УДК 510.63

А.П. Горюшкин
кандидат физико-математических наук, доцент
Камчатский государственный университет имени Витуса Беринга, г. Петропавловск-Камчатский, Россия

О ПРИМЕНЕНИИ КОМПЬЮТЕРНОЙ ТЕХНИКИ ПРИ ИЗУЧЕНИИ ДИСЦИПЛИНЫ «МАТЕМАТИЧЕСКАЯ ЛОГИКА И ТЕОРИЯ АЛГОРИТМОВ»

Аннотация. В статье обсуждается методика использования пакета математических символьных вычислений Maple при изучении вузовского курса «Математическая логика и теория алгоритмов». Рассматриваются особенности синтаксиса подпакета with(Logic). Показана методика компьютерного нахождения конъюнктивной, дизъюнктивной и полиномиальной форм булевой функции. Отмечено, что при машинном нахождении полиномиальной нормальной формы для функции необходимо использовать, кроме команд логического подпакета, команды для работы с многочленами. На конкретном примере детально разобран алгоритм проверки функциональной полноты системы булевых функций.

Ключевые слова: булева функция; закон логики; противоречие; нормальная форма; полином Жегалкина; логическая равносильность; логическое следствие; функциональная полнота; Maple; машинная команда.

DOI: 10.25206/2307-5430-2020-8-97-105

В учебном плане направления подготовки бакалавриата 09.03.04 «Программная инженерия» две дисциплины «Дискретная математика» и «Математическая логика и теория алгоритмов» имеют «непустое пересечение». Алгебра логики высказываний и алгебра булевых функций занимают видное место в каждой из этих дисциплин. Конечно, в первый раз изучение этих тем носит ознакомительный характер (как правило, обсуждение не доходит даже до теоремы Стоуна о связи между булевыми кольцами и булевым алгебрами), но ощущение: «я это знаю» явно мешает студентам при повторном прохождении знакомого материала.

Опыт автора, в течение более десяти лет читающего курс «Математическая логика и теория алгоритмов» в техническом вузе, показывает, что неплохо «оживляет» обучаемых проведение практического занятия в компьютерной аудитории.

Рассмотрим в качестве примеров машинное решение нескольких задач с помощью математического пакета символьных вычислений Maple. Синтаксис подпакета «Логика» имеет свои особенности. Командой Export формула изображается без скобок, с использованием только трёх операций: конъюнкция, дизъюнкция и отрицание и с обычным договором о приоритетности логических операций.

ПРИМЕР 1. Ввод формулы, ее представление и вычисление в заданной точке.

> *with*(*Logic*): # вход в пакет «Логика»
> *f*:=&*and*(*a*,*b*,*c*)&*or*(&*nota*)&*or* (&*notb*) &*or*(&*notc*); # ввод формулы *f*
f := ((((*a*&*andb*) &*andc*) &*or*&*not*(*a*)) &*or*&*not*(*b*)) &*or*&*not*(*c*)
> *Export*(*f*); # представление формулы *f*
a and b and c or not a or not b or not c

> *g*:=(*a*&*iffb*) &*or*(&*notc*)&*orb*; # ввод формулы *g*
g := ((*a* &*iff b*)&*or* &*not*(*c*))&*or b*
> *Export*(*g*); # представления формулы *g*
a and b or not (*a or b*) *or not c or b*

> *g*:=(*a*&*iffb*) &*or*(&*notc*)&*orb*: # ввод формулы*g*
> *g*:=*Export*(*g*): # представление формулы *g*
> *a*:=*true*: *b*:=*false*: *c*:=*true*: *g*; # ввод значений переменных *a*, *b*, *c*
> *g*; # вывод значения формулы *g*
false

ПРИМЕР 2. Проверить, являются ли законами логики формулы *f* и *g* из примера 1.

> *Tautology*(*f*); # формула *f* – тавтология
true
> *Tautology*(*g*); # формула *g* – тавтология
false

Итак, формула *f* – закон логики, а формула *g* – нет. Найдём хотя бы одно место, где формула *g* принимает значение Л.

> *Tautology*(*g*, '*w*'); # формула *g* – тавтология
> *w*; # вывод набора значений, при которых формула *g* принимает значение Л
false
{*a* = *true, b* = *false, c* = *true*}

Впрочем, это значение было уже найдено ранее другим способом. Может быть, вторая формула вообще не принимает значение «И»?

Иначе говоря, выясним, не является ли формула *g* противоречием?

> *Contradiction*(*g*, '*s*'); # формула *g* – противоречие

false

> *s*; # вывод набора значений, при которых формула *g* принимает значение И

{*a = false, b = false, c = false*}

ПРИМЕР 3. Найти полиномиальное представление для тех же формул *f* и *g*.

