ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

УДК 519.7

АЛГОРИТМЫ РАБОТЫ С ROBDD КАК С БАЗАМИ БУЛЕВЫХ ОГРАНИЧЕНИЙ

А. С. Игнатьев, А. А. Семенов

Институт динамики систем и теории управления СО РАН, г. Иркутск, Россия

E-mail: aign@icc.ru, biclop@rambler.ru

Исследуются алгоритмические свойства сокращенных упорядоченных диаграмм решений (ROBDD) при их рассмотрении в роли баз булевых ограничений в гибридном (SAT+ROBDD)-выводе. Приведены ROBDD-аналоги основных алгоритмических процедур, используемых в DPLL-выводе (подстановки, правило единичного дизъюнкта, CL-процедура, механизмы отсроченных вычислений). Описан новый алгоритм изменения порядка в ROBDD. Для всех алгоритмов приводятся оценки их трудоемкости.

Ключевые слова: *логические уравнения, двоичные диаграммы решений, гибридный вывод.*

Введение

В последние годы заметен рост интереса к символьным алгоритмам, эффективным на практически важных классах логических уравнений. Особенно впечатляет прогресс в разработке таких алгоритмов для решения задачи ВЫПОЛНИМОСТЬ (SAT-задачи, см. [1]). Этим вопросам посвящены многочисленные конференции и специализированные издания (одной из центральных тем издающегося в Нидерландах журнала JSAT [2] является алгоритмика SAT-задач).

SAT-задачи находят широкое применение в различных разделах прикладной дискретной математики и кибернетики: синтез и верификация дискретных автоматов [3], верификация программных логик [4, 5], криптоанализ [6], задачи информационной биологии [7] и во многих других областях.

Наибольшую эффективность в решении SAT-задач (по результатам специализированных конкурсов [1]) демонстрируют методы, использующие в своей основе алгоритм DPLL (Devis, Putnam, Logemann, Loveland, см. [8]) и последующие его модернизации [9 – 11]. Главное отличие поздних версий от классического DPLL в том, что в них информация о ходе вывода хранится в форме булевых ограничений-дизъюнктов, запрещающих конфликтные присвоения.

Одной из центральных проблем в современных высокоскоростных SAT-решателях является проблема переполнения памяти генерируемыми в процессе поиска булевыми ограничениями. Вообще говоря, несложно построить семейства противоречивых конъюнктивных нормальных форм (КНФ), для опровержения которых перечисленным алгоритмам потребуется порождать экспоненциальное от размерности КНФ число конфликтных дизъюнктов, однако такие тесты искусственны и далеки по своей природе от реальных практических задач.

На практике проблема переполнения памяти решается при помощи процедур чистки баз ограничений, в результате которых признаваемые нерелевантными ограничения попросту отбрасываются. Однако все заключения о релевантности носят характер эвристик и, как следствие, отбрасывание ограничений в общем случае приводит к потере алгоритмом полноты ввиду невозможности гарантировать окончание его работы на произвольных КНФ. Эффект потери полноты наблюдался на некоторых криптографических тестах и приводил при крупноблочном распараллеливании соответствующих SAT-задач к «сверхлинейному» ускорению (см. [12]).

В работе [13] для решения проблемы переполнения памяти на задачах обращения дискретных функций был предложен гибридный подход, в котором нехронологический DPLL-вывод сочетается с выводом на двоичных диаграммах решений (BDD), а точнее, на сокращенных упорядоченных BDD, или ROBDD.

Настоящая статья посвящена описанию основных алгоритмов работы с ROBDD, рассматриваемыми в роли баз булевых ограничений, порожденных в процессе нехронологического DPLL-вывода. Для всех представленных в работе алгоритмов построены оценки их трудоемкости. Приведем краткий план статьи.

В п. 1 содержатся необходимые сведения об алгоритмах логического вывода на основе DPLL и их применении к задачам обращения полиномиально вычислимых дискретных функций. Здесь же кратко описываются двоичные диаграммы решений и основные алгоритмы манипулирования булевыми функциями с их помощью.

В п. 2 кратко описаны результаты работы [13] по основам гибридного (SAT+ROBDD)-подхода к обращению дискретных функций. Приведены (без доказательств) теорема о ядре DPLL-вывода и теорема, обосновывающая использование ROBDD в роли базы булевых ограничений в гибридном (SAT+ROBDD)-выводе на SAT-задачах, кодирующих проблемы обращения полиномиально вычислимых дискретных функций.

В п. 3 описаны основные алгоритмы работы с ROBDD как с базами булевых ограничений в гибридном (SAT+ROBDD)-выводе и даны оценки их трудоемкости. Приведен новый алгоритм изменения порядка в ROBDD на произвольный новый порядок. Описаны также ROBDD-аналоги основных алгоритмов, используемых в современных SAT-решателях, базирующихся на DPLL (подстановка значения переменной, правило единичного дизъюнкта, CL-процедура, механизмы отсроченных вычислений при работе с булевыми ограничениями).

1. Базовые понятия, конструкции и алгоритмы

1.1. SAT-задачи и их алгоритмика

К SAT-задачам относятся задачи поиска решений логических [14] (булевых) уравнений вида

$$C(x_1, \ldots, x_k) = 1,$$

где $C(x_1, \ldots, x_k)$ — формула исчисления высказываний (ИВ), имеющая вид КНФ; $x_1, \ldots, x_k$ — булевы переменные, образующие множество X. В общей постановке SAT-задачи NP-трудны, однако к ним сводится настолько обширное множество практически важных задач, что построение эвристических алгоритмов, эффективных на тех или иных классах SAT-задач, является одной из актуальных областей современной компьютерной алгебры.

Наиболее эффективные на сегодняшний день SAT-решатели используют в своей основе алгоритм DPLL и дальнейшие его модификации. В [15] описана архитектура современных скоростных SAT-решателей, базирующихся на DPLL. Далее кратко перечислены их основные алгоритмические компоненты.

1) Алгоритм DPLL и его составляющие: правило единичного дизъюнкта, стратегия распространения булевых ограничений (BCP) [8].
2) Графы анализа вывода, процедура «Clause Learning» (далее CL-процедура), смысл которой заключается в конъюнктивном приписывании к текущей КНФ новых ограничений-дизъюнктов, запрещающих приведшие к конфликтам присвоения [9].
3) Процедуры анализа конфликтов и построения конфликтных дизъюнктов [9, 16]. Их предназначение — синтез новых булевых ограничений-дизъюнктов.
4) Механизмы отсроченных вычислений (типа «watched literals», [17]). С их помощью сокращается время, затрачиваемое на подстановку в КНФ значений переменных (в некоторые дизъюнкты, не удовлетворяющие определенным признакам, подстановка не осуществляется).

Использование SAT-решателей с перечисленными компонентами оказалось оправданным даже на таких аргументированно трудных задачах, как задачи обращения некоторых криптографических функций (см. [6, 18]). При этом следует отметить выгодные свойства перечисленных алгоритмов при их крупноблочном распараллеливании (см. [18, 19]).

Здесь мы очень кратко опишем общую схему сведения проблем обращения полиномиально вычислимых дискретных функций к SAT-задачам (более подробное описание можно найти, например, в [20]).

Обозначим через $\Im$ класс, образованный натуральными семействами вычислимых за полиномиальное время дискретных функций вида $f : \{0,1\}^n \to \{0,1\}^*$. Проблема обращения $f \in \Im$ в произвольной точке $y \in \operatorname{range} f$ ставится следующим образом: зная y и алгоритм вычисления f, найти $x \in \{0,1\}^n$, такой, что $f(x) = y$. Используя фундаментальную идею С. Кука (см. [21]), можно свести данную задачу к задаче поиска выполняющего набора выполнимой КНФ $C(f)$ над множеством булевых переменных $\tilde{X} = \{x_1, \ldots, x_{q(n)}\}$, $q(\cdot)$ — некоторый полином. Именно этот факт лежит в основе многочисленных примеров применения SAT-решателей к задачам обращения дискретных функций. Про КНФ $C(f)$ (которую также будем обозначать через $C(x_1, \ldots, x_{q(n)})$) будем говорить, что она кодирует задачу обращения функции f в точке $y \in \operatorname{range} f$.

