ДИСКРЕТНЫЕ ФУНКЦИИ И АВТОМАТЫ

УДК 519.7

А.С. Игнатьев, А.А. Семенов, А.Е. Хмельнов

ИСПОЛЬЗОВАНИЕ ДВОИЧНЫХ ДИАГРАММ РЕШЕНИЙ В ЗАДАЧАХ ОБРАЩЕНИЯ ДИСКРЕТНЫХ ФУНКЦИЙ[1]

Рассматривается возможность использования двоичных диаграмм решений (BDD) в задачах обращения дискретных функций из одного класса, известного своими многочисленными приложениями. Описывается метод решения систем логических уравнений, сочетающий элементы SAT-подхода и подхода, использующего ROBDD-представления характеристических функций рассматриваемых систем. Приводится описание архитектуры «типового» BDD-решателя систем логических уравнений.

Ключевые слова**:** *двоичные диаграммы решений*, *задачи обращения дискретных функций.*

Двоичные диаграммы решений (Binary Decision Diagram, BDD) – это специальный вид направленных помеченных графов, посредством которых можно представлять булевы функции. Образно говоря, BDD (а точнее ROBDD) решают проблему компактного представления булевых функций «в классе графов». Несмотря на то, что первое упоминание BDD (под названием «двоичные решающие программы») встречается в статье [1] (1959 г.), осознание значимости BDD для математической и прикладной кибернетики пришло после фундаментальной работы Р.Е. Брайанта [2] (1986 г.).

В настоящей статье анализируется применимость BDD в задачах обращения дискретных функций из класса $\mathfrak{I}$ (см. [3]), известного своими многочисленными приложениями в дискретной математике и математической кибернетике (в частности, в криптографии).

В первом разделе приводятся основные понятия и определения. Во втором разделе описан BDD-подход к решению систем логических уравнений (особый акцент сделан на системах, кодирующих задачи обращения функций из класса $\mathfrak{I}$). В третьем разделе приведено краткое описание «гибридного» подхода к решению логических уравнений – в рамках данного подхода BDD используются в качестве модифицируемых структур данных, сохраняющих информацию о процессе DPLL-вывода. В четвертом разделе анализируются особенности архитектуры «типового» BDD-решателя систем логических уравнений.

[1] Работа выполнена при поддержке гранта РФФИ № 07-01-00400-а и при поддержке гранта Президента РФ НШ-1676.2008.1.

1. Основные определения и понятия

Логическими уравнениями называются выражения вида

$$L(x_1,...,x_n)=0,\ L(x_1,...,x_n)=1,$$

где через $L(x_1,...,x_n)$ обозначается формула (исчисления высказываний), задающая некоторую булеву функцию над множеством булевых переменных $X=\{x_1,...,x_n\}$. Тот факт, что при подстановке $x_1=\alpha_1,...,x_n=\alpha_n$, $\alpha_i\in\{0,1\}$, $i\in\{1,...,n\}$, формула $L(x_1,...,x_n)$ принимает значение $\beta\in\{0,1\}$ обозначается через $L(x_1,...,x_n)|_{(\alpha_1,...,\alpha_n)}=\beta$ (или более кратко через $L|_{(\alpha_1,...,\alpha_n)}=\beta$). Если существует такой набор $(\alpha_1,...,\alpha_n),\alpha_i\in\{0,1\},i\in\{1,...,n\}$, значений переменных $x_1,...,x_n$, что $L|_{(\alpha_1,...,\alpha_n)}=\beta$, то данный набор называется решением логического уравнения $L(x_1,...,x_n)=\beta$. Системой логических уравнений называется выражение вида

$$\begin{cases} L_1(x_1,...,x_n)=\beta_1, \\ .....\ , \\ L_m(x_1,...,x_n)=\beta_m \end{cases} \qquad (1)$$

при фиксированных значениях булевых констант $\beta_j, j\in\{1,...,m\}$. Решением системы (1) называется набор значений переменных $(x_1,...,x_n)$, являющийся решением всех уравнений системы. Если такого набора не существует, то говорят, что система (1) несовместна. Очевидно, что произвольная система логических уравнений вида (1) может быть эффективно преобразована к виду

$$\begin{cases} L'_1(x_1,...,x_n)=1, \\ .....\ , \\ L'_m(x_1,...,x_n)=1. \end{cases} \qquad (2)$$

Система (2) эквивалентна одному логическому уравнению вида

$$L'_1(x_1,...,x_n)\cdot...\cdot L'_m(x_1,...,x_n)=1.$$

Введем в рассмотрение булеву функцию $L:\{0,1\}^n\to\{0,1\}$, определяемую следующим образом:

$$L(\alpha)=(L'_1(x_1,...,x_n)\cdot...\cdot L'_m(x_1,...,x_n))|_\alpha,\ \alpha\in\{0,1\}^n.$$

Данную функцию назовем характеристической функцией системы (1).

Двоичная диаграмма решений (BDD) (см., например [4]) определяется как направленный ациклический граф, в котором выделена одна вершина с входной степенью 0, называемая корнем, и две вершины с выходной степенью 0, называемые терминальными. Терминальные вершины помечены константами 0 и 1. Все остальные вершины помечаются переменными из множества $X=\{x_1,...,x_n\}$. Из любой вершины, за исключением терминальных, выходят в точности 2 дуги. Одну дугу обычно рисуют пунктирной, а другую – сплошной линией. Дуга, обозначенная пунктиром, называется low-ребром, дуга, обозначенная сплошной линией, – high-ребром. Простейшие примеры BDD получаются объединением одноименных листьев в двоичных деревьях решений.

