Об алгоритмах проверки правильности семейств функций

Д. О. Рыков

В работе изучается вопрос проверки правильности семейств функций. Построен алгоритм проверки правильности, учитывающий информацию о графе семейства. Сложность приведенного алгоритма зависит от порядков сильных компонент графа. Построен алгоритм проверки правильности семейств монотонных функций. В основе алгоритма лежат два свойства правильных семейств монотонных функций: наличие константы и функциональная замкнутость класса монотонных функций в P_2. Сложность алгоритма значительно меньше сложности стандартных алгоритмов проверки правильности.

Ключевые слова: правильное семейство функций, алгоритм проверки правильности семейств функций, граф существенной зависимости семейства функций, семейство монотонных функций, критерий правильности семейства функций.

1. Введение

Данная статья продолжает серию работ, посвященных изучению правильных семейств булевых функций, которые применяются для функционального задания латинских квадратов. Одной из центральных задач, стоящих перед исследователем в этой области, является задача проверки семейства на правильность. Установлено, что эта задача относится к числу NP-полных задач. Тем не менее, необходимость в эффективных алгоритмах проверки правильности остается. Известно несколько способов проверки семейства булевых функций на правильность. Первый способ — проверять на наличие этого свойства по определению правильного семейства. Второй способ опирается на критерий правильности, сформулированный в работе [1]. Оба

эти способа проверки нельзя назвать эффективными. Однако, если ввести некоторые ограничения на семейство функций, задача проверки правильности может значительно упроститься. Так, например, для семейств функций определенных классов (линейных, мультиаффинных) были установлены критерии правильности, в которых используется цикловая структура графов существенной зависимости. Алгоритмы проверки, основанные на этих критериях, работают значительно быстрее. В данной работе на основе предложенного в статье [1] критерия правильности будет построен модифицированный критерий, в котором также будет учтена информация о структуре графа существенной зависимости, что приведет к уменьшению требуемого на проверку времени в некоторых случаях. После этого будет описан алгоритм проверки правильности, использующий эту идею. Также будет построен алгоритм проверки правильности для семейств монотонных функций и оценена его сложность.

2. Правильные семейства булевых функций и графы существенной зависимости

Напомним основные понятия.

Определение 1. Семейство булевых функций $f = (f_1, f_2, \ldots, f_n)$ от переменных $x_1, x_2, \ldots, x_n$ называется правильным, если для любых двух различных наборов значений переменных $x' = (x'_1, x'_2, \ldots, x'_n)$ и $x'' = (x''_1, x''_2, \ldots, x''_n)$ существует $\alpha \in \overline{1,n}$ такое, что выполнено

$$x'_\alpha \neq x''_\alpha, \quad f_\alpha(x') = f_\alpha(x'').$$

Важную роль при изучении правильных семейств играют графы существенной зависимости этих семейств.

Определение 2. Графом существенной зависимости семейства функций $f = (f_1, \ldots, f_n)$ от переменных $x_1, \ldots, x_n$ называется ориентированный граф $G_f = (V, E)$, где $V = \{1, 2, \ldots, n\}$, а $E = \{(i, j) : f_j$ существенно зависит от $x_i\}$.

По структуре графа существенной зависимости семейства функций (в дальнейшем для краткости будем называть его графом семейства) можно делать определенные выводы о правильности семейства.

Так, например, если граф семейства не содержит циклов, то это семейство правильно. Обратное неверно.

3. Критерии правильности

В работе [1] доказан критерий правильности через вид полиномов Жегалкина всевозможных произведений функций из семейства (в рамках данной работы будем называть его Общим критерием правильности).

Теорема 1 (Общий критерий правильности [1]). *Семейство булевых функций $f = (f_i)$, $i \in [1, n]$ правильно тогда и только тогда, когда для любого подмножества I, $I \subseteq [1, n]$, произведение функций $\prod\limits_{i \in I} f_i$ не зависит существенно от множества переменных $x_I = (x_i)$, $i \in I$.*

Смысл этого утверждения такой: чтобы проверить семейство функций на правильность, надо рассмотреть все возможные 2^n произведений функций из семейства и следить за коэффициентами при членах, содержащих соответствующие произведения переменных в полиноме Жегалкина. Если хотя бы один из коэффициентов будет равен 1, семейство не будет правильным. Если все коэффициенты равны 0, то семейство правильно.