> *f*1:=*Export*(*f*, *form*=*MOD*2); # представление формулы *f* в виде многочлена

*f*1 := 1 + (*abc* + 1)*abc*

> *g*1:=*Export*(*g*, *form*=*MOD*2); # представление формулы *g* в виде многочлена

*g*1 := 1 + (*a*+ *b*) *c* (*b* + 1)

Многочлены получены, но в этих многочленах оказались не раскрытыми скобки. Раскроем скобки и приведём подобные члены тоже с помощью техники.

> *f*2:=*evala*(*f*1) *mod* 2; # раскрытие скобок в многочлене *f*1

$f2 := 1 + a^2b^2c^2 + abc$

> *g*2:=*evala*(*g*1) *mod* 2; # раскрытие скобок в многочлене *g*1

$g2 := 1 + abc + ac + b^2c + bc$

И после раскрытия скобок окончательное решение не получено: многочлены – это ещё не полиномы Жегалкина: кольцо коэффициентов $\mathbf{Z}_2$ – идемпотентно, а в выписанных многочленах содержатся вторые степени переменных. Учтем идемпотентность кольца булевых функций. с помощью обычных для многочленов команд.

> *f*3:=*subs*(*a***a*=*a*,*b***b*=*b*,*c***c*=*c*, *f*2)*mod* 2; # замена в многочлене *f*2 с учётом тождества $x^2=x$

*f*3 := 1

> *g*3:=*subs*(*a***a*=*a*,*b***b*=*b*,*c***c*=*c*, *g*2)*mod* 2; # замена в многочлене *g*2 с учётом тождества $x^2=x$

*g*3 := 1 + *abc* + *ac*

Теперь получены полиномы Жегалкина для каждой из формул, и ясно видно, что формула *f* действительно закон логики, а формула *g* опровержима.

Впрочем, эту же цель можно было достигнуть быстрее, используя команды для нахождения совершенных нормальных форм.

Пример 4. Найти полиномиальную, дизъюнктивную и конъюнктивную формы для формул *f* и *g*.

> *Canonicalize*(*f*, *form=MOD*2); # нахождение полинома Жегалкина для формулы *f*

1

> *Canonicalize*(*g*, *form=MOD*2); # нахождение полинома Жегалкина для формулы *g*

1 + *AB* c + *A* c

> *Normalize*(*f*, *form=CNF*); # нахождение конъюнктивной нормальной формы для формулы *f*

true

> *Normalize*(*g*, *form=CNF*); # нахождение конъюнктивной нормальной формы для формулы *g*

(*b &or &not*(*a*))*&or &not*(*c*)

> *Normalize*(*g*, *form=DNF*); # нахождение дизъюнктивной нормальной формы для формулы *g*

(((((((*a &and b*)*&and c*)*&or* ((*&not*(*c*)*&and a*)*&and b*))*&or* ((*&not*(*a*)*&and b*)*&and c*))*&or* ((*&not*(*a*)*&and b*)*&and &not*(*c*)))*&or* ((*&not*(*a*)*&and &not*(*b*))*&and c*))*&or* ((*&not*(*a*)*&and &not*(*b*))*&and &not*(*c*)))*&or* ((*&not*(*b*)*&and a*)*&and &not*(*c*))

В пакете *Logic* кроме «стандартного набора» логических операций (*and* – конъюнкция, *or* – дизъюнкция, *not* – отрицание, *implies* – импликация, *iff* – эквиваленция) содержатся: разделительная дизъюнкция (*xor*), штрих Шеффера (*nand*) и стрелка Пирса (*nor*). Убедимся, что это действительно так, представив эти три последние функции полиномами Жегалкина:

> *Export*(*a&xorb*, *form=MOD*2); # нахождение полинома Жегалкина

a + *b*

> *Export*(*a&nandb*, *form=MOD*2); # нахождение полинома Жегалкина

ab + 1

> *Export*(*a&norb*, *form=MOD*2); # нахождение полинома Жегалкина

(*a* + 1)(*b* + 1)

Может потребоваться и обратное преобразование – от полинома Жегалкина к представлению с использованием трёх операций: конъюнкция, дизъюнкция и отрицание.

> *Import*(*y*+*z*+*x***y***z*+1, *form=MOD*2); # представление формулы *y*+*z*+*xyz*+1

&not((*y &xor z*)*&xor* ((*x &and y*)*&and z*))

ПРИМЕР 5. Выяснить, равносильны ли две формулы алгебры логики, является ли одна формула следствием другой.

> *Equivalent*(*f*, *g*, '*p*'); # формулы *f* и *g* равносильны

false

> *p*; # набор значений переменных, при которых значения формул *f* и *g* не совпадают

{*a = true, b = false, c = true*}

Формулы не равносильны, компьютер указал и строчки набора переменных, где значения формул различны.

