УДК 519.713

А.В. Цыганов

ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ПОСТРОЕНИЯ ПОЛНОГО АВТОМАТА С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИЙ OPENMP И MPI

Цыганов Андрей Владимирович, *кандидат физико-математических наук, доцент кафедры высшей математики Ульяновского государственного педагогического университета им. И.Н. Ульянова. Имеет научные публикации, монографию, учебно-методические пособия и свидетельства о регистрации программ в области параллельных эвристических алгоритмов и алгоритмов минимизации недетерминированных конечных автоматов. [e-mail: andrew.tsyganov@gmail.com].*

Аннотация

Полный (COM) автомат, являющийся одним из инвариантов регулярного языка, используется в алгоритмах минимизации недетерминированных конечных автоматов (НКА), однако алгоритм его построения по каноническим автоматам имеет экспоненциальную сложность. Одним из способов ускорения построения полного автомата является использование параллелизма. В работе рассматривается параллельный алгоритм построения COM-автомата с использованием технологий параллельного программирования OpenMP и MPI, реализованный в программе для минимизации НКА ReFaM. Приводится описание алгоритма и его программной реализации, а также результаты численных экспериментов.

Ключевые слова: недетерминированные конечные автоматы, вершинная минимизация, полный автомат, параллелизм, OpenMP, MPI.

Andrey Vladimirovich Tsyganov, *Candidate of Physics and Mathematics; Associate Professor at the Chair of Higher Mathematics of Ulyanovsk State Pedagogical University named after Ilya N. Ulyanov; author of scientific publications, a monograph, teaching guides and certificates of program registration in the fields of parallel heuristic algorithms and algorithms for minimizing nondeterministic finite automata. e-mail: andrew.tsyganov@gmail.com.*

Abstract

Complete (COM) automaton which is one of the regular Language invariants, is used in nondeterministic finite automata minimization algorithms, but the algorithm for its construction from canonical automata has an exponential complexity. One of the ways to speed up the construction of a COM-automaton is to use parallelism. In the present paper, the author considers the parallel algorithm for constructing COM-automaton using parallel programming techniques OpenMP and MPI, which is implemented in a software tool for minimizing nondeterministic finite automata called ReFaM. The author also provides a description of the algorithm and its implementation as well as the results of numerical experiments.

Key words: nondeterministic finite automata, state minimization, complete automaton, parallelism, OpenMP, MPI.

ВВЕДЕНИЕ

Конечные автоматы (КА) широко применяются для моделирования и автоматизации различных процессов, в частности, в автоматном программировании [1], для синтеза систем управления [2] и др. Приведем основные определения из теории КА и формальных языков, которые нам понадобятся в дальнейшем (более подробную информацию можно найти, например, в [3]).

Недетерминированным конечным автоматом называется пятерка $A = (Q, E, A, I, F)$, где Q - это конечное множество состояний (вершин), E - конечный алфавит, $A \subseteq Q \times E \times Q$ - множество переходов (дуг), $I \subseteq Q$ и $F \subseteq Q$ - множества начальных и конечных состояний соответственно. Переходы автомата A часто описывают с помощью функции переходов $\delta: Q \times E \to 2^Q$. НКА называется *детерминированным* (ДКА) тогда и только тогда, когда $|I| = 1$ и $\forall q \in Q, \forall a \in E: |\delta(q,a)| = 1$ (или $|\delta(q,a)| \le 1$).

В теории формальных языков КА часто применяются для распознавания языков (множеств строк, состоящих из символов заданного алфавита), в связи с чем их называют автоматами-распознавателями, а языки, распознаваемые такими автоматами, называют *регулярными*. Два КА называют *эквивалентными,* если они распознают один и тот же язык. Для любого НКА с помощью процедуры *детерминизации* можно построить эквивалентный ему ДКА. Для данного языка *зеркальным* языком называют язык, который состоит из всех строк исходного языка, записанных «задом наперед», а для данного КА *зеркальным автоматом* будем называть такой автомат, который получается из него изменением направления дуг, при этом входные состояния становятся выходными и наоборот. КА обычно изображают в виде ориентированного графа или таблицы переходов.