Деревья решений (см, например [4]) суть помеченные деревья, используемые в различных разделах дискретной математики. Обозначим через $\{0,1\}^n$, $n \in \mathrm{N}$, множество всех двоичных слов (последовательностей) длины n. Функции вида $f_n : \{0,1\}^n \to \{0,1\}$ называются булевыми функциями над множеством булевых переменных $X = \{x_1,...,x_n\}$. Пусть f_n – произвольная всюду определенная булева функция. Сопоставим ей двоичное дерево решений, построенное в соответствии с перечисленными ниже правилами. Вершины (узлы) дерева соответствуют булевым переменным. Пунктирное ребро, исходящее из узла, помеченного переменной $x_i, i \in \{1,...,n\}$, означает, что данная переменная принимает значение 0, сплошное ребро соответствует тому, что x_i принимает значение 1. Листья дерева помечены значениями рассматриваемой функции при соответствующих наборах значений булевых переменных. Произвольный путь из корня в лист, помеченный $\alpha \in \{0,1\}$, определяет набор или семейство наборов значений переменных, на которых рассматриваемая функция принимает значение α. Пусть $f_n : \{0,1\}^n \to \{0,1\}$ – произвольная всюду определенная булева функция над множеством булевых переменных $X = \{x_1,...,x_n\}$ и $T(f_n)$ – ее дерево решений. Если объединить в одну вершину все листья дерева $T(f_n)$, помеченные нулем, и то же самое проделать с листьями, помеченными единицей, получится BDD. Если произвольный путь π в BDD из корня в терминальную вершину не содержит вершин, помеченных одинаковыми переменными, и его прохождение подчинено общему для всех путей порядку (например, $x_1 \prec x_2 \prec ... \prec x_{n-1} \prec x_n$), то такая BDD называется упорядоченной (OBDD). В записи «$x_1 \prec ... \prec x_n$» здесь и далее подразумевается, что корень рассматриваемой OBDD помечен переменной x_1.

В произвольной OBDD можно выделять фрагменты (подграфы), которые сами являются OBDD. Для этой цели достаточно объявить соответствующую нетерминальную вершину корнем BDD. Идея сокращенной OBDD (кратко «ROBDD») заключается в «склейке» повторяющихся фрагментов: ROBDD-граф не должен содержать одинаковых OBDD-подграфов меньших размерностей. Р. Брайантом в [2] было показано, что произвольная всюду определенная булева функция при фиксированном порядке означивания переменных имеет однозначное (с точностью до изоморфизма соответствующих графов) ROBDD-представление.

2. ROBDD-подход к решению логических уравнений. Обращение дискретных функций

Пусть $L(\cdot)$ – характеристическая функция системы (1) и $B(L)$ – ее ROBDD-представление. Очевидно, что за линейное от числа вершин $B(L)$ время можно найти некоторое решение системы (1) либо убедиться в ее несовместности. Если имеются ROBDD-представления булевых функций, находящихся в левых частях уравнений системы (2), то ROBDD-представление характеристической функции системы (1) можно получить при помощи алгоритма Apply, описанного в работе [2]. Данный алгоритм по паре ROBDD $B\left(f^1\right)$, $B\left(f^2\right)$, представляющих булевы функции f^1 и f^2 над множеством булевых переменных $X = \{x_1,...,x_n\}$, позволя-

ет построить ROBDD, представляющую булеву функцию $f^3 = f^1 * f^2$, где «$*$» – произвольная бинарная логическая связка. Порядок переменных в $B\left(f^1\right)$ и $B\left(f^2\right)$ при этом обязательно должен совпадать. При выполнении этого требования сложность процедуры Apply ограничена сверху величиной $O\left(|B\left(f^1\right)| \cdot |B\left(f^2\right)|\right)$; через $|B|$ здесь и далее обозначается число вершин в ROBDD B. Факт построения посредством Apply ROBDD $B\left(f^3\right)$ по известным $B\left(f^1\right)$ и $B\left(f^2\right)$ обозначается следующим образом:

$$B\left(f^3\right) = \mathrm{Apply}\left(B\left(f^1\right) * B\left(f^2\right)\right).$$

Таким образом, если известны ROBDD-представления булевых функций, стоящих в левых частях уравнений системы (2) (соответственно $B(L'_1)$, $B(L'_2)$, …, $B(L'_m)$), то ROBDD $B(L)$ характеристической функции системы (1) может быть найдена, например, в соответствии со следующей рекурсивной схемой:

$$B_1 = \mathrm{Apply}(B(L'_1) \cdot B(L'_2)), B_2 = \mathrm{Apply}(B_1 \cdot B(L'_3)), \ldots$$
$$\ldots, B_{m-1} = \mathrm{Apply}(B_{m-2} \cdot B(L'_m)),\ B_{m-1} = B(L).$$

Сложность данной процедуры в общем случае, конечно же, есть экспонента от числа переменных, фигурирующих в системе (1).

Дискретной функцией (вообще говоря, частичной) называется любое отображение вида $f_n : \{0,1\}^n \to \{0,1\}^*$, где $\{0,1\}^* = \bigcup_{n \in \mathrm{N}} \{0,1\}^n$ (дискретные функции вида $f_n : \{0,1\}^n \to \{0,1\}$ называются булевыми). Через $Dom\ f_n$ и $Ran\ f_n$ обозначаются соответственно область определения и область значений функции f_n. Если $Dom\ f_n = \{0,1\}^n$, то функция f_n называется всюду определенной или кратко «тотальной» (см. [5]). Если существует программа $M(f)$ для детерминированной машины Тьюринга с входным алфавитом $\Sigma = \{0,1\}$, которая вычисляет любую дискретную функцию из некоторого натурального семейства $f = \{f_n\}_{n \in \mathrm{N}}$, то про f говорят, что оно образовано алгоритмически вычислимыми дискретными функциями.

