УДК 510.5
DOI: 10.15827/0236-235X.131.396-403
Дата подачи статьи: 05.03.20
2020. Т. 33. № 3. С. 396–403

Структуры данных и модификация метода Квайна–МакКласки при минимизации нормальных форм

Н.И. Гданский [1], *д.т.н., профессор, al-kpp@mail.ru*
А.А. Денисов [1], *аспирант, aadenisov88@gmail.com*
Н.Л. Куликова [2], *к.т.н., доцент, kulikovanl@mpei.ru*

[1] *Московский государственный университет технологий и управления им. К.Г. Разумовского (Первый казачий университет), г. Москва, 109004, Россия*
[2] *Национальный исследовательский университет "МЭИ", г. Москва, 111250, Россия*

Логические методы анализа и синтеза систем самой различной природы обычно основаны на использовании описаний их структур и процессов в них в виде булевых функций, которые для унификации представления эквивалентно приводятся к конъюнктивным нормальным формам. Для исходных систем и процессов, как правило, основным критерием оптимальности является минимальное число составляющих их компонент, при котором упрощается структура, уменьшается стоимость и повышается надежность, поэтому для модельных конъюнктивных нормальных форм большое практическое значение имеет задача их минимизации.

Эффективность алгоритмов, обрабатывающих сложные объекты (к которым относятся и конъюнктивные нормальные формы), существенно зависит от основных и вспомогательных структур данных, используемых для представления этих объектов, поэтому на основе анализа существующих структур разработана новая комплексная трехуровневая структура данных для представления конъюнктивных нормальных форм.

Нижний уровень всей комбинированной структуры представляет собой структура данных для одиночного дизъюнкта. Средний уровень образуют списки дизъюнктов с одним и тем же числом литер. На верхнем уровне вся конъюнктивная нормальная форма задается массивом списков, упорядоченных по длинам входящих в них дизъюнктов.

Использование списков, указателей на них и отдельные элементы, упорядоченность дизъюнктов по длинам дают возможность радикально сократить операции по перезаписи информации, ее упорядочению в процессе преобразования конъюнктивных нормальных форм.

На основе разработанной комплексной структуры данных разработана модификация известного метода Квайна–МакКласки, применяемого для сокращения совершенных нормальных форм.

Совместное применение предложенной структуры данных и модифицированного метода позволяет существенно уменьшить общее число операций при минимизации конъюнктивных нормальных форм по сравнению с базовой версией метода Квайна–МакКласки. Оно достигается за счет сокращения повторной обработки данных, а путем использования специальных логических условий достигается дополнительное сокращение общего числа проверок литер в дизъюнктах при их сравнении.

Ключевые слова: *дизъюнкт, конъюнктивная нормальная форма, структуры данных, метод Квайна–МакКласки.*

С использованием аппарата *конъюнктивных нормальных форм* (КНФ) моделируется решение широкого круга задач в технике, системном анализе и искусственном интеллекте: автоматическая генерация тестов (ATPG – Automatic Test Patten Generation) [1, 2], проверка эквивалентности схем (combinational circuit equivalence checking) [3, 4], проверка последовательности (sequential property checking) [5], проверка микропроцессоров (microprocessor verification) [6], проверка моделей (model checking) [7, 8], анализ достижимости (reachability analysis) [9, 10], временной анализ (timing analysis) [11], маршрутизация (routing) [12], описание задач искусственного интеллекта [13] и др.

Формула КНФ имеет вид $F = D_1 \& D_2 \& \ldots \& D_k$, где $\forall s\ (1 \le s \le k)$ $D_s = x_{i1}^{\alpha i1} \vee x_{i2}^{\alpha i2} \vee \ldots \vee x_{ils}^{\alpha ils}$, $\alpha_{ilp} = 0$ или 1, $x_{ip}^{0} = \neg x_{ip}$, $x_{ip}^{1} = x_{ip}$.

Длина формулы равна сумме длин всех ее дизъюнктов: $L = l_1 + \ldots + l_k$.