Одной из важнейших задач теории КА является задача их минимизации, например, по числу состояний *(вершинная минимизация)*. В алгоритмах минимизации НКА важную роль играют автоматы, являющиеся инвариантами регулярных языков. Самым известным инвариантом регулярного языка является так называемый *канонический* автомат - минимальный (по числу состояний) ДКА, распознающий данный язык. В качестве примера другого инварианта можно привести *универсальный* автомат [4].

Как правило, в программном обеспечении для работы с КА и родственными структурами, например, в [5-9] точные алгоритмы минимизации НКА не реализуются, однако в них имеются процедуры для построения тех или иных инвариантов регулярных языков. В данной работе будет рассмотрен алгоритм и программная реализация, построения, так называемого полного (COM) автомата, описанного в [3] и использующегося в некоторых алгоритмах вершинной минимизации НКА,реализованных в программе ReFaM.

ОПРЕДЕЛЕНИЕ COM-АВТОМАТА

В статье [3] COM-автомат для данного языка, заданного некоторым автоматом A, определяется с помощью специального бинарного отношения #, которое может быть записано в виде бинарной таблицы, однако строгое определение и описание алгоритма построения этого отношения достаточно громоздки, поэтому мы рассмотрим лишь общую схему его построения.

Пусть дан некоторый автомат A. Обозначим через $\overline{A}$ зеркальный ему автомат. Эти автоматы задают в общем случае разные языки (исходный и зеркальный), для которых мы построим канонические автоматы X и Усоответственно, то есть подвергнем их последовательно процедурам детерминизации и минимизации (Y при этом называют *зеркальным каноническим автоматом)*. Заметим, что после выполнения процедур канонизации состояниями канонических автоматов являются подмножества состояний детерминированных автоматов, полученных из исходных автоматов с помощью процедуры детерминизации. Отношение # устанавливает соответствие между состояниями канонического автомата и зеркального канонического автомата. При этом строки таблицы отношения # соответствуют состояниям канонического автомата, а столбцы - состояниям зеркального канонического автомата. Если для какой-либо строки i и столбца j пересечение соответствующих подмножеств является непустым множеством, то соответствующий элемент таблицы помечается символом # (при этом рассматриваются только «непустые» состояния канонических автоматов).

Таблица 1

Таблица переходов автомата A

	a	b
→ a_0	$\{a_2\}$	{a j
← a_1	Ц }	Ц }
← a_2		$\{a_0, a_2\}$

Таблица 2

Таблица переходов канонического автомата X

	a	b
← x_0	x_0	x_0
← x_1	x_2	x_0
← x_2		x_1
→ x_3	x_2	x_0

Таблица 3

Таблица переходов зеркального канонического автомата Y

	a	b
← Л	У1	У2
У1	У1	У0
← Л	У0	У2
→ У3	У0	У2

В качестве примера, рассмотрим автомат A над алфавитом E = (a, b}, заданный таблицей переходов (табл. 1). Канонический и зеркальный канонический автоматы Xи Y приведены для него в таблицах 2 и 3 соответственно. Таблица бинарного отношения # для автомата A приведена в таблице 4 (детали построения опущены).

Таблица 4

Таблица бинарного отношения #

	0	1	2	з
0	#	#	#	#
1	#		#	#
2			#	#
з	#		#	

Следующим важным понятием, необходимым для определения COM-автомата, является понятие грида (блока) таблицы отношения #. Пусть R - подмножество строк, а C - подмножество столбцов таблицы отношения #. Тогда декартово произведение R x C называется гридом (блоком), если оно удовлетворяет следующим двум условиям:

1) на пересечениях всех его строк и столбцов располагаются #;

2) множества R и C нельзя расширить без нарушения первого условия.

Для рассматриваемого нами примера в таблице 4 имеется 5 гридов: {0} x {0, 1, 2, 3}, {0, 1, 2} x {2, 3}, {0, 1, 3} x {0, 2}, {0, 1} x {0, 2, 3} и {0, 1, 2, 3} x {2}.

Состояниями COM-автомата являются блоки таблицы отношения #, а алфавитом - алфавит исходного автомата (языка). Для того чтобы полностью описать автомат, требуется определить его функцию переходов 5, а также множества входных и выходных состояний.