Зададимся вопросом: обязательно ли для проверки правильности расчитывать все возможные 2^n произведений функций из семейства или этот процесс можно оптимизировать? Как уже было сказано знание графа семействапозволяет делать о нем некоторые выводы. Предположим теперь, что нам известно не только семейство, но и его граф. Для дальнейших рассуждений необходимо ввести дополнительные определения.

Определение 3. Ориетированный граф называется сильно связным, или сильным, если любые две его вершины взаимно достижимы.

Определение 4. Сильной компонентой графа называется максимальный сильный подграф.

Определение 5. Конденсацией (факторграфом, графом сильных компонент) графа G называется граф G^*, множеством вершин которого служит множество $\{S_1, S_2, \ldots, S_m\}$ всех сильных компонент

графа G, а дуга идет из S_i к S_j, если в графе G имеется по крайней мере одна дуга, идущая из некоторой вершины компоненты S_i к вершине компоненты S_j.

Определение 6. Для любого подмножества вершин S графа G порожденным этим множеством подграфом (вершинным подграфом) $< S >$ называется максимальный подграф графа G, множеством вершин которого является S.

Теперь можно сформулировать новый критерий правильности. Он опирается на общий критерий, но в нем используется информация о графе семейства.

Теорема 2 (Критерий правильности). *Семейство булевых функций $f = (f_i)$, $i \in [1, n]$ правильно тогда и только тогда, когда для любого подмножества I, $I \subseteq [1, n]$, такого, что вершинный подграф $< I >$ — сильный, произведение функций $\prod\limits_{i \in I} f_i$ не зависит существенно от множества переменных $x_I = (x_i)$, $i \in I$.*

Для доказательства теоремы мы будем пользоваться следующими двумя утверждениями из теории графов.

Лемма 1 ([5]). *Конденсация G^* любого графа G не содержит ориентированных циклов.*

Лемма 2 ([5]). *В любом ацикличном графе (графе без циклов) есть вершина, которая не является началом (концом) ни одной из дуг графа.*

Доказательство теоремы. Необходимость. Следует из общего критерия правильности.

Достаточность. Предположим, что $\exists I$, $I \subseteq [1, n]$, такое, что $\prod\limits_{i \in I} f_i$ зависит существенно от $x_I = (x_i)$, $i \in I$. Пусть в $< I >$ ровно k сильных компонент $I_1, I_2, \ldots, I_k$. Достаточно показать, что среди $I_1, \ldots, I_k$ существует компонента J такая, что $\prod\limits_{i \in J} f_i$ зависит существенно от множества переменных $x_J = (x_i)$, $i \in J$. Если $k = 1$, то $J = I_1 = I$. Пусть $k > 1$. Из леммы следует, что из вершин одной из сильных компонент не идет ни одной дуги в вершины других компонент. Для

определенности будем считать, что эта компонента — I_1. Тогда все функции, соответствующие вершинам из компонент $I_2, \ldots, I_k$, не зависят существенно от переменных x_i, $i \in I_1$. Значит, и их произведение не зависит от переменных x_i, $i \in I_1$. Так как в канонической полиномиальной записи разложения произведения $\prod\limits_{i\in I} f_i$ по переменным x_i, $i \in I$, коэффициент у произведения $\prod\limits_{i\in I} x_i$ не равен тождественно нулю, коэффициент при $\prod\limits_{i\in I_1} x_i$ у произведения $\prod\limits_{i\in I_1} f_i$ также не равен нулю. Теорема доказана.

Таким образом, для проверки правильности достаточно рассматривать произведения функций, образующих сильные графы. Следующая теорема является более узким утверждением, но, тем не менее, полезным. Для краткости формулировки этой теоремы обобщим понятие правильного семейства следующим образом.

Определение 7. Семейство булевых функций $f_I = (f_i)$, $i \in I$, от переменных $x_J = (x_j)$, $j \in J$, $I \subseteq J$, будем называть правильным, если при любых значениях вектора $x^0_{J\setminus I} = (x^0_k)$, $k \in J\setminus I$, семейство $\tilde{f}_I = (\tilde{f}_i)$, $i \in I$, где $\tilde{f} = f|_{x_{J\setminus I}=x^0_{J\setminus I}}$, правильно в смысле первоначального определения.