Унифицированный способ представления позволяет при решении задач применять к таким формулам более эффективные с вычислительной точки зрения специализированные алгоритмы обработки. За счет эквивалентной минимизации модельных КНФ можно значительно улучшить свойства моделируемых с их

помощью реальных объектов, систем и процессов, поскольку при минимизации упрощается их структура, уменьшаются материалоемкость и стоимость, повышаются технологичность и надежность [14]. В отдельных случаях в процессе минимизации КНФ может быть установлена тождественная истинность или тождественная ложность данных формул, что снимает задачу их последующего анализа.

Как следует из формулы, КНФ является довольно сложным объектом, в котором дизъюнкты – внутренние логические суммы переменных и их отрицаний (литер) соединены между собой внешней операцией логического умножения. Поэтому, как и для других сложных объектов, эффективность алгоритмов их анализа во многом определяется структурами данных, которые используются для программного представления КНФ. Рассмотрим их.

Для входного представления КНФ обычно используется стандартный формат DIMACS CNF, в котором дизъюнкты формулы задают в виде строк в текстовом файле. Однако при компьютерной обработке в качестве основных структур данных, используемых при хранении и последующем преобразовании КНФ, применяют другие структуры: разреженные матрицы (sparse matrix), структуры из указателей (GRASP [15], relsat [16]), в отдельном массиве (chaff), дерево (SATO) [17, 18].

Общим недостатком данных структур является то, что они позволяют оптимизировать только часть базовых операций, используемых при обработке КНФ, и неэффективны при выполнении других. Для достижения максимального эффекта необходимо комплексное сочетание структур различного типа, в наибольшей степени соответствующее как структуре КНФ, так и основным операциям, выполняемым с ними.

В статье рассмотрены две основные задачи:

– разработка комплексной структуры данных для представления КНФ, позволяющей максимально сократить общее число операций, выполняемых при минимизации;

– модификация стандартного варианта алгоритма Квайна–МакКласки, учитывающая свойства таких структур данных.

Структура данных для представления КНФ

Рассмотрим КНФ, у которых дизъюнкты не содержат дублирующие литеры и не допускаются контрарные пары литеры (вида x_i и $\neg x_i$). Такие дизъюнкты для определенности будем называть *существенными*. При необходимости КНФ можно привести к данному виду за счет линейного прохода по ней (это можно выполнить непосредственно при считывании очередного дизъюнкта).

Для максимального упрощения и ускорения всех необходимых преобразований КНФ предложена комплексная трехуровневая структура данных. Рассмотрим ее подробнее.

Поскольку основу структуры КНФ составляют дизъюнкты, прежде всего введен класс Clause, представляющий дизъюнкт, вспомогательные данные и операции с ними. Данные представлены следующими структурами:

```
std::vector<int> m_ch_vector; // характеристический вектор
std::vector<int> m_variables; // входящие переменные
int m_negs; // число отрицаний
int m_sum; // сумма номеров переменных
bool glued = false; // участвовал ли в «склейке»
```

Использование массивов m_ch_vector и m_variables позволяет ускорить операции проверки включения дизъюнктов, наличия в их составе переменных, способ их включения и другие базовые операции.

Для реализации основных операций в класс включены следующие методы:

```
Clause(int vars); // инициализирующий конструктор
int length() const; // число литер
int negatives() const; // число отрицаний
int var_sum() const; // сумма номеров переменных
bool add_literal(int x); // добавление литеры
void remove_literal(int x); // удаление литеры с номером x
friend bool absorbs(const Clause &lhs, const Clause &rhs); // определение поглощений
int operator[](int i) const;
int& operator[](int i);
int operator()(int i) const;
friend bool includes(const Clause &lhs, const Clause &rhs);
friend int neighbours(const Clause &lhs, const Clause &rhs);
void clear() // удаление дизъюнкта.
```

Для ускорения последующих действий все дизъюнкты КНФ сгруппированы в списки по длинам. Поскольку вся КНФ представлена структурой данных в виде совокупности списков существенных дизъюнктов одинаковой

длины, рассмотрен также класс ClauseList, реализующий список дизъюнктов и операции над ним. Его единственная структура:

```
std::list<Clause> m_list; // массив списков по длинам
```