Пусть $f = \{f_n\}_{n \in \mathrm{N}}$ – семейство тотальных алгоритмически вычислимых дискретных функций. Стандартным образом (см. [6]) можно определить временную сложность программы $M(f)$, вычисляющей функции данного семейства как функцию от n (длины входа). Класс $\Im$ образован всеми натуральными семействами дискретных функций, которые вычислимы за полиномиальное время. Задача обращения произвольной функции $f_n \in f, f \in \Im$, «в точке» $y \in Ran\ f_n$ заключается в нахождении $x \in Dom\ f_n$, такого, что $f_n(x) = y$. Практическая значимость класса $\Im$ обосновывается хотя бы тем фактом, что в него попадает бо́льшая часть функций, используемых в криптографических алгоритмах в качестве шифрующих преобразований.

Интересные результаты применительно к задачам обращения некоторых типов функций из $\Im$ дает пропозициональный подход (см. [7, 8]). В соответствии с дан-

ным подходом алгоритм вычисления функции f_n представляется в виде системы логических уравнений вида $C(f_n)=1$. Здесь через $C(f_n)$ обозначена КНФ над множеством булевых переменных $U=X\bigcup X'$, причем $X=\{x_1,...,x_n\}$, где $x_i, i\in\{1,...,n\}$ – булевы переменные, символизирующие аргументы функции f_n и называемые далее переменными функции f_n, а множество X' образовано булевыми переменными, вводимыми в результате так называемых преобразований Цейтина (см. [7, 9]). Переменные из X' далее будем называть цейтиновскими. В [7] показано, что переход к уравнениям вида КНФ = 1 в задачах обращения функций из класса $\Im$ осуществляется эффективно (в общем случае за полиномиальное время). Подстановка в уравнение $C(f_n)=1$ вектора $y\in Ran\ f_n$ дает уравнение $C_y(f_n)=1$, которое всегда имеет решение. Из произвольного решения данного уравнения можно эффективно выделять прообраз y в $\{0,1\}^n$ относительно отображения f_n. Далее будем говорить, что $C_y(f_n)$ есть КНФ, кодирующая задачу обращения функции f_n в точке y.

Для решения логических уравнений вида КНФ = 1 применяются специальные программы (SAT-решатели), в основе которых лежат алгоритмы логического вывода.

3. ROBDD как модифицируемые базы булевых ограничений

Одним из наиболее удачных алгоритмов логического вывода в применении к задаче пропозициональной выполнимости можно считать алгоритм DPLL, дополненный процедурой Clause Learning [10]. Процедура Clause Learning (CL) представляет собой механизм учета информации о предыстории поиска посредством итеративного добавления к рассматриваемой КНФ новых ограничений-дизъюнктов. Отрицательной стороной этого процесса является возможность «переполнения памяти» из-за чрезмерного роста базы ограничений. На практике данная проблема решается за счет чистки базы ограничений путем удаления из нее дизъюнктов, имеющих малую значимость. Однако все известные критерии значимости имеют характер эвристик. Данный факт, вообще говоря, не гарантирует завершение поиска за конечное число шагов (то есть алгоритм может потерять полноту). Перечисленные вопросы отражены в работе [11].

Прямое применение описанной выше ROBDD-стратегии для решения систем логических уравнений, кодирующих задачи обращения функций из $\Im$, представляется малоперспективным. В данном пункте мы описываем «гибридный» алгоритм решения задач обращения функций из $\Im$, использующий преимущества SAT- и ROBDD- подходов.

Основой данного подхода является понятие ядра логического вывода (в частности, нас будет интересовать ядро DPLL-вывода).

Определение 1. *Ядром DPLL-вывода в задаче поиска решений уравнения вида* $C(u_1,...,u_k)=1$, *где* $C(u_1,...,u_k)$ – КНФ *над множеством булевых переменных* $U=\{u_1,...,u_k\}$, *назовем такое множество* $U^{\ker}(C)\subseteq U$, *произвольное означивание переменных которого либо индуцирует вывод значений всех остальных переменных из* $U\setminus U^{\ker}(C)$ *по правилу единичного дизъюнкта, либо индуцирует вывод*

конфликта. Ядро $U^{\ker}(C):U^{\ker}(C)=U$ *называется тривиальным. Ядро наименьшей мощности называется минимальным и обозначается через* $U_*^{\ker}(C)$.

Для многих важных в практическом отношении классов КНФ процедура выявления некоторого нетривиального ядра не представляет трудности. В данном случае речь идет, прежде всего, о КНФ, кодирующих задачи обращения дискретных функций. Пусть f_n – дискретная функция из класса $\Im$ и $C_y(f_n)$ – КНФ над множеством булевых переменных U, кодирующая задачу обращения функции f_n в некоторой (вообще говоря, произвольной) точке $y \in Ran\, f_n$. Прямым следствием свойств преобразований Цейтина [7] является тот факт, что $U_*^{\ker}\left(C_y(f_n)\right) \subseteq X$, где X – множество переменных функции f_n (тем самым X образует некоторое ядро DPLL-вывода для уравнения $C_y(f_n)=1$). Детальное доказательство данного результата мы здесь не приводим, ввиду его неоправданно большого размера. Основная его идея, тем не менее, совершенно прозрачна и заключается в том, что подстановка в $C_y(f_n)$ произвольного вектора значений переменных входа функции f_n либо индуцирует вывод по правилу единичного дизъюнкта значений всех цейтиновских переменных в $C_y(f_n)$ и дает тем самым выполняющий $C_y(f_n)$ набор, либо приводит к выводу конфликта.

