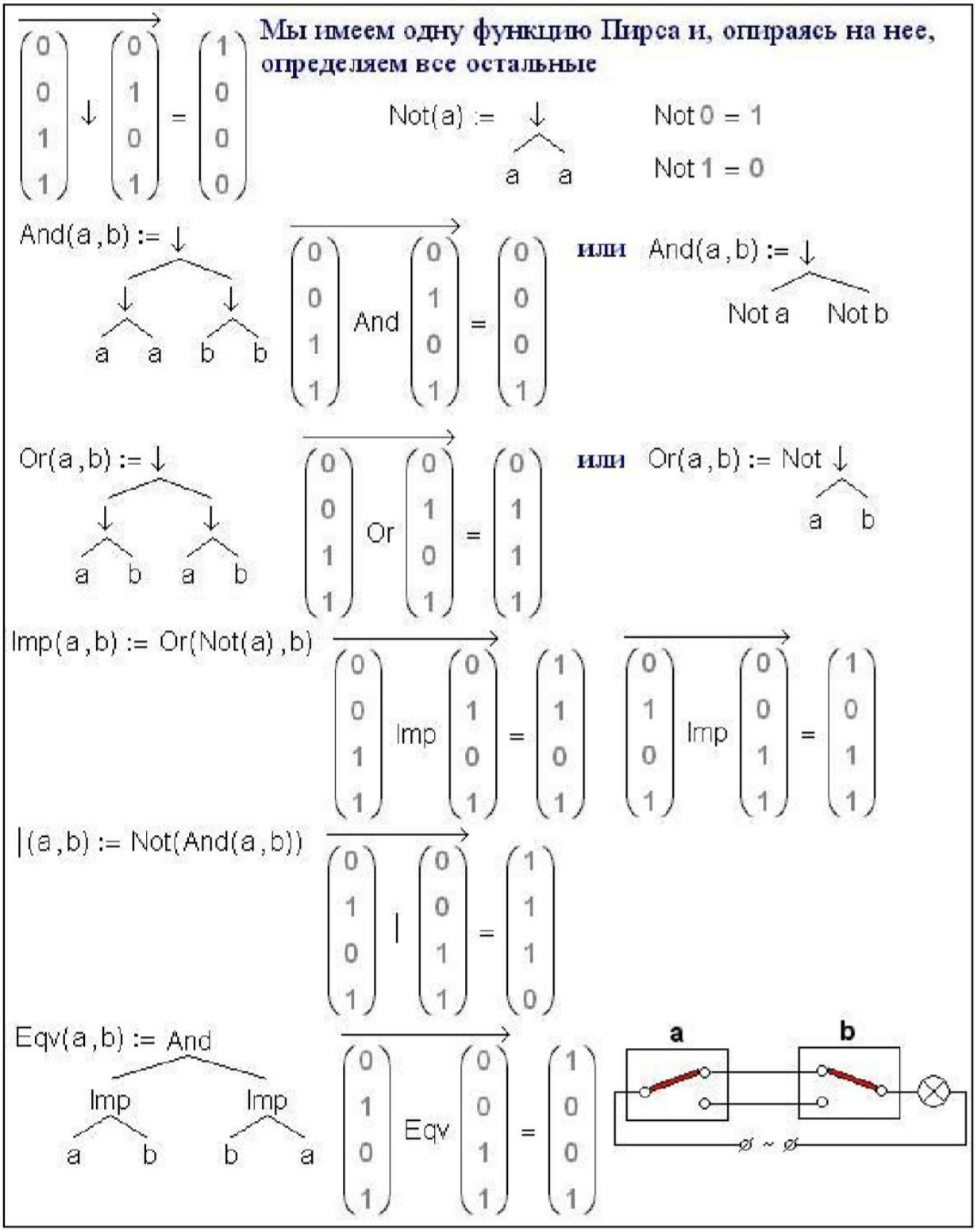


Рис. 9. Построение булевых функций с опорой на функцию Пирса

На рисунке 9 показан документ Mathcad 15, где с опорой на функцию Пирса (↓ - см. рис 8) построены другие булевы функции: одна функция одного аргумента (отрицание, инверсия – Not) и пять функций двух аргументов: And, Or, Imp, штрих Шеффера (|) и Eqv. Последние три функции (Imp, штрих Шеффера и Eqv) определены с использованием ранее определенных функций. Это сделано для большей компактности рисунка, но от механизма вложения пользовательских функций (Imp(a, b) := Or(Not(a), b), например, можно отказаться и оперировать «для чистоты эксперимента» только функцией (штрихом) Пирса.

На алгебру Пирса и алгебру Шеффера возлагали большие надежды в смысле построения цифрового компьютера из однотипных элементов. Потом от этой идеи отказались по ряду причин, главная из которых в том, что любой компьютер и так состоит только из однотипных элементов – из транзисторов, объединенных в интегральные микросхемы (чипы).