Пусть I — некоторое множество индексов, где $I \subset [1, n]$ и $\varepsilon_I = (\varepsilon_\alpha)$, $\alpha \in I$, $\varepsilon_\alpha \in \{0, 1\}$ — семейство констант с индексами из множества I. Определим семейство функций $f^0_{CI} = (f^0_i)$, $i \in CI$, где CI — дополнение множества I в $[1, n]$, полагая для любого $\lambda \in CI$

$$f^0_\lambda(x) = f_\lambda(x)|_{x_\alpha=\varepsilon_\alpha}, \quad \alpha \in I.$$

Другими словами, f^0_λ — это функции семейства f с индексами из множества CI, в которых переменные с индексами из I замещены константами семейства ε_I.

Лемма 3 ([1]). *Семейство булевых функций f правильно тогда и только тогда, когда для любого множества $I \subset [1, n]$ и любого семейства констант ε_I семейство f^0_{CI} правильно.*

Проще говоря, если семейство булевых функций правильно, то при фиксации любого набора переменных константами получается правильное семейство функций.

Следствие 1 (из Леммы 3). *Если семейство $f = (f_i)$, $i \in I$, правильно в некотором смысле, то для любого подмножества $K \subset I$ семейство $\hat{f} = (f_k)$, $k \in K$, правильно в смысле нового определения.*

Теорема 3 (Критерий правильности). *Пусть $f = (f_i)$, $i \in [1, n]$, — семейство булевых функций, конденсация G_f^* графа G_f состоит из k вершин $I_1, I_2, \ldots, I_k$. Тогда f правильно тогда и только тогда, когда правильны семейства $f_{I_j} = (f_i)$, $i \in I_j$, $j = 1, \ldots, k$.*

Доказательство. Необходимость. Если f правильно, то по следствию из леммы правильны также семейства $f_{I_j} = (f_i)$, $i \in I_j$, $j = 1, \ldots, k$.

Достаточность. Предположим, что семейство f не является правильным. Тогда по модифицированному критерию правильности $\exists I \subseteq [1, n]$: $< I >$ — сильный граф и $\prod\limits_{i \in I} f_i$ зависит существенно от множества переменных $x_I = (x_i)$, $i \in I$. Значит, I — подмножество некоторой сильной компоненты K и, следовательно, семейство $\hat{f}_I = (f_i)$, $i \in I$, правильно. Значит, в частности, $\prod\limits_{i \in I} f_i$ не зависит существенно от множества переменных $x_I = (x_i)$, $i \in I$. Противоречие.

Теорема доказана.

Итак, проверка правильности семейства функций сводится к проверке правильности в смысле нового определения на сильных компонентах графа этого семейства. Следующее утверждение, являющееся на самом деле переформулировкой последней теоремы позволяет понять, за счет чего можно уменьшить сложность проверки.

Теорема 4. *Пусть $f = (f_i)$, $i \in [1, n]$, — семейство булевых функций, конденсация G_f^* графа G_f состоит из k вершин $I_1, I_2, \ldots, I_k$, занумерованных таким образом, что дуги могут идти только от вершин с меньшим индексом к вершинам с большим индексом. Тогда f правильно тогда и только тогда, когда одновременно выполняются следующие условия:*

- *Семейство $f_{I_1} = (f_i)$, $i \in I_1$ правильно.*
- *Семейства $f_{I_t}^0 = (f_i^0)$, $i \in I_t$, правильны при любой фиксации переменных $x_S = (x_i)$, $i \in S = I_1 \bigsqcup I_2 \bigsqcup \cdots \bigsqcup I_{t-1}$.*

Проще говоря, семейство $f = (f_i)$, $i \in [1, n]$, правильно тогда и только тогда когда правильны семейства функций сильных компонент графа существенной зависимости при фиксации соответствующих переменных произвольными константами.

Замечание. В этой теореме учитывается, что фиксировать необходимо не все переменные, а только те, от которых функции компонент могут зависеть существенно.

Теперь можно описать алгоритм проверки правильности, использующий последнюю теорему.

