Алгоритм поиска произвольного запрета был применён ко всем функциям от 2, 3 и 4 переменных; результаты приведены в табл. 1.

Таблица 1

Количество функций, имеющих запрет

n	Кол-во функций с запретом	Кол-во функций без запрета
2	10	6
3	226	30
4	64954	582

В табл. 2 приведены данные о среднем времени работы алгоритма поиска произвольного запрета для функций с количеством аргументов больше 5.

Таблица 2

Время работы алгоритма

n	Среднее время, с	n	Среднее время, с
5	0,00019	13	0,128
6	0,0004	14	0,307
7	0,00091	15	0,698
8	0,0021	16	1,548
9	0,0047	17	3,461
10	0,011	18	7,552
11	0,025	19	16,528
12	0,061	20	33,312
		21	77,999

В дальнейшем предполагается разработать и реализовать алгоритмы решения задач 1–3 с помощью обхода дерева в ширину и в глубину (поиск с возвратом) и сравнить трудоёмкости этих алгоритмов (по количеству построенных вершин дерева и сложности их обработки).

ЛИТЕРАТУРА

1. *Колегов Д. Н.* О булевых функциях без запрета // Вестник Томского государственного университета. Приложение. 2005. № 14. С. 58–60.

УДК 519.7

ОБ ЭФФЕКТИВНОМ ПРЕДСТАВЛЕНИИ ДИЗЪЮНКТИВНЫХ НОРМАЛЬНЫХ ФОРМ ДИАГРАММАМИ СПЕЦИАЛЬНОГО ВИДА[1]

А. А. Семенов

Рассматриваются диаграммы нового типа, используемые для представления дизъюнктивных нормальных форм. Показано, что данный класс диаграмм можно применять для компактного представления баз конфликтных дизъюнктов, накапливаемых в процессе нехронологического DPLL-вывода.

Ключевые слова: *ДНФ, решающие диаграммы, BDD, ZDD, дизъюнктивные диаграммы.*

[1]Работа выполнена при поддержке гранта РФФИ № 11-07-00377.

Рассматривается произвольная всюду определённая булева функция $g : \{0,1\}^n \to \{0,1\}$ и произвольная ДНФ $D(g)$, представляющая g:

$$D(g) = K_1 \vee \ldots \vee K_m,$$

где K_j, $j \in \{1,\ldots,n\}$, — конъюнкты над множеством булевых переменных $X = \{x_1, \ldots, x_n\}$. Далее ДНФ $D(g)$ будем рассматривать также как множество конъюнктов $\mathscr{D}(g) = \{K_1,\ldots,K_m\}$. Зафиксируем на X порядок τ ($x_1 \prec x_1 \prec \ldots \prec x_n$). Каждый конъюнкт K_j, $j \in \{1,\ldots,m\}$, рассматривая его как множество литералов, упорядочим относительно τ.

Выделим в множестве $\mathscr{D}(g)$ подмножество конъюнктов, первый литерал в которых принадлежит множеству $L_{x_1} = \{x_1, \overline{x_1}\}$. Обозначим полученное множество конъюнктов через $\mathscr{D}_{x_1}$. Рассмотрим множество $\mathscr{D}^1 = \mathscr{D}(g)\backslash\mathscr{D}_{x_1}$ (полагаем, что оно не пусто). Обозначим через X^1 множество булевых переменных, фигурирующих в конъюнктах из $\mathscr{D}(g)\backslash\mathscr{D}_{x_1}$. Очевидно, что $X^1 \subseteq X\backslash\{x_1\}$. Пусть x_{i_1} — первая в смысле порядка τ переменная в X^1. Выделим в $\mathscr{D}(g)\backslash\mathscr{D}_{x_1}$ подмножество конъюнктов, первый литерал в которых принадлежит множеству $L_{x_{i_1}} = \{x_{i_1}, \overline{x_{i_1}}\}$. Обозначим полученное множество конъюнктов через $\mathscr{D}_{x_{i_1}}$. Рассматриваем множество $\mathscr{D}^2 = \mathscr{D}^1\backslash\mathscr{D}_{x_{i_1}}$. Далее поступаем по аналогии. Процесс продолжается до тех пор, пока для некоторого $k \geqslant 1$ не выполнится $\mathscr{D}^k = \varnothing$, $\mathscr{D}^k = \mathscr{D}^{k-1}\backslash\mathscr{D}_{x_{i_{k-1}}}$ (считаем, что $\mathscr{D}^0 = \mathscr{D}(g)$, $x_{i_0} = x_1$). В итоге имеем представление $\mathscr{D}(g)$ в виде

$$\mathscr{D}(g) = \bigcup_{k\in\{1,\ldots,t\}} \mathscr{D}_{x_k}, \tag{1}$$

$\{1,\ldots,t\} \subseteq \{1,\ldots,n\}$, которому соответствует следующее представление $D(g)$:

$$D(g) = \bigvee_{k\in\{1,\ldots,t\}} D_{x_k}, \tag{2}$$

где D_{x_k} — ДНФ, составленная из конъюнктов, входящих в $\mathscr{D}_{x_k}$.