Методы:

```
void insert(const Clause &c); // вставка дизъюнкта
std::list<Clause>::iterator ClauseList::erase(std::list<Clause>::iterator pos); // удаление по итератору
bool insert_unique(const Clause &c); // вставка с предварительной проверкой на уникальность
std::list<Clause>::iterator begin(); // возвращает итератор на начало списка
std::list<Clause>::iterator end(); // возвращает итератор на конец списка
bool empty(); // определяет, пуст ли список
void clear(); // удаляет все элементы списка
size_t size(); // возвращает длину списка
friend void splice_append(ClauseList &dest, ClauseList &src); // присоединяет список src к dest
friend void splice_append(ClauseList &dest, ClauseList &src, std::list<Clause>::iterator &src_it); // присоединяет один элемент из src, на который указывает src_it, к списку dest без копирования данных
friend bool append_unique(ClauseList &dest, ClauseList &src, std::list<Clause>::iterator &src_it); // вставка с предварительной проверкой на уникальность
};
```

Вся КНФ в целом задается массивом списков ClauseList, упорядоченных по длинам дизъюнктов, входящих в них.

Использование списков, указателей на них и отдельные элементы, упорядоченность дизъюнктов по длинам дают возможность радикально сократить операции по перезаписи информации и ее упорядочению в процессе преобразования КНФ. Применение такой специализированной комплексной структуры данных позволяет наряду с минимизацией упростить выполнение и других операций с формулами КНФ.

Общая идея модифицированного алгоритма Квайна

Сокращение избыточности в формуле КНФ, обусловленной наличием дизъюнктов с соседними наборами литер. Традиционный метод Квайна. На первом этапе обычного метода Квайна решается задача перехода от *совершенной конъюнктивной нормальной формы* (СКНФ, представляющей собой произведение конституент нулей $K(x_1, x_2, \ldots, x_n)$) булевой функции $f(x_1, x_2, \ldots, x_n)$, к ее *сокращенной конъюнктивной нормальной форме* (СкКНФ), состоящей из всех простых наборов функции.

Если на наборе $\bar{\alpha}^n = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ значение $f(\bar{\alpha}^n) = 0$, то в соответствующей конституенте $K_1(x_1, x_2, \ldots, x_n)$ СКНФ, представляющей собой логическую сумму, переменные назначаются по следующему правилу. Если $\alpha_i = 0$, то переменная x_i входит в конституенту $K_1(x_1, x_2, \ldots, x_n)$ в виде тождественной функции. Если же $\alpha_i = 1$, то переменная x_i входит с отрицанием. Например, если $f(1, 0, 0, 1, 0) = 0$, то конституента, соответствующая набору переменных (1, 0, 0, 1, 0), будет иметь вид $K_1(x_1, x_2, x_3, x_4, x_5) = \neg x_1 \vee x_2 \vee x_3 \vee \neg x_4 \vee x_5$.

При переходе к СкКНФ вначале на всех возможных парах соседних дизъюнктов D_i и D_j, в которых разные значения принимает переменная x_k, выполняется операция *полного склеивания*, когда в форму вводится новый, более короткий дизъюнкт, в котором отсутствует переменная x_k. Например, пара соседних дизъюнктов-конституент $K_1(x_1, x_2, x_3, x_4, x_5) = \neg x_1 \vee x_2 \vee x_3 \vee \neg x_4 \vee x_5$ и $K_2(x_1, x_2, x_3, x_4, x_5) = \neg x_1 \vee x_2 \vee \neg x_3 \vee \neg x_4 \vee x_5$, соседних по переменной x_3, при применении операции полного склеивания даст новый дизъюнкт с сокращенным набором переменных $K'(x_1, x_2, x_4, x_5) = \neg x_1 \vee x_2 \vee \neg x_4 \vee x_5$. В нем все слагаемые, кроме x_3, совпадают со слагаемыми в исходных соседних дизъюнктах K_1 и K_2. В операции полного склеивания реализуется одно из правил расщепления алгебры логики, имеющее вид: $(x \vee y)\ \&\ (x \vee \neg y) = x$.