Пусть B - блок таблицы отношения #. Через а (B) будем обозначать множество строк данного блока (некоторое подмножество состояний канонического автомата X), а через P(B) - множество его столбцов (некоторое

подмножество состояний канонического автомата Y). Функция переходов COM-автомата определяется следующим образом. Пусть B_1 и B_2 - два состояния (блока) COM-автомата, тогда B 2 e 5(B p a) при одновременном выполнении следующих двух условий:

$$\left(\forall q \in \alpha(B_1)\right)\left(\delta_X(q,a) \subseteq \alpha(B_2)\right) \text{ и}$$

(Ч д е Р (В 2)) (Ь г (д , а) с Щ)), (1)

где 5_X и $5_Г$ - это функции переходов канонического и зеркального канонического автоматов соответственно. Множества начальных и конечных состояний COM-автомата определяются следующим образом: состояние B является начальным, если a(B) содержит (единственное) начальное состояние автомата X, и конечным, если P(B) содержит (единственное) начальное состояние автомата Y.

В таблице 5 приведена таблица переходов COM-автомата для рассматриваемого примера. Данный автомат имеет 5 состояний, соответствующих перечисленным выше гридам.

Таблица 5

Таблица переходов COM-автомата

	a	*b*
^ co	{c_0, c_1, c_2, c_3, c_4^	^ S c_2 ^ c_4}
^ c		{c_1, c_2 ^ c_4}
^ c2	c c j	K S c_2 ^ c_4}
^ c3	c c j	{c$_0$, c_1, c_2, c_3 c_4^
^ c4		{c$_1$, c_2, c_3 c_4^

Заметим, что нами рассмотрено одно из определений COM-автомата. Построенный таким образом COM-автомат может применяться в задачах вершинной минимизации НКА. В другом возможном определении состояния COM-автомата соответствуют не блокам, а *псевдоблокам* таблицы отношения #, для которых отсутствует требование максимальности (второе условие в определении блока).

АЛГОРИТМЫ ПОСТРОЕНИУ COM-АВТОМАТА

С учетом определения COM-автомата последовательный алгоритм его построения для произвольного КА A может быть сформулирован следующим образом.

Алгоритм построения COM-автомата.

1. Построить канонические автоматыXи Y.
2. Построить таблицу отношения #.
3. Найти все гриды таблицы отношения #.
4. Построить функцию переходов, используя условия (1).
5. Разметить начальные и конечные состояния.

В сформулированном алгоритме шаг 1 имеет теоретически экспоненциальную сложность, но на практике канонизация НКА обычно выполняется достаточно быстро. Как показано далее, шаг 3 алгоритма всегда имеет экспоненциальную сложность, так как связан с полным перебором всех подмножеств строк или столбцов таблицы отношения #, остальные шаги (2, 4 и 5) не приводят к комбинаторному взрыву, но их трудоемкость зависит от результатов выполнения шагов 1 и 3.

Построение канонического автомата X заключается в применении процедуры детерминизации к автомату A и последующей минимизации полученного ДКА, построение зеркального канонического автомата Y выполняется аналогично, но автомат A предварительно обращается. Все эти операции являются классическими.

Процедуру поиска всех гридов таблицы отношения # (шаг 3) можно описать следующим алгоритмом.

Алгоритм поиска гридов.

Для каждого $к$ = 1, 2, ..., n, где n - число столбцов таблицы:

1. Сгенерировать все возможные подмножества столбцов таблицы из $к$ элементов.
2. Для каждого сгенерированного подмножества столбцов C выбрать такое подмножество строк R, чтобы пересечения всех строк и столбцов содержали знак #.
3. Проверить, может ли подмножество C быть расширено без нарушения предыдущего условия, и, если нет, добавить грид R^{x} C к множеству найденных гридов.

Данный алгоритм нахождения гридов «по столбцам» применяется, если число столбцов в таблице не превышает числа строк, в противном случае применяется аналогичный алгоритм «по строкам». Кроме того, заметим, что основное время в этом алгоритме тратится на шаги 2 и 3.