Рассмотрим систему логических уравнения вида (2) над множеством булевых переменных $U=\{u_1,...,u_k\}$. Предположим, что построена декомпозиция этой системы вида

$$\begin{cases} W_1(y_1,...,y_s)=1, \\ ..... , \\ W_k(y_1,...,y_s)=1, \\ C(z_1,...,z_t)=1, \end{cases} \tag{3}$$

$U=\{y_1,...,y_s\}\bigcup\{z_1,...,z_t\}$. Здесь $C(z_1,...,z_t)$ – КНФ, для которой известно некоторое нетривиальное ядро $Z^{\ker}(C)$, причем $Z^{\ker}(C)\subseteq\{y_1,...,y_s\}$. Через $B(W)$ обозначим ROBDD-представление характеристической функции системы

$$\begin{cases} W_1(y_1,...,y_s)=1, \\ ..... , \\ W_k(y_1,...,y_s)=1. \end{cases} \tag{4}$$

Определение 2. *Пусть* $B(W)$ *– ROBDD-представление булевой функции* $W:\{0,1\}^s\to\{0,1\}$ *над множеством булевых переменных* $Y=\{y_1,...,y_s\}$. *Назовем путь в* $B(W)$ *от корня к терминальной вершине* «1» *полным относительно множества* $Y', Y'\subseteq Y$, *если прохождение этого пути задает присвоение соответствующих значений всем переменным из* Y'.

Следующий результат в несколько иной трактовке был получен в работе [12] .

Теорема 1. Прохождение любого пути в ROBDD $B(W)$ из корня в терминальную вершину «1», который полон относительно множества $Z^{\ker}(C)$, индуцирует

подстановку в КНФ $C(z_1,...,z_t)$ значений переменных из $Z^{\ker}(C)$. Результатом этой подстановки является либо вывод по правилу единичного дизъюнкта конфликта, либо вывод выполняющего набора для $C(z_1,...,z_t)$. В последнем случае имеем некоторое решение исходной системы.

Краткое доказательство. Рассмотрим произвольный путь в ROBDD $B(W)$ из корня в терминальную вершину «1», который полон относительно множества $Z^{\ker}(C)$. Его прохождение индуцирует присвоение значений истинности всем переменным из $Z^{\ker}(C)$. Поскольку $Z^{\ker}(C)$ – ядро DPLL вывода, то подстановка произвольного вектора значений переменных из $Z^{\ker}(C)$ в $C(z_1,...,z_t)$ и последующая стратегия распространения булевых ограничений (BCP-стратегия, см. [10]) либо приводит к выводу конфликта, либо порождает вывод значений всех остальных переменных из $Z \setminus Z^{\ker}(C)$. В последнем случае имеем набор значений переменных из Z, который, с одной стороны, соответствует некоторому пути в ROBDD $B(W)$ из корня в терминальную вершину «1» и в этом смысле «определяет» некоторое решение системы (4). С другой стороны, данный набор выполняет КНФ $C(z_1,...,z_t)$ и тем самым дает некоторое решение системы (3). Отметим, что время работы процедуры распространения булевых ограничений при подстановке в $C(z_1,...,z_t)$ значений переменных из $Z^{\ker}(C)$ в общем случае линейно от числа вхождений переменных в $C(z_1,...,z_t)$. Перечисленные факты доказывают справедливость утверждения теоремы.

Данная теорема дает основу для следующего алгоритма (алгоритм А1) поиска решений систем уравнений вида (3) с явно выделенным нетривиальным ядром DPLL-вывода для КНФ $C(z_1,...,z_t)$ (как было сказано выше, именно такого рода системы возникают при пропозициональном кодировании задач обращения дискретных функций из класса $\Im$).

Алгоритм А1. Выбирается очередной путь π в ROBDD $B(W)$ и осуществляется подстановка соответствующих значений ядерных переменных в $C(z_1,...,z_t)$. Если прохождение пути π порождает вывод конфликта в $C(z_1,...,z_t)$, то этот путь исключается из ROBDD $B(W)$. ROBDD с исключенным (запрещенным) путем π представляет собой базу ограничений, в которую занесена информация о некотором множестве конфликтных присвоений.

Тривиальная схема исключения путей в ROBDD может быть реализована посредством алгоритма Apply. Действительно, запрещение некоторой последовательности одновременных присвоений $y_{i_1}=\alpha_{i_1},...,y_{i_r}=\alpha_{i_r}$ равносильно конъюнктивному приписыванию к системе (3) нового ограничения – уравнения $\left(y_{i_1}^{\neg\alpha_{i_1}} \vee ... \vee y_{i_r}^{\neg\alpha_{i_r}}\right)=1$, и соответственно модификацией ROBDD $B(W)$, в которой данный запрет учтен, является ROBDD

$$\mathrm{Apply}\left(B(W)\cdot B\left(y_{i_1}^{\neg\alpha_{i_1}} \vee ... \vee y_{i_r}^{\neg\alpha_{i_r}}\right)\right),$$

где через $B\left(y_{i_1}^{\neg\alpha_{i_1}} \vee ... \vee y_{i_r}^{\neg\alpha_{i_r}}\right)$ обозначено ROBDD-представление булевой функ-

ции $\left(y_{i_1}^{\neg\alpha_{i_1}} \vee \ldots \vee y_{i_r}^{\neg\alpha_{i_r}}\right)$. В соответствии с оценкой сложности алгоритма Apply описанная процедура может потребовать $O\left(|B(W)| \cdot |Z^{\ker}(C)|\right)$ шагов.

