Секция 8

ПРИКЛАДНАЯ ТЕОРИЯ КОДИРОВАНИЯ, АВТОМАТОВ И ГРАФОВ

УДК 519.713+519.766

О ПОСТРОЕНИИ МИНИМАЛЬНЫХ ДЕТЕРМИНИРОВАННЫХ КОНЕЧНЫХ АВТОМАТОВ, РАСПОЗНАЮЩИХ ПРЕФИКСНЫЙ КОД ЗАДАННОЙ МОЩНОСТИ

И. Р. Акишев, М. Э. Дворкин

В настоящее время префиксные коды находят широкое применение в различных областях информационных технологий. В связи с этим изучение их свойств представляет большой интерес.

Наиболее естественный подход к распознаванию префиксных кодов основан на применении конечных автоматов. Имеется ряд статей [1, 2], посвященных исследованию задачи минимизации числа состояний автомата, распознающего некоторый заданный префиксный язык.

В данной работе исследуется следующая задача: дано некоторое натуральное число n, необходимо построить детерминированный конечный автомат, принимающий некоторый префиксный код мощности n над алфавитом $\Sigma = \{0, 1\}$ и имеющий наименьшее возможное число состояний.

Имеют место следующие свойства искомого автомата.

Теорема 1. Пусть детерминированный конечный автомат A — минимальный автомат, принимающий некоторый непустой язык. Этот язык является префиксным тогда и только тогда, когда в автомате A ровно одно терминальное состояние, причем все переходы из него ведут в тупиковое состояние.

Теорема 2. Пусть детерминированный конечный автомат A — минимальный автомат, принимающий конечный префиксный язык. Тогда в этом автомате нет циклов, кроме петель, ведущих из тупикового состояния в само себя.

Теорема 3. В детерминированном конечном автомате, принимающем некоторый префиксный код заданной мощности n и имеющем минимальное число состояний, нет переходов в тупиковое состояние, кроме как из единственного терминального и тупикового состояний.

Так как искомый автомат не содержит циклов, можно выписать его состояния (кроме тупикового) в порядке обратной топологической сортировки:

$$q_0 = f, q_1, q_2, \ldots, q_{k-2} = s.$$

Рассматривается соответствующая последовательность мощностей правых контекстов для выписанных состояний:

$$a_0, a_1, a_2, \ldots, a_{k-2},$$

где $a_i = |R_{q_i}|$ (R_{q_i} — правый контекст состояния q_i).

Из равенств $R_{q_0} = R_f = \{\varepsilon\}$ следует, что $a_0 = 1$. Так как состояния автомата топологически отсортированы и из каждого выходит два перехода в нетупиковые состояния, то все последующие элементы последовательности a_i являются суммой каких-либо двух предыдущих. Последний элемент a_{k-2} соответствует R_s и равен мощности языка, принимаемого автоматом.

Определение 1. *Аддитивной цепочкой* [3] называется последовательность элементов a_i, такая, что

1) $a_0 = 1$;
2) $a_i = a_j + a_k$ для некоторых $j, k < i$ при всех $i > 0$.

Итак, искомому конечному автомату соответствует некоторая аддитивная цепочка. Из минимальной аддитивной цепочки посредством обратного построения можно получить конечный автомат, принимающий некоторый префиксный код заданной мощности. Длина аддитивной цепочки на 2 меньше числа состояний в соответствующем автомате. Отсюда вытекает следующая теорема.

Теорема 4. Задача нахождения детерминированного конечного автомата с минимальным числом состояний, принимающего некоторый префиксный код заданной мощности n, эквивалентна задаче построения кратчайшей аддитивной цепочки, заканчивающейся числом n.

Из теоремы 4 следует дополнительное свойство искомого префиксного кода.

Теорема 5. Префиксный код заданной мощности, соответствующий автомату с минимальным числом состояний, является полным.

Задача о нахождении кратчайшей аддитивной цепочки является классической задачей дискретной математики [3]. Наиболее известным ее применением является задача об оптимальном алгоритме возведения произвольного числа в заданную степень, представляющая интерес для криптографии [4].

Полиномиального решения данной задачи на данный момент неизвестно. Простым и достаточно эффективным приближенным решением является классический бинарный метод возведения в степень [3]. Активно применяются методы поиска приближенного ответа, в том числе ведутся исследования по применению генетических алгоритмов [5] и «муравьиных алгоритмов» [6]. В работе [7] показано, что задача нахождения кратчайшей аддитивной цепочки, которая содержит в качестве подпоследовательности данную последовательность $b_1, b_2, \ldots, b_k$, является NP-полной. То есть естественное обобщение рассматриваемой задачи не имеет полиномиального решения, если $P \neq NP$.