Таким образом, нахождение всех гридов таблицы отношения # связано с перебором всех подмножеств столбцов или строк. Если исходный автомат A имеет N состояний, то после процедуры канонизации число строк/столбцов, соответствующих непустым состояниям канонических автоматов, теоретически может равняться $2^{N} - 1$ (на практике это число обычно значительно меньше). Отсюда легко получается верхняя оценка для числа гридов матрицы: $2^{2^{N}-1} - 1$. Несмотря на то, что приведенные теоретические оценки на практике обычно не достигаются, они дают представление о вычислительной сложности задачи.

Для ускорения выполнения отдельных шагов рассмотренного алгоритма построения COM-автомата могут быть использованы различные технологии параллельного программирования, например, OpenMP или MP1 [10], [11]. Рассмотрим, как это реализовано в программе 1TePaM.

ReFarn (Rational Expressions and Finite Automata M^'тпґаЬ'оп) - кросс-платформенная программа с открытым исходным кодом, написанная на языке C++ и предназначенная для минимизации НКА. ReFarn является частью библиотеки параллельных метаэвристик Ь^О [12].

В программе ReFarn с помощью технологий OpenMP и MP1 распараллелена самая трудоемкая часть построения COM-автомата - поиск гридов (шаг 3 алгоритма). При использовании обеих технологий параллельного программирования каждый из потоков генерирует всю последовательность подмножеств столбцов/строк, но обрабатывает только подмножества, начиная с номера i с шагом h, где i - номер потока, а h - общее число потоков (фактическая нумерация подмножеств не производится, так как

это потребовало бы использования библиотек для работы с числами произвольной точности). Таким образом, все подмножества столбцов равномерно распределяются между потоками. Для генерации подмножеств используется алгоритм Чейза [13]. В OpenMP-версии алгоритма гриды, найденные каждым потоком, помещаются в общий вектор grids_. Для MPI-версии вектор grids_ является локальным для каждого потока, а по окончании поиска найденные каждым потоком гриды пересылаются потоку с номером 0 (для этого в программе реализованы методы сериализации и десериализации гридов).

Алгоритм поиска гридов для каждого потока в OpenMP-версии программы выглядит следующим образом.

Поиск гридов потоками с использованием технологии OpenMP.

1. Определить свой номер потока i.
2. Сгенерировать первые i подмножеств столбцов.
3. Если генерация не закончилась, то попытаться построить грид по последнему сгенерированному подмножеству. Если грид построен, то добавить его в вектор grids_.
4. Сгенерировать следующие h подмножеств столбцов и перейти к п. 3.

Алгоритм поиска гридов в MPI-версии программы отличается дополнительным шагом 5: переслать найденные гриды потоку с номером 0.

В OpenMP-версии также распараллелены основные циклы шагов 4 и 5 алгоритма.

ЧИСЛЕННЫЕ ЭКСПЕРИМЕНТЫ

Для построения COM-автоматов в программе ReFaM предусмотрена команда build_com, которая имеет следующий формат: build_com in_dir out_dir. Программа сканирует все найденные в каталоге in_dir файлы с описаниями автоматов (в формате xml), обрабатывает их и записывает результаты в каталог out_dir.

Рассмотрим результаты построения COM-автоматов для случайной выборки из 100 попарно неэквивалентных НКА без недостижимых и бесполезных состояний со следующими параметрами: число состояний $|Q| = 6$, размер алфавита $|\mathrm{E}| = 3$, число начальных состояний $|I| = 2$, число конечных состояний $|F| = 2$, плотность переходов $D = \frac{T}{|Q|^2|\wedge|'} = 0,3$, где T- число переходов в автомате. Эксперименты проводились с использованием следующей программно-аппаратной конфигурации: Intel Core 2 Quad Q6600 @ 2.40 GHz, 4 Gb RAM, Microsoft Windows 7 Максимальная.

В таблице 6 представлены результаты статистической обработки времени работы алгоритма в целом и наиболее трудоемких его частей в зависимости от числа потоков для OpenMP-версии программы. Приведены: среднее время построения COM-автомата t, среднее время поиска гридов t_3 и среднее время построения функции переходов и разметки начальных и конечных состояний $t_{4,5}$. Из таблицы следует, что для рассматриваемой выборки в среднем около двух третей всего времени работы алгоритма тратится на поиск гридов и практически все время работы алгоритма приходится на шаги 3-5. Кроме того, видно, что рассматриваемая реализация имеет хорошее ускорение.