4. Алгоритм проверки правильности с использованием информации о графе

Алгоритм проверки правильности путем перебора. Для начала опишем алгоритм, основанный на определении правильного семейства. Чтобы проверить семейство размера n на правильность необходимо сравнить значения аргументов и функций на всех парах булевских наборов значений переменных. Если на некотором шаге будет найдена пара наборов, на которой свойство правильности нарушается, алгоритм останавливается с ответом «семейство не является правильным». Если же на всех парах наборов свойство правильности будет выполнено, то алгоритм останавливается с ответом «семейство правильно».

Число различных пар булевских наборов длины n равно $\frac{2^n(2^n-1)}{2} = O(2^{2n})$. А так как длина каждого набора равна n, то всего для проверки правильности надо будет сделать $f(n) = O(n2^{2n})$ операций.

Модифицированный алгоритм проверки правильности. Теперь опишем алгоритм с использованием информации о графе семейства. Пусть нам известно семейство $f = (f_1, \ldots, f_n)$ из n функций и его граф. Предположим, что он содержит k сильных компонент $S_1, \ldots, S_k$, состоящих из $l_1, \ldots, l_k$ вершин соответственно. Будем предполагать, что они занумерованы в таком порядке, что в конденсации G^* графа существенной зависимости G дуги могут идти только от сильных компонент с меньшим номером к сильным компонентам с большим номером.

Шаг 1.. Проверяем на правильность семейство $f_{S_1} = (f_{s_1})$, $s_1 \in S_1$. Функции сильной компоненты S_1 могут зависеть только от переменных из компоненты S_1. Поэтому для семейства функций сильной компоненты S_1 проверка правильности сводится к перебору по парам наборов длины l_1(перебор такой же как и в первоначальном алгоритме, только для другого размера данных). В случае, если семейство f_{S_1} оказалось правильным, переходим к шагу 2. Если же оно не оказалось правильным, то и семейство $f = (f_1, \ldots, f_n)$ не является правильным.

Шаг 2.. Проверяем на правильность семейство $f_{S_2} = (f_{s_2})$, $s_2 \in S_2$. Функции сильной компоненты S_2 могут зависеть только от переменных сильных компонент S_1 и S_2. Поэтому для семейства функций сильной компоненты S_2 проверка правильности сводится к перебору по парам наборов длины $l_1 + l_2$, в которых первые l_1 значений совпадают. В случае, если семейство f_{S_2} оказалось правильным, переходим к шагу 3. Если же оно не оказалось правильным, то и семейство $f = (f_1, \ldots, f_n)$ не является правильным.

Шаг r $(r = 3, \ldots, k-1)$**.** Проверяем на правильность семейство $f_{S_r} = (f_{s_r})$, $s_r \in S_r$. Функции сильной компоненты S_r могут зависет только от переменных сильных компонент $S_1, \ldots, S_r$. Поэтому для семейства функций сильной компоненты S_r проверка правильности сводится к перебору по парам наборов длины $l_1 + l_2 + \cdots + l_r$, в которых первые $l_1 + \cdots + l_{r-1}$ значений совпадают. В случае, если семейство f_{S_r} оказалось правильным, переходим к шагу $r + 1$. Если же оно не оказалось правильным, то и семейство $f = (f_1, \ldots, f_n)$ не является правильным.

Шаг k**.** Проверяем на правильность семейство $f_{S_k} = (f_{s_k})$, $s_k \in S_k$. Функции сильной компоненты S_k могут зависетm от любых переменных. Поэтому для семейства функций сильной компоненты S_r проверка правильности сводится к перебору по парам наборов длины $l_1 + l_2 + \cdots + l_k = n$, в которых первые $l_1 + \cdots + l_{k-1}$ значений совпадают. В случае, если семейство f_{S_k} оказалось правильным, семейство f также правильно. Если же оно не оказалось правильным, то и семейство $f = (f_1, \ldots, f_n)$ не является правильным.

Оценим сложность этого алгоритма. На r-ом шаге необходимо сделать $O(l_r 2^{l_1+l_2+\cdots+l_{r-1}+2l_r})$ действий. Значит, сложность алгоритма равна $f(n) = O(\sum\limits_{t=1}^{k} l_t 2^{\sum\limits_{p=1}^{t}(l_p+l_t)})$, где $l_1 + \cdots + l_k = n$.