В [12] предложена альтернативная схема исключения путей из ROBDD, названная *D*-стратегией. Сложность данной процедуры оценивается величиной $O(|B(W)|)$. Кроме этого, в [12] отмечено, что для каждого пути в ROBDD можно естественным образом определить меру эффективности исключающей его *D*-стратегии. Путь, применительно к которому *D*-стратегия имеет наибольшую эффективность, называется *D*-оптимальным. Довольно неожиданный факт, касающийся сложности поиска *D*-оптимальных путей, состоит в следующем.

Теорема 2 (см. [12]). В произвольной ROBDD B *D*-оптимальный путь находится детерминированным алгоритмом, трудоемкость которого оценивается величиной $O\left(|B|^2\right)$.

Упоминаемый в данной теореме алгоритм использует в качестве основы известный алгоритм Дейкстры поиска кратчайшего пути в ориентированном взвешенном графе (см., например [13]). Исходной ROBDD B при этом сопоставляется ориентированный граф $G(B)$ с начальной и конечной вершинами, ребрам которого специальным образом приписываются натуральные веса. Путь кратчайшей длины (суммарного веса) из корня в терминальную вершину графа $G(B)$ определяет в ROBDD B некоторый оптимальный путь D.

4. Особенности архитектуры BDD-решателя систем логических уравнений

В данной части мы описываем архитектуру «типового» BDD-решателя систем логических уравнений. Общая схема работы такого решателя может включать перечисленные ниже пункты.

1. Система логических уравнений подается на вход решателю в форме текстового описания.

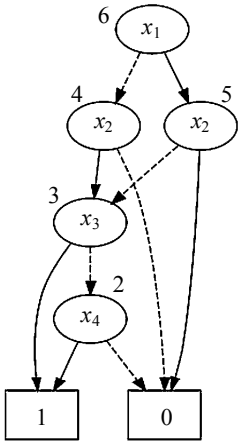
2. Используя данное описание, решатель переводит исходную систему в специальный формат, удобный для последующего построения ROBDD. В этом формате характеристические функции уравнений системы представляется в виде бинарных деревьев (более детальное описание данного этапа приводится ниже).

3. Далее к полученным древовидным представлениям характеристических функций уравнений системы последовательно применяется описанная выше «идеология» ROBDD-подхода (пункты 1, 2).

4. На выходе решатель выдает ROBDD характеристической функции исходной системы логических уравнений, а также одно из решений системы.

Следует отметить, что оптимальным способом «машинного» представления двоичных диаграмм решений является обычная таблица. Таблица, представляющая ROBDD, состоит из трех столбцов. Каждая строка таблицы соответствует определенной вершине ROBDD и, таким образом, содержит три позиции: номер переменной данной вершины, номер нижнего потомка и номер верхнего потомка. Вообще говоря, можно использовать различные схемы нумерации вершин BDD. Наиболее естественной представляется нумерация, при которой терминальные вершины получают номера 0 и 1 (в соответствии с представляемыми ими значениями булевой функции). Последующие вершины (от терминальных к корню) получают номера, начиная с 2.

Пример 1. На следующем рисунке приведена ROBDD для функции $(x_1 \oplus x_2)\cdot(x_3 \vee x_4)$ с порядком переменных $x_1 \prec x_2 \prec x_3 \prec x_4$ и представляющая ее в памяти ЭВМ таблица.



№	var	low	high
0	5		
1	5		
2	4	0	1
3	3	2	1
4	2	0	3
5	2	3	0
6	1	4	5

Рис. 1. Представление ROBDD в памяти ЭВМ

Таблицу, которая представляет ROBDD в памяти ЭВМ, далее обозначим через T. В тех случаях, когда ROBDD является результатом работы некоторого алгоритма, оперирующего либо с рассматриваемой булевой функцией, либо с несколькими ROBDD, помимо таблицы T используется хеш-таблица, которую будем обозначать через H. Данная таблица является хеш-образом T. Хеш-таблица H позволяет ускорять процедуру построения итоговой ROBDD за счет возможности быстрой проверки наличия в H хеш-значения соответствующей вершины.

Представление уравнений рассматриваемой системы также характеризуется рядом особенностей. Как было сказано выше, весьма привлекательным форматом представления характеристических функций логических уравнений являются бинарные деревья. На рис. 2 приведено древовидное представление булевой функции, находящейся в левой части уравнения $(x_1 \oplus (x_2 \vee x_3))\cdot x_4 = 1$.

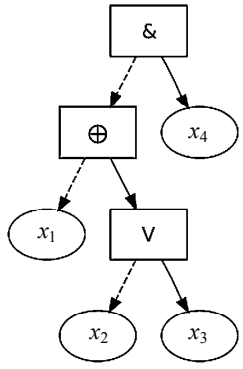


Рис. 2. Древовидное представление функции $(x_1 \oplus (x_2 \vee x_3))\cdot x_4$

Представления такого рода дают ряд преимуществ. Они удобны как для эвристических алгоритмов получения «оптимального» порядка означивания переменных, так и для базовых алгоритмов работы с BDD (алгоритмы Build и Apply).

Интерпретация деревьев в памяти ЭВМ осуществляется при помощи связанных списков. Каждый компонент списка представляет узел дерева, соответствующий функциональному элементу. Такой список может быть реализован, например, при помощи структуры «term», описание которой приведено ниже.

```
typedef struct term {
  operation op;
  int priority;
  struct term *left;
  struct term *right;
  int value;
} term;
```

Одним из важнейших элементов всякого BDD-решателя является механизм выбора порядка означивания переменных, поскольку данный фактор зачастую оказывает очень существенное (а иногда решающее) влияние на размер итоговой ROBDD. В качестве примера приведем ROBDD для функции $(x_1 \oplus x_2)\cdot(x_3 \vee x_4)$ с порядком переменных $x_1 \prec x_3 \prec x_2 \prec x_4$ (рис. 3).