ЛИТЕРАТУРА

1. *Golin M. J., Na H.* Optimal prefix-free codes that end in a specified pattern and similar problems: The uniform probability case (extended abstract) // Data Compression Conference. 2001. P. 143–152.
2. *Han Y.-S., Salomaa K., Wood D.* State complexity of prefix-free regular languages // Proc. of the 8th Int. Workshop on Descriptional Complexity of Formal Systems. 2006. P. 165–176.
3. *Кнут Д. Э.* Искусство программирования. Т. 2. Получисленные алгоритмы. М.: Вильямс, 2004. 832 с.
4. *Bleichenbacher D.* Efficiency and security of cryptosystems based on number theory. Zürich, 1996.
5. *Cruz-Cortes N., Rodriguez-Henriquez F., Juarez-Morales R., Coello C. A.* Finding optimal addition chains using a genetic algorithm approach // LNCS. 2005. V. 3801. P. 208–215.

6. *Nedjah N., de Macedo M. L.* Finding minimal addition chains using ant colony // IDEAL / ed. by R. Y. Zheng, R. M. Everson, Y. Hujun. LNCS. 2004. V. 3177. P. 642–647.
7. *Downey P., Leong B., Sethi R.* Computing sequences with addition chains // SIAM J. Computing. 1981. V. 10. No. 3. P. 638–646.

УДК 519.171

О СООТНОШЕНИЯХ СТЕПЕНИ И ПЛОТНОСТИ НЕКОТОРЫХ ГРАФОВ

И. А. Бадеха, П. В. Ролдугин

В данной работе наиболее важным является понятие реберного покрытия графа кликами (РПК). РПК — это такой набор клик (полных подграфов) $K_1, ..., K_r$, что любое ребро графа G лежит хотя бы в одной из этих клик. В качестве клик, входящих в РПК, подразумеваются только максимальные по включению клики. Кроме того, будем отождествлять клику и множество ее вершин, то есть выражение «множество вершин R образует клику в графе G» означает, что множество вершин R порождает максимальный полный подграф в графе G. Назовем ребро e графа G собственным ребром клики K, если оно лежит в этой клике и не лежит ни в какой другой максимальной по включению клике графа G. Соответственно клику K, имеющую хотя бы одно собственное ребро, назовем зафиксированной.

Утверждение 1. Клика K входит в любое РПК графа G тогда и только тогда, когда она является зафиксированной.

Собственные ребра и соответственно зафиксированные клики допускают простую характеризацию, позволяющую легко распознать их в графе.

Утверждение 2. Ребро $e \in E(G)$ является собственным ребром некоторой клики K тогда и только тогда, когда множество вершин графа G, смежных одновременно с обоими концами ребра e, порождает полный подграф в G. Кроме того, этот полный подграф в объединении с концами ребра e образует клику K.

Из данного утверждения следует возможность нахождения всех зафиксированных клик графа за полиномиальное время (трудоемкость не более $O(n^4)$, где $n = |V(G)|$). Отсюда следует, что в определенном смысле графами, в которых сложно строить минимальное РПК, являются графы, не содержащие зафиксированных клик, или, что эквивалентно, собственных ребер. Далее такие графы, то есть графы, в которых каждое ребро лежит не менее чем в двух кликах, назовем графами, свободными от собственных ребер. Введем на множестве вершин графа G отношение эквивалентности. Две вершины x и y называются эквивалентными, если они смежны и их окружения совпадают, то есть $N(x) = N(y)$. Сжатым графом назовем граф, в котором все вершины попарно неэквивалентны.

Основное содержание работы отражает следующая теорема.

Теорема 1. Предположим, что G является связным сжатым графом, свободным от собственных ребер. Тогда

1) $\rho(G) \leqslant \Delta(G) - 1$;
2) если $\rho(G) = \Delta(G) - 1$, то $\Delta(G) = 4$, и граф G эквивалентен графу B;
3) если $\rho(G) = \Delta(G) - 2$, то в графе G существует не менее двух вершин степени $\Delta(G)$, либо граф G получается из графа B добавлением одной доминирующей вершины.

С помощью введения специальных операций над сжатыми графами, свободными от собственных ребер, сохраняющих данные свойства, и с использованием данной теоремы доказывается следующее утверждение.

Утверждение 3. Существуют непустые сжатые графы, свободные от зафиксированных клик, имеющие $\rho(G) = \rho$ и $\Delta(G) = \Delta$, где Δ и ρ — произвольные натуральные числа, удовлетворяющие ограничениям: $\rho \geqslant 3$, $\Delta \geqslant \rho + 1$.

