Программный и аналитический способы построения и заполнения таблиц истинности логических функций

В. С. Попов, email: popov_vlad@mail.ru [1]
П. А. Леляев, email: petrleliaev@gmail.com [2]

[1] Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)
[2] ГАУ «Центр цифровизации образования»

Аннотация. *В данной работе рассматриваются методы построения таблиц истинности логических функций с использованием языков программирования высокого уровня, а также на основе логических умозаключений, и особенности использования языков программирования при решении подобных задач.*

Ключевые слова: *Логическая функция, булева функция, логическая операция, логическая формула, таблица истинности, булева алгебра, алгебра высказываний, алгебра логики, двоичная логика, логическая переменная, полный перебор значений логических переменных.*

Введение

При работе с логическими функциями одним из способов задания логической функции является таблица истинности [1], задающая соответствие выходных значений логической функции комбинациям её входных логических переменных (табл. 1).

Таблица 1

Общий вид таблицы истинности логической функции F от n переменных

x1	**…**	**xn-1**	**xn**	**F(x1, …, xn)**
0	…	0	0	F(0, …, 0, 0)
0		0	1	F(0, …, 0, 1)
0		1	0	F(0, …, 1, 0)
…		…	…	…
1	…	1	1	F(1, …, 1, 1)

Для логической функции F от n переменных имеется 2^n строк таблицы истинности, содержащих различные комбинации значений логических переменных x_1, …, x_n.

В данной статье рассматривается способ построения таблиц истинности логических функций с использованием компьютерной программы на языке программирования, а также пример аналитического заполнения таблицы истинности.

Пример логической функции для построения таблицы истинности

В качестве примера логической функции для построения её таблицы истинности рассмотрим задание № 2 демонстрационной версии контрольно-измерительных материалов ЕГЭ по информатике 2023 года [2]:

«Миша заполнял таблицу истинности логической функции F, но успел заполнить лишь фрагмент из трёх различных её строк, даже не указав, какому столбцу таблицы соответствует каждая из переменных w, *x, y, z.*

$$\neg(y \to x) \vee (z \to w) \vee \neg z \qquad (1)$$

Определите, какому столбцу таблицы соответствует каждая из переменных *w, x, y, z* (табл. 2).

Таблица 2

Частично заполненная таблица истинности

				F
	0			0
0	1			0
1			0	0

В ответе напишите буквы *w, x, y, z* в том порядке, в котором идут соответствующие им столбцы (сначала буква, соответствующая первому столбцу; затем буква, соответствующая второму столбцу, и т.д.). Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно».

2. Построение таблицы истинности логической функции переборным алгоритмом

Пример программы для построения таблицы истинности логической функции приведён в Листинге 1, вывод данной программы приведён в Листинге 2. Данная программа использует вложенные циклы for для получения всех комбинаций значений логических переменных x, y, z, w. Комбинации значений переменных x, y, z, w перебираются в порядке счёта в двоичной системе счисления: (0, 0, 0, 0), (0, 0, 0, 1), (0, 0, 1, 0), (0, 0, 1, 1), (0, 1, 0, 0), …, (1, 1, 1, 1). Используемый порядок логических переменных (например, x, y, z, w) должен соблюдаться при каждом выводе и переборе логических значений в циклах. Для построения таблицы истинности логической функции (1) при заданном

значении логической функции в цикл максимальной вложенности можно включить условный оператор if.

Листинг

Программа на языке Python для построения таблицы истинности логической функции F при любых значениях F и при F=0

```
# Построение полной таблицы истинности F
print('x y z w F')
for x in 0, 1:
    for y in 0, 1:
        for z in 0, 1:
            for w in 0, 1:
                F = not(y <= x) or (z <= w) or not z
                print(x, y, z, w, F)

# Построение таблицы истинности при F=0
print('x y z w F')
for x in 0, 1:
    for y in 0, 1:
        for z in 0, 1:
            for w in 0, 1:
                F = not(y <= x) or (z <= w) or not z
                if F == 0:
                    print(x, y, z, w, F)
```

Листинг 2

Вывод программы при F=0

```
x y z w F
0 0 1 0 False
1 0 1 0 False
1 1 1 0 False
```

Перебор в каждом из циклов for осуществляется путём перебора элементов кортежа 0, 1. Также возможны следующие способы записи, приведённые в Листинге 3.