1.2. Двоичные диаграммы решений и алгоритмы манипулирования булевыми функциями на их основе

Двоичные диаграммы решений (BDD) были введены К. Ли в [22]. Фундаментальность этой структуры данных для дискретной математики была осознана после выхода работы Р. Брайанта [23], в которой он описал семейство алгоритмов «манипулирования» булевыми функциями при помощи BDD.

Двоичные диаграммы решений — это ориентированные ациклические помеченные графы, представляющие булевы функции. Вообще говоря, произвольную BDD можно рассматривать как графическую интерпретацию рекурсивного применения к некоторой булевой функции разложения Шеннона. Однако более наглядным является переход к BDD от двоичных деревьев решений (см. [24]).

Пусть $T(f)$ — некоторое двоичное дерево решений, представляющее всюду определенную булеву функцию $f : \{0,1\}^n \to \{0,1\}$. Для определенности можно считать, что f выражена формулой ИВ $L(f)$. Корень и внутренние вершины $T(f)$ помечаются переменными из множества $X = \{x_1, \ldots, x_n\}$, листья помечаются константами из $\{0,1\}$. Каждый путь в $T(f)$ из корня в лист определяет некоторую последовательную подстановку значений переменных из X в $L(f)$. Константа, приписанная листу, есть значение

функции f, полученное в результате этой подстановки. Из каждой вершины, помеченной переменной из X, выходят два ребра: пунктирное, что соответствует принятию переменной значения 0, и сплошное, что соответствует принятию ею значения 1. Традиционно используются термины «low-ребра» (для пунктирных) и «high-ребра» (для сплошных). Порядок выбора переменных из X при осуществлении последовательной подстановки будем называть порядком означивания переменных в соответствующем пути. Если в дереве $T(f)$ все пути подчинены общему порядку означивания переменных, то назовем это дерево упорядоченным. Если склеить все листья, помеченные «0», упорядоченного дерева в один и то же самое проделать с листьями, помеченными «1», то получится OBDD (упорядоченная двоичная диаграмма решений). Можно заметить, что каждая вершина в произвольной OBDD определяется тройкой «координат»: переменной из X, которой данная вершина помечена, low-ребенком и high-ребенком (то есть детьми данной вершины по соответствующим ребрам). Вершины, определяемые одинаковыми тройками координат в OBDD, можно склеить. Кроме этого, можно удалить (с сохранением структуры) из OBDD вершины, у которых low-ребенок совпадает с high-ребенком. В результате получим сокращенную упорядоченную диаграмму решений, или ROBDD. Каноническая теорема Р. Брайанта [23] утверждает, что произвольная всюду определенная булева функция f при фиксированном порядке означивания переменных единственным образом (с точностью до изоморфизма графов) представляется ROBDD $B(f)$.

Представление булевых функций в виде ROBDD имеет ряд привлекательных свойств. Во-первых, в силу канонической теоремы, произвольную ROBDD можно рассматривать как «сжатое» представление соответствующей булевой функции в специальном классе графов. Во вторых, используя ROBDD-представления, можно оперировать с булевыми функциями при помощи алгоритмов, большая часть которых была приведена в статье Р. Брайанта [23] (ниже перечислены основные).

1) Алгоритм *Apply* позволяет на основе ROBDD-представлений функций $B(f_1)$ и $B(f_2)$ построить ROBDD-представление функции $f_1 * f_2$, где $*$ — произвольная бинарная логическая связка. При совпадении порядка означивания переменных в $B(f_1)$ и $B(f_2)$ сложность *Apply* ограничена сверху величиной $O(|B(f_1)|\cdot|B(f_2)|)$ (здесь и далее через $|B|$ обозначается число вершин в ROBDD B).
2) Алгоритм *Restrict* по ROBDD, представляющей функцию f, выраженную формулой $L(f)$, строит ROBDD-представление функции $f|_{x=\alpha}$, выраженной формулой $L(f)|_{x=\alpha}$, для произвольных $x \in X$ и $\alpha \in \{0,1\}$. Сложность *Restrict* есть $O(|B(f)|)$.
3) Алгоритм *Satcount* позволяет по ROBDD $B(f)$, представляющей функцию f, подсчитать число векторов значений переменных из X, на которых f принимает значение 1. Сложность *Satcount* есть $O(|B(f)|)$.

Рассмотрим произвольную систему логических уравнений S следующего вида:

$$\begin{cases} U_1(x_1,\ldots,x_n) = 1, \\ \ldots \\ U_m(x_1,\ldots,x_n) = 1. \end{cases}$$

Под характеристической функцией системы S будем понимать булеву функцию $\chi_S : \{0,1\}^n \to \{0,1\}$, заданную формулой

$$U_1(x_1,\ldots,x_n)\cdot\ldots\cdot U_m(x_1,\ldots,x_n).$$

Несложно понять, что, имея ROBDD-представление функции χ_S, можно за линейное от числа вершин в данной ROBDD время предъявить решение системы S, либо констатировать ее несовместность.

Алгоритмы работы с ROBDD можно использовать для решения задач обращения полиномиально вычислимых дискретных функций напрямую, не переходя к логическим уравнениям (см. [25]). Однако проведенные вычислительные эксперименты показывают, что ROBDD-подход проигрывает по эффективности SAT-подходу на большинстве криптографических тестов. Кроме этого, можно показать (см., например, [26]), что задачи построения ROBDD-представлений булевых функций, заданных хорновскими КНФ (т. е. КНФ, состоящими из двухбуквенных дизъюнктов), не могут быть в общем случае решены за полиномиальное время в предположении, что $P \neq NP$. При этом известно, что SAT-задачи для таких КНФ решаются за полиномиальное время.

2. Гибридный (SAT+ROBDD)-вывод в применении к задачам обращения дискретных функций

Как было отмечено выше, использование ROBDD «в чистом виде» оправдано лишь на обращении сравнительно простых функций (криптоанализ простейших генераторов типа Геффе). Однако ROBDD привлекательны как структуры данных, наиболее экономным образом представляющие булевы функции в специальном классе графов. Этот факт приводит к идее использования ROBDD в качестве структур, представляющих булевы ограничения, накапливаемые в процессе нехронологического DPLL-вывода. Данная идея представляется перспективной именно в отношении задач обращения полиномиально вычислимых функций. И этому есть целый ряд причин.

Одна из главных причин состоит в том, что если рассматривать задачу обращения некоторой полиномиально вычислимой функции $f : \{0,1\}^n \to \{0,1\}^*$, $f \in \Im$, как SAT-задачу, то в соответствующей КНФ $C(f)$ можно выделить подмножество булевых переменных, от которых в некотором смысле «функционально зависят» все остальные переменные, фигурирующие в данной КНФ. Это множество, обозначаемое далее через X, образовано булевыми переменными, кодирующими входное слово из $\{0,1\}^n$. Как правило, число n существенно меньше общего числа переменных в $C(f)$. Однако можно показать, что для решения соответствующей SAT-задачи достаточно оперировать (в указанном ниже смысле) только с переменными множества X. Кратко остановимся на перечисленных моментах (результаты, представленные в данном пункте, подробно изложены в [13]).