ЛИТЕРАТУРА

1. *Cavers M. S.* Clique partitions and coverings of graphs. University of Waterloo, 2005.
2. *Kou L. T., Stockmeyer L. J., Wong C. K.* Covering edges by cliques with regard to keyword conflicts and intersection graphs // Communicat. ACM. 1978. V. 21. No. 2. P. 135–139.
3. *Orlin J.* Contentment in graph theory: Covering graphs with cliques // Indagationes Math. 1977. V. 39. P. 406–424.

УДК 519.7

БЕНТ-ФУНКЦИИ И ЛИНЕЙНЫЕ КОДЫ В CDMA[1]

А. В. Павлов

Булева функция от четного числа переменных называется *бент-функцией*, если она максимально удалена от класса аффинных булевых функций. Задача построения бент-функций возникает во многих областях, в том числе в теории кодирования, где находит свое применение в системах коллективного доступа, таких, как стандарты цифровой сотовой связи CDMA. Данные стандарты используют бент-функции для построения кодов постоянной амплитуды, что позволяет предельно снизить коэффициент отношения пиковой и средней мощностей сигнала. Такие коды состоят из векторов значений бент-функций. И как известно, предпочтение отдается линейным кодам, так как они довольно просты в реализации.

Так возникла задача построения максимального линейного кода на основе заданной бент-функции, такого, что при сдвиге данной бент-функции на любое кодовое слово не нарушалось бы свойство «бент». В [1] предлагается использовать для построения кода конструкцию Мак-Фарланда [2] $f(x, y) = \langle x, \pi(y)\rangle + g(y)$, где $x, y \in E^{n/2}$; $g(y)$ — булева функция от $n/2$ переменных; π — подстановка на $E^{n/2}$; $E^{n/2}$ — булев куб размерности $n/2$. Рассмотрим линейный код длины 2^n, состоящий из векторов значений функций $h(x, y) = g(y)$ и всех аффинных функций от n переменных. Размерность данного кода равна $k = 2^{n/2} + n/2$, кодовое расстояние равно $d = 2^{n/2}$. Например, для любой бент-функции из класса Мак-Фарланда от 6 переменных имеем линейный код с параметрами $[2^6, 11, 8]$, а для 8 переменных — с параметрами $[2^8, 20, 16]$.

В [3] было доказано, что две бент-функции находятся на минимальном расстоянии $2^{n/2}$ друг от друга тогда и только тогда, когда они отличаются на аффинном подпространстве размерности $n/2$ и обе на нём аффинны. Исходя из этого критерия, предложен следующий алгоритм построения максимального линейного кода.

Алгоритм

1) Вход: бент-функция f.
2) Добавляем f в список функций $functionList$.

[1]Работа выполнена при финансовой поддержке гранта Президента РФ для молодых российских ученых (грант МК № 1250.2009.1).

3) Строим все аффинные подпространства размерности $n/2$, на которых данная бент-функция аффинна (см. [3]), и добавляем их в список $list$.
4) Далее вызываем рекурсивную функцию **findCode**(f, $list$, $functionList$).

findCode(f, $list$, $functionList$)

1) Вход: бент-функция f; список аффинных подпространств, на которых f аффинна; список бент-функций.
2) Для каждого аффинного подпространства из $list$ строим бент-функцию g на минимальном расстоянии от f.
3) Если g линейно независима со всеми функциям из $functionList$, то добавляем g в $functionList$.
4) Далее формируем список аффинных подпространств $newList$. Для каждого аффинного подпространства из $list$ проверяем условие: если функция g аффинна на данном аффинном подпространстве, добавляем это аффинное подпространство в $newList$.
5) Вызываем рекурсивную функцию **findCode**(g, $newList$, $functionList$).

С помощью данного алгоритма удалось построить линейные коды для бент-функций от 6 переменных и для некоторых бент-функций от 8 переменных. Стоит заметить, что построенные по предлагаемому алгоритму коды не обязательно являются оптимальными.