Выясним теперь, не является ли одна из формул логическим следствием другой, а если нет, то в каком месте логическое следствие нарушается.

> *Implies*(*f*, *g*, '*w*'); # формула *g* логически следует из формулы *f*

false

> *w*; # набор значений переменных, при которых нарушается логическое следствие

{*a = true*, *b = false*, *c = true*}

> *Implies*(*g*, *f*); # формула *f* логически следует из формулы *g*

true

Заметим, что при проверке равносильности одна из функций может быть константой, и в этом случае будет найдена соответствующая строка таблицы истинности.

> *Equivalent*(*f*, *false*, '*w*'); # формула *f* тождественно ложна

false

> *w*; # набор значений переменных, при которых формула *f* принимает значение И

{*f = true*}

> *Equivalent*(*g*, *true*, '*u*'); # формула *g* тождественно истинна

false

> *u*; # набор значений переменных, при которых формула *g* принимает значение Л

{*a = true*, *b = false*, *c = true*}

> *Equivalent*(*g*, *false*, '*w*'); *w*; # формула *g* тождественно ложна

false

> *w*; # набор значений переменных, при которых формула *g* принимает значение И

{*a = false*, *b = false*, *c = false*}

Основополагающей теоремой при проверке полноты системы булевых функций является теорема Э. Поста о функциональной полноте. Наиболее сложной частью является проверка, не является ли функция самодвойственной. Для этого достаточно уметь находить двойственную функцию.

ПРИМЕР 6. Нахождение двойственной функции и проверка самодвойственности.

> *Dual*(*a*&*andb*); # двойственная функция для $a \,\&\, b$
a &or b
> *Dual*(*a*&*orb*); # двойственная функция для $a \vee b$
a & and b

Выясним, какая функция двойственна импликации. Найдём представление функции $a \to b$ и её двойственной в виде полиномов Жегалкина.

> *t*:=*a*&*impliesb*; # ввод функции$t = a \to b$
t := *a & implies b*
> *t*1:=*Dual*(*t*); # *t*1 – функция, двойственная функция для $a \to b$
t1 := *&not*(*a*) *& implies & not*(*b*)
> *Canonicalize*(*t*, *form*=*MOD*2); # нахождение полинома Жегалкина для функции *t*
1 + *BA* + *A*
> *Canonicalize*(*t*1, *form*=*MOD*2); # нахождение полинома Жегалкина для функции *t*1
1 + *ba* + *b*

Представления Жегалкина для функций оказались различны; функция $a \to b$ – несамодвойственна.

Проверим, не является ли самодвойственной появившаяся ранее функция *g* (функция *f*– константа, и её не самодвойственность очевидна).

> *g*:=(*a*&*iffb*) *&or*(*&notc*)*&orb*: # ввод функции *g*
> *g*1:=*Dual*(*g*); *g*1 – функция, двойственная функция для *g*
g1 := ((*a &xor b*)*&and &not*(*c*))*&and b*
> *Equivalent*(*g*, *g*1); формулы *g* и *g*1 равносильны
false
> *Equivalent*(*g*, *g*1,'*w*'): *w*; # набор значений переменных, при которых значения формула *g* и *g*1 не совпадают
{*a = false, b = false, c = false*}

Функция *g* несамодвойственна. Кстати, она не линейна и не сохраняет нуль, но полную систему всё-таки не образует, так как она сохраняет единицу.

В некоторых задачах математической логики желательно иметь всю таблицу истинности исследуемой формулы алгебры логики. Найдём такую таблицу для формулы *g*:

> *g*:=(*a*&*iffb*) &*or*(&*notc*)&*orb*: # ввод функции *g*
> *T*1:=*TruthTable*(*g*, [*a*, *b*, *c*]); # табулирование функции *g*
T1 := *table*([(*false, false, false*) = *true,* (*true, true, true*) = *true,* (*false, false, true*) = *true,* (*false, true, false*) = *true,* (*true, true, false*) = *true,* (*true, false, true*) = *false,* (*true, false, false*) = *true,* (*false, true, true*) = *true*])

Для работы, впрочем, удобней таблица значений соответствующей булевой функции:

> *T*2:=*TruthTable*(*g*, [*a*,*b*,*c*], *form*=*MOD*2); # табулирование полинома Жегалкина, представляющего функцию *g*
T3 := *table*([(1, 1, 0) = 1, (1, 0, 1) = 0, (1, 0, 0) = 1, (1, 1, 1) = 1,
(0, 0, 1) = 1, (0, 1, 0) = 1, (0, 0, 0) = 1, (0, 1, 1) = 1])

Оказалось, что функция *g* при всех наборах значений переменных, кроме одного, принимает значение 1. Таким образом, нарушение равносильности формул *f* и *g* происходит в единственном месте $a = 1$, $b = 0$, $c = 1$; то же самое машина показала раньше при проверке равносильности: *a = true, b = false, c = true*.