После всех возможных операций полного склеивания в СДНФ выполняется операция поглощения, при которой в полученной формуле удаляются все исходные дизъюнкты, которые хотя бы один раз участвовали в операции полного склеивания. В рассмотренном примере будут удалены оба соседних дизъюнкта – $K_1(x_1, x_2, x_3, x_4, x_5) = \neg x_1 \vee x_2 \vee x_3 \vee \neg x_4 \vee x_5$ и $K_2(x_1, x_2, x_3, x_4, x_5) = \neg x_1 \vee x_2 \vee \neg x_3 \vee \neg x_4 \vee x_5$, а новый дизъюнкт с сокращенным набором переменных $K'(x_1, x_2, x_4, x_5) = \neg x_1 \vee x_2 \vee \neg x_4 \vee x_5$ останется. В операции поглощения реализуется одно из правил поглощения алгебры логики, имеющее вид: $x\ \&\ (x \vee y) = x$.

В случае единичных дизъюнктов применение правила полного склеивания дает в итоге 0,

то есть невыполнимость, тождественную ложность КНФ, поскольку по закону противоречия $x\&\vee\neg x = 0$.

Если после применения первых двух операций будут получены дизъюнкты, у которых переменные имеют одинаковые номера и значения их совпадают (дубликаты), то к ним применяют операцию удаления дубликатов, при которой из всех дубликатов в формуле оставляют только один. В данной операции реализуется один из законов идемпотентности алгебры логики, имеющий вид: $x \vee x = x$.

В случае появления новых пар соседних дизъюнктов в КНФ все три рассмотренные операции повторяются до тех пор, пока таких пар уже не будет. В результате получится искомая сокращенная форма исходной СКНФ – СкКНФ. По сравнению с совершенной формой у нее отсутствует избыточность внутри дизъюнктов в том смысле, что их литеры образуют простые наборы, у которых нельзя удалить ни одной литеры без потери эквивалентности вновь полученной формулы и исходной.

Модификация метода Квайна. В чистом виде метод Квайна предусматривает перебор и сравнительный анализ значительного количества пар дизъюнктов, которое растет квадратично по числу проверяемых дизъюнктов. Если число дизъюнктов одинаковой длины равно k, то число всех различных проверяемых пар равно $k(k-1)/2$. Предложено существенно сократить его за счет проверки достаточно простых необходимых условий. Исследуем их с учетом хранения дизъюнктов в объектах класса Clause.

Рассмотрим применение метода для КНФ произвольного вида, необязательно являющихся совершенными формами (СКНФ) булевых функций.

Для сокращения средних затрат операций рассмотрим дополнительные необходимые условия, выполняющиеся для пары существенных соседних дизъюнктов D_i и D_j.

1. Общее число литер одинаково. При этом для объектов, соответствующих D_i и D_j, должно выполняться условие

$$\text{nvars}(D_i) = \text{nvars}(D_j). \quad (1)$$

2. Состав переменных в D_i и D_j должен быть одинаков. В этом случае необходимо выполнение условия

$$\text{sum\_nv}(D_i) = \text{sum\_nv}(D_j). \quad (2)$$

3. Разность чисел переменных, входящих в дизъюнкты D_i и D_j с отрицанием, должна отличаться ровно на 1. При этом должно выполняться условие

$$\left|\text{sum\_ch}(D_i) - \text{sum\_ch}(D_j)\right| = 1. \quad (3)$$

Условия (1)–(3) ни по одному, ни все одновременно не дают достаточного условия соседства рассматриваемых в них дизъюнктов D_i и D_j. Поэтому при выполнении необходимых условий точную проверку соседства с определением контрарной переменной необходимо осуществлять по характеристическим векторам D_i и D_j.

В случае соседства D_i и D_j по некоторой переменной x_p строится дизъюнкт D' с сокращенным набором переменных, в котором все слагаемые, кроме x_p, совпадают со слагаемыми в исходных соседних дизъюнктах D_i и D_j.

После склеивания двух дизъюнктов длины *len* новый сокращенный дизъюнкт длины *len* – 1 вносится в соответствующий список дизъюнктов КНФ. Поскольку при этом возможно появление дубликатов, перед записью сокращенных дизъюнктов в список их достаточно проверить на несовпадение только с уже внесенными в него на данной итерации новыми дизъюнктами длины *len* – 1.

Также предложено минимизировать число операций при образовании новых сокращенных дизъюнктов длины *len* – 1 и удалении образующего его дизъюнкта длины *len* за счет совмещения данных операций.

