Посвящается памяти Глеба Павловича Акимова. Его широкие взгляды и плодотворные идеи столь же полезны для дискретной, как и для непрерывной математики

УДК 519.7

ЛЕЗВИЕ ОККАМА И ЧАСТИЧНЫЕ БУЛЕВЫ ФУНКЦИИ

Р.Е.Кричевский

1. Средневековый английский философ Оккам выдвинул принцип лезвия: "не умножай сущности сверх необходимости". Среди всех возможных объяснений какого-либо явления следует выбирать самое простое.

Одна из формализаций этого принципа приводит к задаче о простейшей реализации не всюду определенной булевской функции.

Пусть E^n, $n > 0$, – множество двоичных слов длины n, f – частичная булевская функция, т.е. отображение из E^n во множество $\{0,1\}$. Область определения f обозначим через $\mathrm{dom}f$, $\mathrm{dom}f = \{x_1,\ldots,x_T\}$, через T обозначена мощность $\mathrm{dom}f$, $x_t \in E^n$, $t=1,\ldots,T$. Слово

$$\mathrm{can}f = x_1 * f(x_1) * x_2 * f(x_2) \ldots x_T * f(x_T)$$

назовем каноническим представлением частичной функции f.

Допустим, мы обучили студента принимать решение $f(x_t)$ на слове x_t, $t = 1,\ldots,T$. И вот обучение окончено, студент столкнулся с ранее не встречавшейся ситуацией x, $x \notin \mathrm{dom}f$. Какое же решение должен принять студент в ситуации x ?

В процессе обучения студент должен сформулировать некоторое правило, преобразующее $x_t \in \mathrm{dom}f$ в $f(x_t)$. Из всех таких правил надлежит выбрать самое простое. Принцип Оккама рекомендует принять на неизвестном слове x решение, получаемое в результате приложения к x этого простейшего правила.

Под правилом будем понимать программу вычисления f на компьютере с произвольным доступом к памяти. Если программа и слово x загружены в память, то компьютер вычислит $f(x)$ в реальное время, т.е. спустя $O(n)$ единиц времени, где n является длиной слова x и адресуемого компьютерного слова. Программа - бинарное слово, а ее сложность равна ее длине.

Конечно, в слово "правило" можно вкладывать и иной смысл. Скажем, это может быть программа на машине Тьюринга, или схема из функциональных элементов, или что-то еще. Однако асимптотическое поведение результатов не очень существенно зависит от выбранной модели вычислений.

Очевидный метод конструирования программ вычисления частичных булевых функций состоит в следующем. Упорядочим слова $x_1,\ldots,x_T$, на которых определена функция f, что потребует $O(T \log T)$ единиц времени, $|\mathrm{dom}f| = T$. Возьмем слово $\mathrm{can}f$ и добавим к нему команду: "Ищите вхождение слова x в слово $\mathrm{can}f$. Если $x = x_t$, $1 \leqslant t \leqslant T$, положите $f(x) = f(x_t)$. Если же ни одно из слов $x_1,\ldots,x_T$ не равно x, положите $f(x) = 0$". Слово $\mathrm{can}f$ с таким добавлением образует программу вычисления f, длина которой равна $T(n + 1) + O(1)$ бит, где n - длина слов $x_1,\ldots,x_T$. Время вычисления $f(x)$ с помощью этой программы равно $O(n)$, если разыскивать x методом дихотомического деления.

Пионером в этой области был Э.И.Нечипорук. Хотя в его работе [1] речь шла о вентильных схемах, результаты нетрудно перефразировать для компьютерных программ. Они таковы. Во-первых, если

$$\lim \frac{T}{\log n} = \infty, \qquad (1.1)$$

то для почти всех функций f от n переменных, $|\mathrm{dom} f| = T$, длина кратчайшей программы не меньше $T(1+o(1))$. Во-вторых, имеется алгоритм, преобразующий слово $\mathrm{can} f$ в программу для вычисления f. Эта программа асимптотически минимальна, имея длину $T(1+o(1))$.

К сожалению, время работы этого преобразующего алгоритма чрезвычайно велико.

Важные результаты, относящиеся к этой тематике, получены в работах Л.А.Шоломова [2], Н.Пиппенжера [3], А.Е.Андреева [4]. В них речь идет о реализации булевых функций схемами из функциональных элементов. Скажем, в работе [4] доказано, что для реализации любой частичной функции достаточно иметь $\rho \, {}^{T}/_{\log T}$ функциональных элементов, где ρ – константа, зависящая от базиса. Как и в [1], предлагаемые в этих работах алгоритмы преобразования слова $\mathrm{can} f$ в схему весьма трудоемки.