Таблица 6

Зависимость времени работы алгоритма (в сек) от числа потоков

	t_3	$t_{4,5}$	t
1	5,5546	2,9552	8,5133
2	3,1378	1,5090	4,6504
3	2,2791	0,9991	3,2817
4	1,9467	0,8147	2,7649

В таблице 7 представлены результаты статистической обработки характеристик полученных COM-автоматов (ц - среднее значение, о - среднеквадратическое отклонение).

Таблица 7

Статистические характеристики COM-автоматов

	min	max	μ	o
Число состояний	1	1270	121,2800	216,3910
Число начальных состояний	1	479	42,7700	75,2816
Число конечных состояний	1	476	36,3900	66,6138
Число переходов	3	1607305	73035,2300	229738,3453
Плотность переходов	0,3184	1,0000	0,5942	0,1625

ЗАКЛЮЧЕНИЕ

В данной работе был рассмотрен параллельный алгоритм построения COM-автомата, который является одним из инвариантов регулярного языка и используется в некоторых алгоритмах минимизации НКА. Наиболее трудоемкие шаги последовательного алгоритма были распараллелены с использованием технологий параллельного программирования OpenMP и MPI. Результаты численных экспериментов подтверждают вычислительную сложность задачи и показывают, что предложенная реализация алгоритма обладает хорошим ускорением.

СПИСОК ЛИТЕРАТУРЫ

1. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. - СПб. : Питер, 2008. - 167 с.

2. Вашкевич Н.П., Вашкевич С.Н. Недетерминированные автоматы и их использование для синтеза систем управления. Часть 1. Эквивалентные преобразования недетерминированных автоматов : учеб. пособие. - Пенза : Изд-во Пенз. гос. техн. ун-та, 1996. - 88 с.

3. Мельников Б.Ф. Недетерминированные конечные автоматы. - Тольятти : Изд-во ТГУ, 2009. - 160 с.

4. Po1бк L., Minimalizations of NFA using the universal automaton // Int. J. Found. Comput. Sci., Vol. 16, No. 5, pp. 999-1010, 2005.

5. Kell V., Maier A., Potthoff A. et al. AMORE: a system for computing automata, monoids and regular expressions // Proceedings of the 6th Annual Symposium on Theoretical Aspects of Computer Science on STACS 89. New York, NY, USA: Springer-Verlag New York, Inc., 1989. pp. 537-538.

6. Mohri M., Pereira F., Riley M. AT&T General-purpose finite-state machine software tools.

7. Raymond D.,Wood D. Grail: Engineering Automata in C++: Tech. Rep. HKUST-CS96-24: Hong Kong University of Science and Technology, 1996.

8. Lombardy S., Poss R., Rйgis-Gianas Y., Sakarovitch J. Introducing VAUCANSON // Implementation and Application of Automata, 8th International Conference, CIAA 2003, Santa Barbara, California, USA, July 16-18, 2003, Proceedings / Ed. by O. H. Ibarra, Z. Dang. Vol. 2759 of Lecture Notes in Computer Science. Springer, 2003. pp. 96-107.

9. Rodger S.H. JFLAP: An Interactive Formal Languages and Automata Package. USA: Jones and Bartlett Publishers, Inc., 2006.

10. Гергель В.П. Высокопроизводительные вычисления для многоядерных многопроцессорных систем : учеб. пособие. - Нижний Новгород : Изд-во ННГУ им. Н.И. Лобачевского, 2010.

11. Гергель В.П. Теория и практика параллельных вычислений. - М. : Изд-во Бином. Лаборатория знаний, Интернет-университет информационных технологий, 2007. - 424 с.

12. Цыганов А.В., Булычов О. И. НеО: библиотека метаэвристик для задач дискретной оптимизации // Программные продукты и системы. - 2009. - № 4. - С. 148-151.

13. Chase P.J. Algorithm 382: Combinations of M out of N Objects [G6] // Communications of the Association for Computing Machinery 13:6:368, 1970.