Алгоритм Mod_Quine

Рассмотрим алгоритм Mod_Quine, реализующий переход от произвольной КНФ, заданной в виде набора списков объектов класса Clause, в которых дизъюнкты сгруппированы по длинам, к эквивалентной ей КНФ, имеющей аналогичную структуру, но не содержащей дизъюнктов с одинаковыми наборами переменных. Он построен на основе обычного алгоритма Квайна с учетом дополнительных свойств соседних наборов (1)–(3), а также совмещения образования нового сокращенного дизъюнкта длины *len* – 1 с удалением образующего его дизъюнкта длины *len*.

Входная структура данных: совокупность lists однонаправленных списков типа ClauseList, состоящих из узлов, имеющих структуру Clause, и описывающих дизъюнкты одинаковой длины. Она задает КНФ произвольного вида некоторой булевой функции.

Выходная структура данных: функция возвращает логическое значение false, если в процессе сокращения выявилась ложность исходной КНФ, и true при возможной выполнимости ее сокращенной формы (она не пуста);

в случае возможной выполнимости КНФ ее результирующая модификация по-прежнему содержится в структуре lists.

Решаемая задача: переход от произвольной КНФ булевой функции, возможно, содержащей дизъюнкты с соседними наборами переменных, к ее СкКНФ, не содержащей их, в том случае, если выведен пустой дизъюнкт, возврат ложности исходной КНФ.

Вспомогательные структуры данных: массив neib_literals[$k - i - 1$] номеров переменных, по которым текущий дизъюнкт имеет соседство с остальными; neib – общее число дизъюнктов, соседних с текущим; вспомогательный список shorted_list дизъюнктов длины $len - 1$.

Алгоритм Mod_Quine, реализущий переход от произвольной КНФ к ее СкКНФ:

```
bool mod_Quine(int n, lists &lists) {
// Итерации по всем спискам объектов класса clause
  for (length = n - 1; length > 0; --length) {
    lst = lists[length]; // lst – текущий список
    if (!lst.empty()) {
      shorted_list = {}; // Создание пустого списка укороченных дизъюнктов
    // Итерации по всем дизъюнктам it из lst, которые сравниваем со всеми последующими
      it = lst.begin;// создание указателя it на текущий проверяемый дизъюнкт
      while (it != lst.end) {// проверка окончания текущего списка
        neib_literals<int> = {}; // создание пустого списка соседних дизъюнктов
        // Проход по всем последующим дизъюнктам jt
        for (jt = it->next(); jt != lst.end(); jt=jt->next()) {
        // 1. проверка условий соседства, определение номера контрарной переменной neib
        delta = it->negatives - jt->negatives;
          if ((delta*delta == 1) && (it->var_sum == jt->var_sum)
                && ((neib = neighbours(*it, *jt)) != -1)) {
            neib_literals.add(neib); // сохранение номера литеры в массив neib
            jt->glued = true; // внесение отметки о том, что дизъюнкт jt – соседний
          }
        }
        if (neib_literals.empty()) {// проверка отсутствия соседних дизъюнктов
          if (it->glued)erase(it); // если it участвовал ранее в склейке, то удаляется
        }
        else {// у it есть соседние дизъюнкты – перебор всех соседств
          for (unsigned int kk = 0; kk < neib_literals.size() - 1; ++kk) {
            shorted = it; // создание нового дизъюнкта shorted
            shorted.remove_literal(neib_literals[kk]); // удаление из него литеры
            if (it.vars == 1) return false; // получен пустой дизъюнкт
    // вставка shorted в shorted_list с проверкой уникальности shorted
    shorted_list.insert_unique(shorted);
          }
          // Обработка дизъюнкта it
          it->remove_literal(neib_literals[neib_literals.size() - 1]);
          if (it.vars == 1) return false;
            append_unique(shorted_list, lst, it);
          }
        } // окончание перебора последующих дизъюнктов
// добавление shorted_list к списку с длиной дизъюнктов на 1 меньше текущей splice_append(lists[length-1], shorted_list);
      } // окончание перебора дизъюнктов в текущем списке lst
    } // окончание перебора списков
    return true;
}
```

В алгоритме для сокращения числа всех выполняемых операций существенно используются возможности введенной выше комплексной структуры данных.