Можно отметить *недостаточность* набора математических инструментов, отображенных в таблицах 1 и 2. Возьмем, например, наверное, самую популярную функцию булевой алгебры – конъюнкцию (следствие ее популярности и в том, что у нее больше всего имен и символов для обозначения – см. столбец f_1 в таблице 2). Столбец истинности конъюнкции в таблице истинности (а так называют таблицы 1 и 2.) по идее должен быть такой, как показано в таблице 3.

Таблица 3. Уточненная конъюнкция

a b	a And b (f_1)
0 –	0
0 –	0
1 0	0
1 1	1

Прочерк рядом с нулем в первом столбце таблицы 3 означает то, что если первый (a) аргумент равен нулю, то незачем проверять, чему равен второй аргумент (b), и наоборот. Такое предусматривают, создавая некоторые («неленивые») языки программирования – язык C, например. При программировании в среде языка BASIC условный переход по конъюнкции можно записать так:

If a And b Then… (1-й способ)

но лучше так:

If a Then If b Then… или If b Then If a Then… (2-й способ)

Второй способ записи позволяет не только ускорять расчеты, но и избегать некоторых ошибок – логическое выражение b может иметь смысл только в том случае, если на альтернативный вопрос a дан положительный ответ. Вот типичный пример такой «программистской» ситуации:

If I> 0 Then If V(i) >V(i-1) Then…

Можно сказать, что в языке BASIC есть две конъюнкции: And и Then If.

В таблицах 1 и 2 мы, повторяем, собрали булевы функции одного (таблица 1) и двух (таблица 2) аргументов. Но, возвращаясь к конъюнкции, можно сказать эта функция имеет не два, а… полтора аргумента – см. таблицу 3.

Такую же нецелочисленность количества аргументов можно отметить и по другим булевым функциям:

Таблица 4. Булевы функции полутора, одного и нуля аргументов

a b	a Or b (f_2)
0 0	0
0 1	1
1 –	1
1 –	1

a b	a (f_{11})	¬a (f_{13})
0 –	0	1
0 –	0	1
1 –	1	0
1 –	1	0

a b	b (f_{12})	¬b (f_{14})
– 0	1	0
– 1	1	1
– 0	0	0
– 1	0	1

a b	1 (f_{15})	0 (f_{16})
– –	1	0
– –	1	0
– –	1	0
– –	1	0

Можно отметить, что в таблицу 2 попали операторы, изначально предназначенные для работы не с двоичными, а с вещественными операндами: «>», «<», «≥», «≤», «=» и «≠». Но если принять во внимание тот факт, что нуль и единица входят во множество вещественных чисел, то включение этих операторов в таблицу 2 вполне закономерно. В этом ряду («>», «<», «≥», «≤», «=» и «≠») также можно отметить и избыточность и недостаточность. С избыточностью все более-менее ясно («больше», например, – это инверсия от оператора «меньше или равно» и т.д.). Недостаточность же можно наблюдать в том, например, что при работе с вещественными переменными вместо оператора «равно» более уместно использовать оператор «примерно равно», которого нет в списках встроенных. Можно также вспомнить о существовании понятий «намного больше» или «намного меньше». Эти операторы соотношения также возвращают двоичные значения, но имеют фактически уже не два, а три аргумента (операнда): сравниваемую пару вещественных чисел и некое контекстное наше представление о том, что такое «примерно» или «намного».

На рис. 10 показано создание в среде Mathcad 15 булевого оператора Примерно равно.