Нетрудно показать, что для аффинно эквивалентных функций размерности таких кодов будут одинаковыми. Далее приведём таблицу с аффинно неэквивалентными бент-функциями и размерностями кодов, построенных для них.

n	Бент-функция	Размерность кода
6	$x_1x_2x_3 \oplus x_2x_4x_5 \oplus x_3x_4x_6 \oplus x_1x_4 \oplus x_2x_6 \oplus x_3x_4 \oplus$ $\oplus x_3x_5 \oplus x_3x_6 \oplus x_4x_5 \oplus x_4x_6$	15
6	$x_1x_2x_3 \oplus x_2x_4x_5 \oplus x_1x_2 \oplus x_1x_4 \oplus x_2x_6 \oplus x_3x_5 \oplus x_4x_5$	15
6	$x_1x_2x_3 \oplus x_1x_4 \oplus x_2x_5 \oplus x_3x_6$	15
6	$x_1x_2 \oplus x_3x_4 \oplus x_5x_6$	15
8	$x_1x_2x_3 \oplus x_2x_4x_5 \oplus x_3x_4x_6 \oplus x_1x_4x_7 \oplus x_3x_5 \oplus$ $\oplus x_2x_7 \oplus x_1x_5 \oplus x_1x_6 \oplus x_4x_8$	29
8	$x_1x_2x_3 \oplus x_2x_4x_5 \oplus x_3x_4x_6 \oplus x_3x_5 \oplus x_2x_6 \oplus$ $\oplus x_2x_5 \oplus x_1x_7 \oplus x_4x_8$	28
8	$x_1x_2x_3 \oplus x_2x_4x_5 \oplus x_3x_4x_6 \oplus x_3x_5 \oplus x_1x_3 \oplus$ $\oplus x_1x_4 \oplus x_2x_7 \oplus x_6x_8$	30
8	$x_1x_2x_3 \oplus x_2x_4x_5 \oplus x_3x_4x_6 \oplus x_3x_5 \oplus x_2x_6 \oplus$ $\oplus x_2x_5 \oplus x_1x_2 \oplus x_1x_3 \oplus x_1x_4 \oplus x_7x_8$	30
8	$x_1x_2x_3 \oplus x_2x_4x_5 \oplus x_3x_4x_6 \oplus x_3x_5 \oplus x_1x_6 \oplus x_2x_7 \oplus x_4x_8$	28
8	$x_1x_2x_7 \oplus x_3x_4x_7 \oplus x_5x_6x_7 \oplus x_1x_4 \oplus x_3x_6 \oplus$ $\oplus x_2x_5 \oplus x_4x_5 \oplus x_7x_8$	29
8	$x_1x_2x_3 \oplus x_2x_4x_5 \oplus x_3x_4 \oplus x_2x_6 \oplus x_1x_7 \oplus x_5x_8$	28
8	$x_1x_2x_3 \oplus x_2x_4x_5 \oplus x_1x_3 \oplus x_1x_5 \oplus x_2x_6 \oplus x_3x_4 \oplus x_7x_8$	30
8	$x_1x_2x_3 \oplus x_1x_4 \oplus x_2x_5 \oplus x_3x_6 \oplus x_7x_8$	29
8	$x_1x_2 \oplus x_3x_4 \oplus x_5x_6 \oplus x_7x_8$	28

Так же как и в кодах, основанных на конструкции Мак-Фарланда, код, полученный с помощью предложенного алгоритма, имеет то же кодовое расстояние, равное

минимальному расстоянию между бент-функциями $2^{n/2}$. Особенностью метода является то, что он зависит от конкретного вида бент-функции, в отличие от конструкции Мак-Фарланда.

ЛИТЕРАТУРА

1. *Paterson K. G.* Sequences For OFDM and Multi-code CDMA: two problems in algebraic Coding Theory // Sequences and their applications. Seta 2001. Second Int. Conference (Bergen, Norway, May 13–17, 2001). Proc. Berlin: Springer, 2002. P. 46–71.
2. *McFarland R. L.* A family of difference sets in non-cyclic groups // J. Combin. Theory. Ser. A. 1973. V. 15. No. 1. P. 1–10.
3. *Коломеец Н. А., Павлов А. В.* Свойства бент-функций, находящихся на минимальном расстоянии друг от друга // Прикладная дискретная математика. 2009. № 4. С. 5–20.

УДК 519.175.1

О НОВОМ ПОЛНОМ ИНВАРИАНТЕ АЦИКЛИЧЕСКИХ ГРАФОВ

А. В. Пролубников

В задаче проверки изоморфизма графов (задача ИГ) даны два обыкновенных графа с одинаковым числом вершин и ребер. Необходимо ответить на вопрос, существует ли такое биективное отображение (изоморфизм) множества вершин одного графа на множество вершин второго, которое сохраняло бы смежность соответствующих вершин?

Поскольку любой алгоритм решения задачи ИГ представляет собой проверку равенства некоторых инвариантных относительно изоморфизма количественных характеристик графов, неясный статус задачи ИГ в иерархии теории сложности непосредственно связан с вопросом сложности вычисления полного инварианта графа. На данный момент не доказано, что задача ИГ является NP-полной, равно как и не найдено полиномиальных алгоритмов решения общего случая задачи.