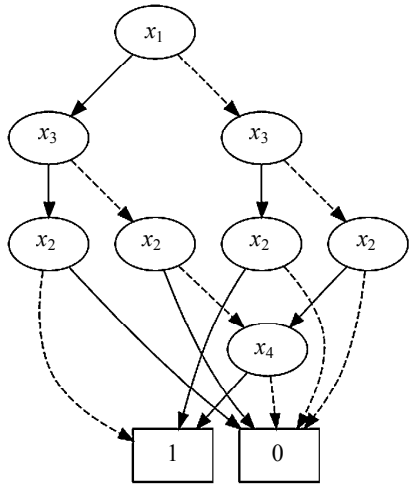


Рис. 3. ROBDD-представление функции $(x_1 \oplus x_2)\cdot(x_3 \vee x_4)$ с порядком $x_1 \prec x_3 \prec x_2 \prec x_4$

Как видно из рисунка, число вершин полученной ROBDD равно десяти, в то время как представленная выше (рис. 1) ROBDD для этой же функции с порядком означивания переменных $x_1 \prec x_2 \prec x_3 \prec x_4$ состоит из семи вершин. Механизмы выбора порядка означивания переменных носят, как правило, характер эвристик. Ниже приведена общая схема эвристики формирования порядка означивания переменных. Данная эвристика выстраивает «иерархию влияния» одних переменных на другие, анализируя статистику появления переменных в уравнениях исходной системы.

1) Производится обход всех деревьев, представляющих булевы функции в левых частях уравнений системы, с присвоением переменным весов (натуральных чисел), учитывающих степень взаимного влияния переменных (см. пример 2).

2) Производится сортировка переменных по убыванию весов. Сформированный порядок – это порядок означивания переменных при построении ROBDD-представления характеристической функции системы.

Пример 2. Далее приведен пример реализации эвристики формирования порядка, которая оценивает степень «взаимного влияния» переменных. Для произвольной переменной x, встречающейся в некотором уравнении системы, ее вес вычисляется как функция от соответствующего дерева T: инициальное значение веса равно 0, затем в процессе обхода дерева T данное значение изменяется при помощи некоторой рекурсивной процедуры. Примером задания подобного рода процедуры является следующая формула:

$$wt_T(x) := wt_T(x) + g_T(d_T(x)).$$

В данной формуле фигурирует функция $g_T(\cdot)$, аргументом которой является «глубина расположения» переменной x в дереве T. Обычно функция $g_T(\cdot)$ подбирается эвристически.

Для большей иллюстративности сказанного рассмотрим следующую систему из двух логических уравнений:

$$\begin{cases}(x_1 \oplus (x_2 \vee x_3)) \cdot x_4 = 1,\\ x_2 \oplus (x_1 \vee (x_3 \cdot x_4)) = 1.\end{cases}$$

Дерево T_1, «представляющее» первое уравнение, приведено на рис. 2. В данном дереве $d_{T_1}(x_4) = 1$, $d_{T_1}(x_1) = 2$, $d_{T_1}(x_2) = d_{T_1}(x_3) = 3$. Допустим, что вычислены следующие значения весов переменных, входящих в первое уравнение: $wt_{T_1}(x_1) = 597$, $wt_{T_1}(x_2) = 594$, $wt_{T_1}(x_3) = 594$, $wt_{T_1}(x_4) = 600$ (если бы система состояла только из первого уравнения, то следовало бы строить ROBDD, используя следующий порядок означивания переменных: $x_4 \prec x_1 \prec x_2 \prec x_3$).

Дерево T_2, «представляющее» второе уравнение рассматриваемой системы, приведено на рис. 4.

Предположим, что в соответствии с описанной процедурой найдены значения весов $wt_{T_2}(x_1) = 597$, $wt_{T_2}(x_2) = 600$, $wt_{T_2}(x_3) = wt_{T_2}(x_4) = 594$. В качестве итоговых значений весов переменных могут быть взяты суммы весов по отдельным деревьям ($wt(x) = \sum_T wt_T(x)$). В рассматриваемом примере имеем $wt(x_1) = 1194$, $wt(x_2) = 1197$, $wt(x_3) = 1188$, $wt(x_4) = 1197$. Откуда следует, что порядок означивания переменных при построении ROBDD-представления характеристической функции рассматриваемой системы в соответствии с описанным подходом должен быть следующим: $x_2 \prec x_4 \prec x_1 \prec x_3$.

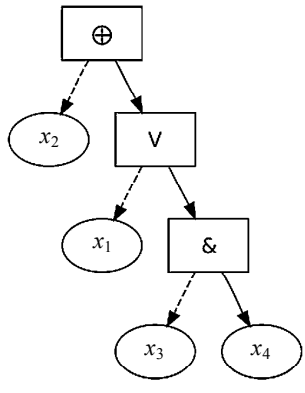


Рис. 4. Древовидное представление функции $x_2 \oplus (x_1 \vee (x_3 \cdot x_4))$

Предположим, что выбран некоторый порядок означивания переменных (например, в соответствии с представленной выше эвристикой) – $x_{i_1} \prec x_{i_2} \prec \ldots \prec x_{i_{n-1}} \prec x_{i_n}$. Число $r \in \{1,\ldots,n\}$ называем индексом переменной x_{i_r} относительно выбранного порядка. На следующем шаге рассматриваемая система логических уравнений вида разбивается на подмножества, называемые слоями. Первый слой образован всеми уравнениями системы, в которые входит переменная x_{i_1}. Второй слой образован всеми оставшимися уравнениями, содержащими переменную x_{i_r}, причем x_{i_r} – переменная наименьшего индекса по уравнениям, не входящим в первый слой. И так далее. Пусть R – число определяемых описанным образом слоев системы. Очевидно, что $1 \le R \le n$. ROBDD каждого слоя B_j, $j \in \{1,\ldots,R\}$ строится по следующей рекурсивной схеме:

$$B_j = \mathrm{Apply}\left(B_{j_1} \cdot \mathrm{Apply}\left(B_{j_2} \cdot \ldots \cdot \mathrm{Apply}\left(B_{j_{k-1}} \cdot B_{j_k}\right)\right)\right),$$

