УДК 510.6, 519.7

Научная статья

Математическая логика: построение логических схем из логических элементов в Maple

А. А. Оленев, К. А. Киричек, Е. В. Потехина

Ставропольский государственный педагогический институт, 355029,

Ставропольский край, г. Ставрополь, ул. Ленина, д. 417А, Россия

E-mail: olenevalexandr@gmail.com, kirichekka@mail.ru, ekapotekhina@yandex.ru

В статье рассматриваются возможности применения библиотеки Logic системы компьютерной алгебры Maple в аспекте компьютерного моделирования логических схем в различных базисах. Смоделированы основные логические элементы в Maple. На конкретном примере детально представлен алгоритм построения логической схемы в различных базисах.

Ключевые слова: булева функция, самодвойственность, основной базис, универсальные базисы, логические элементы.

DOI: 10.26117/2079-6641-2021-36-3-155-164

Поступила в редакцию: 03.09.2021

В окончательном варианте: 03.10.2021

Для цитирования. Оленев А. А., Киричек К. А., Потехина Е. В. Математическая логика: построение логических схем из логических элементов в Maple // *Вестник КРАУНЦ. Физ.-мат. науки.* 2021. Т. 36. № 3. С. 155-164. DOI: 10.26117/2079-6641-2021-36-3-155-164

Контент публикуется на условиях лицензии Creative Commons Attribution 4.0 International (<https://creativecommons.org/licenses/by/4.0/deed.ru>)

© Оленев А. А., Киричек К. А., Потехина Е. В., 2021

Введение

С технической и технологической точек зрения использование языка двоичных чисел (нулей и единиц), то есть булевой алгебры, удобно для хранения и обработки числовой, а также символьной информации вычислительными средствами.

Известно, что всякую булеву функцию от n аргументов можно выразить через конъюнкцию, дизъюнкцию и отрицание, которые составляют так называемый основной базис с технологической точки зрения [1]-[2]. В целом такой подход означает, что при наличии устройств, реализующих конъюнкцию, дизъюнкцию и отрицание, можно составить некую функциональную схему, которая реализует любую заданную булеву функцию. Такие устройства являются базовыми логическими элементами или функциональными элементами и, как сказано выше, образуют основной логический базис.

Финансирование. Исследование выполнялось без финансовой поддержки фондов.

Система компьютерной алгебры (СКА, CAS) Maple в настоящее время является незаменимым помощником для проведения эвристических экспериментов в демонстрации (наглядном представлении)[3]-[5], а также построения и анализа логических схем из функциональных элементов [6]-[7]. В данной статье с использованием результатов работ [8]-[11] рассмотрен порядок моделирования функциональных схем в основном и универсальном базисах, а также описано проведение соответствующих компьютерных экспериментов. Каждый логический элемент смоделирован нами как отдельная процедура.

1. Библиотека базовых логических элементов

1.1. Моделирование элементов основного базиса

Элементами для создания цифровых узлов при использовании основного базиса являются: инвертор, элементы "И" и "ИЛИ".

Воспользуемся СКА Maple для моделирования логических элементов и построения из них в дальнейшем функциональных схем. Для преобразования функциональной схемы в выражение используем операторы библиотеки Logic СКА Maple. При построении моделей основных логических элементов, кроме команд библиотеки Logic, необходимо использовать общие команды для создания процедур.

1.2. Инвертор

Этот элемент должен обладать следующими функциональными характеристиками:

элемент "НЕ"(инвертор), реализующий отрицание (инверсию), имеет всегда один вход и один выход; сигнал (true или 1) на выходе появляется, когда на входе нет сигнала (присутствует значение – false или 0), и не появляется (на входе значение – false или 0), когда на вход подан сигнал (true или 1).

Таким образом, процедура для инвертора должна возвращать логическое отрицание своего входа.