Заключение

Для повышения вычислительной эффективности алгоритмов обработки КНФ предложено применять специальную комбинированную структуру данных, основу которой составляет упорядоченный массив из списков существенных дизъюнктов одинаковой длины.

В базовый класс Clause входят основные и вспомогательные структуры для представления отдельных дизъюнктов КНФ, а также типовые методы для их обработки. В производном классе ClauseList представлены списки дизъюнктов одинаковой длины, ссылки на которые в порядке возрастания длин вносятся в специальный массив указателей. Также класс ClauseList содержит методы, предназначенные для обработки таких составных списочных структур дизъюнктов.

Главная цель данной структуры заключается в сокращении трудоемкости базовых операций с КНФ за счет замены операций с массивами на операции со списками и их элементами, которые выполняются путем изменения указателей, исключая перемещения самих структур в памяти вычислительного устройства. Также перебор сокращается благодаря упорядоченности дизъюнктов по длинам.

На основе такой комплексной структуры данных предложена модификация Mod_Quine алгоритма Квайна–МакКласки, которая в полной мере позволяет использовать преимуще-

ства такого представления КНФ за счет устранения перезаписи информации, значительного сокращения количества полных сравнений дизъюнктов. Практически алгоритм реализован на языке программирования C++. Его использование позволит ускорить решение задачи эквивалентной минимизации формул КНФ, применить эти методы к формулам повышенного размера, имеющим десятки и сотни переменных и дизъюнктов.

Литература

1. Larrabee T. Test pattern generation using Boolean satisfiability. IEEE Transactions on Computer-Aided Design, 1992, vol. 11, no. 1, pp. 4–15. DOI: 10.1109/43.108614.

2. Stephan P.R., Brayton R.K., Sangiovanni-Vincentelli A.L. Combinational test generation using satisfiability. IEEE Transactions on Computer-Aided Design, 1996, pp. 1167–1176. DOI: 10.1109/43.536723.

3. Goldberg E.I., Prasad M.R., and Brayton R.K. Using SAT for combinational equivalence checkin. Proc. IEEE/ACM Design, Automation and Test in Europe, 2001, pp. 114–121. DOI: 10.1109/DATE.2001.915010.

4. Marques-Silva J., Glass T. Combinational equivalence checking using satisfiability and recursive learning. Proc. Conf. Design, Automation and Test in Europe, 1999, pp. 145–149. DOI: 10.1145/307418.307477.

5. Sheeran M., Singh S., Stalmark G. Checking safety properties using induction and a SAT-solver. Proc. 3rd FMCAD2000, Springer-Verlag Berlin Heidelberg, 2000, pp. 108–125. DOI: 10.1007/3-540-40922-X_8.

6. Velev M.N., Bryant R.E. Effective use of Boolean satisfiability procedures in the formal verification of superscalar and VLIW microprocessors. Proc. DAC, 2001, pp. 226–231. DOI: 10.1145/378239.378469.

7. Biere A., Cimatti A., Edmund C., Zhu Y. Symbolic model checking without BDDs. Proc. TACAS, 1999, pp. 193–207. DOI: 10.1007/3-540-49059-0_14.

8. McMillan K.L. Applying SAT methods in unbounded symbolic model checking. Proc. 14th CAV2002, Springer-Verlag Berlin Heidelberg, 2002, pp. 250–264.

9. Gupta A., Yang Z., Ashar P., Gupta A. SAT-based image computation with application in reachability analysis. Proc. 3rd Int. Conf. FMCAD, 2000, pp. 391–408. DOI: 10.1007/3-540-40922-X_22.

10. Parosh Aziz Abdulla, Per Bjesse, Niklas Een. Symbolic reachability analysis based on SATsolvers. Proc. 6th Int. Conf. TACAS, 2000, pp. 411–425. DOI: 10.1007/3-540-46419-0_28.

11. Marques-Silva J.P., Sakallah K.A. Efficient and robust test generation-based timing analysis. Proc. IEEE ISCAS'94, 1994. DOI: 10.1109/ISCAS.1994.408815.