Единственным известным полным инвариантом графа является его канонический код — максимальное число, двоичная запись которого может быть получена путем некоторой конкатенации строк верхне-(нижне-)треугольной подматрицы матрицы смежности графа [1].

Полные инварианты известны лишь для немногих относительно простых классов графов, поскольку наличие полиномиально вычислимого полного инварианта для графов из некоторого класса эквивалентно полиномиальной разрешимости задачи ИГ для графов из этого класса. Так, в работе [2] представлен полный инвариант для деревьев и в целом класса ациклических графов, в работе [3] — для планарных графов. Однако в этих работах, как и в большинстве работ, нацеленных на нахождение полного инварианта для ограниченного класса графов, полный инвариант ищется как результат канонизации графа — процесс, который может быть описан следующим образом. Пусть $\mathbf{G}$ — некоторый класс графов. Пусть $f : \mathbf{G} \to \{0,1\}^*$ — функция, отображающая граф в пространство битовых строк (канонических кодов), такая, что для всех $G, H \in \mathbf{G}$ имеем $G \simeq H \Leftrightarrow f(G) = f(H)$, то есть f — полный инвариант для графов из $\mathbf{G}$. Если f дает для G граф $f(G)$ такой, что $G \simeq f(G)$, то $f(G)$ — канонический код графа, по которому восстанавливается сам граф.

В этой работе предлагается алгебраический полный инвариант ациклических графов, который не получается в результате канонизации графа, а представляет собой множество из $1 + n(n+1)/2$ числовых значений, каждое из которых есть произведе-

ние собственных значений из спектра графа и спектров его подграфов. Этот подход позволяет рассмотреть вопрос поиска полного инварианта для классов графов, используя алгебраические свойства матриц, представляющих графы, и допускает обобщения на более широкие классы графов. Вариант такого обобщения также рассматривается в работе.

ЛИТЕРАТУРА

1. *Balasubramanian K., Parthasarathy K. R.* In search of a complete invariant for graphs // Lect. Notes Mathem. 1981. V. 885. P. 42–59.
2. *Lindell S.* A Logspace Algorithm for Tree Canonization // Proc. of the 24th Annual ACM Symposium on the Theory of Computing. New York: ACM, 1992. P. 400–404.
3. *Datta S., Limaye N., Nimbhorkar P., Thierauf T., Wagner F.* Planar Graph Isomorphism is in Log-Space // 24th Annual IEEE Conference on Computational Complexity. Paris, France, July 15 – July 18, 2009. ISBN: 978-0-7695-3717-7.

УДК 519.17

О КАРКАСЕ АВТОМАТА

В. Н. Салий

В [1] было введено понятие каркаса автомата. Это упорядоченное множество, которое образуют слои автомата вместе с отношением обратной достижимости. Оно сыграло весьма существенную роль в описании автоматов, у которых каждая конгруэнция является ядром подходящего эндоморфизма. В работе устанавливаются некоторые свойства каркаса, связанные с основными алгебраическими конструкциями для автоматов, такими, как подавтомат, гомоморфизм, конгруэнция.

Автомат — это тройка $\mathbf{A} = (S, X, \delta)$, где S и X — конечные непустые множества, соответственно множество состояний и множество входных сигналов, а $\delta : S \times X \to S$ — отображение, называемое функцией переходов.

Подмножество $S' \subseteq S$ называется устойчивым в автомате $\mathbf{A}$, если $\delta(s, x) \in S'$ для любых $s \in S'$ и $x \in X$. Если S' устойчиво в $\mathbf{A}$, то, ограничивая функцию переходов δ на $S' \times X$, получают автомат $\mathbf{A'} = (S', X, \delta)$ — подавтомат автомата $\mathbf{A}$, соответствующий S'. Совокупность Sub$\mathbf{A}$ всех подавтоматов автомата $\mathbf{A}$, упорядоченная отношением $\mathbf{A_1} \leqslant \mathbf{A_2} \iff S_1 \subseteq S_2$, где $\mathbf{A_i} = (S_i, X, \delta)$, $i = 1, 2$, является дистрибутивной решеткой.

Пусть $\mathbf{A} = (S, X, \delta)$ и $\mathbf{B} = (T, X, \delta)$ — сравнимые автоматы, т. е. автоматы с одним и тем же множеством входных сигналов. Отображение $\varphi : S \to T$ по определению является гомоморфизмом автомата $\mathbf{A}$ в автомат $\mathbf{B}$, если $\varphi(\delta(s, x)) = \delta(\varphi(s), x)$ для любых $s \in S$, $x \in X$. Взаимно однозначные гомоморфизмы автоматов называются вложениями, а их биективные гомоморфизмы — изоморфизмами. Эндоморфизмы автомата — это его гомоморфизмы в себя, автоморфизмы — изоморфизмы на себя.

