Серия «Математика»
2011. Т. 4, № 1. С. 83—96

Онлайн-доступ к журналу:
http://isu.ru/izvestia

ИЗВЕСТИЯ
Иркутского
государственного
университета

УДК 519.7

Преобразования алгоритмов вычисления дискретных функций в булевы уравнения

И. В. Отпущенников
Институт динамики систем и теории управления СО РАН, Иркутск

А. А. Семёнов
Институт динамики систем и теории управления СО РАН, Иркутск

Аннотация. Статья посвящена проблеме преобразования алгоритмических описаний дискретных функций в эквациональные описания, имеющие вид булевых уравнений. Основу развиваемого в работе аппарата составляют процедуры пропозиционального кодирования программ для двоичной машины с произвольным доступом. На базе этих процедур строятся преобразования высокоуровневых описаний алгоритмов вычисления дискретных функций в булевы уравнения.

Ключевые слова: дискретные функции; булевы уравнения; машина с произвольным доступом.

Введение

При изучении дискретных систем часто оказывается возможным описать некоторый процесс в рассматриваемой системе в виде дискретной функции, то есть функции, преобразующей двоичные слова в двоичные слова. Для широкого класса систем такие функции оказываются вычислимыми за полиномиальное время. Многие важные свойства дискретных систем можно устанавливать, решая задачи обращения соответствующих функций, то есть находя неизвестный прообраз по известному образу и известному алгоритмическому описанию функции. Такого рода задачи возникают, например, при исследовании автоматных моделей дискретных систем, когда требуется, наблюдая результат эволюции системы в дискретном времени и зная механизм этой эволюции, восстановить начальное или некоторое промежуточное состояние системы. Множество примеров задач обращения дает криптография, так как разнообразные постановки задач криптоанализа являются частными случаями проблемы обращения полиномиально вычислимых функций.

Сказанное означает принципиальную важность разработки вычислительных методов, применимых к решению задач обращения дискретных функций, возникающих в практических приложениях. В настоящей работе мы рассматриваем основы «пропозиционального подхода» к обращению полиномиально вычислимых дискретных функций. Данный подход включает две составляющих. Во-первых, это преобразования алгоритмов вычисления дискретных функций в булевы уравнения, и, во-вторых, это символьные алгоритмы поиска решений получаемых уравнений.

По методам решения булевых уравнений имеется весьма обширная библиография, однако статей, специально посвященных технологиям сводимости к булевым уравнениям различных комбинаторных проблем, сравнительно мало (можно сослаться на обзор [13] и список литературы к нему). К тому же в подавляющем большинстве эти результаты имеют характер наглядных примеров и правдоподобных рассуждений — самой строгой в этом смысле продолжает оставаться процедура, фигурирующая в оригинальном и последующих доказательствах теоремы Кука [9].

В настоящей статье описаны основные принципы преобразования программ для двоичной машины с произвольным доступом в булевы уравнения. На основе данных принципов построены механизмы трансляции в булевы уравнения высокоуровневых процедурных описаний алгоритмов, вычисляющих дискретные функции.

1. Преобразования формальных программ вычисления дискретных функций в булевы уравнения

Многочисленные комбинаторные задачи тесно связаны со следующей общей проблемой, которую мы называем далее проблемой обращения полиномиально вычислимых дискретных функций.

Дана дискретная функция $f : \{0,1\}^* \to \{0,1\}^*$, описанием которой является программа M_f детерминированной машины Тьюринга (ДМТ). Предполагаем, что $dom\, f = \{0,1\}^*$, то есть M_f останавливается на произвольном двоичном слове, и сложность M_f растет как некоторый полином с ростом длины входа. Тем самым M_f задает счетное семейство функций вида $f_n : \{0,1\}^n \to \{0,1\}^*$, $dom\, f_n = \{0,1\}^n$, $n \in N$. Проблемой обращения произвольной функции f_n из данного семейства в точке $y \in range\, f_n$ называется следующая задача: зная M_f, число n и $y \in range\, f_n$, найти такое $x \in \{0,1\}^n$, что $f_n(x) = y$.

Пропозициональный подход к данной задаче основан на следующем факте: процесс работы программы M_f на произвольном входе можно эффективно (в общем случае за полиномиальное от n время) представить в виде формулы исчисления высказываний. Истинность данной

формулы при некоторых дополнительных условиях, характеризующих конкретный выход $y \in range\, f_n$, означает существование такого $x \in \{0,1\}^n$, результат трансформации которого посредством программы M_f есть y. Фактически в этом состоит теорема Кука.