Ниже представлена процедура математической модели инвертора:

```
> restart;  
> with(Logic); # библиотеки Logic  
> элемент_НЕ := proc(a) # начало процедуры  
> return &not(a); # выполнение отрицания входного значения  
end proc; # завершение процедуры
```

1.3. Элемент "И"

Этот элемент должен обладать следующими функциональными характеристиками:

элемент "И" реализует конъюнкцию, имеет не менее двух входов и один выход; сигнал на выходе (true или 1) появляется тогда и только тогда, когда на все входы поданы сигналы. Процедура для элемента "И":

```
> элемент_И := proc(a::seq(anything)) # начало процедуры  
if nops([a]) < 2 then
```

```

error "Элемент И должен иметь не менее двух входов.";
end if; # проверка наличия как минимум двух входных значений
&and(op([a])); # выполнение операции конъюнкция для входных значений
end proc; # завершение процедуры

```

1.4. Элемент "ИЛИ"

Этот элемент должен обладать следующими функциональными характеристиками:

элемент "ИЛИ" реализует дизъюнкцию, имеет не менее двух входов и один выход; сигнал (true или 1) появляется на выходе тогда и только тогда, когда сигнал (true или 1) подан хотя бы на один из входов.

Процедура для элемента "ИЛИ":

```

> элемент_ИЛИ := proc(a::seq(anything)) # начало процедуры
if pops([a]) < 2 then error "Элемент ИЛИ должен иметь не менее двух входов.";
end if; # проверка наличия как минимум двух входных значений
&or(op([a])); # выполнение операции дизъюнкция для входных значений
end proc; # завершение процедуры

```

Приведем пример использования разработанных процедур.

Если x , y и z являются входными данными для логического элемента "И" необходимо ввести следующее:

```

> K2:=элемент_И (x,y,z); # моделирование элемента "И" на три входа

```

На что будет получен ответ от СКА Maple:

```

K2:=(x &and y)&and z

```

Если результат от этого логического элемента "И" отправить на инвертор:

```

> K3:=элемент_НЕ(K2); # моделирование элемента отрицания "НЕ" элемента «И»
на три входа

```

то получим результат:

```

K3:=&not((x &and y)&and z)

```

который затем соединим с логическим элементом "ИЛИ" и одним входом x , получим:

```

> элемент_ИЛИ (K3,x); # моделирование элемента "ИЛИ" на два входа, одним из
входов является элемент K3 (И-НЕ)

```

```

&not((x &and y)&and z)&or x

```

Таким образом, предлагаемые три отдельные процедуры позволяют осуществить построение функциональной схемы в основном базисе, то есть в базисе, состоящем из элементов "И" "ИЛИ" "НЕ".

2. Построение функциональной схемы в основном базисе

Рассмотрим процесс моделирования функциональной схемы, представленной на рисунке 1, с использованием разработанных выше процедур для базовых логических элементов.

Воспользуемся процедурами для нахождения логического выражения для схемы, приведенной на рис. 1.

Рис. 1. Схема для исследования в нормальной дизъюнктивной форме

Шаг 1. Схему будем формировать слева направо. Реализуем первоначально два инвертора с входными параметрами a и c соответственно, присвоим им имена $K1$ и $K2$:

```
> K1:=элемент_НЕ(a); # моделирование первого элемента "НЕ"
&not(a)
> K2:=элемент_НЕ(c); # моделирование второго элемента "НЕ"
&not(c)
```

Реализуем элемент "И" на два входа, также находящийся на первой вертикальной линейке:

```
> K13:=элемент_И (a,b); # моделирование элемента "И" на два входа
a &and b
```

Шаг 2. Реализуем функциональный элемент "И" ($K21$) на три входа и его входными значениями будут являться выходы элементов $K11$, $K12$ и b :

```
> K13:=элемент_И (a,b); # моделирование элемента "И" на два входа
K13:=(&not(a)&and b)&and &not(c)
```

Шаг 3. На завершающем этапе реализуем функциональный элемент "ИЛИ" ($K3$) на два входа, где входами будут являться выходы соответственно реализованных элементов $K21$ и $K13$:

```
> K3:=элемент_ИЛИ(K21,K13); # моделирование элемента "ИЛИ" на два входа
```

Результатом будет логическое выражение:

```
K3:=((&not(a)&and b)&and &not(c))&or(a &and b)
```

3. Построение функциональной схемы в универсальных базисах

Известно, что можно произвести построение функциональной схемы простейшего цифрового устройства, реализующего все возможные булевы функции, с использованием логических элементов лишь одного типа [3], [7]. Это так называемые универсальные базисы, которые носят название базис Штрих Шеффера ("И-НЕ") и Стрелка Пирса ("ИЛИ-НЕ").