где $B_{j_1},\ldots,B_{j_k}$ – ROBDD булевых функций в левых частях уравнений системы, образующих j-й слой. Итоговая ROBDD характеристической функции системы строится по аналогичной рекурсивной схеме:

$$B = \mathrm{Apply}\left(B_1 \cdot \mathrm{Apply}\left(B_2 \cdot \mathrm{Apply}\left(B_3 \cdot \ldots \cdot \mathrm{Apply}\left(B_{R-1} \cdot B_R\right)\right)\right)\right).$$

Одним из серьезных препятствий к применению BDD в решении систем логических уравнений является проблема чрезмерного роста используемой BDD-решателем памяти. Вообще говоря, объем используемой BDD-решателем памяти растет как некоторая экспонента от числа применения алгоритма Apply. Использование разного рода эвристик (в частности, описанной выше эвристики слоев) позволяет лишь уменьшить основание экспоненты, но не решить проблему в принципе. Задачи обращения криптографических функций из класса $\mathfrak{I}$ интересны тем, что кодирующие их системы логических уравнений выполнимы, как правило, на одном наборе. На практике это означает, что при построении ROBDD характеристической функции такой системы до некоторого момента будет наблюдаться экспоненциальный рост числа вершин в текущей ROBDD, а затем начнется «экспоненциальное сокращение» ее объема (в результате должна получиться ROBDD, представляющая элементарную конъюнкцию n литералов). Качество эвристик формирования порядка и разбиения на слои можно оценивать на основе наблюдаемых коэффициентов роста (предполагая примерно те же коэффициенты убывания).

Еще одна возможность повышения эффективности BDD-подхода в решении систем логических уравнений состоит в использовании некоторых фрагментов рассматриваемой системы в качестве «шаблонов» для построении на их основе более сложных конструкций. Данные рассуждения иллюстрируются приведенным ниже примером.

Пример 3. Пусть дана функция $f(x_1,x_2,x_3,x_4,x_5) = x_1 \cdot (x_2 \oplus x_3 \oplus x_4) \cdot x_5$. И пусть необходимо построить ROBDD данной функции, используя порядок означивания переменных $x_1 \prec x_2 \prec x_3 \prec x_4 \prec x_5$. Применяя для построения ROBDD алгоритм Build, получим следующую ROBDD (рис. 5).

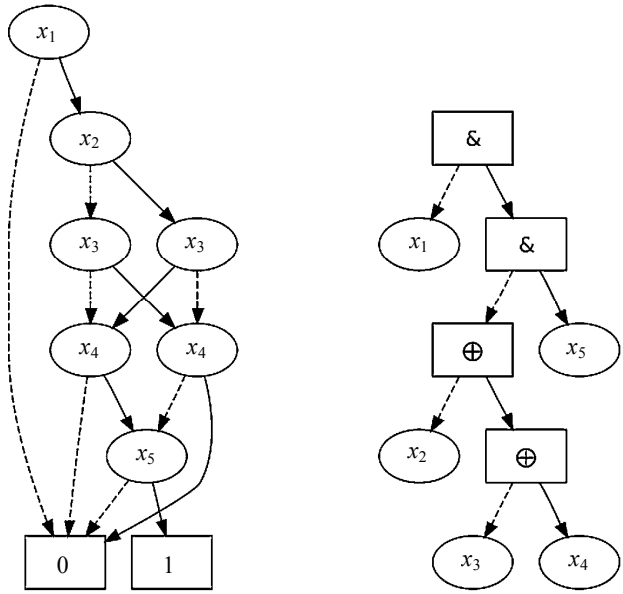


Рис. 5. ROBDD функции $x_1 \cdot (x_2 \oplus x_3 \oplus x_4) \cdot x_5$ и ее древовидное представление

Алгоритм Build при этом произвел построение и последующее усечение полного дерева решений данной функции, состоящего из 63 вершин. Можно построить диаграмму решений этой функций значительно эффективнее. Дерево, представляющее данную функцию в памяти ЭВМ, изображено на рис. 5 справа. Анализируя это дерево, приходим к выводу, что зная «типовую» структуру ROBDD для конъюнкции и сложения по модулю 2, можно построить ROBDD исходной функции, не прибегая к помощи алгоритмов Build или Apply. Действительно, ROBDD функции $y(x_2, x_3, x_4) = x_2 \oplus x_3 \oplus x_4$ с порядком означивания переменных $x_2 \prec x_3 \prec x_4$ и функции $z(x_1, y, x_5) = x_1 \cdot y \cdot x_5$ с порядком означивания переменных $x_1 \prec y \prec x_5$ имеют соответственно вид (рис. 6):

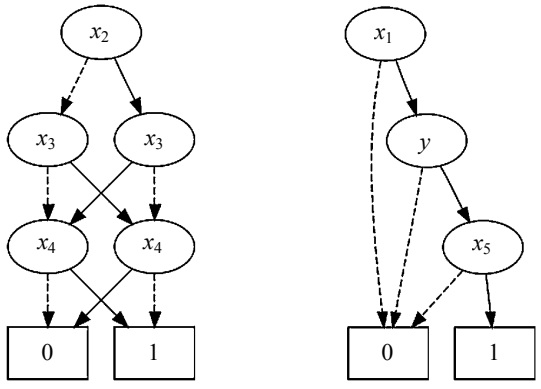


Рис. 6. ROBDD функций $x_2 \oplus x_3 \oplus x_4$ и $x_1 \cdot y \cdot x_5$

Подставляя в последней ROBDD вместо переменной y ROBDD для функции $y(x_2, x_3, x_4) = x_2 \oplus x_3 \oplus x_4$, получим ROBDD исходной функции.