Не ограничивая общности, рассмотрим D_{x_1}. Пусть $\left\{x_1^1, x_2^1, \ldots, x_{p_1}^1\right\}$ — множество переменных, фигурирующих в D_{x_1}, $x_1^1 = x_1$. Представим D_{x_1} в виде

$$D_{x_1} = x_1 \cdot \Psi_{x_1} \vee \overline{x_1} \cdot \Psi_{\overline{x_1}}, \tag{3}$$

где Ψ_{x_1} и $\Psi_{\overline{x_1}}$ — некоторые ДНФ над множеством переменных $\left\{x_2^1, \ldots, x_{p_1}^1\right\}$. Представим каждую из этих ДНФ в форме (2), используя порядок τ.

Сделаем аналогичные действия для всех остальных D_{x_k}, входящих в правую часть (2). Продолжим описанный процесс рекурсивно.

Отметим, что каждое $\mathscr{D}_{x_k}$, входящее в (1), можно представить в виде r-арного дерева T_{x_k}, $r \leqslant n-1$. Листья дерева T_{x_k} помечаются либо символом «1», либо символом «?». Корень дерева T_{x_k} помечается x_k. Внутренние вершины каждого пути из корня в лист помечаются переменными из множества X. Для обозначения вершин с приписанными им переменными будем использовать выражения вида $v(x_s)$. Каждый путь в T_{x_k} из корня в лист, помеченный «1», соответствует конъюнкту из $\mathscr{D}_{x_k}$. Рёбра, выходящие из вершин, могут быть двух типов — пунктирные и сплошные. Если путь, проходящий через вершину $v(x_s)$ из корня T_{x_k} в лист, помеченный «1», определяет конъюнкт, в который x_s входит в виде литерала $\overline{x_s}$, то соответствующее ребро из $v(x_s)$ является пунктирным. Если же данный путь определяет конъюнкт, в который x_s входит без

отрицания, то соответствующее ребро является сплошным. Выходная степень каждой вершины (за исключением листьев) не меньше 2, причём из каждой вершины выходят как пунктирные, так и сплошные ребра.

Из (1) и определения T_{x_k}, $k \in \{1, \ldots, t\}$, следует, что ДНФ $D(g)$ представима в виде леса, состоящего из t деревьев. Данный лес называется дизъюнктивным лесом.

Пример 1. Построим дизъюнктивный лес по ДНФ

$$x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 \vee x_1 \cdot x_2 \cdot \overline{x_3} \cdot \overline{x_4} \vee \overline{x_1} \cdot x_2 \cdot x_4 \vee \overline{x_1} \cdot \overline{x_4} \vee x_2 \cdot \overline{x_3} \cdot \overline{x_4} \vee \overline{x_2} \cdot x_4 \vee x_3 \cdot \overline{x_4}.$$

Фиксируем порядок $x_1 \prec x_2 \prec x_3 \prec x_4$. Представление (2) имеет вид

$$D_{x_1} = x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 \vee x_1 \cdot x_2 \cdot \overline{x_3} \cdot \overline{x_4} \vee \overline{x_1} \cdot x_2 \cdot x_4 \vee \overline{x_1} \cdot \overline{x_4},$$
$$D_{x_2} = x_2 \cdot \overline{x_3} \cdot \overline{x_4} \vee \overline{x_2} \cdot x_4,$$
$$D_{x_3} = x_3 \cdot \overline{x_4}.$$

Представление (3), например, для ДНФ D_{x_1} выглядит следующим образом:

$$x_1 \cdot \left(x_2 \cdot \overline{x_3} \cdot \overline{x_4} \vee \overline{x_2} \cdot \overline{x_3} \cdot x_4\right) \vee \overline{x_1} \cdot \left(x_2 \cdot x_4 \vee \overline{x_4}\right).$$

Итогом применения описанной рекурсивной процедуры к рассматриваемой ДНФ является дизъюнктивный лес (рис. 1).