3.1. Элемент "И-НЕ"

Этот элемент должен обладать следующими функциональными характеристиками:

элемент "И-НЕ" реализует отрицание конъюнкции, имеет не менее двух входов и один выход; сигнал на выходе (false или 0) появляется тогда, и только тогда, когда на все входы поданы сигналы (true или 1).

Процедура для элемента "И-НЕ":

```
>элемент_И_НЕ := proc(a::seq(anything)) # начало процедуры
if pops([a]) < 2 then
  error "Элемент И-НЕ должен иметь не менее двух входов."; # проверка наличия
  как минимум двух входных значений
end if; &nand(op([a])); # выполнение операции "И-НЕ" для входных значений
end proc; # конец процедуры
```

Реализация отрицания:

```
>K1:=элемент_И_НЕ (x,x);
```

```
K1:=x &nand x
```

Проверим эквивалентность представления, для чего воспользуемся функцией библиотеки Logic:

```
>ШШ_Инвертор:=элемент_И_НЕ (x,x); # моделирование элемента "НЕ" в базисе
штрих Шеффера
```

```
ШШ_Инвертор:=x &nand x
```

```
Equivalent(ШШ_Инвертор,&(x)); # проверка эквивалентности
```

```
true
```

При реализации описания логической схемы в универсальном базисе Штрих Шеффера необходимо выполнять следующее правило:

1. Представлению подлежит только дизъюнктивная нормальная форма (ДНФ, дизъюнкция элементарных конъюнкций).

2. Если элементарная конъюнкция входит в ДНФ как функция одного аргумента, необходимо взять его инверсию (отрицание).

3. Если дизъюнктивная нормальная форма представлена одной элементарной конъюнкцией, то необходимо взять её инверсию (отрицание).

Используя вышеприведенные правила, опишем функциональную схему, представленную на рисунке 1, с использованием универсального базиса Штрих Шеффера.

Шаг 1. Схему будем формировать слева направо. Рассматриваемая функциональная схема описывается булевой функцией, представленной в дизъюнктивной нормальной форме, то есть к данной схеме могут быть применены выше приведенные правила. Реализуем первоначально два инвертора с входными параметрами а и с, используя процедуру для построения элемента "И-НЕ" соответственно и присвоим им имена ШШ1 и ШШ2:

```
>ШШ_11:=элемент_И_НЕ(a,a); # моделирование первого элемента "НЕ" с использованием
элементов «И-НЕ»
```

```
ШШ_11:=a &nand a
```

Реализуем элемент И на два входа, также находящийся на первой вертикальной линейке:

```
>ШШ_12:=элемент_И_НЕ(c,c); # моделирование второго элемента "НЕ" с использованием
элементов «И-НЕ»
```

```
ШШ_12:=c &nand c
```

Шаг 2. Реализуем функциональный элемент "И" на три входа путем представления (замены) на элемент "И-НЕ"(ШШ2) на три входа, его входными значениями будут являться выходы элементов ШШ_11, ШШ_12 и b:

```
>ШШ_2:=элемент_И_НЕ (a,b); # моделирование элемента "И" на два входа
```

```
ШШ_2:=((a &nand a)&nand b)&nand(c &nand c)
```

Шаг 3. На завершающем этапе реализуем функциональный элемент "ИЛИ" путем замены на элемент "И-НЕ"(ШШ3) на два входа, где входами будут являться выходы соответственно реализованных элементов ШШ_2 и ШШ_13:

```
>ШШ_3:=элемент_И_НЕ(ШШ_2,ШШ_13); # моделирование элемента «И-НЕ» на два входа
```

Таким образом, булево выражение, представленное в универсальном базисе штрих Шеффера, описывающее функциональную схему, представленную на рисунке 1, будет иметь вид:

```
ШШ_3:=((a &nand a)&nand b)&nand(c &nand c)&nand(a &nand b)
```

Сравним его с выражением, полученным ранее:

```
>Equivalent(ШШ&1,K3); # проверка эквивалентности
```

```
true
```

Из этого следует вывод, что созданные процедуры можно использовать для описания и анализа функциональных схем представленных с помощью базовых логических элементов "И-НЕ".