Отношение эквивалентности $\theta \subseteq S \times S$ называется конгруэнцией автомата $\mathbf{A} = (S, X, \delta)$, если оно согласовано с функцией переходов в том смысле, что $(\forall s, t \in S)(\forall x \in X)((s, t) \in \theta \implies (\delta(s, x), \delta(t, x)) \in \theta)$. Каждая конгруэнция θ автомата $\mathbf{A}$ определяет его фактор-автомат $\mathbf{A}/\theta = (S/\theta, X, \delta)$, где $\delta(\theta(s), x) := \theta(\delta(s, x))$ для любых $s \in S$, $x \in X$.

Пусть X^* — множество всех конечных слов над алфавитом X. Продолжим функцию переходов δ на множество $S \times X^*$, полагая $\delta(s, e) = s$, где $e \in X^*$ — пустое слово, и $\delta(s, px) = \delta(\delta(s, p), x)$ для любых $s \in S$, $x \in X$, $p \in X^*$. Говорят, что состояние t

достижимо в автомате $\mathbf{A}$ из состояния s, если найдется входное слово $p \in X^*$, такое, что $\delta(s,p) = t$. Записывая это в виде $(s,t) \in \tau$, вводим отношение достижимости τ в автомате $\mathbf{A}$.

Симметричная часть $\sigma = \tau \cap \tau^{-1}$ отношения достижимости называется отношением взаимной достижимости в $\mathbf{A}$. Классы этой эквивалентности называют слоями автомата $\mathbf{A}$. Каркасом автомата $\mathbf{A}$ назовем упорядоченное множество $F(\mathbf{A}) = (S/\sigma, \tau^{-1})$. Его элементами являются слои автомата $\mathbf{A}$, а порядком — отношение, обратное достижимости, перенесенное на слои: $(\sigma(t), \sigma(s)) \in \tau^{-1}$ равносильно тому, что $(s,t) \in \tau$.

Теорема 1. Каждое конечное упорядоченное множество изоморфно каркасу подходящего автомата с двумя входными сигналами.

Теорема 2. Конечное упорядоченное множество тогда и только тогда изоморфно каркасу автономного автомата, когда у каждого его элемента имеется не более чем один нижний сосед.

Теорема 3. Если $\mathbf{A}$ и $\mathbf{B}$ — произвольные автоматы, то $\mathrm{Sub}\mathbf{A} \cong \mathrm{Sub}\mathbf{B}$ тогда и только тогда, когда $F(\mathbf{A}) \cong F(\mathbf{B})$.

Теорема 4. Если φ — вложение автомата $\mathbf{A}$ в автомат $\mathbf{B}$ и $F(\mathbf{A}) \cong F(\mathbf{B})$, то φ — изоморфизм $\mathbf{A}$ на $\mathbf{B}$.

Следствие 1. Если $\mathbf{A}'$ — подавтомат автомата $\mathbf{A}$ и $F(\mathbf{A}') \cong F(\mathbf{A})$, то $\mathbf{A}' = \mathbf{A}$.

Следствие 2. Если φ — эндоморфизм автомата $\mathbf{A}$ и $F(\varphi(\mathbf{A})) \cong F(\mathbf{A})$, то φ — автоморфизм.

Теорема 5. Если θ — конгруэнция автомата $\mathbf{A}$, то $F(\mathbf{A}/\theta) \cong F(\mathbf{A})$ тогда и только тогда, когда $\theta \subseteq \sigma$.

ЛИТЕРАТУРА

1. *Салий В. Н.* Автоматы, у которых все конгруэнции — внутренние // Изв. вузов. Математика. 2009. № 9. С. 36–45.

УДК 519.5

КЛАССЫ ГРАФОВ, ВОССТАНАВЛИВАЕМЫЕ С ЛИНЕЙНОЙ ВРЕМЕННОЙ СЛОЖНОСТЬЮ

Е. А. Татаринов

Рассматривается задача [1] восстановления конечного связного неориентированного графа G без петель и кратных ребер при помощи агента, который премещается по ребрам графа G, считывает и изменяет метки на вершинах и инциденторах. На основе собранной информации агент строит граф H, изоморфный графу G с точностью до меток на вершинах и инциденторах графов. Необходимо найти метод обхода и разметки графа G с целью его восстановления.