Замечание. В общем случае новый алгоритм работает с той же скоростью, что и первоначальный (если граф семейства сильный, то улучшений нет). Однако, если граф семейства не является сильным, с помощью этого алгоритма можно добиться значительного уменьшения времени проверки.

Итак, мы предъявили алгоритм проверки правильности на тот случай, когда нам известны сильные компоненты графа семейства (то есть, какие вершины в каких компонентах содержатся) и граф сильных компонент. Однако, обычно нам эти характеристики неизвестны. Поэтому возникает необходимость в быстрых алгоритмах построения графа семейства (по формулам, задающим функции семейства), нахождения его сильных компонент и упорядочивания сильных компонент в соответствии с теоремой 4.

Построение графа семейства.

Графы можно задавать двумя способами: с помощью матриц смежности или с помощью списков смежности. В данном случае будем пользоваться списками смежности, поскольку такая структура данных требует меньше памяти.

Будем также предполагать, что функции семейства представлены в виде полиномов Жегалкина. В случае такого представления функций легко делать выводы о существенной зависимости функции от переменной.

Опишем способ построения списков смежности.

Каждой вершине графа i сопоставим 2 массива, в которые будем последовательно записывать соответственно вершины смежные из i (массив Out[i]) и вершины, смежные к i (массив In[i]). Запись в эти массивы будет происходить следующим образом. В начале списки пусты. Рассмотрим функцию f_1. Последовательно рассматриваем все коэфициенты полинома Жегалкина этой функции. Если коэфициент полинома Жегалкина равен 0, переходим к следующему коэфициенту. Если коэфициент равен 1 (предположим, что это коэфициент при $x_{i_1} \dots x_{i_k}$), то производим следующие действия: записываем в массив

In[1] вершины $i_1, \ldots, i_k$, а в массивы $Out[i_1], \ldots, Out[i_k]$ записываем вершину 1. После этого переходим к следующему коэфициенту. По аналогии продолжаем эту процедуру для функций $f_2, \ldots, f_n$. В итоге будут построены оба списка смежности для каждой вершины. Алгоритм построения списков смежности потребует не более, чем $O(n^2 2^n)$ операций.

Замечание. Мы предполагали, что функции семейства заданы нам в виде полиномов Жегалкина. Существует алгоритм перевода функции из табличного представления в представление в виде полинома Жегалкина (или в обратную сторону) за $O(n2^n)$ операций. С помощью этого алгоритма мы могли бы перевести все функции семейства из табличного представления в полиномы Жегалкина за $O(n^2 2^n)$ действий. Таким образом, асимптотическая сложность алгоритма построения списков смежности не меняется при переходе от представления в виде полинома к табличному представлению.

Алгоритм Косарайю нахождения сильных компонент и топологическая сортировка

Для нахождения сильных компонент орграфа можно пользоваться известными алгоритмами Косарайю, Тарьяна или Габова. Все эти алгоритмы находят сильные компоненты графа $G = (V, E)$ за время $\Theta(|V| + |E|)$. Опишем один из этих алгоритмов.

В основе алгоритма Косарайю лежит метод поиска в глубину. Чтобы найти сильные компоненты заданного графа, сначала выполняется поиск в глубину для обратного графа (то есть графа, получающегося из исходного путем инвертирования его ребер). Затем выполняется поиск в глубину на исходном графе, причем вершины берутся в порядке, обратном тому, который получился посредством нумерации вершин при обратном проходе при первом запуске поиска в глубину. В итоге получится лес, деревья которого представляют сильные компоненты графа. Сложность алгоритма Косарайю складывается из сложности алгоритмов поиска в глубину, который используется дважды, и нахождения обратного графа. Поскольку оба эти алгоритма имеют линейную сложность, алгоритм Косарайю работает за линейное время. Немного модифицировав этот алгоритм, можно не только найти все сильные компоненты графа, но и определить все дуги в графе его компонент. Таким образом, мы будем знать и граф компонент исходного графа.

Теперь необходимо упорядочить вершины графа компонент в необходимом порядке. Для этого можно использовать алгоритм топологической сортировки. Этот алгоритм также основывается на алгоритме поиска в глубину. Нумерация при обратном обходе при поиске в глубину обратна нумерации топологической сортировки. Таким образом, необходимая нумерация вершин графа компонент достигается также за линейное время.