Пусть $C = C(x_1, \ldots, x_k)$ — произвольная КНФ над множеством булевых переменных $\tilde{X} = \{x_1, \ldots, x_k\}$. Рассмотрим некоторое множество $X' = (x'_1, \ldots, x'_r)$, являющееся подмножеством $\tilde{X}$. Пусть $(\alpha_1, \ldots, \alpha_r)$ — произвольный вектор значений переменных из X'. Осуществим подстановку в КНФ C значения $x'_1 = \alpha_1$. Данная подстановка заключается в вычеркивании из C некоторых литералов и дизъюнктов. При этом отслеживаются возможности срабатывания правила единичного дизъюнкта с последующими подстановками в C соответствующих индуцированных значений. Если в результате не выведен конфликт или выполняющий C набор, то в КНФ $C|_{x'_1=\alpha_1}$ осуществляется подстановка $x'_2 = \alpha_2$. И так далее.

Описанная процедура определяет последовательную подстановку в C вектора $(\alpha_1, \ldots, \alpha_r)$ относительно порядка $x'_1 \prec \ldots \prec x'_r$. Очевидно, что последовательная подстановка в общем случае реализуется эффективно. Возможны различные исходы этой процедуры. Во-первых, может оказаться, что данная подстановка выводит конфликт. Во-вторых, что ее результатом является нахождение некоторого выполняющего C на-

бора. Наконец, возможен переход к некоторой КНФ, относительно которой нельзя сказать ничего. Если имеет место первая или вторая ситуация, то будем говорить, что данная подстановка индуцирует детерминированный DPLL-вывод соответственно конфликта или выполняющего набора.

Предположим, что относительно некоторой КНФ C над $\tilde{X}$ и некоторого множества $X' \subset \tilde{X}$ можно показать, что результатом последовательной подстановки любого вектора значений переменных из X' в C относительно некоторого порядка могут быть только первая или вторая ситуации. Несложно видеть, что в этом случае достаточно перебрать $2^{|X'|}$ всевозможных значений переменных из X', чтобы решить рассматриваемую SAT-задачу в отношении КНФ C.

Определение 1. Пусть $\tilde{X} = \{x_1, \ldots, x_k\}$ — множество булевых переменных и $X' \subseteq \tilde{X}$. Проекцией произвольного вектора $\alpha = (\alpha_1, \ldots, \alpha_k)$ значений переменных из $\tilde{X}$ на множество X' называется вектор, образованный теми компонентами α, которые являются значениями переменных из X'. Проекцию вектора α на множество X' обозначим через $\alpha_{X'}$.

Определение 2. Ядром DPLL-вывода для КНФ $C = C(x_1, \ldots, x_k)$ над множеством булевых переменных $\tilde{X} = \{x_1, \ldots, x_k\}$ называется такое множество $X^{\ker}(C) \subseteq \tilde{X}$ с введенным на нем порядком τ, что имеют место следующие свойства:

1) для любого вектора $\alpha = (\alpha_1, \ldots, \alpha_k)$, выполняющего C, последовательная подстановка в C вектора $\alpha_{X^{\ker}(C)}$ относительно τ индуцирует детерминированный DPLL-вывод α;
2) для любого вектора $\beta = (\beta_1, \ldots, \beta_k)$, такого, что $C|_\beta = 0$, последовательная подстановка в C вектора $\beta_{X^{\ker}(C)}$ относительно τ индуцирует детерминированный DPLL-вывод конфликта.

Ядро $X^{\ker}(C) = \tilde{X}$ называется тривиальным. Ядро наименьшей мощности называется минимальным и обозначается через $X^{\ker}_*(C)$.

Рассмотрим произвольную функцию $f : \{0,1\}^n \to \{0,1\}^*$, $f \in \Im$. Функция f — это дискретная функция, которая может быть выражена формулой от булевых переменных $x_1, \ldots, x_n$. Назовем множество $X = \{x_1, \ldots, x_n\}$ множеством переменных входа функции f. Рассматриваем задачу обращения f в произвольной точке $y \in \operatorname{range} f$. При помощи преобразований Цейтина (см. [27]) и техники, описанной, например, в [20], сводим за полиномиальное время данную проблему к задаче поиска выполняющего набора выполнимой КНФ $C(f)$ от булевых переменных, образующих множество $\tilde{X} = \{x_1, \ldots, x_{q(n)}\}$, $q(\cdot)$ — некоторый полином. Справедлива следующая теорема.

Теорема 1 [13]. Рассмотрим произвольную функцию $f : \{0,1\}^n \to \{0,1\}^*$ из класса $\Im$. Обозначим через X_τ множество переменных входа f с зафиксированным на нем порядком τ (вообще говоря, произвольным). Пусть $C(x_1, \ldots, x_{q(n)})$ — КНФ, кодирующая задачу обращения функции f в произвольной точке $y \in \operatorname{range} f$. Тогда $X^{\ker}_*(C(x_1, \ldots, x_{q(n)})) \subseteq X_\tau$.

Фактически данная теорема означает следующее. Будем рассматривать задачу обращения функции f в произвольной точке $y \in \operatorname{range} f$ как SAT-задачу в отношении КНФ $C(x_1, \ldots, x_{q(n)})$, решаемую при помощи любого алгоритма, основанного на DPLL. Тогда в качестве переменных уровней решения достаточно выбирать переменные входа рассматриваемой функции. Очень важен тот факт, что порядок выбора может быть произвольным. Это означает, что стратегия выбора переменных только из X гарантирует решение задачи обращения f в произвольной точке $y \in \operatorname{range} f$ алгоритмом DPLL, использующим CL-процедуру и рестарты [28].

Рассмотрим систему логических уравнений вида

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

над множеством булевых переменных $U = \{y_1,\ldots,y_s\} \cup \{z_1,\ldots,z_t\}$. Предположим, что $C(z_1,\ldots,z_t)$ — КНФ, для которой известно некоторое нетривиальное ядро DPLL-вывода $Z^{\ker}(C)$, причем $Z^{\ker}(C) \subseteq \{y_1,\ldots,y_s\}$. Через $B(W)$ обозначим ROBDD-представление характеристической функции системы

$$\begin{cases} W_1(y_1,\ldots,y_s) & = 1, \\ \ldots & \\ W_k(y_1,\ldots,y_s) & = 1. \end{cases} \tag{2}$$

Определение 3. Пусть $B(W)$ — ROBDD-представление булевой функции $W : \{0,1\}^s \to \{0,1\}$ от s булевых переменных $y_1,\ldots,y_s$. Назовем путь в $B(W)$ от корня к терминальной вершине «1» полным относительно множества $Y' \subseteq Y = \{y_1,\ldots,y_s\}$, если прохождение этого пути задает присвоение соответствующих значений всем переменным из Y'.

Справедлива следующая теорема.

Теорема 2 [29]. Прохождение любого пути в ROBDD $B(W)$ из корня в терминальную вершину «1», который полон относительно множества $Z^{\ker}(C)$, индуцирует подстановку в КНФ $C(z_1,\ldots,z_t)$ значений переменных из $Z^{\ker}(C)$. Результатом этой подстановки является либо вывод по правилу единичного дизъюнкта конфликта, либо вывод набора, выполняющего $C(z_1,\ldots,z_t)$. В последнем случае имеем некоторое решение исходной системы.