Листинг 3

Другие способы перебора логических значений в цикле

```
for x in 0, 1:
for x in [0, 1]:
for x in range(2):
for x in [False, True]:
```

Соотнеся вывод программы и данную в задании частично заполненную таблицу истинности, получим ответ yxzw.

При использовании логических значений False, True вместо целочисленных значений 0, 1 для логических переменных x, y, z, w при

их выводе будет удобным приведение логического типа к целочисленному: print(int(x), int(y), int(z), int(w), F).

Условие «F == 0» может быть записано в виде «not F», условие «F == 1» может быть записано в виде «F» (if F:).

3. Операторы языков программирования для сокращения записи логических операций

В большинстве языков программирования присутствуют логические операторы «логическое не» (not), «логическое и» (and), «логическое или» (or), а также побитовые операторы «побитовое не», «побитовое и», «побитовое или», «побитовое исключающее или». Запись данных операций в различных языках программирования приведена в табл. 3.

Таблица 3

Операторы логических и побитовых операций в различных языках программирования

Операция	**Python**	**C++**	**Pascal**
Логическое не	not	!	not
Логическое и	and	&&	and
Логическое или	or	||	or
Побитовое не	~	~	not
Побитовое и	&	&	and
Побитовое или	|	|	or
Побитовое исключающее или	^	^	xor

В то же время логические операции импликации, эквиваленции, исключающего или могут быть записаны в виде дизъюнктивных или конъюнктивных нормальных форм (ДНФ и КНФ соответственно) [3], примеры записи которых приведены в табл. 4.

Таблица 4

Дизъюнктивные и конъюнктивные нормальные формы для некоторых логических операций

Операция	**ДНФ**	**КНФ**
Импликация $x \rightarrow y$	$\neg x \vee y$	$\neg x \vee y$
Эквиваленция $x \equiv y$	$x \wedge y \vee \neg x \wedge \neg y$	$(\neg x \vee y) \wedge (x \vee \neg y)$

Окончание таблицы 4

Исключающее или $x \oplus y$	$\neg x \wedge y \vee x \wedge \neg y$	$(\neg x \vee \neg y) \wedge (x \vee y)$

Запись ДНФ и КНФ на языках программирования может быть громоздкой, а учащиеся могут испытывать сложности с пониманием и запоминанием нормальных форм логических функций. Для короткой записи логических операторов рекомендуется использовать операторы языков программирования, приведённые в [4]. Некоторые часто используемые логические операторы и их возможная запись на языке программирования в виде операторов сравнения приведены в табл. 5.

Таблица 5

Соответствие некоторых логических операторов и операторов языков программирования

Название логической операции	**Обозначение логической операции**	**Обозначение соответствующей операции в языке программирования**
Отрицание, логическое не	$\neg$	not
Конъюнкция, логическое и	$\wedge$	and
Дизъюнкция, логическое или	$\vee$	or
Импликация	$\rightarrow$	<=
Эквиваленция	$\equiv$	==
Исключающее или	$\oplus$	!=

Операция импликации $\rightarrow$ может быть представлена в виде оператора сравнения <= по причине соответствия таблицы истинности логической операции $x \rightarrow y$ и результатов операции сравнения x <= y (см. табл. 6): единственная строка таблицы истинности, в которой результат импликации $\rightarrow$ и сравнения <= окажется ложен, – строка со значениями переменных x = 1 и y = 0, в остальных строках таблицы истинности результат операций $\rightarrow$ и <= равен 1.

Таблица 6

Соответствие таблицы истинности импликации и операции сравнения «меньше или равно»

x	**y**	**$x \rightarrow y$**	**x <= y**
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

Аналогичным образом доказывается соответствие эквиваленции ≡ операции сравнения ==, а исключающего или ⊕ операции «не равно» !=.