Построение процедуры пропозиционального кодирования ДМТ-программ имеет чисто теоретический интерес. Гораздо более близкими к современным реальным вычислителям являются машины с произвольным доступом (RAM) [1]. Далее мы используем вариант RAM, синтаксис программ которой фактически совпадает с синтаксисом программ для машины с неограниченными регистрами. Данная модель включает потенциально бесконечную вправо ленту, разбитую на ячейки, которые пронумерованы натуральными числами. В каждой ячейке может быть записан только один бит. Произвольная бинарная RAM-программа — это нумерованный список команд, каждая из которых может быть командой одного из следующих двух типов:

1. команды записи в ячейку с номером k бита «0» или бита «1» — соответственно $B_0(k)$ и $B_1(k)$;
2. команды условного перехода $J(k,l,m)$: сравнить содержимое ячеек с номерами k и l, в случае совпадения перейти к команде с номером m в списке, в противном случае перейти к команде, которая следует в списке за командой $J(k,l,m)$.

Вычисление останавливается либо после выполнения последней команды в программе (если данная команда не является командой условного перехода), либо если происходит ссылка на несуществующую команду.

Пусть f — произвольная тотальная на $\{0,1\}^*$ полиномиально вычислимая дискретная функция, рассматриваемая как семейство

$$f = \{f_n\}_{n\in N} : f_n : \{0,1\}^n \to \{0,1\}^*,$$

и M_f — ДМТ-программа, вычисляющая f. В соответствии с [2] значение функции сложности $\rho(n)$ программы M_f равно максимуму шагов ДМТ, выполняющей M_f, по всевозможным входам из $\{0,1\}^n$. Пусть R_{f_n} — произвольная программа бинарной RAM, вычисляющая функцию f_n. Сопоставим ей значение функции $\vartheta(n)$, равное максимальному по всевозможным входам из $\{0,1\}^n$ числу обращений к регистрам RAM в процессе выполнения R_{f_n}.

Мы приводим здесь два утверждения, являющиеся теоретической базой описываемых далее процедур преобразования.

Лемма 1 (о моделировании)**.** *Пусть M_f — ДМТ-программа, вычисляющая тотальную дискретную функцию $f : \{0,1\}^* \to \{0,1\}^*$, и функция сложности программы M_f ограничена некоторым полиномом от n. Существует тотальная алгоритмически вычислимая функция g, которая за полиномиальное от n время по тексту программы M_f*

и числу n выдает текст программы R_{f_n}, вычисляющей функцию f_n. Функция $\vartheta(n)$, сопоставляемая получаемому семейству МНР-программ, ограничена сверху полиномом от n.

Данный факт означает наличие эффективной процедуры перехода от ДМТ-программы, вычисляющей f, к семейству двоичных RAM-программ, каждая из которых вычисляет функцию $f_n, n \in N$. Как уже отмечалось, синтаксис RAM-программ близок к ассемблерным программам, что крайне важно при построении практичных процедур пропозиционального кодирования алгоритмов. В техническом плане доказательство леммы 1 стандартно и здесь не приводится (работа ДМТ с входным алфавитом $\{0,1\}$ на всевозможных входах длины n моделируется на описанной выше RAM в соответствии с принципами, изложенными, например, в [1]).

Теорема 1. *Пусть $f = \{f_n\}_{n\in N}$ — семейство алгоритмически вычислимых за полиномиальное время дискретных функций и $\{R_{f_n}\}_{n\in N}$ — семейство бинарных RAM-программ, сопоставляемое f в соответствии с леммой о моделировании. Существует алгоритмически вычислимая тотальная функция h, которая, получая на входе текст программы R_{f_n}, за полиномиальное в общем случае от n время строит такую систему булевых уравнений $S(f_n)$, что:*

1. *для произвольного $y \in range\, f_n$ система $S(f_n)|_y$ совместна;*
2. *если $\tilde{x}$ — решение системы $S(f_n)|_y$, то за линейное от $|\tilde{x}|$ время возможен переход от $\tilde{x}$ к $x \in \{0,1\}^n : f_n(x) = y$.*

Здесь через $S(f_n)|_y$ обозначена система булевых уравнений, которая получается из системы $S(f_n)$ в результате подстановки в нее вектора $y \in range\, f_n$.