Пусть уравнение $C(x_1,\ldots,x_{q(n)}) = 1$ кодирует задачу обращения функции f из класса $\Im$ в некоторой точке $y \in \operatorname{range} f$. Через $X^{\ker}(C)$ обозначено некоторое нетривиальное ядро DPLL-вывода КНФ C (для рассматриваемой задачи в качестве $X^{\ker}(C)$ всегда можно взять X — множество переменных входа функции f). Организуем решение задачи поиска набора, выполняющего C, при помощи алгоритма DPLL, дополненного CL-процедурой (см. п. 1). При этом, руководствуясь теоремой 1, будем выбирать в качестве переменных уровней решения только те переменные, которые находятся в $X^{\ker}(C)$. Допустим, что осуществлено Q итераций такого выбора с последующим распространением булевых ограничений и реализацией CL-процедуры, но выполняющий набор при этом не найден. Обозначим через

$$D_1\left(x_1^1,\ldots,x_{r_1}^1\right),\ldots,D_Q\left(x_1^Q,\ldots,x_{r_Q}^Q\right)$$

конфликтные дизъюнкты, выведенные в данных итерациях (очевидно, что $\bigcup\limits_{i=1}^{Q}\{x_1^i,\ldots,x_{r_i}^i\} \subseteq X^{\ker}(C)$). Рассмотрим следующую систему логических уравнений:

$$\begin{cases} D_1(x_1^1,\ldots,x_{r_1}^1) & = 1, \\ \ldots & \\ D_Q(x_1^Q,\ldots,x_{r_Q}^Q) & = 1. \end{cases} \tag{3}$$

Определение 4. Пусть S — произвольная совместная система логических уравнений, $B(S)$ — ROBDD-представление ее характеристической функции и π — произвольный путь из корня $B(S)$ в терминальную вершину «1». Данный путь определяет некоторое множество решений S, обозначаемое через $A(\pi)$, $|A(\pi)| \geqslant 1$. Про любое $\alpha \in A(\pi)$ говорим также, что путь π содержит α.

Теорема 3 [13]. Обозначим через $\alpha = (\alpha_1, \ldots, \alpha_n)$ произвольное решение задачи обращения функции f из класса $\Im$ в некоторой точке $y \in \operatorname{range} f$. Пусть B — ROBDD-представление характеристической функции системы (3) в контексте рассматриваемой задачи. Тогда существует такой путь π из корня B в терминальную вершину «1» , что $\alpha \in A(\pi)$.

Отметим, что данная теорема определяет конкретный вид систем (1)–(2) в контексте задачи обращения функции f в точке $y \in \operatorname{range} f$. Роль $C(z_1, \ldots, z_t)$ в этом случае играет КНФ $C(f) = C(x_1, \ldots, x_{q(n)})$, а роль системы (2) — система (3), образованная уравнениями вида $D_i = 1, i \in \{1, \ldots, Q\}$, где D_i — конфликтные дизъюнкты, полученные в результате DPLL-вывода, примененного к КНФ $C(x_1, \ldots, x_{q(n)})$, с выбором переменных уровней решения из $X^{\ker}(C)$.

Теоремы 1–3 дают теоретическую базу для нового подхода к решению задач обращения полиномиально вычислимых дискретных функций, основная идея которого состоит в совместном использовании SAT и ROBDD именно в тех «частях» задачи обращения, где они могут дать ощутимый выигрыш. Как уже говорилось, вряд ли можно за счет использования только ROBDD обогнать SAT-подход на задачах обращения криптографических функций. Однако ROBDD могут использоваться для решения другой важной проблемы — потери полноты SAT-решателем в результате чистки баз ограничений. Именно ROBDD представляются оптимальными структурами данных для хранения массивов конфликтных дизъюнктов, накапливаемых SAT-решателем в процессе вывода (и это целиком подтверждается численными экспериментами). Особо отметим, что никакие из синтезированных в процессе вывода ограничений при этом не удаляются (в отличие от практики, принятой в большинстве современных SAT-решателей).

Все сказанное означает необходимость описания и изучения ROBDD-аналогов механизмов логического вывода, используемых в современных SAT-решателях, базирующихся на DPLL. Соответствующие процедуры будут применяться к базам булевых ограничений, представленным не в виде конъюнкций дизъюнктов, а в виде ROBDD.

3. Алгоритмы работы с ROBDD как с базами булевых ограничений

3.1. Алгоритмы логического вывода на ROBDD

Рассматривается проблема обращения функции $f : \{0,1\}^n \to \{0,1\}$ из класса $\Im$. Пусть B — ROBDD-представление характеристической функции системы логических уравнений (3). Дальнейшая цель состоит в описании процесса логического вывода на ROBDD B. Для этого потребуется определить аналоги таких компонент DPLL-вывода, как правило единичного дизъюнкта, CL-процедура, процедуры «отсроченной» работы с данными (head-tail literals и watched literals). Все приводимые далее результаты справедливы в отношении произвольных ROBDD.

Определение 5. Пусть B — ROBDD-представление произвольной булевой функции от булевых переменных $x_1, \ldots, x_n$. Каждой переменной $x_i, i \in \{1, \ldots, n\}$, и терминальным вершинам «0», «1» поставим в соответствие множества значений данной переменной, задаваемых всевозможными путями в B из корня в соответствующую терминальную вершину. Данные множества обозначим через $\Delta^0(x_i)$, $\Delta^1(x_i)$.

Предположим, что в некоторой ROBDD B выполнены следующие условия:

1) Для некоторой переменной $x_k \in X = \{x_1, \ldots, x_n\}$ любой путь π из корня B в терминальную вершину «1» обязательно проходит через некоторую вершину, помеченную переменной x_k.
2) Справедливо $|\Delta^1(x_k)| = 1$.

Результатом данной ситуации является заключение о том, что в любом наборе значений переменных из множества X, на котором значение функции f, представленной B, равно 1, переменная x_k может принимать только одно значение (соответствующее значение в $\Delta^1(x_k)$).

Определение 6. Определяемую условиями 1–2 ситуацию далее называем ROBDD-следствием соответствующего значения для переменной x_k.

Возникновение в B, представляющей базу булевых ограничений-дизъюнктов, ROBDD-следствия для некоторой переменной является аналогом правила единичного дизъюнкта в DPLL-выводе. Покажем, что справедлив следующий факт.

Лемма 1. Выполнимость 1–2 относительно некоторой переменной x_k означает, что выполнено в точности одно из следующих условий:

1) для каждой вершины, помеченной x_k, ее high-ребенком является терминальная вершина «0»;
2) для каждой вершины, помеченной x_k, ее low-ребенком является терминальная вершина «0».

Доказательство. Прежде всего отметим, что из любой нетерминальной вершины ROBDD B достижима как вершина «0», так и вершина «1».

Предположим теперь, что выполнены условия 1–2. Это означает, что для некоторой переменной x_k любой путь в ROBDD B содержит вершину, помеченную данной переменной, и пути в B задают x_k одно и то же значение (поскольку $|\Delta^1(x_k)| = 1$) — либо 0, либо 1. Не ограничивая общности, будем считать, что все пути задают x_k значение 1 ($\Delta^1(x_k) = \{1\}$). Если предположить, что ребенком некоторой $v(x_k)$ является терминальный «0», то это может быть только low-ребенок. Действительно, в противном случае из $v(x_k)$ по low-ребенку достижима терминальная вершина «1», и соответствующий путь задает x_k значение 0, что противоречит сделанному предположению.

Предположим, что найдется такая $v'(x_k)$, что ее low-ребенком не является терминальный «0», и обозначим через $v'(x_l)$ вершину, являющуюся low-ребенком $v'(x_k)$. Но из $v'(x_l)$ достижима терминальная вершина «1», поэтому существует путь из корня в «1», задающий x_k значение 0, что противоречит условиям 1–2. Итак, если выполнены условия 1–2 и любой путь в B из корня в «1» задает x_k значение 1, то low-ребенок любой вершины, помеченной x_k, — это терминальный «0». Аналогично, если выполнены 1–2 и любой путь из корня B в «1» задает x_k значение 0, то high-ребенок любой вершины, помеченной x_k, — это терминальный «0». Лемма 1 доказана. ■

Лемма 2. Процедура проверки возникновения ROBDD-следствий в ROBDD B требует детерминированного времени, ограниченного сверху величиной $O(n \cdot |B|)$.