3.2. Элемент "ИЛИ-НЕ"

Этот элемент должен обладать следующими функциональными характеристиками:

элемент "ИЛИ-НЕ" реализует отрицание дизъюнкции, имеет не менее двух входов и один выход; сигнал на выходе (true или 1) появляется тогда, и только тогда, когда на все входы поданы сигналы (false или 0).

Процедура для элемента "ИЛИ-НЕ":

```
> элемент_ИЛИ_НЕ := proc(a::seq(anything)) # начало процедуры
```

```
if nops([a]) < 2 then
```

```
error "Элемент ИЛИ-НЕ должен иметь не менее двух входов.";
```

```
end if; # проверка наличия как минимум двух входных значений
```

```
&nor(op([a])); # выполнение операции ИЛИ-НЕ для входных значений
```

```
end proc: # конец процедуры
```

Реализация отрицания:

```
>S1:=элемент_ИЛИ_НЕ(y,y); # моделирование элемента "ИЛИ" на два входа
```

```
S_1:=y &nor y
```

Проверим эквивалентность представления, для чего воспользуемся функцией библиотеки Logic:

```
>СП_инвертор:=элемент_ИЛИ_НЕ(y,y); # моделирование элемента "НЕ" с использованием элемента «ИЛИ-НЕ»
```

```
СП_инвертор:=y &nor y
```

```
>Equivalent(СП_инвертор,&not(y)); # проверка эквивалентности
```

```
true
```

При реализации описания логической схемы в универсальном базисе стрелка Пирса необходимо выполнять следующее правила:

1. Представлению подлежит только конъюнктивная нормальная форма (КНФ, конъюнкция элементарных дизъюнкций).

2. Если элементарная дизъюнкция входит в КНФ как функция одного аргумента, необходимо взять его инверсию (отрицание).

3. Если конъюнктивная нормальная форма представлена одной элементарной дизъюнкцией, то необходимо взять её инверсию (отрицание).

Используя вышеприведенные правила, опишем функциональную схему, представленную на рис. 2, с использованием универсального базиса стрелка Пирса.

Рис. 2. Схема для исследования в нормальной конъюнктивной форме

Шаг 1. Схему будем формировать слева направо. Представленная функциональная схема описывается булевой функцией, представленной в дизъюнктивной нормальной форме, то есть к данной схеме могут быть применены выше приведенные правила.

Функциональная схема, представленная на рисунке 2, описанная как конъюнктивная нормальная форма будет содержать один инвертор, два базовых логических элемента "ИЛИ" на два входа и один базовый элемент "И" на два входа. Исходя из этого, произведем описание инвертора с входным параметром s , используя процедуру для построения элемента "ИЛИ-НЕ" соответственно, и присвоим ему имя СП_12, входы - x_1 .

> СП_12:=элемент_ИЛИ_НЕ($x[1]$, $x[1]$); # моделирование элемента "НЕ" через элемент "ИЛИ-НЕ"

СП_12:= x_1 & not x_1

Реализуем два элемента "ИЛИ" на два входа, путем представления (замены) на элемент "ИЛИ-НЕ" (СП_11 и СП_2) на 2 входа, и их входными значениями будут являться для СП_21 - (x_1 и x_2).

> СП_11:=элемент_ИЛИ_НЕ($x[1]$, $x[2]$); # моделирование элемента первого "ИЛИ" на два входа

СП_11:= x_1 & not x_2

и соответственно для СП_2 – (x_3 и СП_12)

>СП_2:=элемент_ИЛИ_НЕ(СП_12, $x[3]$); # моделирование второго элемента "ИЛИ"на два входа

СП_11:=(x_1 ¬ x_1)¬ x_3

Шаг 2. На завершающем этапе реализуем функциональный элемент И путем замены на элемент "ИЛИ-НЕ"(СП_3) на два входа, где входами будут являться выходы соответственно реализованных элементов СП_11 и СП_2:

>СП_3:=элемент_ИЛИ_НЕ(СП_11,СП_2); # моделирование элемента "И"на два входа

Таким образом, булево выражение, представленное в универсальном базисе стрелка Пирса, описывающее функциональную схему, представленную на рисунке 2, будет иметь вид:

СП_3:=(x_1 ¬ x_2)¬((x_1 ¬ x_1)¬ x_3)

Логическое выражение описывающее функциональную схему, представленное на рисунке 2 будет следующим:

> G:= ($x[1]$ &or $x[2]$) &and ($x[3]$ &or (¬ $x[1]$)); # общее логическое выражение для рисунка 2 в конъюнктивной форме

G:=(x_1 &or x_2)&and(x_3 &or ¬ x_1)

Сравним его с выражением, полученным при описании схемы, представленной в универсальном базисе стрелка Пирса:

> Equivalent(СП_3,G); # проверка эквивалентности
true

Из этого следует вывод, что созданные процедуры можно использовать для описания и анализа функциональных схем представленных с помощью базовых логических элементов "ИЛИ-НЕ".

Заключение

Установлено, что СКА Maple можно применять для изучения логических схем, построенных с использованием разработанной библиотеки функциональных элементов.

Авторами разработаны три отдельные процедуры, позволяющие осуществить построение функциональных схем в основном базисе (с помощью элементов "И" "ИЛИ" "НЕ"), а также две отдельные процедуры, позволяющие осуществить построение, описание и анализ функциональных схем в универсальном базисе (с помощью элементов "И-НЕ" и "ИЛИ-НЕ").

Это благоприятствует тому, что можно существенно сократить учебные временные затраты при обучении разделу вузовского курса «Математическая логика и теория алгоритмов», посвященного применению булевых функций к логическим схемам из функциональных элементов, уменьшить количество ошибок при проведении экспериментов, а также обеспечить наглядность представляемого материала.

Конкурирующие интересы. Авторы заявляют, что конфликтов интересов в отношении авторства и публикации нет.

Авторский вклад и ответственность. Все авторы участвовали в написании статьи и полностью несут ответственность за предоставление окончательной версии статьи в печать. Окончательная версия рукописи была одобрена всеми авторами.