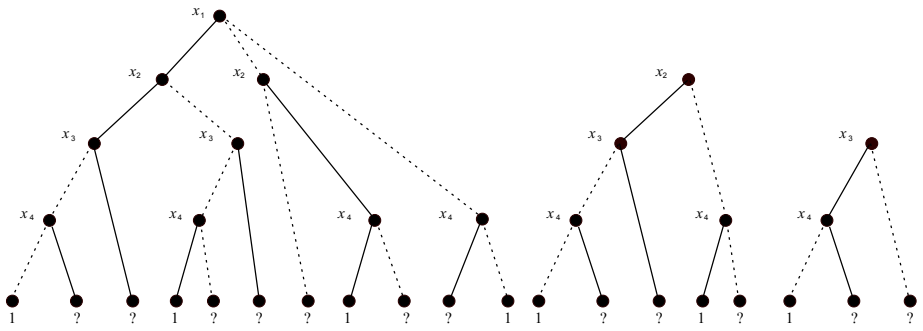

Рис. 1. Дизъюнктивный лес ДНФ из примера 1

Отметим, что идея представления ДНФ в форме дизъюнктивных лесов близка к идеям использования древовидных структур для представления булевых функций и множеств (соответственно BDD [1] и ZDD [2]). Однако имеются и существенные отличия. Во-первых, число путей в дизъюнктивном лесе заведомо ограничено полиномом от $n \cdot m$. Во-вторых, дизъюнктивный лес нельзя рассматривать как дерево решений (decision tree) булевой функции — пути, ведущие в ?-листья, вообще говоря, не дают наборы значений, на которых функция g принимает значение 0. В следующем частном случае, однако, дизъюнктивный лес совпадает с деревом решений.

Утверждение 1. Пусть $D(g)$ — СДНФ булевой функции g. Тогда дизъюнктивный лес, построенный по $D(g)$ в соответствии с описанными выше правилами, — это бинарное дерево, совпадающее с деревом решений g. В этом случае любой путь из корня в ?-лист определяет множество наборов значений переменных, на которых функция g принимает значение 0.

Используя структуры данных для представления КНФ/ДНФ, описанные в [3], можно показать, что справедлив следующий факт.

Теорема 1. Существует алгоритм, который по произвольной ДНФ $D(g)$ строит представляющий её дизъюнктивный лес за время $\mathrm{O}(|D(g)|)$, где $|D(g)|$ — объём двоичного кода $D(g)$.

Пусть $D(g)$ — произвольная ДНФ и $F(D(g))$ — представляющий её дизъюнктивный лес. Каждую вершину в $F(D(g)$, за исключением терминальных вершин «1» и «?», можно задать следующим набором координат:

$$v(x) = \left\langle x, v_1^l, \ldots, v_{k_l}^l, v_1^h, \ldots, v_{k_h}^h \right\rangle .$$

Здесь $v_1^l, \ldots, v_{k_l}^l$, $k_l \leqslant n-1$, — вершины, которые являются детьми $v(x)$ по пунктирным рёбрам (по аналогии с используемой при описании BDD терминологией называем эти вершины low-детьми); $v_1^h, \ldots, v_{k_h}^h$, $k_h \leqslant n-1$, — вершины, которые являются детьми вершины $v(x)$ по сплошным рёбрам (high-дети).

По аналогии с процедурами построения BDD и ZDD можем определить операции склеивания вершин произвольного дизъюнктивного леса — склеиваются вершины, имеющие одинаковые наборы координат. Подобным же образом определяются операции сокращения. Результатом применения к дизъюнктивному лесу $F(D(g))$ склеивания и сокращения является дизъюнктивная диаграмма, обозначаемая через $\Delta(D(g))$.

Теорема 2. Существует алгоритм, который строит по произвольной ДНФ представляющую её дизъюнктивную диаграмму за время $\mathrm{O}(n \cdot |D(g)|)$.

На рис. 2 приведена дизъюнктивная диаграмма, полученная в результате применения операций склеивания и сокращения к дизъюнктивному лесу рис. 1.

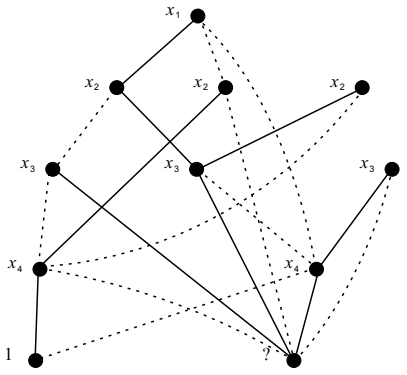

Рис. 2. Дизъюнктивная диаграмма для ДНФ из примера 1

Основное предназначение дизъюнктивных диаграмм — компактное представление конфликтных баз, накапливаемых в процессе нехронологического DPLL-вывода. Напомним, что в соответствии с [4] нехронологический DPLL сохраняет информацию о пройденных участках дерева поиска в виде булевых ограничений, называемых конфликтными дизъюнктами. Если C — исходная КНФ, к которой применяется DPLL, и D — конфликтный дизъюнкт, то C выполнима тогда и только тогда, когда выполнима КНФ $C \cdot D$. Современные SAT-решатели, использующие нехронологический DPLL, могут накапливать очень большие объёмы конфликтных дизъюнктов. Заполнение конфликтной информацией оперативной памяти негативно сказывается на эффективности вывода. Таким образом, актуальна разработка таких приёмов модификации конфликтных баз, которые позволяют существенно снижать объём оперативной памяти, занимаемой этими базами.