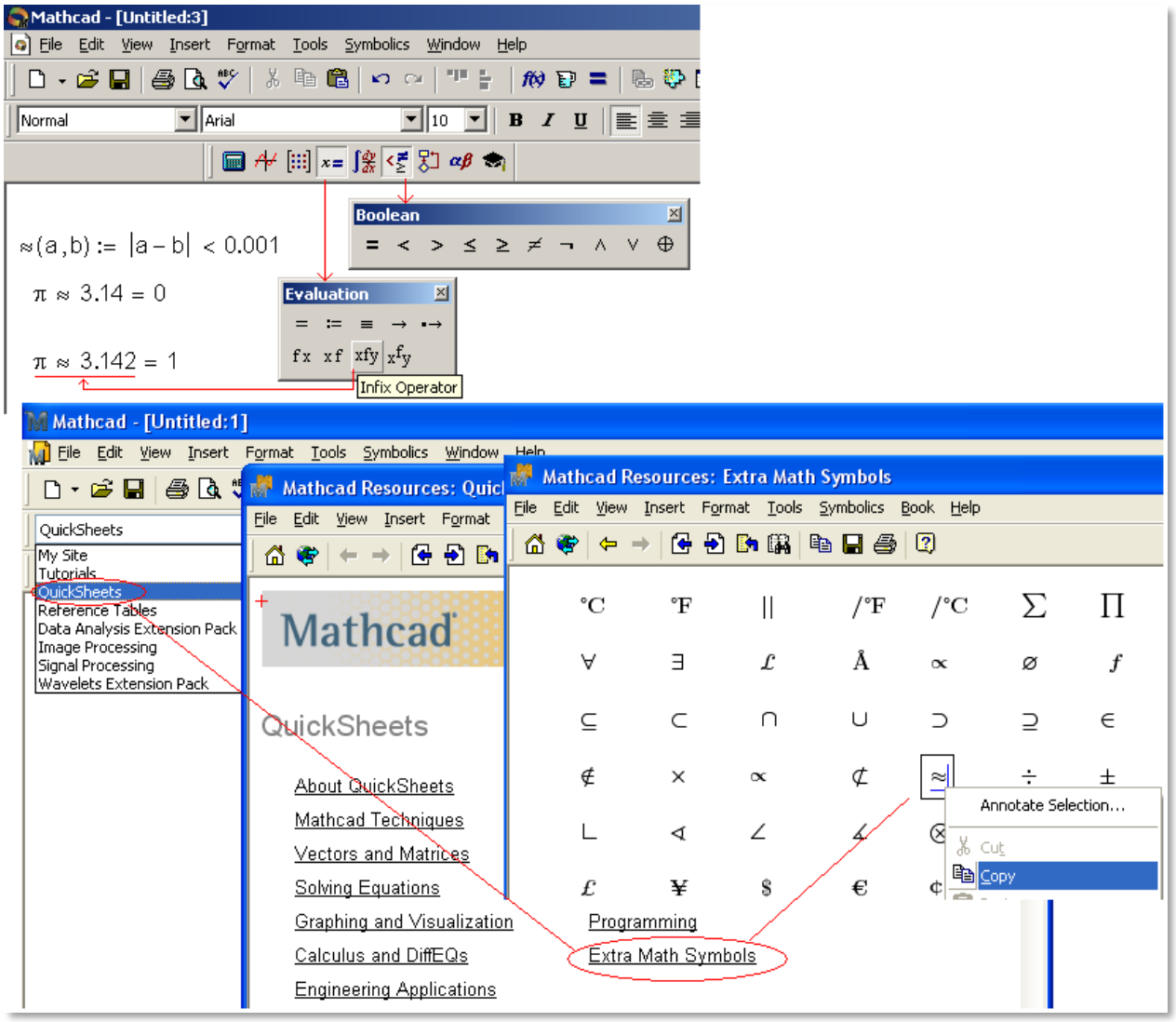


Рис. 10. Создание в среде Mathcad 15 булевого оператора **Примерно равно**

Если говорить не о классической булевой алгебре, а о реальной практике программирования, то следует признать, что переменные, фигурирующие в таблицах 1 и 2, могут принимать не два (0 или 1), а три значения: 0, 1 и неопределенно. Эту особенность мы уже зафиксировали в таблицах 3 и 4, где вместо конкретных значений аргументов (0 или 1) стоит прочерк. В языках программирования есть инструменты, обработки таких «прочерков» в таблицах истинности. Если аргумент булевой функции не определен, то расчет может, либо прерываться сообщением об ошибке, либо идти по третьему пути.

Аргументы булевых функций могут принимать не два и не три, а… бесконечное множество вещественных значений. Это множество делится на две предельно неравные части: на нуль и на ненуль (¬0, если говорить языком таблицы 1 – на числа, отличные от нуля, которые, повторяем, булевыми функциями пакета Mathcadвоспринимаются как единицы). Бывает и так, что булева функция возвращает не только нули и единицы. Вот, например, как работает функция Or в одной из реализаций языка BASIC (язык BASIC, которым комплектовалась популярная в свое время ПЭВМ «Искра-226» - http://www.ic.kz/~ksxi/musei/int6.htm калька машины «Wang-2000»): см. таблицу 5.

Таблица 5. «Дизъюнктивная конъюнкция»

a	b	a Or b
0	0	0
0	¬0	1
¬0	0	1
¬0	¬0	2

Можно допустить и такую работу расширенного оператора Or: см. таблицу 6.

Таблица 6. Расширенная конъюнкция (дизъюнкция)

a	b	a Or b
0	0	0
0	¬0	1
¬0	0	2
¬0	¬0	3

Одно дело, когда первый аргумент (операнд) не равно нулю, а другое – когда второй, и третье – когда оба одновременно.