Итак, мы установили, что суммарная сложность построения графа семейства по этому семейству составляет $O(n^2 2^n)$. Также мы установили, что за линейное время по заданному графу можно найти все сильные компоненты и граф сильных компонент, в котором все вершины занумерованы таким образом, что дуги могут идти только от вершин с меньшим номером к вершинам с большим номером. Все это оправдывает усилия по модификации алгоритма проверки правильности за счет использования информации о графе семейства.

Замечание. Более подробно об алгоритмах поиска в глубину, Косарайю и топологической сортировки можно прочитать в работах [7, 8, 9].

5. Толерантность

Определение 8. Назовем функцию f_n толерантной по отношению к семейству функций $(f_1, \ldots, f_{n-1})$ (все функции зависят от переменных $x_1, \ldots, x_n$), если для любых двух наборов α^1 и α^2 таких, что $\alpha^1_n \neq \alpha^2_n$, существует $j \in [1, n]$ такое, что $\alpha^1_j \neq \alpha^2_j$, а $f_j(\alpha^1) = f_j(\alpha^2)$.

Замечание. В частности, функция f_n, толерантная семейству $(f_1, \ldots, f_{n-1})$ не зависит существенно от переменной x_n.

Пусть $f = (f_1, \ldots, f_n)$ — семейство булевых функций от переменных $x_1, \ldots, x_n$. Обозначим $f' = (f'_1, \ldots, f'_{n-1})$ семейство, в котором $f'_i = f_i|_{x_n=0}$, и $f'' = (f''_1, \ldots, f''_{n-1})$ семейство, в котором $f''_i = f_i|_{x_n=1}$.

Утверждение 1. *Семейство $f = (f_1, \ldots, f_n)$ правильно тогда и только тогда, когда семейства f' и f'' правильны и функция f_n толерантна семейству $(f_1, \ldots, f_{n-1})$.*

Доказательство. Необходимость следует из леммы.

Достаточность. Рассмотрим два различных набора α^1 и α^2.

Случай 1. $\alpha^1_n = \alpha^2_n$. Пусть для определенности $\alpha^1_n = \alpha^2_n = 0$. Тогда существует $j \in [1, n-1]$ такое, что $\alpha^1_j \neq \alpha^2_j$, $f'_j(\alpha^1_1, \ldots, \alpha^1_{n-1}) = f'_j(\alpha^2_1, \ldots, \alpha^2_{n-1})$. Следовательно, $f_j(\alpha^1) = f_j(\alpha^2)$.

Случай 2. $\alpha^1_n \neq \alpha^2_n$. Так как функция f_n толерантна семейству $(f_1, \ldots, f_{n-1})$, существует $j \in [1, n]$ такое, что $\alpha^1_j \neq \alpha^2_j$, $f_j(\alpha^1) = f_j(\alpha^2)$.

Значит, f — правильное семейство. Утверждение доказано.

Другими словами, проверка семейства функций f на правильность сводится к проверке на правильность двух семейств f' и f'' меньшего размера и проверке функции f_n на толерантность к семейству $(f_1, \ldots, f_{n-1})$. С другой стороны, если семейства f' и f'' от переменных $x_1, \ldots, x_{n-1}$ правильны, то семейство $f = (f_1, \ldots, f_n)$, где $f_j = \overline{x_n} f'_j \vee x_n f''_j$, правильно тогда и только тогда, когда функция f_n толерантна к семейству $(f_1, \ldots, f_{n-1})$. Таким образом, по двум правильным семействам размера $n-1$ можно строить правильное семейство размера n путем подбора толерантной функции f_n.

Замечание. Нетрудно заметить, что константа толерантна любому семейству функций.

Следствие 2. *Семейство функций $f = (f_1, \ldots, f_n)$, содержащее константу (пусть, для определенности $f_n = \mathrm{const}$) правильно тогда и только тогда, когда семейства функций f' и f'' правильны.*

6. Алгоритм проверки правильности семейства монотонных функций

Как уже было отмечено, для семейств функций некоторых классов (линейных, мультиаффинных) установлены эффективные критерии правильности. С помощью этих критериев можно было бы построить быстрые алгоритмы проверки правильности в соответствующих случаях. В данной работе мы не будем описывать эти алгоритмы, а зададимся целью построить быстрый алгоритм проверки правильности семейства монотонных функций.