Доказательство. Базовая идея пропозиционального кодирования алгоритмов состоит в построении символьного описания всех возможных эволюций соответствующего вычисления на произвольных входах. В нашем случае требуется описать поведение RAM-вычисления на всевозможных входных двоичных словах длины n. Это можно сделать, если описывать каждую последующую конфигурацию набором функций от предыдущей конфигурации. Если к некоторой конфигурации K_i применяется оператор условного перехода, то, вообще говоря, возможен переход как в K'_{i+1}, так и в K''_{i+1}, однако можно рассмотреть «абстрактную» конфигурацию K_{i+1}, ячейкам которой сопоставлены булевы переменные, принимающие (в зависимости от соответствующего условия) либо значения, которые формируют конфигурацию K'_{i+1}, либо значения, которые формируют конфигурацию K''_{i+1}. Тем самым, биты в ячейках регистров RAM в конфигурации K_{i+1} можно считать значениями булевых функций от содержимого регистров RAM в конфигурации

K_i. Данные функции можно записать в явном виде, используя только текст программы R_{f_n}. Очень важно то, что при этом мы точно знаем, сколько потребуется булевых переменных, кодирующих содержимое регистров RAM в конфигурации K_{i+1}. Это число заведомо не больше максимального номера ячейки, к которой обращается программа R_{f_n} в процессе вычисления.

Итак, процесс пропозиционального кодирования программы R_{f_n} состоит в следующем. Предполагается, что R_{f_n} получает на вход произвольный двоичный вектор $(x_1, \ldots, x_n)$ (стартовая конфигурация K_0). Допустим, что первой командой в R_{f_n} является команда условного перехода. Для описания конфигурации K_0 введем булевы переменные $x_1, \ldots, x_\phi$, $\phi \geq n$, где $\phi = \phi(n)$ — максимальное натуральное число, фигурирующее в тексте программы R_{f_n} (рост функции $\phi(n)$ ограничен некоторым полиномом от n). Конфигурация K_1 описывается переменными $x_1^1, \ldots, x_\phi^1$, которые связаны с переменными из множества X^0, $X^0 = \{x_1, \ldots, x_\phi\}$, некоторыми формулами исчисления высказываний (ИВ)

$$x_j^1 \equiv \theta_j^1(x_1, \ldots, x_\phi).$$

Далее действуем по аналогии: переход из конфигурации K_1 в конфигурацию K_2 кодируется переменными x_j^2, $j \in \{1, \ldots, \phi\}$, которые связаны с переменными из X^1, $X^1 = \{x_1^1, \ldots, x_\phi^1\}$, формулами ИВ

$$x_j^2 \equiv \theta_j^2(x_1^1, \ldots, x_\phi^1).$$

И так далее.

Пусть $X^0, \ldots, X^e$ — множества переменных, соответствующие в указанном выше смысле конфигурациям $K_0, \ldots, K_e$ (K_e — заключительная конфигурация). Сопоставим программе R_{f_n} систему булевых уравнений (обозначаемую через $S(f_n)$)

$$\begin{cases} \mathop{\&}\limits_{j=1}^{\phi} \left(x_j^1 \equiv \theta_j^1(x_1, \ldots, x_\phi)\right) = 1 \\ \quad \ldots \\ \mathop{\&}\limits_{j=1}^{\phi} \left(x_j^e \equiv \theta_j^e\left(x_1^{e-1}, \ldots, x_\phi^{e-1}\right)\right) = 1 \end{cases} \tag{1.1}$$

Система $S(f_n)|_y$ является результатом подстановки в (1.1) значений некоторых переменных из множества $\left\{x_1^e, \ldots, x_\phi^e\right\}$, образующих вектор $y \in range\, f_n$. С применением подхода, описанного в [4], можно показать, что система $S(f_n)|_y$ совместна, и если $\tilde{x}$ — произвольное ее решение, то за линейное от $|\tilde{x}|$ время возможен переход к $x \in \{0,1\}^n : f_n(x) = y$. $\square$

От произвольной системы вида $S(f_n)|_y$ возможен эффективный (в общем случае за полиномиальное от n время) переход к одному урав-

нению вида КНФ=1. Этот переход осуществляется при помощи преобразований Цейтина [8]. При этом множество переменных разрастается (не более чем полиномиально), однако между множеством решений $S(f_n)|_y$ и множеством решений получаемого уравнения вида КНФ=1 существует биекция [5].

2. Преобразования в булевы уравнения высокоуровневых описаний алгоритмов, вычисляющих дискретные функции

2.1. Основные принципы и механизмы преобразования высокоуровневых программ в булевы уравнения.

Ядром пропозиционального подхода является идея представления булевыми уравнениями переходов из одной конфигурации вычислительной модели в другую. В «высокоуровневом случае» булевы переменные в новой конфигурации могут выражаться в виде суперпозиций булевых функций от многих переменных предыдущей конфигурации. Соответственно и условия перехода могут выражаться сложными булевыми функциями. При этом ситуация «Условие выполнено» эквивалентна принятию некоторой булевой функцией значения «1», ситуация «Условие не выполнено» эквивалентно принятию этой функцией значения «0».