12. Nam G., Sakallah K.A., Rutenbar R.A. Satisfiability-based layout revisited: Routing complex FPGAs via search-based Boolean SAT. Proc. Int. Symp. FPGAs, 1999, pp. 167–175. DOI: 10.1145/296399.296450.

13. Люгер Д.Ф. Искусственный интеллект. Стратегии и методы решения сложных проблем; [пер. с англ.]. М.: Вильямс, 2003. 864 с.

14. Гданский Н.И. Прикладная дискретная математика. Логика. Графы. Автоматы. Алгоритмы. Кодирование. М.: Вузовская книга, 2011. 508 с.

15. Kim J., Marques-Silva J.P., Savoj H., Sakallah K.A. RID-GRASP: Redundancy identification and removal using GRASP. Proc. IEEE/ACM Int. Workshop on Logic Synthesis, 1997, pp. 1–4.

16. Lynce I., Marques-Silva J.P., Computadores S. Efficient data structures for fast SAT solvers. Proc. 5th Int. Sympos. SAT, 2001.

17. Bayardo R.J., Schrag R.C. Using csp look-back techniques to solve real-world sat instances. Proc. 14th Conf. AAAI/IAAI, 1997, pp. 203–208.

18. Zhang H. SATO: An efficient propositional prover. Proc. 14th Int. CADE, 1997, pp. 272–275. DOI: 10.1007/3-540-63104-6_28.

Software & Systems
DOI: 10.15827/0236-235X.131.396-403

Received 05.03.20
2020, vol. 33, no. 3, pp. 396–403

Data structures and the Quine–McCluskey method modification for minimizing normal forms

N.I. Gdanskiy [1], *Dr.Sc. (Engineering), Professor, al-kpp @ mail.ru*
A.A. Denisov [1], *Postgraduate Student, aadenisov88@gmail.com*
N.L. Kulikova [2], *Ph.D. (Engineering), Associate Professor, kulikovanl@mpei.ru*

[1] *K.G. Razumovsky Moscow State University of Technology and Management (First Cossack University), Moscow, 109004, Russian Federation*
[2] *National Research University "Moscow Power Engineering Institute", Moscow, 111250, Russian Federation*

Abstract. Logical methods of analysis and synthesis of systems of various nature are usually based on the use of descriptions of their structures and processes in them in the form of Boolean functions, which are equivalently reduced to conjunctive normal forms to unify the representation. For initial systems and processes, as a rule, the main criterion for optimality is the minimum number of components that make up their components, which simplifies the structure, reduces cost and increases reliability, so for model conjunctive normal forms, the problem of minimizing them is of great practical importance.

The algorithm's efficiency that processes complex objects (including conjunctive normal forms) significantly depends on the main and auxiliary data structures used to represent these objects. Therefore, based on the analysis of existing structures, a new complex three-level data structure for representing conjunctive normal forms has been developed.

The lower level of the entire combined structure is the data structure for a single clause. Lists of clauses with the same number of letters form the middle level. An array of lists ordered by the lengths of their clauses sets the entire conjunctive normal form at the top level.

The using lists, pointers to them and single elements, and the ordering of clauses by lengths make it possible to radically reduce operations for rewriting information and ordering it in the process of converting conjunctive normal forms.

Based on the developed complex data structure, the authors developed a modification of the well–known Quine-McCluskey method used to reduce perfect normal forms.

The combined use of the proposed data structure and the modified method makes it possible to significantly reduce the total number of operations while minimizing conjunctive normal forms compared to the basic version of the Quine–McCluskey method.

This is achieved by reducing data re-processing, and by using special logical conditions, an additional reduction is achieved in the total number of checks of letters in clauses when comparing them.

Keywords: clause, conjunctive normal form, data structures, Quine-McCluskey method.

References

1. Larrabee T. Test pattern generation using Boolean satisfiability. *IEEE Transactions on Computer-Aided Design*, 1992, vol. 11, no. 1, pp. 4–15. DOI: 10.1109/43.108614.

2. Stephan P.R., Brayton R.K., Sangiovanni-Vincentelli A.L. Combinational test generation using satisfiability. *IEEE Transactions on Computer-Aided Design*, 1996, pp. 1167–1176. DOI: 10.1109/43.536723.