Цель этой статьи состоит в построении быстрого алгоритма, преобразующего слово $\mathrm{can} f$ в асимптотически кратчайшую программу для вычисления f. При условии (1.1) длина этой программы равна T бит, вместо $T(n+1) + O(1)$, как в случае тривиальной программы. Время работы алгоритма почти квадратичное по отношению к длине $|\mathrm{can} f|$ перерабатываемого им слова. С одной стороны, этот алгоритм гораздо быстрее, чем алгоритмы Нечипорука, Шоломова, Пиппенжера, Андреева. С другой стороны, он все же медленнее, чем упоминавшийся очевидный алгоритм: это расплата за то, что, в отличие от очевидного, он продуцирует наилучшую программу.

Итак, если мы хотим, согласно принципу Оккама, найти простейшее правило (программу), порождающее частичную функцию, то для почти всех функций можем сделать это за почти квадратичное время относительно длины канонической записи функции.

Наш алгоритм применим к решению многих комбинаторных задач. Среди них построение k-независимых множеств, качественно независимых разбиений, быстрое нахождение подслов данного слова. Не будем останавливаться здесь на этих задачах. Дадим только идею алгоритма и проиллюстрируем его примерами. Полное доказательство будет опубликовано в другом месте.

2. Алгоритм нахождения программы для частичной булевой функции использует полиномы Галуа, лексикографические разбиения и, наконец, полный перебор для функций от небольшого числа переменных.

Во-первых, о полиномах Галуа. Пусть $h(x)$ - неприводимый над полем $GF(2)$ полином степени n_2, $n_2 > 0$. Установим с помощью $h(x)$ изоморфизм между E^{n_2} и полем Галуа $GF(2^{n_2})$, отождествляя двоичные слова длины n_2 из E^{n_2} с наборами коэффициентов полиномов по модулю $h(x)$. Так, если n_2=2, то можно взять неприводимый полином $h(x) = x^2+x + 1$. Тогда

$$00 \longrightarrow 0 + 0\cdot x,\ 01 \longrightarrow 0 + x,\ 10 \longrightarrow x + 0,\ \ 11 \longrightarrow x + 1.$$

Возьмем произвольный двоичный вектор некоторой длины n_1, $n_1 > n_2$. Разделим его на n_2-мерные подвекторы $W_1,\ldots,W_{]n_1/n_2[}$, дополняя последний из них до длины n_2 нулями, если это необходимо. Здесь $]x[$ - наименьшее целое число, не меньшее x. Любой элемент $x \in GF(2^{n_2})$ определяет следующее полиномиальное отображение g_x множества E^{n_1} в E^{n_2}:

$$g_x(W) = W_1 + W_2\cdot x +\ldots+ W_{]n_1/n_2[}\, x^{]n_1/n_2[-1}. \qquad (2.1)$$

Множество всех таких отображений обозначим через $G(n_1,n_2)$. Например, множество $G(4,2)$ содержит 4 отображения. Их таблицы приведены в табл.1. Двоичные векторы представлены соответствующими целыми числами: 11=3, 110=6 и т.д., на пересечении строки x и столбца W стоит $g_x(W)$. Так,

$$g_3(9) = g_{11}(1001) = (10) + (01)(11) = x + x + 1 = 1.$$

Множество $G(4,2)$ является универсальным нумератором 3-элементных подмножеств E^4. Это значит, что для любых трех элементов из E^4 найдется такой x в E^2, что отображение g_x принимает на этих трех элементах попарно различные значения. Скажем, если W_1 = 3, W_2=4, W_3=9, то x=0. Этот факт установлен в [5].

Таблица 1

Множество $G(4,2)$

переменные	отображения			
	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	2	2
4	1	1	1	1
5	1	0	3	2
6	1	3	2	0
7	1	2	0	3
8	2	2	2	2
9	2	3	0	1
10	2	0	1	3
11	2	1	3	0
12	3	3	3	3
13	3	2	1	0
14	3	1	0	2
15	3	0	2	1

Во-вторых, о лексикографических разбиениях. Пусть $\mathrm{dom} f = S$, $S \subseteq E^n$, $S = A_1 \cup A_2$, $A_1 \cap A_2 = \emptyset$, причем все слова A_1 лексикографически предшествуют словам из A_2. Обозначим через f_{A_t} функцию, совпадающую с f на A_t, $t=1,2$, и принимающую любые значения вне F_t. Тогда (A_1, A_2) называется лексикографическим разбиением. Задачу построения программы для f мы сводим к задачам построения программ для f_{A_t}, $t=1,2$.