Пример 1. Предположим, что вычисляется функция $f_3 : \{0,1\}^3 \to \{0,1\}^2$, заданная следующей программой
if $(x_1 \cdot x_2 \vee x_3 = 1)$ then $(y_1 := (x_1 \downarrow x_2),\ y_2 := x_2)$
else $(y_1 := (x_1 \to x_2),\ y_2 := x_2)$.
В соответствии с результатами, представленными выше, данной программе сопоставляется следующая система $S(f_3)$:

$$\begin{cases} \Big(y_1 \equiv \Big((x_1 \downarrow x_2) \cdot (x_1 \cdot x_2 \vee x_3) \vee (x_1 \to x_2) \cdot \overline{(x_1 \cdot x_2 \vee x_3)}\Big)\Big) = 1 \\ (y_2 \equiv x_2) = 1. \end{cases}$$

Задача обращения рассматриваемой функции, например, в точке $(y_1 = 1, y_2 = 1)$, таким образом, сводится к поиску решений системы $S(f_3)|_{y_1,y_2}$.

Преобразования программ, содержащих несколько вложенных условных операторов, осуществляется аналогичным образом.

Пример 2. Рассмотрим программу
if $g(x_{i_1}, \ldots, x_{i_k}) = 1$ then S
else if $h(x_{j_1}, \ldots, x_{j_l}) = 1$ then T
else U
В соответствии с описанными ранее принципами, в процессе преобразования данного оператора вводятся булевы переменные $y_1, \ldots, y_r$, для

значений которых имеются три альтернативы, определяемые блоками команд S, T или U. Предположим, что значение переменных y_i, $i \in \{1,\dots,r\}$, в блоке S определяется функциями $\lambda_i^S(x_1,\dots,x_n)$, в блоке T — функциями $\mu_i^T(x_1,\dots,x_n)$, в блоке U — функциями $\nu_i^U(x_1,\dots,x_n)$. Сказанное означает, что результатом преобразования данной программы является следующая система булевых уравнений

$$\begin{cases} \left(y_1 \equiv g\cdot\lambda_1^S \vee \overline{g}\cdot h\cdot\mu_1^T \vee \overline{g}\cdot\overline{h}\cdot\nu_1^U\right) = 1 \\ \qquad\qquad \dots \\ \left(y_r \equiv g\cdot\lambda_r^S \vee \overline{g}\cdot h\cdot\mu_r^T \vee \overline{g}\cdot\overline{h}\cdot\nu_r^U\right) = 1. \end{cases}$$

Далее мы более подробно останавливаемся на основных механизмах, используемых при преобразовании в булевы уравнения описаний алгоритмов на высокоуровневых процедурных языках.

Прежде всего отметим, что действия с памятью, которые выполняются в любом современном вычислительном устройстве, в целом аналогичны действиям с регистрами RAM. Поэтому вычисление, которое осуществляет преобразуемая высокоуровневая программа, можно рассматривать как последовательность изменений данных в памяти вычислительного устройства в моменты времени $0, 1, \dots, e$. В каждый момент i, $i \in \{0,\dots,e\}$, данные в памяти кодируются булевыми переменными, образующими множество X^i. Таким образом, множество X^0 содержит переменные, кодирующие входные данные, а множество X^e — переменные, кодирующие выходные данные рассматриваемого дискретного преобразования.

Из сказанного выше следует, что ключевую роль при пропозициональном кодировании RAM-программ играют механизмы связывания областей памяти RAM с булевыми переменными, кодирующими содержимое этих областей в определенные моменты времени. В применении к высокоуровневым программам это означает необходимость построения процедур, которые бы связывали некоторые переменные, встречающиеся в тексте программы (далее мы называем такие переменные переменными программы), с булевыми переменными, попадающими в итоговую систему булевых уравнений (эти переменные далее называются переменными кода).

Особо подчеркнем, что переменные программы и переменные кода представляют разные сущности. Переменные программы понимаются в традиционном для программирования смысле — это идентификаторы областей памяти. Переменные кода — это символы некоторого конечного алфавита. Продуманная организация связи между этими видами переменных может существенно сократить объем получаемой системы булевых уравнений.

Пример 3. Предположим, что требуется преобразовать в систему булевых уравнений программу, которая реализует регистр сдвига с ли-

нейной обратной связью (РСЛОС, [11]), заданной полиномом над GF(2) $Z^{19} + Z^{18} + Z^{17} + Z^{14} + 1$ (Z — формальная переменная).

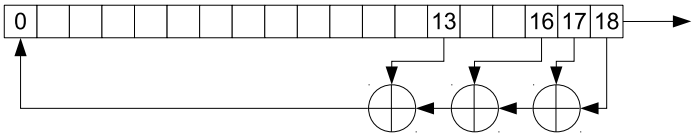

Рис. 1. РСЛОС с полиномом обратной связи $Z^{19} + Z^{18} + Z^{17} + Z^{14} + 1$.