Доказательство. Сам по себе данный факт не является трудным. Поэтому представляется целесообразным привести в процессе доказательства полное описание алгоритма отслеживания ROBDD-следствий. Осуществляется это при помощи представленной ниже процедуры *check_ impl*().

```
check_impl(u - текущая вершина,
           inf_vector - вектор выведенных значений переменных)
{
    if u == 1:
        // u - терминальная вершина
        // проверить «целостность пути» (проверка условия 1)
        check_path_consistency(u, inf_vector);
        return;

    if not already_in_cache():
        // если такой вершины еще нет в кэше,
        // производим рекурсивный спуск по исходной ROBDD,
        // вызывая check_impl() для потомков данной вершины
        check_impl(low(u),  inf_vector);
        check_impl(high(u), inf_vector);

        // проверить «целостность пути» (проверка условия 1)
        check_path_consistency(u, inf_vector);

        // проверить наличие нулевого ребенка (проверка условия 2)
        check_for_zero_child(u, inf_vector, 0);

        // добавить в кэш запись о том, что данная вершина пройдена
        put_to_cache(u);

    return;
}
```

Как видно из представленного выше псевдокода, процедура *check_ impl*() рекурсивна и осуществляет полный обход B. Для каждой вершины u проверяются условия 1–2 относительно переменной, соответствующей u. Это осуществляется посредством следующих двух процедур: *check_ path_ consistency*() и *check_ for_ zero_ child*().

Процедура *check_ path_ consistency*(), находясь в вершине u, помеченной некоторой переменной $x_k, k \in \{2, \ldots, n\}$, просматривает родителей данной вершины. Если среди них имеется вершина, помеченная переменной $x_j, j < k - 1$, то принимается решение о невозможности возникновения ROBDD-следствий в отношении переменных $x_{j+1}, \ldots, x_{k-1}$.

Процедура *check_ for_ zero_ child*() проверяет для вершины u, помеченной переменной x_k, выполняется ли в ее отношении утверждение леммы 1.

В представленном псевдокоде использована специальная процедура кэширования, назначение которой в хранении информации о пройденных вершинах (ее использование исключает повторное прохождение вершин).

Результатом работы *check_ impl*() является *inf_ vector* — вектор длины n с компонентами из множества $\{-1, 0, 1\}$. На начальном шаге все компоненты данного вектора нулевые. Компонента с номером $k, k \in \{1, \ldots, n\}$, принимает значение 1 или -1, если для переменной x_k по ROBDD-следствию выведено значение соответственно $x_k = 1$ или $x_k = 0$.

Оценим сложность процедуры *check_ impl*(). Заметим, что данная процедура один раз обходит ROBDD B, модифицируя при необходимости вектор *inf_ vector*, при этом работа с произвольной вершиной u может потребовать (в процедуре *check_ path_ consistency*()) просмотра всего текущего вектора *inf_ vector*. Применение процедуры *check_ for_ zero_ child*() в отношении произвольной вершины требует времени $O(1)$. Таким образом, сложность процедуры *check_ impl*() ограничена сверху величиной $O(n \cdot |B|)$. Лемма 2 доказана. ■

Для дальнейшей работы потребуется модифицированный алгоритм *Restrict*, который позволяет осуществлять в ROBDD B подстановку набора значений некоторых переменных из X. Как отмечает Р. Брайант [23, 30], данный алгоритм имеет ту же сложность, что и обычный *Restrict*, то есть $O(|B|)$. Установим теперь справедливость следующей теоремы.

Теорема 4. Пусть в ROBDD B подставляются значения переменных

$$x_{i_1} = \alpha_{i_1}, \ldots, x_{i_m} = \alpha_{i_m}, m \leqslant n, \alpha_{i_j} \in \{0, 1\}, j \in \{1, \ldots, m\}.$$

Сложность детерминированной процедуры, осуществляющей данную подстановку и проверяющей наличие всевозможных ROBDD-следствий, ограничена сверху величиной $O(n \cdot |B|)$.

Доказательство. Используя результаты леммы 2 и модифицированный *Restrict*, можно записать процесс подстановки набора значений переменных в ROBDD с проверкой возникновения ROBDD-следствий в виде следующей процедуры:

```
assign(oldbdd        - исходная ROBDD,
       newbdd        - новая ROBDD,
       assign_vector - вектор подставляемых значений переменных,
       inf_vector    - вектор выведенных значений переменных)
{
    // подстановка в ROBDD oldbdd значений переменных
    // из вектора assign_vector
    restrict_m(oldbdd, newbdd, assign_vector);

    // проверка наличия ROBDD-следствий в ROBDD newbdd
    check_impl(newbdd, inf_vector);
}
```

Учитывая, что сложность процедуры *check_ impl*() ограничена величиной $O(n \cdot |B|)$ (лемма 2), а сложность процедуры *restrict_ m*(), реализующей модифицированный *Restrict*, ограничена величиной $O(|B|)$, заключаем, что верхняя граница сложности для процедуры *assign*() имеет вид $O(n \cdot |B|)$. Теорема 4 доказана. ■

Следствие 1. Если результатом применения процедур *check_ impl*() или *assign*() к B является ROBDD-следствие $x_k = \alpha_k, \alpha_k \in \{0, 1\}$, для некоторой $x_k \in X$, то подстановка в B значения $x_k = \alpha_k$ не может привести к возникновению нового ROBDD-следствия, индуцированного данной подстановкой.

Доказательство. Пусть в результате одной из перечисленных процедур в B возникло ROBDD-следствие $x_k = \alpha_k$. Не ограничивая общности, полагаем, что $\alpha_k = 1$.

В силу леммы 1 сделанное предположение означает, что low-ребенком всех вершин, помеченных x_k, является терминальный «0». Подстановка $x_k = 1$ в B для произвольной вершины $u(x_k)$ означает передачу high-ребенка вершины $u(x_k)$ вершинам, являющимся родителями $u(x_k)$. Но low-ребенок $u(x_k)$, то есть терминальный «0», при этом не передается никаким вершинам. Таким образом, подстановка $x_k = 1$ в B не может привести к *возникновению* в B вершины, ребенком которой является терминальный «0» (что не исключает наличия таких вершин, находившихся в B до осуществления этой подстановки). Аналогичные рассуждения справедливы и в предположении, что $\alpha_k = 0$. Следствие доказано. ■

Данный факт демонстрирует очень привлекательное свойство ROBDD, рассматриваемой в роли базы булевых ограничений. Напомним, что подстановка значения некоторой переменной в КНФ может приводить к выводу по правилу единичного дизъюнкта (unit clause) ряда индуцированных присвоений, подстановка которых также не исключает дальнейших срабатываний unit clause, и т. д. В этом смысл стратегии распространения булевых ограничений (BCP). В общем случае полная реализация BCP может приводить к многократному обходу КНФ, что сопряжено с существенными вычислительными затратами. Полученное свойство ROBDD означает, что порождаемые произвольной подстановкой ROBDD-следствия сами по себе новых ROBDD-следствий породить не могут и, таким образом, вся информация, индуцируемая данной подстановкой, извлекается в результате однократного обхода ROBDD.

На рис. 1 слева показана реализация BCP-стратегии применительно к КНФ $(x_1 \vee x_2)(\overline{x}_2 \vee x_3)(\overline{x}_3 \vee x_4)$ в результате подстановки $x_1 = 0$; справа — результат подстановки $x_1 = 0$ в ROBDD, представляющую булеву функцию, которая выражается той же самой КНФ; требуется единственный обход ROBDD.