Вся таблица может оказаться слишком большой, а в решаемой задаче нужно лишь одно или несколько значений исследуемой формулы. С помощью таблицы (не выводя саму таблицу на экран) удобно вычислить отдельные значения нашей функции. Например,

> *T*1:=*TruthTable*(*g*, [*a*, *b*, *c*]): # табулирование функции *g*
true
> *T*1[*true*, *false*, *true*];]): # вычисление значения функции *g* при *a*=И, *b*=Л, *c*=И
false
> *T*2:=*TruthTable*(*g*, [*a*, *b*, *c*], *form*=*MOD*2): # табулирование полинома Жегалкина, представляющего функцию *g*
>*T*2[0, 0, 1]; # вычисление значения функции *g* при *a*=0, *b*=0, *c*=1

1
> *T*2[1, 0, 1]; # вычисление значения функции *g* при *a* = 1, *b* = 0, *c* = 1
0

Найдём теперь таблицу значений для функции *g*1, двойственной для отрицания функции *g*:

> *T*2:=*TruthTable*(&*not*(*g*1), [*a*, *b*, *c*], *form*=*MOD*2);
T2 := *tAB*le([(0, 0, 1) = 1, (1, 1, 1) = 1, (0, 0, 0) = 1,
(1, 1, 0) = 1, (0, 1, 1) = 1, (1, 0, 0) = 1, (0, 1, 0) = 0,
(1, 0, 1) = 1])

Если бы функция *g* была самодвойственной, то таблицы *T1* и *T2* полностью бы совпадали, однако (правда, лишь в одном месте) эти таблицы различны.

ПРИМЕР 7. Проверка функциональной полноты булевой функции.

Образует ли функция

s (*a*, *b*, *c*) = ((&*not*(*a*) &*andb*) &*andc*) &*or* ((&*not*(*a*) &*and*&*not*(*b*)) &*and*&*not*(*c*))

полную систему булевых функций?

Для начала убедимся, что эта функция не сохраняет ни нуль, ни единицу:

> *s*:= ((&*not*(*a*) &*andb*) &*andc*) &*or* ((&*not*(*a*) &*and*&*not*(*b*)) &*and*&*not*(*c*)):
> *T*:= *TruthTable*(*s*, [*a*,*b*,*c*], *form*=*MOD*2):
> *T*[0, 0, 0];
1
> *T*[1, 1, 1];
0

Функция *s* – не константа, и *s*(0, 0, 0) = 1, поэтому она не монотонна. Проверим её на самодвойственность.

> *Equivalent*(*s*, *Dual*(*s*));
false

Остаётся проверить функцию *s* на линейность.

> *Canonicalize*(*s*, [*a*,*b*,*c*], *form*=*MOD*2);
ab + *ac* + *a* + *b* + *c* + 1

Итак, функция не сохраняет ни нуль, ни единицу, она не монотонна, не самодвойственна и не линейна. Следовательно, по теореме Поста функция *s* образует полную систему.

Уместно спросить, а может быть, эта функция представляет собой лишь обобщение функции Пирса или функции Шеффера, то есть не равна ли *s* функции $s_1(a,b,c)=\bar{a}\,\&\,\bar{b}\,\&\,\bar{c}$ или $s_2(a,b,c)=\bar{a}\vee\bar{b}\vee\bar{c}$?

Техника легко решает этот вопрос.

> *s*1:=&*not*(&*xor*(&*xor*(*b*,*a*),*c*));

s1 := &not((b &xor a)&xor c)
> s2 := &and(&and(&not(a),&not(b)),&not(c));
s2 := (&not(a)&and &not(b))&and &not(c)
> Equivalent(s, s1);
false
> Equivalent(s, s2);
false

Конечно, для формулы с тремя аргументами все это можно проделать за сравнительно недолгое время и вручную. Возможности техники впечатляют при проверке полноты системы, состоящей из функций от большого числа переменных. Например, таблица истинности формулы с восемью переменными занимает 256 строк, но техника проделывает все вычисления, связанные с проверкой полноты этой функции, в несколько секунд.

Сведения об авторе:

Горюшкин Александр Петрович
Служебный адрес: 683032 г. Петропавловск-Камчатский, ул. Пограничная, 4, КамГУ имени Витуса Беринга, офис 28; e-mail: as2021@mail.ru; spin-code: 6283-2930.