В тексте программы, реализующей данный регистр, будут присутствовать 19 переменных программы, идентифицирующие в каждый момент времени те области памяти ЭВМ, которые хранят содержимое ячеек рассматриваемого регистра. Однако в каждый момент i, $i = 0, 1, 2, \ldots$, эти 19 переменных оказываются связанными с 19 переменными кода, образующими множество $X^i = \{x_1^i, \ldots, x_{19}^i\}$. Так, множество $X^0 = \{x_1^0, \ldots, x_{19}^0\} = \{x_1, \ldots, x_{19}\}$ образовано переменными, кодирующими начальное заполнение регистра. Множество

$$X^1 = \left\{x_1^1, \ldots, x_{19}^1\right\} = \{x_{20}, \ldots, x_{38}\}$$

образовано переменными, которые кодируют результат первого сдвига регистра. В соответствии с теоремой 1 переменные из X^0 и X^1 связываются следующей системой булевых уравнений (аналог системы (1.1)):

$$\begin{cases} (x_{20} \equiv x_{19} \oplus x_{18} \oplus x_{17} \oplus x_{14}) = 1 \\ (x_{21} \equiv x_1) = 1 \\ \qquad \ldots \\ (x_{38} \equiv x_{18}) = 1 \end{cases} \tag{2.1}$$

Можно заметить, что пропозициональный код, представляемый системой (2.1), является избыточным, поскольку переменные x_1 и x_{21} кодируют одну и ту же информацию. То же самое справедливо для переменных x_2 и $x_{22}, \ldots, x_{18}$ и x_{38}.

Избежать такого рода избыточности позволяет описываемая далее техника, использующая в процессе преобразования программы специальный словарь термов. Данный словарь, который обозначается через S, содержит термы над переменными кода преобразуемой программы. Словарь S является динамически расширяемым. В начальном состоянии в S находятся только переменные множества X^0 (то есть переменные, кодирующие входную информацию). В дальнейшем каждый новый терм, попадающий в словарь, является результатом интерпретации некоторой операции присваивания в программе следующего вида:

$$z = \Phi\left(z_{j_1}, \ldots, z_{j_r}\right).$$

Здесь z — переменная программы, которая связана через специальную структуру данных (далее «структура связи») с некоторой переменной

кода, а $\Phi(z_{j_1},\ldots,z_{j_r})$ — терм над переменными программы. Результатом обработки терма $\Phi(z_{j_1},\ldots,z_{j_r})$ является некоторый терм φ над переменными кода. Затем осуществляется проверка словаря S на предмет наличия в нем φ. Если $\varphi \notin S$, то данный терм связывается с новой переменной кода и добавляется в словарь S. Если же $\varphi \in S$, это означает, что кодируемая данным термом информация уже учтена в пропозициональном коде программы, и ей соответствует отдельная переменная кода x'. В этом случае переменная программы z связывается с переменной кода x' через структуру связи. Данный прием позволяет избегать ввода переменных кода, кодирующих одну и ту же информацию. При этом каждой переменной кода $\tilde{x} \in X^i$, $i \in \{1,\ldots,e\}$, ставится в соответствие булево уравнение вида

$$(\tilde{x} \equiv \varphi(x_{j_1},\ldots,x_{j_k})) = 1,$$

где $\{x_{j_1},\ldots,x_{j_k}\} \subset \bigcup\limits_{s=0}^{i-1} X^s$, $\varphi(x_{j_1},\ldots,x_{j_k})$ — терм, помещаемый в словарь S.

Пример 4. Снова рассмотрим преобразование алгоритма из примера 3. На начальном шаге в словарь термов включаются переменные кода, образующие множество $X^0 = \{x_1,\ldots,x_{19}\}$. При этом связи между переменными программы и переменными из X^0 выглядят следующим образом: При интерпретации первого сдвига РСЛОС в словарь S заносится

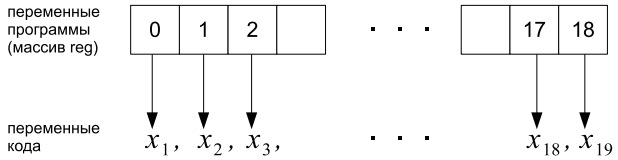

Рис. 2. Связь переменных программы с переменными кода до сдвига регистра.