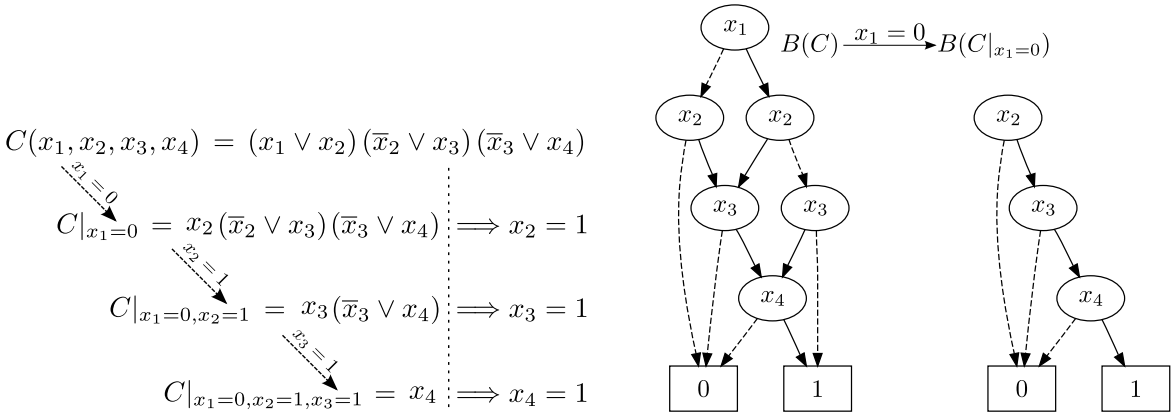

Рис. 1. Пример к следствию теоремы 4

Природа конфликтов в гибридном выводе более разнообразна, чем в DPLL-выводе. Во-первых, это обычные «DPLL-конфликты», возникающие в результате подстановок в КНФ-часть. Во-вторых, это «гибридный конфликт»: ситуация, когда в результате последовательности подстановок в ROBDD-части возникло ROBDD-следствие «$x_k = \alpha$», а в КНФ-части — выведенное по правилу единичного дизъюнкта присвоение «$x_k = \overline{\alpha}$». Однако в целом механизмы разбора конфликтов аналогичны механизмам, используемым в современных SAT-решателях (см. [16]), поэтому здесь дополнительное внимание этим процедурам не уделяется.

Еще одним полезным свойством гибридного вывода является возможность довольно естественной организации на ROBDD т. н. «отсроченных вычислений». Рассмотрим следующие условия, которые определяют ситуацию, в некотором роде двойственную ситуации возникновения ROBDD-следствия:

i) Для некоторой переменной $x_q \in X$ в ROBDD B любой путь π из корня в терминальную вершину «0» обязательно проходит через некоторую вершину, помеченную переменной x_q.

ii) Имеет место $|\Delta^0(x_q)| = 1$.

Установим справедливость следующей теоремы.

Теорема 5. Пусть B — произвольная ROBDD, и относительно некоторой переменной x_q в B справедливы условия *i* и *ii*. Тогда в ROBDD B невозможны ROBDD-следствия ни для каких переменных из множества $X \setminus \{x_q\}$. Трудоемкость процедуры проверки условий *i*–*ii* ограничена сверху величиной $O(n \cdot |B|)$.

Доказательство. Пусть в ROBDD B для некоторой переменной $x_q \in X$ выполняются условия *i* и *ii*. Не ограничивая общности, полагаем, что $\Delta^0(x_q) = \{1\}$. Используя рассуждения, полностью аналогичные тем, посредством которых была доказана лемма 1, можно показать, что в этом случае для любой вершины, помеченной переменной x_q, ее low-ребенком является терминальная вершина «1».

Теперь предположим, что существует такая переменная $x_p \in X \setminus \{x_q\}$, для которой выполнены условия 1–2 вывода некоторого ее ROBDD-следствия. Для данной переменной возможны следующие два варианта ее расположения относительно x_q в порядке означивания переменных в ROBDD B:

$$1 : x_1 \prec \ldots \prec x_q \prec \ldots \prec x_p \prec \ldots$$

$$2 : x_1 \prec \ldots \prec x_p \prec \ldots \prec x_q \prec \ldots$$

Рассмотрим первый случай. Из вышесказанного следует, что low-ребенком любой вершины, помеченной переменной x_q, является терминальная вершина «1». Данный факт означает, что существует обходной путь из корня ROBDD в терминальную вершину «1», не проходящий через вершины, помеченные переменной x_p, то есть для данной переменной вывод ее ROBDD-следствия невозможен.

Рассмотрим второй случай. Как было отмечено выше, условия 1 и 2 означают, что одним из детей любой вершины, помеченной переменной x_p, является «0». Но тогда существуют обходные пути из вершин, помеченных x_p, в «0», не проходящие через вершины, помеченные x_q, что противоречит предположению о выполнимости условий *i* и *ii*. Все проделанные рассуждения переносятся на случай $\Delta^0(x_q) = \{0\}$.

Отметим возможность совмещения процедур подстановки, проверки условий 1–2 и условий *i*–*ii*. В самом деле, проверка условий *i*–*ii* гарантируется использованием аналогов процедур *check_path_consistency*() и *check_for_zero_child*(), которые можно также «встроить» в процедуру *check_imp*(). При этом порядок сложности полученной процедуры останется прежним.

Все сказанное позволяет заключить, что сложность процедуры подстановки значений переменных в ROBDD с отслеживанием ситуаций возникновения ROBDD-следствий и выполнения относительно некоторых переменных (не обязательно одной) условий *i*–*ii* ограничена сверху величиной $O(n \cdot |B|)$. Теорема 5 доказана. ■

Данная теорема позволяет сформировать механизмы отсроченных вычислений при подстановке выведенных в процессе гибридного вывода значений некоторых переменных: если для текущей ROBDD B выполнены *i*–*ii* относительно x_q и из КНФ-части

выведено значение некоторой переменной x_k, $k \neq q$, нет смысла на данном этапе подставлять соответствующее значение в B — ничего нового выведено не будет. После присвоения или вывода из КНФ-части некоторого значения для x_q целесообразно осуществить в B подстановку сразу всех накопленных к этому моменту значений переменных, а также вывод всех возможных ROBDD-следствий, используя для этого процедуру *assign*().

3.2. Процедуры изменения порядка означивания переменных в ROBDD

Порядок означивания переменных существенным образом влияет на число вершин в ROBDD, поэтому одной из центральных проблем при использовании ROBDD в решении практических задач является выбор «хорошего» порядка. Известны различные подходы к этой проблеме. Так, в [24] предлагается изменять порядок означивания динамически — непосредственно в процессе построения ROBDD. Алгоритм, используемый для этой цели в [24], сходен в своей основе с известным методом пузырьковой сортировки (bubble sort, [31]). Сложность данного алгоритма в общем случае экспоненциальна. Более того, нет никакой гарантии, что ее использование даст в конечном счете порядок, лучший, чем заданный изначально.

В гибридном (SAT+ROBDD)-выводе часто есть веские основания считать некоторый порядок лучше текущего. Как правило, это связано со статистикой конфликтности, которая меняется в процессе вывода (переменные с меньшей конфликтностью на предыдущих этапах могут демонстрировать большую конфликтность на последующих). Тем самым возникает следующая задача.

Определение 7. Дана ROBDD $B(f)$, представляющая булеву функцию $f : \{0,1\}^n \to \{0,1\}$, построенная в соответствии с заданным порядком τ означивания переменных из множества $X = \{x_1, \ldots, x_n\}$. Требуется построить ROBDD $B'(f)$, представляющую ту же самую функцию, в которой порядок означивания переменных из X есть τ', отличный от τ. Назовем данную проблему проблемой модификации ROBDD в соответствии с новым порядком.

Для решения данной задачи далее предлагается подход, в корне отличающийся от метода, использующего пузырьковую сортировку. В его основе лежит возможность гарантированного решения за полиномиальное время задачи установки произвольной переменной из X на заданную позицию в новом порядке означивания переменных в ROBDD.