4. Приоритет логических операций при записи выражений на языке программирования

В соответствии с приоритетом операций в языках программирования [5], операции сравнения (например, <=, ==, !=) имеют больший приоритет, чем логические операции not, and, or. Однако в алгебре логики приоритет логических операций отличается, что создаёт проблемы при применении операторов сравнения в качестве замены логических операторов импликации, эквиваленции, исключающего или. Приоритет рассматриваемых логических операторов в порядке от большего приоритета к меньшему в алгебре логики: отрицание, конъюнкция, дизъюнкция и исключающее или (одинаковый приоритет), импликация, эквиваленция. При записи логических формул с использованием как логических операторов, так и операторов сравнения на языке программирования следует использовать скобки для операндов операторов сравнения. Например, неправильная и верная запись формулы ¬x → (y ≡ ¬z) выглядят следующим образом соответственно: not x <= (y == not z) и (not x) <= (y == (not z)). Первый вариант записи приведёт к возникновению ошибки в языке программирования Python: «SyntaxError: bad input». Также отсутствие правильно расставленных скобок для указания приоритетов логических операций приводит к построению неверной таблицы истинности, например, следующая запись формулы ¬x → (y ≡ ¬z) будет тоже неверной из-за отсутствия скобок для левого операнда импликации: not x <= (y == (not z)).

5. Примеры записей логических формул на языке программирования

Примеры записи некоторых логических формул в языке программирования Python показаны в табл. 7.

Таблица 7

Примеры записи логических формул на языке программирования

Логическая формула	**Запись в Python**
¬x → (y ≡ ¬z)	(not x) <= (y == (not z))
¬(¬x ⊕ y) → (z ≡ ¬w)	(not((not x) != y)) <= (z == (not w))
x → (¬y → ¬(¬z → w))	x <= ((not y) <= (not((not z) <= w)))

6. Построение таблиц истинности без использования программирования

Некоторые задачи с условием, аналогичным приведённому в п. 1, не имеют простого решения с использованием переборных алгоритмов. В качестве примера рассмотрим следующую задачу:

«Дана частично заполненная таблица истинности функции

$$((x \to y) \equiv (y \to z)) \wedge (w \vee y) \qquad (2)$$

Определите, какому столбцу соответствует каждая переменная из набора w, x, y, z (табл. 8).

Таблица 8

Частично заполненная таблица истинности

				F
		0	0	1
0	0	0		1
0				1

В ответе запишите буквы w, x, y, z в том порядке, в котором идут соответствующие им столбцы (сначала буква, соответствующая первому столбцу, затем буква, соответствующая второму столбцу, и т.д.). Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.»

Если решать такую задачу методом, предложенным в п. 2, легко установить, что данное логическое выражение (2) имеет истинное значение при 6 из 16 наборов логических переменных. Дальше есть возможность выбрать из них 3 и расставить переменные в нужном порядке, но это может оказаться трудоёмкой задачей (напомним, что время на ЕГЭ по информатике ограничено 3 часами 55 минутами, а кроме данной задачи, есть ещё 26). Поэтому рассмотрим теоретический способ решения — к слову, до 2021 года ЕГЭ по информатике проходил в письменной форме и такие задачи предполагали именно письменное аналитическое решение.

Заметим, что в последнюю очередь в данном выражении (2) выполняется конъюнкция, следовательно, левая и правая части конъюнкции в каждой рассматриваемой строке приведённой частично заполненной таблицы истинности должны быть истинными (равняться 1). Таким образом, в каждой из этих строк хотя бы одна из переменных w и y должна равняться 1, из чего следует, что в 4 столбце таблицы должна находиться одна из них. Это не может быть y, поскольку в таком случае для второй из рассматриваемых строк таблицы истинности слева от конъюнкции получится ложное значение 0. Итак, в 4 столбце данной

частично заполненной таблицы истинности находится переменная w, и её значение во второй строке равно 1 (табл. 9).