Подытоживая наш разбор таблиц 1 и 2, можно сказать, что описываемые булевы функции в реальных компьютерных реализациях могут иметь недвоичные аргументы и возвращать опять же недвоичные результаты. Но особого недвоичного смысла в этом нет. Просто, повторяем, по технологическим причинам вещественные переменные в описываемых реализациях языков программирования (BASIC, Mathcad) «по совместительству» исполняют роль двоичных. При этом

булевы функции воспринимают свои вещественные аргументы «двоично»: нуль есть нуль («Нет», «False»), а все остальное единица («Да», «True»).

Эта, можно сказать, «категоричность» описываемых встроенных функций вступает в противоречие с положениями *теории нечетких множеств* (fuzzy sets) и теории нечеткой логики (fuzzy logic [3, 4]). Необходимо, например, статистически обрабатывать на компьютере не только «черно-белые» (двоичные) ответы анкетируемых типа «Да (1)» – «Нет (0)», но и «цветные» (вещественные) ответы: «Да (1)», «Скорее да, чем нет (0.75, например)», «Ни да, ни нет (0.5)», «Скорее нет, чем да (0.25, например)» и «Нет (0)». Если говорить не о статистике, а об электротехнике и вернуться к электрическим цепям, которыми часто иллюстрируют работу булевых функций (см. рис. 7), то можно упомянуть тот факт, что сейчас в быту получают распространение выключатели, плавно меняющие накал ламп от 100% до нуля. Еще раньше такие устройства стали применять в театрах и кинозалах. Медики уверяют, что плавный переход от света к темноте через полумрак не портит зрение. (В кинотеатрах свет тушат плавно, конечно, не по медицинским соображениям, а по другим причинам – если резко погасить свет, то может возникнуть паника.)

Можно привести множество других примеров, толкающих к тому, что аргументы функций, перечисленных в таблицах 1 и 2, могут и должны быть не только двоичными, но и вещественными числами от нуля до единицы. И функции, перечисленные в таблицах 1 и 2, должны возвращать вещественные значения, плавно меняющиеся от нуля до единицы. Вот как, например, можно задать «плавную» функцию отрицания: Not(a) := 1-a (см. рис. 11).

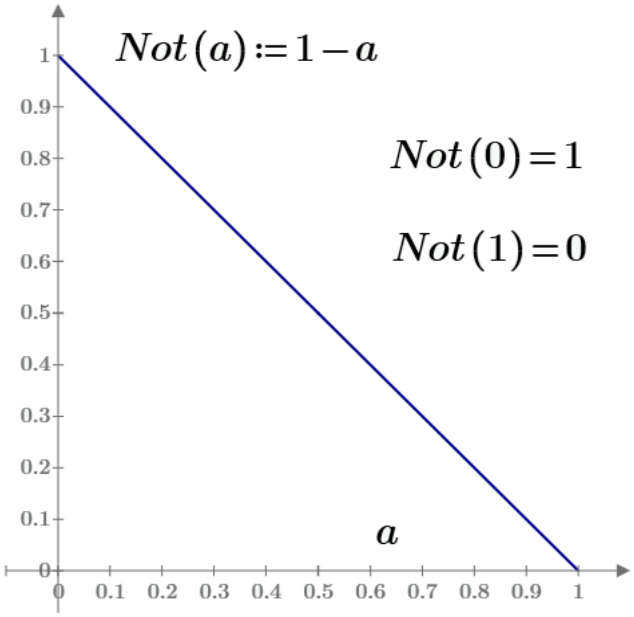


Рис. 11. Нечеткое отрицание

«Плавная» конъюнкции (And) и «плавная» дизъюнкция (Or) получаются сами собой, если вспомнить о том, что одно из обозначений конъюнкции – это min (минимум – см. столбец f_1 в таблице 2), а одно из обозначений дизъюнкции – это max (максимум – см. столбец f_2 в таблице 2):

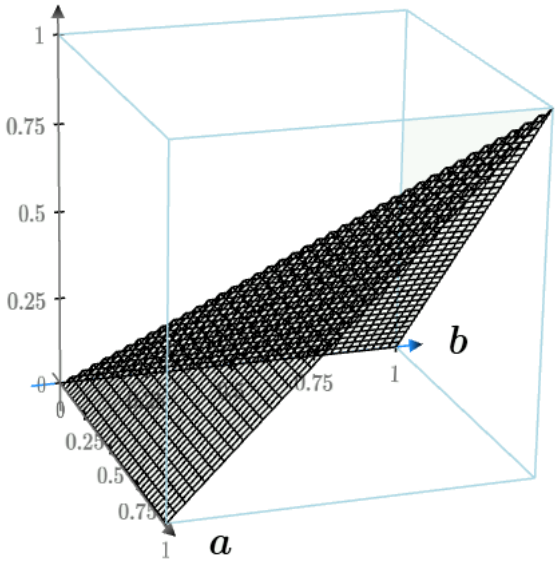
And(a, b) := min(a, b) Or(a, b) := min(a, b)

Примечание.