терм $\varphi = x_{19} \oplus x_{18} \oplus x_{17} \oplus x_{14}$, кодирующий функцию обратной связи. С этим термом связывается новая переменная кода x_{20} посредством булева уравнения $(x_{20} \equiv x_{19} \oplus x_{18} \oplus x_{17} \oplus x_{14}) = 1$. Связи между элементами словаря S, переменными программы и переменными кода обновляются посредством изменений в структурах связи так, как показано на рисунке 4. Особо отметим, что для переменных программы, которые

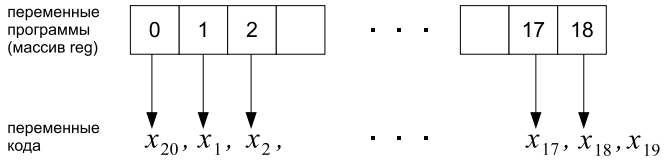

Рис. 3. Связь переменных программы с переменными кода после сдвига регистра.

идентифицируют содержимое ячеек РСЛОС, помеченных числами от 1 до 18, новые термы не создаются — эти переменые связываются с переменными кода $x_1, \ldots, x_{18}$ соответственно.

2.2. Преобразование условных переходов.

Преобразование условного оператора начинается с анализа условного выражения. Условный оператор — это оператор вида (в псевдокоде)

if $\Phi(z_{j_1}, \ldots, z_{j_r}) = 1$ then Ветвь 1;
else Ветвь 2;

здесь терм условного выражения $\Phi(z_{j_1}, \ldots, z_{j_r})$ — терм над переменными программы. Первый шаг преобразования заключается в интерпретации терма условного выражения Φ, результатом чего является терм ψ над переменными кода.

Ветвью условного оператора может быть произвольный блок операторов. Интерпретация ветви — это последовательная интерпретация всех операторов соответствующего блока. Особо отметим, что одна и та же переменная программы может фигурировать в различных ветвях условного оператора. Пусть z — переменная программы, являющаяся левым операндом операции присваивания, которая выполняется как в 1-й ветви, так и во 2-ой ветви. Пусть Δ_1 и Δ_2 — термы над переменными программы, являющиеся правыми операндами соответствующих операций присваивания в 1-й и 2-й ветвях. В этой ситуации переменой z сопоставляется переменная кода $\tilde{x}$, которая связана с термом

$$\psi \cdot \delta_1 \vee \overline{\psi} \cdot \delta_2$$

словаря S при помощи булева уравнения

$$\left(\tilde{x} \equiv \psi \cdot \delta_1 \vee \overline{\psi} \cdot \delta_2\right) = 1,$$

здесь δ_1 и δ_2 — термы над переменными кода, полученные в результате интерпретации термов Δ_1 и Δ_2 соответственно. Для хранения пар вида (z, δ_1) и (z, δ_2) используются отдельные списки (в данном случае списки L_1 и L_2 соответственно).

Далее рассматривается конструкция из нескольких вложенных условных операторов.

if $\Phi_1(\ldots) = 1$ then Ветвь 1;
else if $\Phi_2(\ldots) = 1$ then Ветвь 2;
...
else if $\Phi_n(\ldots) = 1$ then Ветвь n;
else Ветвь $n + 1$;

В соответствии со сказанным выше каждому терму Φ_i, $i = 1, \ldots, n$, сопоставляется терм ψ_i над множеством переменных кода. Каждой ветви с номером $i \in \{1, \ldots, n + 1\}$, ставится в соответствие список L_i,

образованный парами вида (z, δ_i). Пусть z — переменная программы, выступающая в качестве левого операнда операций присваивания в ветвях с номерами от 1 до $n+1$ рассматриваемого условного оператора. Тогда переменной z сопоставляется переменная кода $\tilde{x}$, которая связана с заносимым в словарь S термом

$$\psi_1 \cdot \delta_1 \vee \overline{\psi_1} \cdot \psi_2 \cdot \delta_2 \vee \ldots \vee \overline{\psi_1} \cdot \overline{\psi_2} \cdot \ldots \cdot \overline{\psi_{n-1}} \cdot \psi_n \cdot \delta_n \vee \overline{\psi_1} \cdot \ldots \cdot \overline{\psi_n} \cdot \delta_{n+1},$$

посредством булева уравнения

$$\Big(\tilde{x} \equiv \psi_1 \cdot \delta_1 \vee \overline{\psi_1} \cdot \psi_2 \cdot \delta_2 \vee \ldots \vee \vee \overline{\psi_1} \cdot \overline{\psi_2} \cdot \ldots \cdot \overline{\psi_{n-1}} \cdot \psi_n \cdot \delta_n \vee \overline{\psi_1} \cdot \ldots \cdot \overline{\psi_n} \cdot \delta_{n+1}\Big) = 1.$$

2.3. Кодирование целочисленных операций.

Многие дискретные функции, возникающие в практических приложениях, реализуют операции над целыми числами. При преобразовании соответствующих алгоритмов целые числа представляются двоичными векторами.