Таблица 9

Частично заполненная таблица истинности

			w	**F**
		0	0	1
0	0	0	1	1
0				1

Далее обратимся к первой предложенной строке. Отметим, что если из двух пропущенных ячеек единица только в одной, то она должна соответствовать переменной *w* – это противоречие, так как *w* уже находится в 4 столбце. Таким образом, единиц в первой строке две, и одна из них соответствует *y*. В таком случае другая должна соответствовать z, чтобы выполнялась эквиваленция в левой скобке (2). Методом исключения получаем, что третий столбец соответствует переменной x (табл. 10).

Таблица 10

Частично заполненная таблица истинности

		x	**w**	**F**
1	1	0	0	1
0	0	0	1	1
0				1

Наконец, рассмотрим последнюю строку. Одна из переменных *z* и *y* в ней равна 0. Предположим, что это *z*. В таком случае для истинности левой скобки значение переменной y также должно равняться 0. Это же справедливо и для переменной *x*. Переменная w должна равняться 1 для истинности правой скобки. Таким образом, мы получили строку, совпадающую со второй, что противоречит условию задачи – таблица содержит неповторяющиеся строки. Поэтому первому столбцу соответствует переменная y, и ответ данной задачи – *yzxw*. Значения переменных в третьей строке можно не находить, поскольку по условию задачи это не требуется (табл. 11).

Таблица 11

Частично заполненная таблица истинности

y	**z**	**x**	**w**	**F**
1	1	0	0	1
0	0	0	1	1
0				1

При подобных решениях следует помнить, что если предположение не приводит к противоречию, это ещё не означает, что оно верное, и стараться выдвигать предположения, позволяющие прийти к противоречию, исключающему варианты заполнения таблицы.

7. Проверка таблицы истинности и полученного ответа

Для проверки заполненной таблицы истинности и полученного ответа можно последовательно подставить значения логических переменных для каждой строки таблицы истинности в заданную логическую функцию F (1) или (2), в результате вычисления которой вычисленные значения F должны совпадать с заданными в частично заполненной таблице истинности. Данную проверку также можно выполнить аналитическим или программным способом. Например, для первой строки таблицы истинности задания из п.6: $((0 \rightarrow 1) \equiv (1 \rightarrow 1)) \wedge (0 \vee 1)$ или на языке программирования print(((0 <= 1) == (1 <= 1)) and (0 or 1)).

Заключение

В статье рассмотрены вопросы применения переборных алгоритмов и операторов сравнения языков программирования для построения таблиц истинности логических функций, включающих операции импликации, эквиваленции, исключающего или. Приведены операторы логических и побитовых операций языков программирования, соответствие логических операций и операций языков программирования, примеры записи логических формул на языке программирования. Рассмотрена проблема несоответствия приоритета логических операций и операций языков программирования при использовании операций сравнения в качестве замены логических операций импликации, эквиваленции, исключающего или. Также рассмотрен теоретический алгоритм заполнения таблиц истинности, содержащих пропущенные значения.

Список литературы

1. Новиков, Ф. А. Дискретная математика для программистов: Учебник для вузов / Ф. А. Новиков. – 3-е изд. – СПб: Питер, 2009. – 384 с.
2. Демонстрационный вариант контрольных измерительных материалов единого государственного экзамена 2023 года по информатике // ФГБНУ Федеральный институт педагогических измерений. – М.: 2022. – 11 с.
3. Белоусов, А. И., Ткачёв, С. Б. Дискретная математика : учебник для втузов / Белоусов А. И., Ткачёв С. Б.; ред. Зарубин В. С., Крищенко А. П. – 4-е изд., испр. – М.: Изд-во МГТУ им. Н. Э. Баумана, 2006. – 743 с.
4. Popov, V. S. Equivalence of logical operations and other operations in Python programming language [Manuscript submitted for publication]. Informatics and Control Systems, BMSTU.
5. Expressions – Python 3.11.1 documentation [Электронный ресурс]: Интернет-сайт – Режим доступа: https://docs.python.org/3/reference/expressions.html#operator-precedence