Некоторым пользователям Mathcad трудно запомнить, что ^ – это логическое умножение, а ˅ – логическое сложение. Поэтому не будет лишним ввести в расчет пользовательские функции **And** и **Or** даже в том случае, если мы работаем только с четкой логикой.

На рисунках 12-15 показано графическое отображение нечетких булевых функций **And**, **Or** и **Eqv**. Последняя функция представлена в двух вариантах **Eqv** (рис. 14) и **Eqv1** (рис. 15).

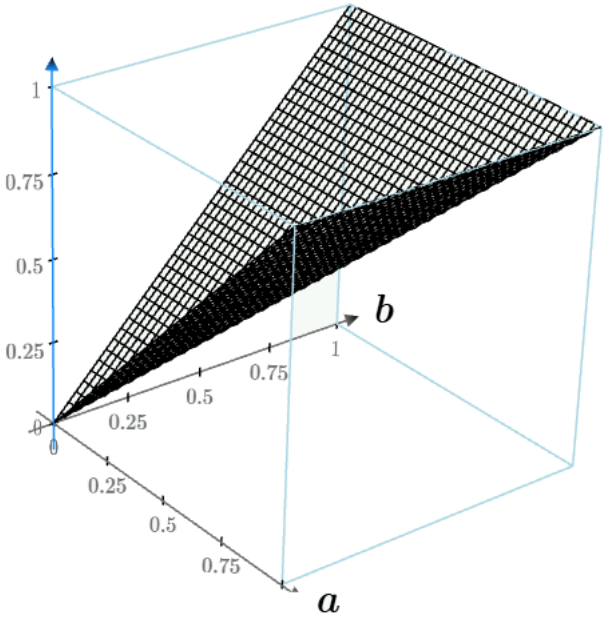
$$And(a,b) := min(a,b)$$



$$\overrightarrow{And\left(\begin{bmatrix}0\\0\\1\\1\end{bmatrix},\begin{bmatrix}0\\1\\0\\1\end{bmatrix}\right)} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

Рис. 12. График нечеткого **And**

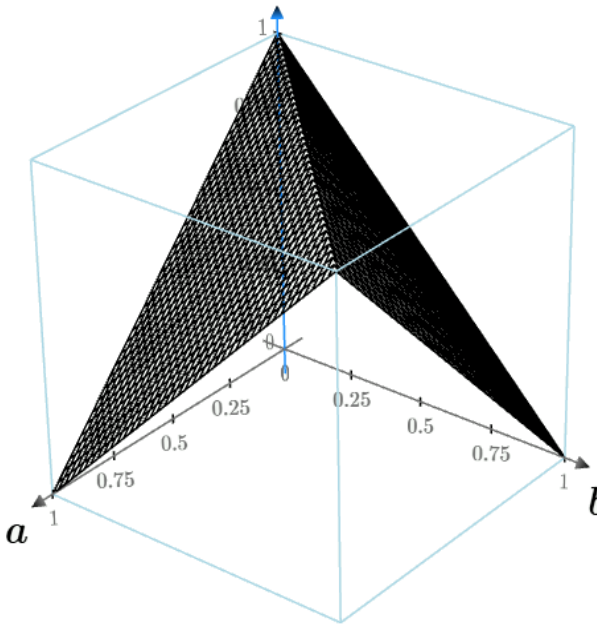
$$Or(a,b) := \max(a,b)$$



$$\overrightarrow{Or\left(\begin{bmatrix}0\\0\\1\\1\end{bmatrix},\begin{bmatrix}0\\1\\0\\1\end{bmatrix}\right)} = \begin{bmatrix}0\\1\\1\\1\end{bmatrix}$$

Рис. 13. График нечеткого **Or**

$$Eqv(a,b) := 1 - |a-b|$$



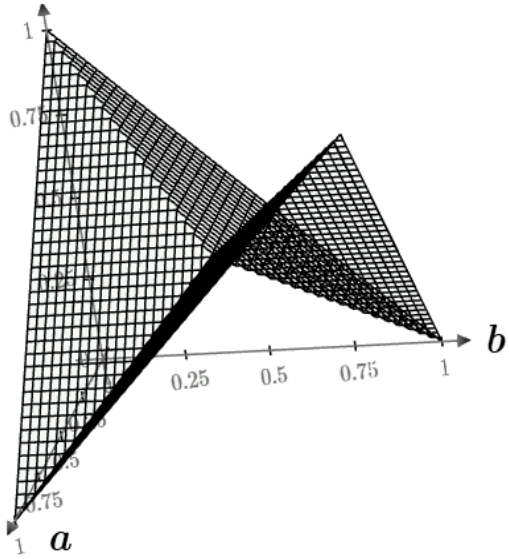
$$\overrightarrow{Eqv\left(\begin{bmatrix}0\\0\\1\\1\end{bmatrix},\begin{bmatrix}0\\1\\0\\1\end{bmatrix}\right)} = \begin{bmatrix}1\\0\\0\\1\end{bmatrix}$$