Замечание. Класс монотонных функций наряду с классами линейных, самодвойственных, сохраняющих 0 и сохраняющих 1 функций

является замкнутым предполным в P_2 классом. Во многом интерес к нему проявлен в связи с этой его исключительной особенностью. Интересно, что из самодвойственных функций составить правильное семейство нельзя.

Для доказательства правильности алгоритма нам потребуется следующее утверждение.

Утверждение 2. *Правильное семейство монотонных функций* $f = (f_1, \ldots, f_n)$ *содержит константу.*

Доказательство. Так как семейство $f = (f_1, \ldots, f_n)$ правильно, то $\exists j \in [1, n] : f_j(0, \ldots, 0) = f_j(1, \ldots, 1)$. А так как f_j — монотонная функция, $f_j = \text{const}$.

Введем операцию удаления функции f_k из семейства $f = (f_i)$, $i \in I$. Под этой операцией будем понимать операцию, в результате которой получается семейство $\hat{f} = (\hat{f}_i)$, $i \in I \backslash \{k\}$, где $\hat{f}_i = f_i|_{x_k=\alpha}$, $\alpha \in \{0, 1\}$.

Заметим, что в результате удаления монотонной функции из семейства, получается семейство, опять целиком состоящее из монотонных функций (так как монотонные функции образуют замкнутый класс в P_2). Поскольку в правильном семействе монотонных функций есть константа(она толерантна любому семейству), получаем следующее следствие.

Следствие 3. *Семейство монотонных функций правильно тогда и только тогда, когда одновременно выполняются следующие условия:*

1) оно содержит константу и

2) при удалении этой константы из семейства получается 2 правильных семейства функций.

Таким образом задаче проверки правильности семейства монотонных функций размера n сведена к 2 таким же задачам для семейств размера $n - 1$. Осталось только описать алгоритм, использующий это соображение.

Алгоритм проверки правильности семейства монотонных функций.

Используя все вышеобозначенное построим алгоритм проверки правильности семейства монотонных функций и оценим его сложность.

Предположим, что нам задано семейство $f = (f_1, \ldots, f_n)$ монотонных функций, которое необходимо проверить на правильность.

Шаг 1. Случай 1. $n = 1$.

1) $f_1 = 0$ или $f_1 = 1$. Алгоритм проверки правильности завершает работу с ответом: «f — правильное семейство».

2) $f_1 \neq 0$ и $f_1 \neq 1$. Алгоритм проверки правильности завершает работу с ответом: «семейство f не является правильным».

Случай 2. $n \neq 1$. Перейти к шагу 2.

Шаг 2. Ищем среди функций семейства f константу путем проверки каждой функции семейства на равенство константам 0 и 1. Рассмотрим случаи:

Случай 1. Если константной функции не найдется, то алгоритм проверки правильности заканчивает свою работу с ответом: «семейство f не является правильным».

Случай 2. $f_k = 0$ или $f_k = 1$. Тогда

Шаг 3. Для $i \in [1, k-1]$ вычислить функции $g_i(x_1, \ldots, x_{n-1}) := f_i(x_1, \ldots, x_{k-1}, 0, x_k, \ldots, x_{n-1})$. Для $i \in [k, n-1]$ вычислить функции $g_i(x_1, \ldots, x_{n-1}) := f_{i+1}(x_1, \ldots, x_{k-1}, 0, x_k, \ldots, x_{n-1})$. Перейти к шагу 4.

Шаг 4. Рекурсивно проверить семейство $g = (g_1, \ldots, g_{n-1})$ на правильность.

1) Семейство g правильно. Перейти к шагу 5.

2) Семейство g не является правильным. Алгоритм проверки правильности завершает работу с ответом: «семейство f не является правильным».

Шаг 5. Для $i \in [1, k-1]$ вычислить функции $g_i(x_1, \ldots, x_{n-1}) := f_i(x_1, \ldots, x_{k-1}, 1, x_k, \ldots, x_{n-1})$. Для $i \in [k, n-1]$ вычислить функции $g_i(x_1, \ldots, x_{n-1}) := f_{i+1}(x_1, \ldots, x_{k-1}, 1, x_k, \ldots, x_{n-1})$. Перейти к шагу 6.