В [5, 6] описан гибридный SAT+ROBDD вывод, в котором для представления конфликтных баз используются ROBDD. В работе [6] определён специальный тип вывода, представляющий собой ROBDD-аналог итеративного применения правила Unit Propagation. Основной минус такого подхода — необходимость согласования двух структур данных, ROBDD и КНФ, что неизбежно сказывается на скорости работы программы. При этом несложно показать, что переход от ROBDD, представляющей конфликтную базу, к КНФ в соответствии со стандартными техниками (см., например, [7]) приводит к тому, что на конфликтной части КНФ в общем случае перестаёт работать FDA-процедура (Failure Driven Assertion [4]), которая является одним из базовых элементов во всех современных нехронологических DPLL-решателях.

Покажем, что данную проблему можно эффективно решить, если вместо ROBDD использовать дизъюнктивные диаграммы.

Теорема 3. Пусть C' — произвольная КНФ над множеством переменных X, D — произвольный дизъюнкт, входящий в C', и $L_D = \left\{x_{i_1}^{\alpha_{i_1}}, \ldots, x_{i_s}^{\alpha_{i_s}}\right\}$ — множество литералов данного дизъюнкта. Пусть $\Delta(\overline{C'})$ — дизъюнктивная диаграмма, представляющая ДНФ $\overline{C'}$, число вершин в которой есть N. Существует алгоритм, который за время $\mathrm{O}(N)$ строит по $\Delta\left(\overline{C'}\right)$ КНФ $\tilde{C}$ над множеством переменных Y, $X \subset Y$, такую, что подстановка в $\tilde{C}$ произвольного множества литералов вида $\left\{x_{j_1}^{\overline{\alpha_{j_1}}}, \ldots, x_{j_{s-1}}^{\overline{\alpha_{j_{s-1}}}}\right\}$, $\{j_1, \ldots, j_{s-1}\} \subset \{i_1, \ldots, i_s\}$, даёт вывод по правилу Unit Propagation литерала из $L_D \setminus \left\{x_{j_1}^{\alpha_{j_1}}, \ldots, x_{j_{s-1}}^{\alpha_{j_{s-1}}}\right\}$.

Предположим, что C' — конфликтная база, накопленная нехронологическим DPLL-выводом при решении проблемы выполнимости КНФ C. Пусть D — произвольный конфликтный дизъюнкт. Теорема 3 утверждает наличие эффективного алгоритма, строящего по дизъюнктивной диаграмме $\Delta(\overline{C'})$ КНФ $\tilde{C}$, которая может сама по себе использоваться в роли базы конфликтных ограничений. При этом на $\tilde{C}$ работает FDA-процедура, то есть подстановка в $\tilde{C}$ отрицаний произвольных $s-1$ литералов дизъюнкта D даёт вывод по правилу Unit Propagation оставшегося литерала из D. Особо отметим, что на практике объём памяти, занимаемый КНФ $\tilde{C}$, может быть существенно (в некоторых ситуациях в разы) меньше, чем объём, занимаемый C'.

ЛИТЕРАТУРА

1. *Bryant R. E.* Graph-based algorithms for Boolean function manipulation // IEEE Trans. Comput. 1986. V. 35. No. 8. P. 677–691.
2. *Minato S.* Zero-suppressed BDDs for set manipulation in combinatorial problems // Proc. DAC-93, June 14–18, 1993, Dallas, Texas. P. 272–277.
3. *Dowling W. and Gallier J.* Linear-time algorithms for testing the satisfiability of propositional Horn formulae // J. Logic Programming. 1984. V. 1. No. 3. P. 267–284.
4. *Marques-Silva J. P. and Sakallah K. A.* GRASP: A search algorithm for propositional satisfiability // IEEE Trans. Comput. 1999. V. 48. No. 5. P. 506–521.
5. *Игнатьев А. С., Семенов А. А.* Алгоритмы работы с ROBDD как с базами булевых ограничений // Прикладная дискретная математика. 2010. № 1. С. 86–104.
6. *Ignatiev A. and Semenov A.* DPLL+ROBDD derivation applied to inversion of some cryptographic functions // LNCS. 2011. V. 6695. P. 76–89.
7. *Een N. and Sorensson N.* Translating pseudo-Boolean constraints into SAT // J. Satisfiability, Boolean Modeling and Computation. 2006. V. 2. P. 1–25.