$$Eqv(0.5,0.5) = 1$$

$$Eqv(0.4,0.6) = 0.8$$

Рис. 14. График нечеткого **Eqv**

$$Eqv1(a,b) := min(\max(Not(a),b),\max(a,Not(b)))$$



$$\overrightarrow{Eqv1\left(\begin{bmatrix}0\\0\\1\\1\end{bmatrix},\begin{bmatrix}0\\1\\0\\1\end{bmatrix}\right)} = \begin{bmatrix}1\\0\\0\\1\end{bmatrix}$$

$$Eqv1(0.5,0.5) = 0.5$$

$$Eqv1(0.4,0.6) = 0.4$$

Рис. 15. График нечеткого**Eqv** (второй вариант)

На рисунке 16 показано создание нечетких булевых функции и их отображение на графиках линий одного уровня (на контурных графиках).

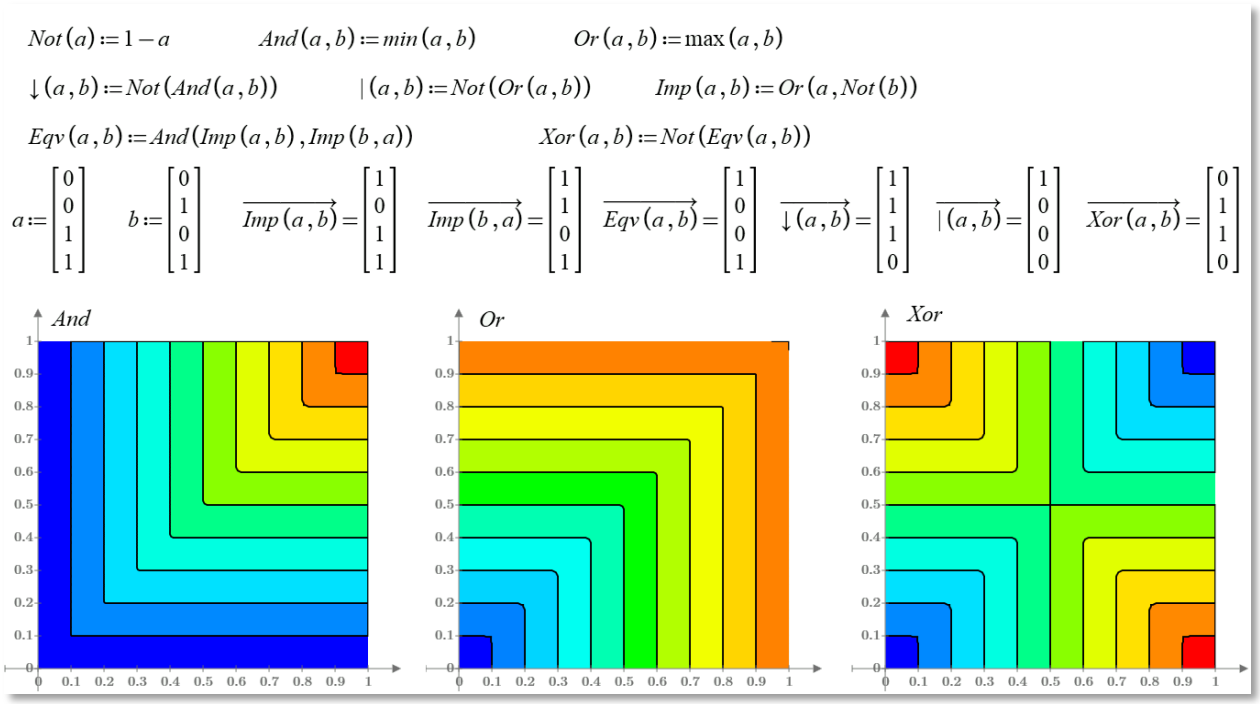


Рис. 16. Нечеткие булевы функции

На рисунках 12-15, а также на рисунке 18 ниже показаны, если так можно выразиться, «булевы кубики» или «булевы оригами».

1.3. Функции многих аргументов

Булевых функций трех аргументов 256, а четырех…Читатель, определи это сам!

Функции And и Or, если их отожествлять с функциями min и max (см. выше), могут иметь переменное число аргументов – с четкими (0 или 1) или нечеткими (от 0 до 1) значениями. Этим свойством обладают и некоторые другие булевы функции. Какие? Читатель, определи опять же это сам!