Известен ряд методов восстановления графа [2, 3], которые используют не более четырех различных красок, однако имеют верхнюю оценку временной сложности восстановления, равную квадратичной и кубической функцией от числа вершин в восстанавливаемом графе соответственно. Для каждого метода нижней оценкой временной сложности восстановления является линейная функция от числа вершин в графе G. Данная работа посвящена выделению классов графов, для которых верхняя

оценка временной сложности восстановления является линейной функцией, и построению операций над графами, порождающих новый граф, для которого верхняя оценка временной сложности не ухудшается.

Методы основаны на обходе графа «в глубину». Для восстановления ребра агент вычисляет M — номера вершин [4], которым оно инцидентно. Для этого агент выполняет некоторое количество переходов по ранее посещенным вершинам. Этот отход порождает нелинейность верхней границы временной сложности восстановления. Для некоторых классов количество и длина таких отходов могут быть заранее известны и ограничены. Очевидны следующие утверждения.

Теорема 1. Для графа вида «дерево» верхняя оценка временной сложности восстановления является линейной функцией от числа вершин G.

Теорема 2. Для графа вида «кольцо» порождается один отход при восстановлении графа G.

Найдены операции над графами, которые не ухудшают верхнюю оценку $T(G)$ временной сложности восстановления графа.

Теорема 3. Операции добавления в граф висячей вершины и введения вершины в ребро графа не ухудшают верхнюю оценку временной сложности восстановления.

Таким образом, если граф G восстанавливается за линейное время, то и граф, полученный из него при помощи этих операций, восстанавливается за линейное время.

Найдена бинарная операция, в результате которой получается граф, для которого верхняя оценка временной сложности восстановления не превосходит суммы верхних оценок временной сложности восстановления исходных графов.

Теорема 4. Если граф G получен путем соединения двух графов G_1 и G_2 через вершину сочленения, то $T(G) \leqslant T(G_1) + T(G_2)$.

Данная операция допускает естественное обобщение.

Теорема 5. Если граф G получен путем древовидного соединения графов $G_1, \ldots, G_i$ через вершины сочленения, то $T(G) \leqslant T(G_1) + \ldots + T(G_i)$.

Если $G_1, \ldots, G_i$ восстанавливаются за линейное время, то G, полученный при помощи этой операции, также восстанавливается за линейное время.

В результате найдены частные случаи классов графов, для которых верхняя оценка временной сложности является линейной функцией от числа врешин в восстанавливаемом графе. Найдены операции, которые не ухудшают верхнюю оценку временной сложности восстановления, и бинарная операция, которая строит граф, для которго верхняя оценка временной сложности не превосходит суммы верхних оценок временной сложности исходных графов.

ЛИТЕРАТУРА

1. *Dudek G., Jenkin M.* Computational principles of mobile robotic. Cambridge Univ. Press. 2000.
2. *Грунский И. С., Татаринов Е. А.* Распознавание конечного графа блуждающим по нему агентом // Вестник Донецкого университета. Сер. А. Естественные науки. 2009. Вып. 1. С. 492–497.
3. *Грунский И. С., Татаринов Е. А.* Алгоритм распознавания графов // Труды Четвертой Междунар. конф. «Параллельные вычисления и задачи управления» PACO '2008. М.: Институт проблем управления им. В. А. Трапезникова РАН, 2008. С. 1483–1498.

4. *Касьянов В. Н., Евстигнеев В. А.* Графы в программировании, визуализация и применение. СПб.: Петербург, 2003.

УДК 519.713

ПЕРЕСТРАИВАЕМЫЕ АВТОМАТЫ С ОБЩЕЙ ПАМЯТЬЮ[1]

В. Н. Тренькаев

Одним из требований к современным цифровым устройствам является гибкость, т. е. возможность внесения изменений в алгоритм функционирования, реализуемая с помощью соответствующей настройки устройства [1, 2]. В работе рассматривается функциональная настройка, когда не изменяются связи между элементами перестраиваемого устройства, но изменяется их функциональность. Предполагается, что часть устройства реализована на «жесткой» логике, а часть — на многофункциональных настраиваемых элементах. Например, к данному классу перестраиваемых устройств можно отнести автоматные шифраторы с ключевой информацией в виде подмножества переходов. Поведение таких устройств предлагается моделировать с помощью совместной работы двух базовых автоматов, один из которых имеет жестко фиксированное поведение, а поведение другого может изменяться (задаваться пользователем). Предложена логическая структура перестраиваемого устройства (рис.1), поведение которого является объединением поведения составляющих его базовых автоматов.