В-третьих, о полном переборе. Для данных чисел n и T через $V_T(n)$ обозначим такое множество всюду определенных булевых функций n переменных, что для любой частичной функции f от n переменных, $|\mathrm{dom} f| = T$, существует функция $g \in V_T(n)$, совпадающая с f на области ее определения $\mathrm{dom} f$. Множество $V_T(n)$ может быть

построено полным перебором. Функции, входящие в $V_3(2)$, приведены в табл.2.

Таблица 2

Векторы из E^2	Номер функции в $V_3(2)$							
	000	001	010	011	100	101	110	111
00	0	1	1	1	0	0	0	1
01	0	1	0	0	1	1	0	1
10	0	0	1	0	0	1	1	1
11	0	0	0	1	1	0	1	1

Программа для вычисления частичной функции строится путем применения трех описанных методов. Во-первых, выберем полином Галуа, инъективный на множестве $\mathrm{dom} f$. Возможность этого доказана в [5]. Там же описан алгоритм выбора. Заменяя множество $\mathrm{dom} f$ на его образ при выбранном полиномиальном отображении, уменьшим число переменных, от которых зависит f. Переходя к лексикографическим разбиениям, мы можем уменьшить мощность множества $\mathrm{dom} f$. Полный перебор используется в конце для функций, зависящих от не очень большого числа переменных.

Проиллюстрируем метод на примере. Пусть f - частичная функция, определенная в табл.3. Четырехмерные бинарные векторы представлены целыми числами от 0 до 15. Если f определена на x, то $f(x)$ помещено напротив x. Эта функция f определена на множестве

Таблица 3

Частично определенная функция f

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$f(x)$				1	0					1		1			1	0

$\mathrm{dom} f = S$ - {3,4,9,11,14,15}. Вектор a=10 (в двоичной записи 1010) определяет лексикографическое разбиение S на два атома, A_1 и A_2, где A_1= {3,4,9}, A_2= {11,14,15}. Функция f_{A_t} совпадает с f на A_t. Как можно проверить, полином Галуа g_{x_1}, x_1= 00, инъективен на A_1, g_{x_2}, x_2= 10, - на A_2. Полином g_{00} отображает A_1 в $g_{00}(A_1)$ = {0,1,2}, а g_{10} отображает A_2 в $g_{00}(A_2)$ = {3,0,2}. Это отображение уменьшает число переменных с четырех до двух. Определим функции двух переменных f'_1 и f'_2 формулой $f'(g_{00}(W)) = f_{A_t}(W)$, если $W \in A_t$, t=1,2. Области определения функций f'_t состоят из трех двумерных векторов. Следовательно, каждая из них совпадает на своей области определения с какой-нибудь функцией из $V_3(2)$. А именно, f'_1 совпадает с функцией номер n_1 = 010, а f'_2 - с функцией номер n_2 = 011.

Возьмем векторы a=1010, векторы x_1=00 и x_2=10, определяющие полиномы Галуа, а также номера n_1=010 и n_2=011 функций f'_{A_t}, t = 1,2. Их конкатенация, четырнадцатимерный вектор $ax_0x_1n_0n_1$ = 10100010010011, и является искомой программой, вычисляющей f.

Программа работает следующим образом. Пусть $W \in E^4$. Выделим из программы первые четыре разряда - вектор a. Сравним W с a. Если $W \geqslant a$, выделим из программы x_1. Найдем $g_{x_1}(W)$. Найдем номер n_1 функции f'_1 из $V_3(2)$ и сосчитаем ее значение на $g_{x_1}(W)$. Это и есть $f(W)$.

В формальном доказательстве будет указано, как действовать в общем случае. Для произвольной частичной булевой функции f за почти квадратичное от $|\mathrm{can} f|$ время будет найдена почти асимптотически кратчайшая программа.

Литература

1. **Нечипорук Э.И.** Сложность реализации булевых матриц с неопределенными элементами вентильными схемами // ДАН СССР. - 1965.-Т.163, № 1. - С.40-42.

2. **Шоломов Л.А.** О реализации недоопределенных булевых функций схемами из функциональных элементов // Проблемы кибернетики. Вып.21. - М.: Наука, 1969. - С.215-226.

3. **Pippenger N.** Information theory and the complexity of Boolean functions // Math. Syst. Theory. - 1977. - V.10. - P.129-167.

4. **Андреев А.Е.** О сложности реализации частичных булевых функций схемами из функциональных элементов // Дискретная математика. - 1989. - Т.1, № 4. - С.36-45.

5. **Кричевский Р.Е.** Сжатие и поиск информации. - М.: Радио и связь, 1989.

Поступила в редакцию
1 ноября 1990 г.