На рисунках 17 и 18 показано формирование в среде Mathcad нечеткой функции трех аргументов, возвращающей… решение жюри присяжных, которые могут выдавать уже не «черно-белые» ответы (виновен – невиновен), а… «цветные»: виновен на 30%, невиновен на 70% и т.д. В электрическом аналоге машинки для голосования выключатели (0 или 1) заменены на реостаты (от 0 до 1).

Печальное примечание.

Говорят, что в США электрический стул приводят в смертельное действие несколько человек. При этом настоящий рубильник приводится в действие только одним человеком. Остальные участники этой экзекуции включают фальшивые рубильники. И никто не знает, где фальшивый, а где настоящий рубильник. Такая несколько ханжеская процедура дает возможность каждому такому палачу думать, что не он, а кто-то другой был причиной смерти человека. Если же все настоящие рубильники заменить на реостаты (см. ниже рис. 17), плавно меняющими напряжение, то смерть можно заменить на нелетальное наказание: приговоренный преступник получит удар током (сильный или слабый), но останется жив.

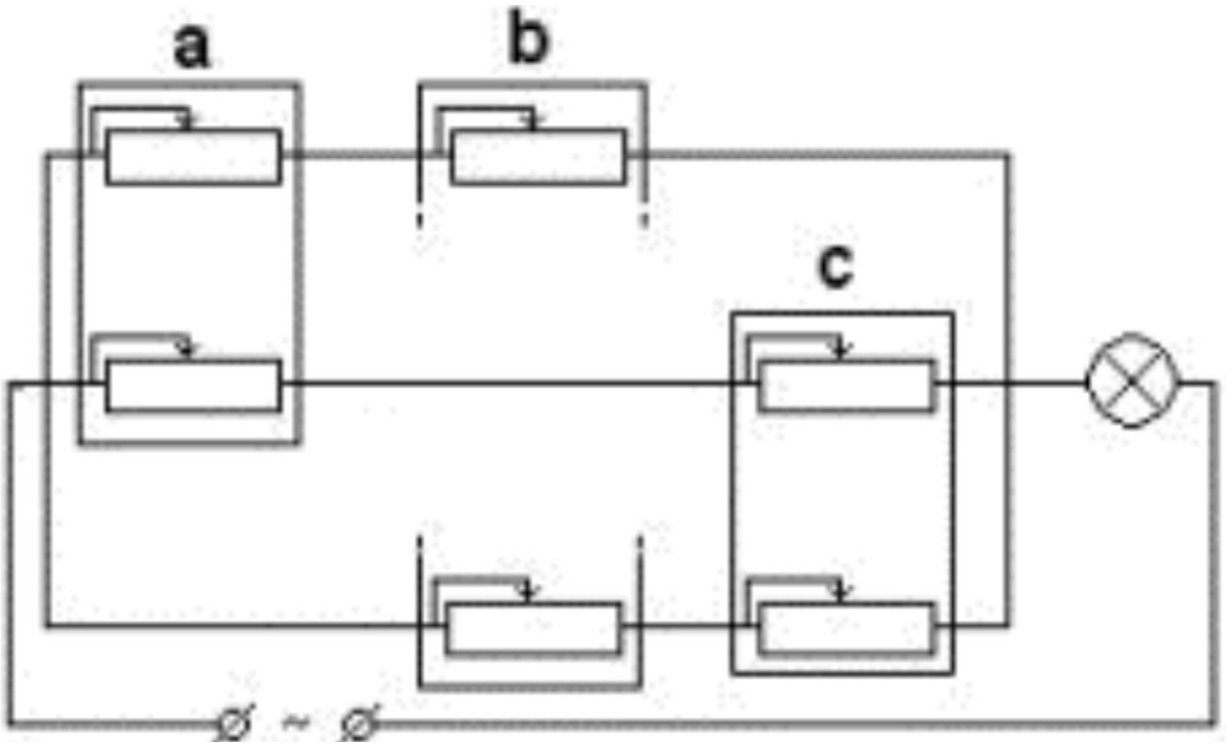


Рис. 17. Машина для голосования: параллельное (Or) соединение последовательно (And) соединенных выключателей (реостатов)