Пусть дана произвольная ROBDD $B(f)$, представляющая булеву функцию f от n переменных и построенная в соответствии с порядком их означивания

$$\tau : x_1 \prec x_2 \prec \ldots \prec x_n. \tag{4}$$

Рассмотрим произвольную подстановку на множестве $\{1, \ldots, n\}$

$$\sigma\,(\tau \to \tau') = \begin{pmatrix} 1 & 2 & \ldots & n \\ \alpha_1 & \alpha_2 & \ldots & \alpha_n \end{pmatrix},$$

$\{\alpha_1, \ldots, \alpha_n\} = \{1, \ldots, n\}$. Будем говорить, что данная подстановка задает изменение исходного порядка τ (4) на порядок τ', подразумевая под этим следующее. Столбец подстановки с номером $i \in \{1, \ldots, n\}$, имеющий вид $\begin{pmatrix} i \\ j \end{pmatrix}$, $j \in \{1, \ldots, n\}$, интерпретирует тот факт, что в новом порядке τ' переменная x_j будет находиться на позиции

с номером i. Например, исходный порядок $\tau : x_1 \prec x_2 \prec x_3$ подстановкой $\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$ изменяется на порядок $\tau' : x_2 \prec x_3 \prec x_1$. Исходному порядку в этом смысле соответствует тождественная подстановка $E_\tau = \begin{pmatrix} 1 & \dots & n \\ 1 & \dots & n \end{pmatrix}$. Также будем говорить, что порядок τ' определяется подстановкой $\sigma\,(\tau \to \tau')$ относительно порядка τ.

Определение 8. Дана ROBDD $B(f)$, построенная в соответствии с заданным порядком τ означивания переменных множества X. Требуется построить ROBDD $B'(f)$, в которой порядок означивания переменных определяется относительно исходного порядка τ произвольной подстановкой следующего вида:

$$\begin{pmatrix} 1 & \dots & i & \dots & n \\ \alpha_1 & \dots & j & \dots & \alpha_n \end{pmatrix}.$$

Данную проблему называем проблемой установки переменной x_j на позицию с номером i.

Теорема 6. Пусть $B(f)$ — произвольная ROBDD с порядком τ (4). Для произвольных $i, j \in \{1, \dots, n\}$ проблема установки переменной x_j на позицию с номером i решается детерминированным образом за время, ограниченное сверху величиной $O\left(|B(f)|^2\right)$.

Доказательство. Пусть дана ROBDD $B(f)$, представляющая булеву функцию от n переменных, образующих множество X. Считаем, что $B(f)$ построена в соответствии с порядком τ (4). Рассмотрим ROBDD B_j^0 и B_j^1, представляющие булевы функции $f^0 = f|_{x_j=0}, f^1 = f|_{x_j=1}$. Данные ROBDD являются результатом применения процедуры *Restrict* к $B(f)$, что требует времени, ограниченного сверху величиной $O\,(|B(f)|)$. Заметим, что B_j^0 и B_j^1 — ROBDD над множеством булевых переменных $X \setminus \{x_j\}$ с заданным на нем порядком

$$\tau^* : x_1 \prec \dots \prec x_{j-1} \prec x_{j+1} \prec \dots \prec x_n.$$

Построим две ROBDD $(B(x_j), B(\overline{x}_j))$ (рис. 2):

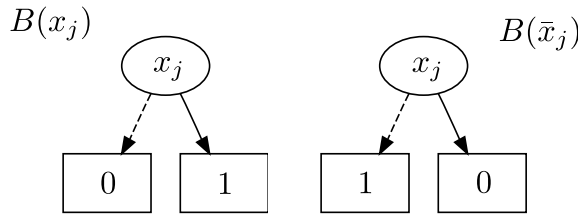

Рис. 2. Элементарные ROBDD

При помощи алгоритма *Apply* построим в соответствии с порядком означивания переменных

$$\tau' : x_1 \prec \dots \prec x_{i-1} \prec x_j \prec x_i \prec \dots \prec x_{j-1} \prec x_{j+1} \prec \dots \prec x_n \tag{5}$$

следующие ROBDD:

$$B^0 = Apply\left(B(\overline{x}_j) \cdot B_j^0\right), B^1 = Apply\left(B(x_j) \cdot B_j^1\right).$$

Рассмотрим ROBDD

$$B'(f) = Apply\left(B^0 \vee B^1\right), \tag{6}$$

построенную в соответствии с порядком τ' (5).

Заметим, что B^0 — ROBDD-представление функции $\overline{x}_j \cdot f|_{x_j=0}$, а B^1 — ROBDD-представление функции $x_j \cdot f|_{x_j=1}$. Таким образом, (6) можно рассматривать как «альтернативный» вид разложения Шеннона булевой функции f по переменной x_j. Тем самым ROBDD $B'(f)$ представляет булеву функцию f в соответствии с порядком (5) означивания переменных множества X.

Поскольку $|B(\overline{x}_j)| = |B(x_j)| = 3$, то сложность построения ROBDD B^0 и B^1 заведомо ограничена сверху величиной $O(|B(f)|)$. Это есть следствие оценки трудоемкости алгоритма *Apply* и того факта, что порядок τ^* подчинен порядку τ' (то есть из $x' \prec_{\tau^*} x''$ следует, что $x' \prec_{\tau'} x''$). В силу сказанного, число вершин в ROBDD B^0 и B^1 также ограничивается сверху величиной $O(|B(f)|)$. Отсюда (снова используя оценку сложности алгоритма *Apply*) имеем: сложность построения ROBDD $B'(f)$ ограничивается сверху величиной $O\left(|B(f)|^2\right)$. Теорема 6 доказана. ■

Данная теорема позволяет предложить следующий алгоритм изменения фиксированного начального порядка τ на новый порядок τ'.

Следствие 2. Пусть $B(f)$ — ROBDD, представляющая булеву функцию f от n переменных в соответствии с порядком означивания переменных τ. Проблема модификации $B(f)$ в соответствии с произвольным порядком τ' сводится к l-кратному ($l \leqslant n$) решению проблемы установки переменной на позицию с заданным номером.

Доказательство. Докажем данный факт, предъявив в явном виде соответствующий алгоритм. Пусть имеется ROBDD $B(f)$, представляющая булеву функцию f от n переменных в соответствии с некоторым порядком означивания переменных τ, и рассматривается некоторый порядок τ', отличный от τ. Тем самым τ' определяется относительно τ некоторой подстановкой, отличной от E_τ. Обозначим вторую строку данной подстановки через $\alpha_{\tau'} = (\alpha_1, \ldots, \alpha_n)$. Описываемый ниже алгоритм получает на входе ROBDD $B(f)$, а также порядки τ и τ'.

0. Пусть $m = 1$.
1. Если $\alpha_m = m$, то полагаем $m = m+1$ и переходим к шагу 1; в противном случае переходим к шагу 2.
2. Решаем проблему установки переменной x_{α_m} на позицию с номером m в текущей ROBDD.
3. Если $m < n$, то полагаем $m = m + 1$ и переходим к шагу 1; в противном случае завершаем выполнение алгоритма.

Теперь покажем, что описанный алгоритм решает проблему модификации ROBDD в соответствии с новым порядком τ'. На каждой итерации алгоритма переменная x_{α_m} устанавливается в ROBDD на позицию с номером m. При этом, как следует из доказательства теоремы 6, все переменные, номера которых меньше m (т. е. установленные на предыдущих итерациях), своего порядка в текущей ROBDD не изменяют. Обобщая сказанное, заключаем, что для корректного решения проблемы модификации ROBDD в соответствии с новым порядком τ' достаточно не более n раз решить проблему установки переменной на заданную позицию, что и делает описанный алгоритм. Следствие доказано. ■

Обратим внимание на то, что данный алгоритм, вообще говоря, не является полиномиальным. Однако фактически он разбит на $l, l \leqslant n$, процедур, каждая из которых

устанавливает некоторую переменную на заданную позицию и сама по себе работает за полиномиальное от числа вершин в подаваемой ей на вход ROBDD время. На практике алгоритм показывает хорошие результаты в задачах модификации порядка ROBDD, возникающих в гибридном (SAT+ROBDD)-выводе. Его эффективность особенно заметна при незначительных отличиях в новом и старом порядках (когда большая часть переменных не меняет своего местоположения). Следует подчеркнуть, что для процедуры, использующей пузырьковую сортировку, в общем случае невозможны полиномиальные оценки даже в отношении задачи установки переменной на заданную позицию.