Пример 5. Результатом преобразования программы, которая выполняет сложение двух неотрицательных целых чисел, представленных двоичными векторами одинаковой длины, будет следующая система булевых уравнений (здесь предполагается, что для сложения используется алгоритм «столбик»):

$$\begin{cases} (c_0 \equiv a_0 \oplus b_0) = 1 \\ (p_0 \equiv a_0 \cdot b_0) = 1 \\ (p_j \equiv \mathrm{maj}(a_j, b_j, p_{j-1})) = 1,\ j = \overline{1,\, n} \\ (c_i \equiv a_i \oplus b_i \oplus p_{i-1}) = 1,\ i = \overline{1,\, n} \\ (c_n \equiv p_{n-1}) = 1. \end{cases}$$

Запись «$\mathrm{maj}(x, y, z)$» обозначает терм $x \cdot y \vee x \cdot z \vee y \cdot z$. Для кодирования битов переноса вводятся новые переменные кода p_i, $i = \overline{0,\, n-1}$. Аналогичным образом выглядя механизмы преобразования операций умножения, вычитания и сравнения пар целых чисел.

3. Программная трансляция алгоритмов вычисления некоторых криптографических функций в булевы уравнения

В данном разделе мы приводим краткий перечень результатов, полученных с применением программного комплекса Transalg, который был

создан в соответствии с принципами, изложенными в предыдущих пунктах.

Одной из наиболее наглядных областей применения описанной техники трансляции алгоритмов является криптография. В статье [10] было отмечено, что пропозициональные коды алгоритмов шифрования можно использовать для построения аргументированно трудных тестов для разнообразных решателей комбинаторных задач (в том числе для SAT-решателей). В дальнейшем криптоанализ, рассматриваемый как процесс поиска решений булевых уравнений (в частности, SAT-задач), стали называть логическим криптоанализом [12]. Логический криптоанализ оказался эффективным в применении к некоторым генераторам ключевого потока [7, 3].

Генераторы поточного шифрования — это быстро вычислимые дискретные функции, преобразующие двоичные последовательности конечной длины (инициализирующие последовательности) в бесконечные периодические двоичные последовательности (ключевой поток). Задача криптоанализа генератора заключается в нахождении инициализирующей последовательности по известному фрагменту ключевого потока и алгоритму функционирования генератора. Транслируя данный алгоритм, записанный на специальном процедурном языке, Transalg строит систему булевых уравнений, кодирующих процесс порождения произвольного фрагмента ключевого потока. Подстановка в полученную систему анализируемого фрагмента ключевого потока дает систему, из решения которой можно эффективно выделить искомый секретный ключ (инициализирующую последовательность). Наиболее значимым на текущий момент практическим результатом в этом направлении является успешный логический криптоанализ генератора поточного шифрования A5/1, реализованный в специально построенной для этой цели Grid-среде [3].

Интересные результаты дало применение комплекса Transalg для кодирования алгоритма DES. Данный алгоритм фигурирует в массе источников (например, в [11]), поэтому его описание здесь не приводится. Первой работой, в которой был приведен пропозициональный код DES, стала статья [12].

Одним из базовых примитивов шифра DES являются перестановки. Перестановки в DES задаются таблицами натуральных чисел, которые не являются секретными. Произвольная перестановка применяется к некоторому множеству битов обрабатываемого слова. При пропозициональном кодировании операции перестановки Transalg не создает новых переменных кода — как видно из материала пункта 2.1, транслятору достаточно обновить связи переменных программы с уже существующими элементами словаря термов S. Данный факт означает, что операции перестановки не вносят в пропозициональный код алгоритма новой информации, никак не усложняя тем самым задачу логического крипто-

анализа. В результате применения транслятора Transalg к кодированию алгоритма DES был получен (см. таблицу 3) пропозициональный код (в КНФ), существенно более экономный, чем код, приведенный в [12].

Таблица 1.
Кодирование процесса шифрования алгоритмом DES одного блока открытого текста длиной 64 бита.

Программный комплекс Transalg				F. Massacci, L. Marraro, [12]	
Без минимизации		Минимизация (Espresso, [14])			
Переменные	Дизъюнкты	Переменные	Дизъюнкты	Переменные	Дизъюнкты
1912	37888	1912	26400	10336	61935

Заключение

Мы полагаем, что преобразования, представленные в работе, найдут применение при исследовании разнообразных систем, поведение которых описывается полиномиально вычислимыми дискретными функциями. Сказанное касается, прежде всего, дискретно-автоматных динамических систем — переходы в последующие состояния в этих системах происходят в дискретные моменты времени, и функции, задающие эти переходы, как правило оказываются эффективно вычислимыми (за полиномиальное от объема входных данных время). Устанавливать некоторые свойства такого рода систем можно, преобразуя алгоритмы вычисления функций переходов в булевы уравнения и добавляя при необходимости к получаемым системам дополнительные ограничения. Для осуществления указанных преобразований можно использовать программный комплекс Transalg. В ближайшее время данный подход будет применен к исследованию динамических свойств дискретных моделей генных сетей [6].