Шаг 6. Рекурсивно проверить семейство $g = (g_1, \ldots, g_{n-1})$ на правильность.

1) Семейство g правильно. Алгоритм проверки правильности завершает работу с ответом: «f — правильное семейство».

2) Семейство g не является правильным. Алгоритм проверки правильности завершает работу с ответом: «семейство f не является правильным».

Оценим теперь сложность этого рекурсивного алгоритма: Введем обозначения:

$f(n)$ — сложность проверки правильности семейства из n функций.
$g(n)$ — сложность проверки равенства функции от n переменных константе.
$h(n)$ — сложность подсчета функции семейства g по функции семейства f размера n (сложность подстановки константы вместо одной из переменных).

Нетрудно убедиться, что $g(n) = h(n) = O(2^n)$. Тогда сложность алгоритма проверки правильности равна:

$$\begin{aligned} f(n) &= ng(n) + 2(n-1)h(n) + 2f(n-1) = \\ &= \sum_{i=0}^{n} 2^i(n-i)g(n-i) + \sum_{i=0}^{n-1} 2^{i+1}(n-i-1)h(n-i) = \\ &= \sum_{i=0}^{n} 2^i(n-i)O(2^{n-i}) + \sum_{i=0}^{n-1} 2^{i+1}(n-i-1)O(2^{n-i}) = \\ &= (\sum_{i=0}^{n}(n-i))O(2^n) + (\sum_{i=0}^{n-1}(n-i-1))O(2^n) = O(n^2 2^n). \end{aligned}$$

Таким образом, предложенный алгоритм проверки правильности монотонных функций работает примерно в $\frac{2^n}{n}$ раз быстрее, чем алгоритм, основанный на переборе всех возможных пар наборов.

7. Заключение

В данной работе изучается вопрос проверки правильности семейств булевых функций. Так как задача проверки правильности в общем случае очень сложная, возникает необходимость в эффективных алгоритмах проверки правильности. На основе стандартных методов проверки правильности построен модифицированный алгоритм проверки, в котором учитывается информация о графе семейства, а именно состав сильных компонент и граф этих компонент. Учитывая то, что извлечение этой информации из семейства займет гораздо меньше времени, чем проверка правильности с помощью стандартных методов, использование модифицированного алгоритма совместно с алгоритмами построения графа семейства, поиска сильных компонент и топологической сортировки, представляется акту-

альным. Сложность приведенного алгоритма выражена через количество вершин в каждой из сильных компонент.

Также в работе построен алгоритм проверки правильности в случае семейств монотонных функций. В основе этого алгоритма лежат два свойства правильных семейств монотонных функций. Первое — такие семейства всегда содержат константную функцию. Второе — класс монотонных функций является функционально замкнутым в P_2. Эти особенности монотонных семейств сильно упрощают анализ правильности. Именно поэтому время работы данного алгоритма в $\frac{2^n}{n}$ раз меньше времени работы стандартных алгоритмов проверки.

Список литературы

[1] Носов В. А. Критерий регулярности булевского неавтономного автомата с разделенным входом // Интеллектуальные системы. Т. 3, вып. 3–4. 1998. С. 269–280.

[2] Носов В. А. О построении классов латинских квадратов в булевой базе данных // Интеллектуальные системы. Т. 4, вып. 3–4. 1999. С. 307–320.

[3] Носов В. А. Построение параметрического семейства латинских квадратов в векторной базе данных // Интеллектуальные системы. Т. 8, вып. 1–4. 2004. С. 517–528.

[4] Носов В. А., Панкратьев А. Е. О функциональном задании латинских квадратов // Интеллектуальные системы. Т. 12, вып. 1–4. 2008. С. 317–332.

[5] Харари Ф. Теория графов. М.: Мир, 1973.

[6] Яблонский С. В. Введение в дискретную математику. М.: Высшая школа, 2003.

[7] Седжвик Р. Фундаментальные алгоритмы на C++. Часть 5. Алгоритмы на графах. М.: Торгово-издательский дом DiaSoft, 2002.

[8] Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. М.: Мир, 1979.

[9] Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы. Построение и анализ / Второе издание. М.: Изд. дом Вильямс, 2005.