3. Goldberg E.I., Prasad M.R., and Brayton R.K. Using SAT for combinational equivalence checkin. *Proc. IEEE/ACM Design, Automation and Test in Europe*, 2001, pp. 114–121. DOI: 10.1109/DATE.2001.915010.

4. Marques-Silva J., Glass T. Combinational equivalence checking using satisfiability and recursive learning. *Proc. Conf. Design, Automation and Test in Europe*, 1999, pp. 145–149. DOI: 10.1145/307418.307477.

5. Sheeran M., Singh S., Stalmark G. Checking safety properties using induction and a SAT-solver. *Proc. 3rd FMCAD2000*, Springer-Verlag Berlin Heidelberg, 2000, pp. 108–125. DOI: 10.1007/3-540-40922-X_8.

6. Velev M.N., Bryant R.E. Effective use of Boolean satisfiability procedures in the formal verification of superscalar and VLIW microprocessors. *Proc. DAC*, 2001, pp. 226–231. DOI: 10.1145/378239.378469.

7. Biere A., Cimatti A., Edmund C., Zhu Y. Symbolic model checking without BDDs. *Proc. TACAS*, 1999, pp. 193–207. DOI: 10.1007/3-540-49059-0_14.

8. McMillan K.L. Applying SAT methods in unbounded symbolic model checking. *Proc. 14th CAV2002*, Springer-Verlag Berlin Heidelberg, 2002, pp. 250–264.

9. Gupta A., Yang Z., Ashar P., Gupta A. SAT-based image computation with application in reachability analysis. *Proc. 3rd Int. Conf. FMCAD*, 2000, pp. 391–408. DOI: 10.1007/3-540-40922-X_22.

10. Parosh Aziz Abdulla, Per Bjesse, Niklas Een. Symbolic reachability analysis based on SATsolvers. *Proc. 6th Int. Conf. TACAS*, 2000, pp. 411–425. DOI: 10.1007/3-540-46419-0_28.

11. Marques-Silva J.P., Sakallah K.A. Efficient and robust test generation-based timing analysis. *Proc. IEEE ISCAS'94*, 1994. DOI: 10.1109/ISCAS.1994.408815.

12. Nam G., Sakallah K.A., Rutenbar R.A. Satisfiability-based layout revisited: Routing complex FPGAs via search-based Boolean SAT. *Proc. Int. Symp. FPGAs*, 1999, pp. 167–175. DOI: 10.1145/296399.296450.

13. Luger G. *Artificial intelligence. Structures and Strategies for Complex Problem Solving*. Addison Wesley, 2003, 754 p. (Russ. ed.: Moscow, 2003, 864 p.).

14. Gdansky N.I. *Applied Discrete Mathematics. Logics. Counts. Automata. Algorithms. Coding.* Moscow, 2011, 580 p. (in Russ.).

15. Kim J., Marques-Silva J.P., Savoj H., Sakallah K.A. RID-GRASP: Redundancy identification and removal using GRASP. *Proc. IEEE/ACM Int. Workshop on Logic Synthesis*, 1997, pp. 1–4.

16. Lynce I., Marques-Silva J.P., Computadores S. Efficient data structures for fast SAT solvers. *Proc. 5th Int. Sympos. SAT*, 2001.

17. Bayardo R.J., Schrag R.C. Using csp look-back techniques to solve real-world sat instances. *Proc. 14th Conf. AAAI/IAAI*, 1997, pp. 203–208.

18. Zhang H. SATO: An efficient propositional prover. *Proc. 14th Int. CADE*, 1997, pp. 272–275. DOI: 10.1007/3-540-63104-6_28.

Для цитирования

Гданский Н.И., Денисов А.А., Куликова Н.Л. Структуры данных и модификация метода Квайна–МакКласки при минимизации нормальных форм // Программные продукты и системы. 2020. Т. 33. № 3. С. 396–403. DOI: 10.15827/0236-235X.131.396-403.

For citation

Gdanskiy N.I., Denisov A.A., Kulikova N.L. Data structures and the Quine–McCluskey method modification for minimizing normal forms. *Software & Systems*, 2020, vol. 33, no. 3, pp. 396–403 (in Russ.). DOI: 10.15827/0236-235X.131.396-403.