Сформулируем условия, при выполнении которых возможно использование описанной процедуры. Пусть некоторая булева функция $f(x_1,...,x_n)$ задана формулой исчисления высказываний $L_f(x_1,...,x_n)$ и ее подформула $L_y\left(x_{j_1},...,x_{j_m}\right)$ задает булеву функцию $y\left(x_{j_1},...,x_{j_m}\right)$, $\left\{x_{j_1},...,x_{j_m}\right\} \subset \{x_1,...,x_n\}$. Пусть $x_{i_1} \prec x_{i_2} \prec ... \prec x_{i_{n-1}} \prec x_{i_n}$ – общий порядок означивания переменных для $f(x_1,...,x_n)$, а $x_{j_1} \prec ... \prec x_{j_m}$ – порядок означивания переменных для функции $y\left(x_{j_1},...,x_{j_m}\right)$. Несложно понять, что использование ROBDD функции $y\left(x_{j_1},...,x_{j_m}\right)$ в роли шаблона, «вставляемого» в ROBDD функции $f(x_1,...,x_n)$ (так, как это сделано в предыдущем примере), возможно тогда и только тогда, когда выполняются перечисленные ниже условия.

1. Условие автономности порядка $x_{j_1} \prec ... \prec x_{j_m}$ относительно общего порядка требует, чтобы в общем порядке порядок $x_{j_1} \prec ... \prec x_{j_m}$ образовывал связный интервал, то есть не существовало переменной x_0, $x_0 \notin \left\{x_{j_1},...,x_{j_m}\right\}$, для которой в общем порядке имеет место следующая картина: $x_{i_1} \prec ... \prec x_{j_1} \prec ... \prec x_0 \prec ... \prec x_{j_m} \prec ... \prec x_{i_n}$ (например, порядок $x_2 \prec x_3 \prec x_4$ подчинен общему порядку $x_1 \prec x_2 \prec x_3 \prec x_5 \prec x_4$, но не удовлетворяет условию автономности).

2. Никакая переменная из множества $\left\{x_{j_1},...,x_{j_m}\right\}$ не должна входить в формулу $L_f(x_1,...,x_n)$ вне формулы $L_y\left(x_{j_1},...,x_{j_m}\right)$.

Заключение

В статье описан подход к решению систем логических уравнений, в основе которого лежит идея представления булевых функций в форме двоичных диаграмм решений (BDD). Особый акцент сделан на применимости BDD к задачам обращения дискретных функций из класса, который используется в различных областях кибернетики. Описан гибридный алгоритм решения логических уравнений, использующий преимущества SAT- и ROBDD- подходов. Приведена типовая архитектура BDD-решателя логических уравнений. Обсуждаются общие принципы организации данных и формирования эвристик в BDD-решателе.

ЛИТЕРАТУРА

1. *Lee C.Y.* Representation of switching circuits by binary-decision programs // Bell Systems Techn. J. 1959. No. 38. P. 985 – 999.
2. *Bryant R.E.* Graph-based algorithms for boolean function manipulation // IEEE Trans. Computers. 1986. V. 35. No. 8. P. 677 – 691.
3. *Семенов А.А.* О сложности обращения дискретных функций из одного класса // Дискретный анализ и исследование операций. 2004. Т. 11. № 4. С. 44 – 55.
4. *Meinel Ch.*, *Theobald T.* Algorithms and Data Structures in VLSI-Design: OBDD-Foundations and Applications. Springer-Verlag, 1998. 276 p.

5. *Катленд Н.* Вычислимость. Введение в теорию рекурсивных функций. М.: Мир, 1983. 256 с.
6. *Гэри М.*, *Джонсон Д.* Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982. 416 с.
7. *Семенов А.А.* Логико-эвристический подход в криптоанализе генераторов двоичных последовательностей // Труды Междунар. науч. конф. ПАВТ'07. Челябинск: ЮУрГУ, 2007. Т. 1. С. 170 – 180.
8. *Заикин О.С.*, *Семенов А.А.* Технология крупноблочного параллелизма в SAT-задачах // Проблемы управления. 2008. № 1. С. 43 – 50.
9. *Een N.*, *Sorensson N.* Translating Pseudo-Boolean constraints into SAT // J. Satisfiability, Boolean Modeling and Computation. 2006. No. 2. P. 1 – 25.
10. *Marqeus-Silva J.P. and Sakallah K.A. GRASP*: A search algorithm for propositional satisfiability // IEEE Trans. on Computers. 1999. V. 48. No. 5 P. 506 – 521.
11. *Семенов А.А.*, *Заикин О.С.* Неполные алгоритмы в крупноблочном параллелизме комбинаторных задач // Вычислительные методы и программирование. 2008. Т. 9. № 1. С. 112 – 122.
12. *Семенов А.А.* Консервативные преобразования систем логических уравнений // Вестник ТГУ. Приложение. 2007. № 23. С. 52 – 59.
13. *Кормен Т.*, *Лейзерсон Ч.*, *Ривест Р.* Алгоритмы. Построение и анализ. М.: МЦНМО, 2001. 955 с.

Статья представлена кафедрой информационных технологий в исследовании дискретных структур радиофизического факультета Томского государственного университета. Поступила в редакцию 27 ноября 2008 г.