Заключение

Настоящая работа посвящена гибридному (SAT+ROBDD)-выводу, используемому в задачах обращения дискретных функций. В соответствии с гибридной стратегией некоторый алгоритм на основе DPLL действует в отношении КНФ $C(f)$, кодирующей задачу обращения полиномиально вычислимой функции f в некоторой точке. После каждого рестарта вместо чистки базы конфликтных дизъюнктов используется процедура построения ROBDD-представления характеристической функции системы вида (3). Дальнейший вывод идет как на исходной КНФ, так и на ROBDD, представляющей соответствующую базу накопленных ограничений.

Построены аналоги основных операций с булевыми ограничениями, используемыми в нехронологическом DPLL-выводе: аналог подстановки с отслеживанием срабатывания правила единичного дизъюнкта — процедура *assign*() (теорема 4); аналог CL-процедуры — применение *Apply* к текущей ROBDD и новому ограничению-дизъюнкту; аналог механизма действия структур watched literals определяется условиями *i*–*ii* и теоремой 5. Для всех предложенных алгоритмов приведены сложностные оценки. Кроме этого, предложен новый алгоритм модификации порядка в заданной ROBDD, показывающий хорошие практические результаты.

Отметим также, что гибридный (SAT+ROBDD)-подход к обращению дискретных функций был программно реализован с использованием стандарта MPI [32] в виде решателя, функционирующего в распределенной вычислительной среде. В решателе используется межпроцессорный обмен накапливаемыми ограничениями, передаваемыми в виде ROBDD. В этом его принципиальное отличие от недавних зарубежных разработок (см. [33]), в которых при решении SAT-задач также используется межпроцессорное взаимодействие. Построенный решатель показал высокую эффективность на задачах обращения некоторых криптографических функций.

Авторы выражают глубокую благодарность А. Е. Хмельнову за внимание к представленным в статье исследованиям, дружеские советы и конструктивные замечания.

ЛИТЕРАТУРА

1. `http://www.satlive.org`
2. `www.isa.ewi.tudelft.nl/jsat`
3. *Hachtel G. D., Somenzi F.* Logic synthesis and verification algorithms. Kluwer Ac. Publ, 2002.
4. *Кларк Э. М., Грамберг О., Пелед Д.* Верификация моделей программ: Model Checking. М.: МЦНМО, 2002.
5. *Гаранина Н. О., Шилов Н. В.* Верификация комбинированных логик знаний, действий и времени в моделях // Системная информатика. 2005. Вып. 10. С. 114–173.

6. *Семенов А. А., Заикин О. С., Беспалов Д. В., Ушаков А. А.* SAT-подход в криптоанализе некоторых систем поточного шифрования // Вычислительные технологии. 2008. Т. 13. № 6. С. 134–150.
7. 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.
8. *Davis M., Logemann G., Loveland D.* A machine program for theorem proving // Comm. ACM. 1962. V. 5. P. 394–397.
9. *Marqeus-Silva J. P., Sakallah K. A.* GRASP: A search algorithm for propositional satisfiability // IEEE Trans. Comp. 1999. V. 48. No. 5. P. 506–521.
10. *Moskewicz M., Madigan C., Zhao Y., et al.* Chaff: Engineering an Efficient SAT Solver // Proc. Design Automat. Conf. (DAC). 2001. P. 530–535.
11. `http://minisat.se/MiniSat.html` — MiniSat.
12. *Семенов А. А., Заикин О. С.* Неполные алгоритмы в крупноблочном параллелизме комбинаторных задач // Вычислительные методы и программирование. 2008. Т. 9. С. 108–118.
13. *Семенов А. А.* Декомпозиционные представления логических уравнений в задачах обращения дискретных функций // Изв. РАН. Теория и системы управления. 2009. № 5. С. 47–61.
14. *Закревский А. Д.* Логические уравнения. Минск: Наука и техника, 1975.
15. *Семенов А. А., Беспалов Д. В.* Технологии решения многомерных задач логического поиска // Вестник Томского госуниверситета. Приложение. 2005. № 14. С. 61–73.
16. *Zhang L., Madigan C., Moskewicz M., Malik S.* Efficient conflict driven learning in a boolean satisfiability solver // Proc. Intern. Conf. on Computer Aided Design (ICCAD). 2001. P. 279–285.
17. *Lynce I., Marques-Silva J. P.* Efficient data structures for backtrack search SAT solvers // Ann. Mathem. Artific. Intellig. 2005. V. 43. P. 137–152.
18. *Семенов А. А., Заикин О. С., Беспалов Д. В. и др.* Решение задач обращения дискретных функций на многопроцессорных вычислительных системах // Труды Четвертой Междунар. конф. PACO'2008 (Москва, 26–29 октября 2008).М., 2008. С. 152–176.
19. *Заикин О. С., Семенов А. А.* Технология крупноблочного параллелизма в SAT-задачах // Проблемы управления. 2008. № 1. С. 43–50.
20. *Семенов А. А.* Трансляция алгоритмов вычисления дискретных функций в выражения пропозициональной логики // Прикладные алгоритмы в дискретном анализе. Сер.: Дискретный анализ и информатика. 2008. Вып. 2. С. 70–98.
21. *Cook S. A.* The complexity of theorem-proving procedures // Proc. 3rd Ann. ACM Symp. on Theory of Computing, ACM. 1971. P. 151–159. [Пер.: *Кук С. А.* Сложность процедур вывода теорем // Кибернетический сборник. Новая серия. 1975. Вып. 12. С. 5–15.]
22. *Lee C. Y.* Representation of Switching Circuits by Binary-Decision Programs // Bell Systems Technical J. 1959. V. 38. P. 985–999.
23. *Bryant R. E.* Graph-Based Algorithms for Boolean Function Manipulation // IEEE Trans. Comp. 1986. V. 35. No. 8. P. 677–691.
24. *Meinel Ch., Theobald T.* Algorithms and Data Structures in VLSI-Design: OBDD-Foundations and Applications. Springer Verlag, 1998.
25. *Хмельнов А. Е., Игнатьев А. С., Семенов А. А* Двоичные диаграммы решений в логических уравнениях и задачах обращения дискретных функций // Вестник НГУ. Сер.: Информационные технологии. 2009. Т. 7. № 4. С. 36–52.
26. *Семенов А. А.* О преобразованиях Цейтина в логических уравнениях // Прикладная дискретная математика. 2009. № 4. С. 28–50.

27. *Цейтин Г. С.* О сложности вывода в исчислении высказываний // Записки научных семинаров ЛОМИ АН СССР. 1968. Т. 8. С. 234–259.
28. *Beame P., Kautz H., Sabharwal A.* Understanding the power of clause learning // Proc. Of 18th Intern. Joint Conf. on Artificial Intellegence (IJCAI). 2003. P. 1194–1201.
29. *Игнатьев А. С., Семенов А. А., Хмельнов А. Е.* Использование двоичных диаграмм решений в задачах обращения дискретных функций // Вестник Томского госуниверситета. Сер.: Управление, вычислительная техника, информатика. 2009. № 1(6). С. 115–129.
30. *Bryant R. E.* Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams // ACM Comp. Surv. 1992. V. 24. No. 3. P. 293–318.
31. *Левитин А.* Алгоритмы. Введение в разработку и анализ. М.: Вильямс, 2006.
32. *Воеводин В. В., Воеводин Вл. В.* Параллельные вычисления. СПб.: БХВ-Петербург, 2002.
33. *Schubert T., Lewis M., Becker B.* PaMiraXT: Parallel SAT Solving with Threads and Message Passing // J. Satisf., Bool. Mod. Comp. 2009. V. 6. P. 203–222.