Список литературы/References

1. Rosen K. H. Discrete Mathematics and Its Applications. New York: Published by McGrawHill, 2012. 1071 pp.
2. Gorjushkin A. P. Diskretnaja matematika v Maple. Kishinev: LAP LAMBERT Academic Publishing, 2016. 436 pp. (In Russian)
3. Оленев А. А., Киричек К. А., Потехина Е. И. *Программа для визуализации основных логических операций*, Свидетельство о регистрации программы для ЭВМ 2021613965. М., 2021. [Olenev A. A., Kirichek K. A., Potehina E. V. Programma dlja vizualizacii osnovnyh logicheskikh operacij, Svidetel'stvo o registracii programmy dlja JeVM 2021613965. M, 2021 (In Russian)].
4. Monagan M. B., et al. Maple Introductory Programming Guide. Canada: Maplesoft, a division of Waterloo Maple, 2006. 398 pp.
5. Сдвижков О. А. *Математика на компьютере: Maple 8*. М.: Солон-Пресс, 2003. 176 с. [Sdvizhkov O. A. Mathematics on the computer: Maple 8. Moscow: SOLON-Press, 2003. 176 pp. (In Russian)]
6. Оленев А. А., Малиатаки В. В. Моделирование логических элементов и простейших узлов ЭВМ в системе компьютерной математики Maple // *Информатика в школе*, 2017. № 8(131), С. 58–63. [Olenev A. A., Maliataki V. V. Modelirovanie logicheskikh jelementov i prostejshih uzlov JeVM v sisteme komp'yuternoj matematiki Maple // Informatika v shkole, 2017. no. 8(131), pp. 58-63 (In Russian)].
7. Оленев А. А., Малиатаки В. В., Тынчеров К. Т. Логические элементы и схемы в SKA Maple / *Современные технологии в нефтегазовом деле - 2017*, Сборник трудов международной научно-технической конференции, Октябрьский, 31 марта 2017, Т. 2. Октябрьский, Уфимский государственный нефтегазовый еухнический университет, 2017, С. 280–283. [Olenev A. A., Maliataki V. V., Tyncherov K. T. Logicheskie jelementy i shemy v SKA Maple / Sovremennye tehnologii v neftegazovom dele — 2017, Sbornik trudov mezhdunarodnoj nauchno-tehnicheskoy konferencii, Oktjabr'skij, 31 marta 2017 g., vol. 2. Oktjabr'skij, Ufinskij gosudarstvennyj neftjanoj tehnikeskij universitet, 2017, pp. 280-283 (In Russian)].
8. Горюшкин А. П. О применении компьютерной техники при изучении дисциплины «Математическая логика и теория алгоритмов» // *Актуальные проблемы преподавания математики в техническом вузе*, 2020. Т. 8, С. 97–105. [Goryushkin A. P. O primenenii komp'yuternoy tekhniki pri izuchenii distsipliny «Matematicheskaya logika i teoriya algoritmov» // Aktual'nyye problemy prepodavaniya matematiki v tekhnicheskom vuze, 2020. vol. 8, pp. 97-105 (In Russian)].
9. Горюшкин А. П. Машинное решение задач дискретной математики // *Вестник КРАУНЦ. Физ.-мат. науки*, 2012. № 2(3), С. 58–68. [Gorjushkin A. P. Mashinnoe reshenie zadach diskretnoj matematiki // Vestnik KRAUNC. Fiziko-matematicheskie nauki, 2011. no. 2(3), pp. 58-68 (In Russian)].
10. Горюшкин А. П. Исследование дискретных объектов компьютерными средствами / *Теория и практика современных гуманитарных и естественных наук*, Материалы ежегодной международной научно-практической конференции, 08-10 февраля 2012, Петропавловск-Камчатский. Петропавловск-Камчатский, КамГУ им. Витуса Беринга, 2012, С. 185–187. [Gorjushkin A. P. Issledovanie diskretnyh obektov komp'yuternymi sredstvami / Teorija i praktika sovremennyh gumanitarnyh i estestvennyh nauk, Materialy ezhegodnoj mezhregional'noj nauchno-prakticheskoy konferencii, Petropavlovsk-Kamchatskij, 08-10 fevralja 2012 g.. Petropavlovsk-Kamchatskij, Kamchatskij gosudarstvennyj universitet im. Vitusa Beringa, 2012, pp. 185-187 (In Russian)].
11. Горюшкин А. П., Горюшкин В. А. *Элементы абстрактной и компьютерной алгебры*. Петропавловск-Камчатский: КамГУ им. Витуса Беринга, 2011. 518 с. [Goryushkin A. P., Goryushkin V. A. Elementy abstraktnoy i komp'yuternoy algebry. Petropavlovsk-Kamchatka: KamGU im. im. Vitusa Beringa, 2011. 518 pp. (In Russian)]

MSC 03B70, 90C90

Research Article

Mathematical logic: construction of logic circuits from logical elements in Maple

A. A. Olenov, K. A. Kirichek, E. V. Potekhina

Stavropol State Pedagogical Institute, 355029, Stavropol region, Stavropol, Lenina str., 417A, Russia

E-mail: olenevalexandr@gmail.com, kirichekka@mail.ru, ekapotekhina@yandex.ru

The article discusses the possibilities of using the Logic library of the Maple computer algebra system in the aspect of computer modeling of logic circuits in various bases. Basic logic gates are modeled in Maple. On a specific example, an algorithm for constructing a logical circuit in various bases is presented in detail.

Key words: boolean function, self-duality, basis, universal bases, logical elements.

DOI: 10.26117/2079-6641-2021-36-3-155-164

Original article submitted: 03.09.2021

Revision submitted: 03.10.2021

For citation. Olenov A. A., Kirichek K. A., Potekhina E. V. Mathematical logic: construction of logic circuits from logical elements in Maple. *Vestnik KRAUNC. Fiz.-mat. nauki.* 2021, **36**: 3, 155-164. DOI: 10.26117/2079-6641-2021-36-3-155-164

Competing interests. The authors declare that there are no conflicts of interest regarding authorship and publication.

Contribution and Responsibility. All authors contributed to this article. Authors are solely responsible for providing the final version of the article in print. The final version of the manuscript was approved by all authors.

The content is published under the terms of the Creative Commons Attribution 4.0 International License (<https://creativecommons.org/licenses/by/4.0/deed.ru>)

© Olenov A. A., Kirichek K. A., Potekhina E. V., 2021

Funding. The study was carried out without financial support from foundations.