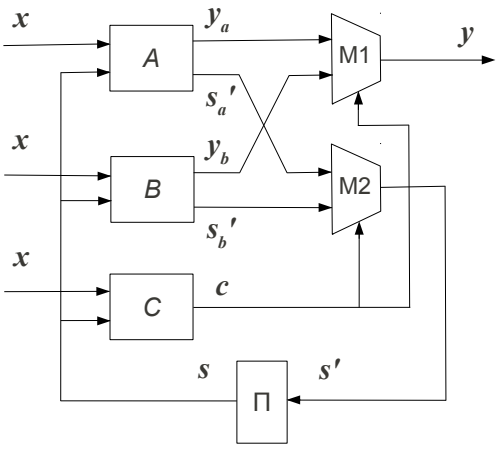

Рис. 1. Структура перестраиваемого устройства

Далее в работе «гибкие» цифровые устройства описываются моделью перестраиваемого автомата. Перестраиваемым автоматом Q называется шестерка $(S, X, Y, K, \psi, \varphi)$, где S, X, Y, K — конечные множества, называемые соответственно множеством состояний, входным алфавитом, выходным алфавитом, множеством настроек, а $\psi : S \times X \times K \to S$ и $\varphi : S \times X \times K \to Y$ — функции переходов и выходов соответственно. Таким образом, перестраиваемый автомат Q задает множество автоматов Мили $\{A_k = (S, X, Y, \psi_k, \varphi_k) : k \in K\}$, где $\psi_k(s, x) = \psi(s, x, k), \varphi_k(s, x) = \varphi(s, x, k)$ для всех $s \in S$, $x \in X$ и $k \in K$.

[1]Работа выполнена в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009–2013 гг. (гос. контракт № П1010).

Рассмотрим предлагаемую логическую схему перестраиваемого устройства (рис.1). Блоки A, B, C имеют входы $x \in X$, $s \in S$. Выходы блока A суть $y_a \in Y$ и $s'_a \in S$ — значения его функций $\mu : S \times X \to Y$ и $\eta : S \times X \to S$ соответственно. Выходы блока B суть $y_b \in Y$ и $s'_b \in S$ — значения его функций $\lambda : S \times X \times K \to Y$ и $\delta : S \times X \times K \to S$ соответственно при фиксированном $k \in K$, т. е. блок B отвечает за настройку устройства. Блок C реализует функцию $\pi : S \times X \times K \to \{a, b\}$ при фиксированном $k \in K$, т. е. блок C также отвечает за настройку устройства. Таким образом, чтобы настроить устройство, требуется зафиксировать $k \in K$. Выход блока C есть управляющий символ c для мультиплексоров M1 и M2, которые в зависимости от c «пропускают дальше» соответствующие выходы либо блока A, либо блока B. Более точно: если $c = a$, то выход M1 есть y_a, т. е. $y = y_a$, а выход M2 есть s'_a, т. е. $s' = s'_a$; если $c = b$, то $y = y_b$ и $s' = s'_b$. Память перестраиваемого устройства реализуется в виде блока П, в котором хранится состояние s, причем $s = s'$ в следующий такт времени работы устройства.

Поведение перестраиваемого устройства формируется в результате совместной работы двух базовых автоматов $A_1 = (S, X, Y, \eta, \mu)$ и $A_2 = (S, X, Y, \delta_k, \lambda_k)$ и может быть описано с помощью перестраиваемого автомата. Такой автомат будем называть перестраиваемым автоматом с общей памятью, поскольку состояние, в которое перейдет автомат, формируется как блоком A, так и блоком B, но в память «закладывается» только одно из двух возможных значений, т. е. память является как бы общей для автоматов A_1 и A_2. Справедливо следующее утверждение.

Утверждение. Пусть заданы два базовых автомата $A_1 = (S, X, Y, \eta, \mu)$ и $A_2 = (S, X, Y, \delta_k, \lambda_k)$, а также функция $\pi : S \times X \times K \to \{a, b\}$. Тогда перестраиваемый автомат с общей памятью $Q = (S, X, Y, K, \psi, \varphi)$ при фиксированной настройке $k \in K$ есть автомат Мили $A_k = (S, X, Y, \psi_k, \varphi_k)$, такой, что для любой пары (s, x) из $S \times X$ верно: если $\pi_k(s, x) = a$, то $\psi_k(s, x) = \eta(s, x)$ и $\varphi_k(s, x) = \mu(s, x)$, иначе $\psi_k(s, x) = \delta_k(s, x)$ и $\varphi_k(s, x) = \lambda_k(s, x)$.

ЛИТЕРАТУРА

1. *Sklyarov V.* Reconfigurable models of finite state machines and their implementation in FPGAs // J. Systems Architecture. 2002. No. 47. P. 1047–1064.
2. *Шидловский С. В.* Автоматическое управление. Перестраиваемые структуры. Томск: Томский государственный университет, 2006. 288 с.