Список литературы

1. Ахо А. Построение и анализ вычислительных алгоритмов / А. Ахо, Дж. Хопкрофт, Дж. Ульман. – М. : Мир, 1979. – 536 с.
2. Гэри М. Вычислительные машины и труднорешаемые задачи / М. Гэри, Д. Джонсон. – М. : Мир, 1982. – 416 с.
3. Решение задач криптоанализа поточных шифров в распределенных вычислительных средах / М. А. Посыпкин, О. С. Заикин, Д. В. Беспалов, А. А. Семенов // Тр. ИСА РАН. – 2009. – Т. 46. – С. 119–137.
4. Семенов А. А. Трансляция алгоритмов вычисления дискретных функций в выражения пропозициональной логики / А. А. Семенов // Прикладные алгоритмы в дискретном анализе. – Иркутск : Изд-во ИГУ, 2008. – Вып. 2. – С. 70–98. – (Дискретный анализ и информатика).
5. Семенов А. А. О преобразованиях Цейтина в логических уравнениях / А. А. Семенов // Прикладная дискретная математика. – 2009. – № 4. – С. 28–50.

6. Системная компьютерная биология / под ред. Н. А. Колчанова, С. С. Гончарова, В. А. Лихошвая, В. А. Иванисенко. - Новосибирск : Изд.-во СО РАН, 2008. – 767 с.
7. SAT-подход в криптоанализе некоторых систем поточного шифрования / А. А. Семенов, О. С. Заикин, Д. В. Беспалов, А. А. Ушаков // Вычисл. технологии. – 2008. – Т. 13, № 6. – С. 134–150.
8. Цейтин Г. С. О сложности вывода в исчислении высказываний / Г. С. Цейтин // Зап. науч. семинаров ЛОМИ АН СССР. – 1968. – Т. 8. – С. 234–259.
9. Cook S. A. The complexity of theorem-proving procedures / S. A. Cook // Proc. 3rd Ann. ACM Symp. on Theory of Computing (STOC 71). – ACM, 1971. – P. 151–159.
10. Cook S. A. Finding hard instances of the satisfiability problem: A survey / S. A. Cook, G. Mitchel // DIMACS Series in Discrete Mathematics and Theoretical Computer Science. – 1997. – Vol. 35. – P. 1–17.
11. Menezes A. Handbook of Applied Cryptography / A. Menezes, P. Oorschot, S. Vanstone. – CRC Press, 1996. – 657 p.
12. Massacci F. Logical Cryptanalysis as a SAT Problem / F. Massacci, L. Marraro // Journal of Automated Reasoning. – 2000. – Vol. 24, N 1–2. – P. 165–203.
13. Prestwich S. CNF encodings // In Handbook of Satisfiability / S. Prestwich ; eds.: A. Biere, M.Heule, H. van Maaren, T. Walsh. – IOS Press, 2009. – P. 75–97.
14. URL: http://embedded.eecs.berkeley.edu/pubs/downloads/espresso

---

I. V. Otpuschennikov, A. A. Semenov

Transformations of discrete functions calculation algorithms to boolean equations

Abstract. The article is devoted to the problem of transforming algorithmic descriptions of discrete functions to their equational descriptions in the form of boolean equations. We propose an approach based on propositional encoding procedures for binary random access machine programs. These procedures are used to build transformations of high-level descriptions of discrete functions calculation algorithms to boolean equations.

Keywords: discrete functions; boolean equations; random access machines.

Отпущенников Илья Владимирович, программист, лаборатория дискретного анализа и прикладной логики, Институт динамики систем и теории управления СО РАН, 664033, Иркутск, ул. Лермонтова, 134, тел.: (3952)453054 (otilya@yandex.ru)

Семёнов Александр Анатольевич, кандидат технических наук, зав. лабораторией дискретного анализа и прикладной логики, Институт динамики систем и теории управления СО РАН, 664033, Иркутск, ул. Лермонтова, 134, тел.: (3952)453054 (biclop@rambler.ru)

Otpuschennikov Ilya, Institute for System Dynamics and Control Theory of SB RAS, Lermontov str., 134, Post Box 292 664033, Irkutsk, Russia Phone: (3952)453054 (otilya@yandex.ru)

Semenov Alexander, Institute for System Dynamics and Control Theory of SB RAS, Lermontov str., 134, Post Box 292 664033, Irkutsk, Russia Phone: (3952)453054 (biclop@rambler.ru)