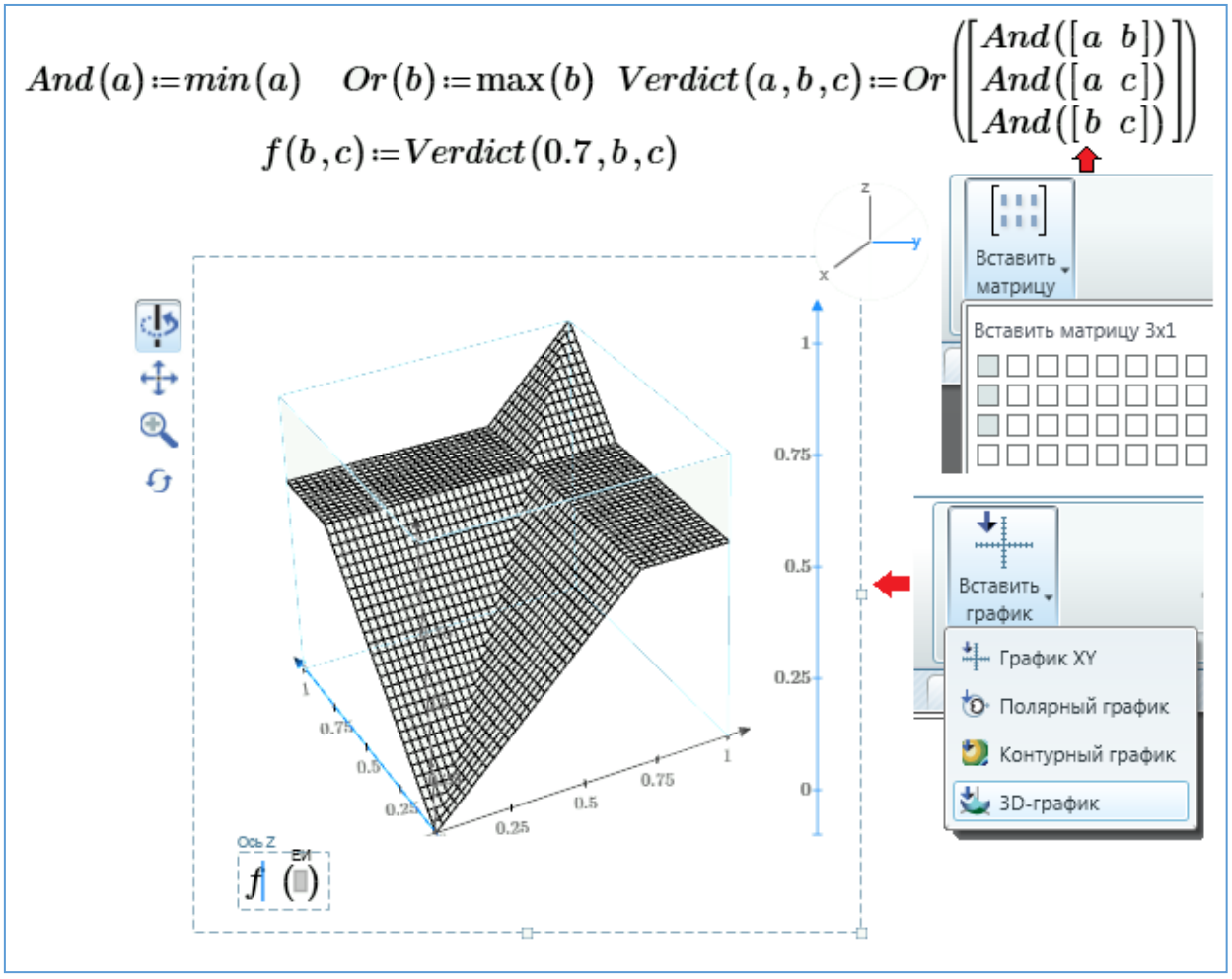


Рис. 18. Функция нечеткого голосование трех членов жюри

Функция Verdict, показанная на рис. 18, при двоичных аргументах возвращает двоичный ответ, а при вещественных – вещественный, естественно. Показан соответствующий «булев кубик» (оригами) при a = 0.7 – мы видим гибрид конъюнкции с дизъюнкцией: нечеткое (от 0 до 1) мнение одного члена жюри плавно переводит вердикт из области Or (см. рис. 13) в область And (см. рис. 12).

Задание для читателей: создайте функцию, подобную той, какая показана на рис. 18, но не для трех, а для любого числа членов жюри присяжных, выносящих свой вердикт нечетко – вещественным числом от 0 до 1.

Литература:

1. Есипов А.С. Логические основы построения и работы компьютеров. // Компьютерные инструменты в образовании (http://www.aec.neva.ru:8081/journal). №. 2000

2. Очков В.Ф. Принцип неопределенности программирования. // КомпьютерПресс. 7’1996 (http://twt.mpei.ac.ru/ochkov/IZBYT.htm)

3. Очков В.Ф. Mathcad и нечеткие множества. // КомпьютерПресс. № 1. 1998 (http://twt.mpei.ac.ru/ochkov/F_sets.htm)

4. Очков В.Ф. Mathcad и нечеткая логика// КомпьютерПресс. № 8. 1998 (http://twt.mpei.ac.ru/ochkov/F_log.htm)