Статья опубликована в сборнике докладов "Международной конференции по мягким вычислениям и измерениям" (SCM`2011). СПб ГЭТУ "ЛЭТИ". 2011. Т. 2, с. 76 – 80

ПРИМЕНЕНИЕ АЛГОРИТМА EDSM ДЛЯ ПОСТРОЕНИЯ УПРАВЛЯЮЩИХ КОНЕЧНЫХ АВТОМАТОВ ПО СЦЕНАРИЯМ РАБОТЫ

А.К. Вихарев, В.И. Ульянцев, А.А. Шалыто

Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Abstract. This paper describes the method of extended finite-state machine induction using state-merging technique named evidence-driven state merging (EDSM). The described algorithm is based on the Blue-Fringe framework. Input data for the induction algorithm is a set of so-called test scenarios. The induction method was tested on alarm clock controlling EFSM induction problem on which it greatly outperformed induction method based on genetic algorithms.

Введение. В последнее время развивающаяся теория автоматного программирования находит все больше применений для решения различных задач. В рамках данной теории поведение программы описывается набором детерминированных конечных управляющих автоматов [1].

Многие задачи можно решить, построив управляющий автомат вручную, однако существуют задачи, в которых построение управляющего автомата вручную затруднительно. В качестве примеров задач такого вида можно привести задачи об «Умном муравье» [2–4], об управлении моделью беспилотного летательного аппарата [5].

Данные задачи успешно решаются с применением генетических алгоритмов [6], в том числе на основе обучающих примеров [7]. Недостатком такого подхода является большое время работы алгоритма, а также недетерминированное поведение алгоритма.

В данной работе предлагается альтернативный метод построения управляющих автоматов с применением метода сливания состояний, лишенный недостатков генетических алгоритмов.

Постановка задачи. Управляющим конечным автоматом будем называть детерминированный конечный автомат, каждый переход которого помечен *событием*, последовательностью *выходных воздействий* и *охранным условием*, представляющим собой логическую формулу от *входных переменных*.

Автомат получает события от так называемых *поставщиков событий* (в их роли могут выступать внешняя среда, интерфейс пользователя и т.д.) и генерирует выходные воздействия для *объекта управления*. При поступлении события автомат выполняет переход в соответствии с охранными условиями и значениями входных переменных. При выполнении перехода генерируются выходные воздействия, которыми он помечен, и автомат переходит в соответствующее состояние. Отметим, что состояния такого автомата не делятся на допускающие и недопускающие.

Формальное определение управляющего автомата дано в книге [1]. Пример управляющего автомата приведен на рис. 1.

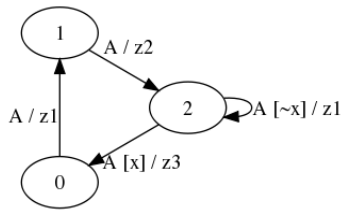

Рис. 1. Пример управляющего автомата.

Для данного автомата множество входных событий равно $\{A\}$, охранные условия зависят от единственной логической входной переменной x, множество выходных воздействий равно $\{z1, z2, z3\}$. Здесь и далее, состояние автомата с номером 0 будем считать начальным.

Далее, путем в автомате будем называть последовательность из n пар $<e_i, f_i>$ и $n+1$ состояния таких, что из i-го состояния пути (для $i = 1..n$) существует переход по вход-

ному событию e_i с охранным условием f_i в состояние с номером i+1. Два пути будем называть *непротиворечивыми*, если при этом последовательность выходных воздействий $A_1..A_n$ первого пути совпадает с последовательностью выходных воздействий $B_1..B_n$ второго пути.

В настоящей работе в качестве исходных данных для построения управляющего конечного автомата используется множество *сценариев работы*. Сценарием работы будем называть последовательность $T_1 \ldots T_n$ троек $T_i = <e_i, f_i, A_i>$, где e_i – входное событие, f_i – булева формула от входных переменных, задающая охранное условие, A_i – последовательность выходных воздействий. В дальнейшем тройки T_i будем называть *элементами сценария*.

Будем говорить, что автомат, находясь в состоянии *state*, *удовлетворяет элементу сценария* T_i, если из *state* существует переход, помеченный событием e_i, последовательностью выходных воздействий A_i и охранным условием, тождественно равным f_i как булева формула. Автомат *удовлетворяет сценарию работы* $T_1 \ldots T_n$, если он удовлетворяет каждому элементу данного сценария, находясь при этом в состояниях пути, образованного соответствующими переходами.

В настоящей работе применяется алгоритм EDSM для решения задачи построения управляющего конечного автомата по заданному множеству сценариев работы *Sc*, которым автомат должен удовлетворять.

Описание алгоритма EDSM. В работе [8] был предложен алгоритм построения автомата-распознавателя по обучающим примерам (набор слов, для каждого из которых известно, должно ли оно допускаться автоматом или не должно). В настоящей работе алгоритм EDSM применен к задаче построения управляющего автомата по сценариям работы. Опишем алгоритм EDSM, адаптированный для поставленной задачи.

На этапе инициализации строится начальный управляющий автомат, удовлетворяющий заданному набору сценариев. Затем на каждом шаге алгоритма рассматриваемый автомат обобщается с помощью слияния выбранной пары состояний. После слияния рассматриваемый управляющий автомат должен удовлетворять множеству сценариев работы *Sc*. Алгоритм заканчивает свою работу, когда на очередном шаге алгоритма не существует пары состояний, которые можно было бы слить без потери данного свойства.

Опишем алгоритмы построения начального автомата, слияния пары вершин автомата и выбора пары вершин для слияния на очередном шаге алгоритма (стратегии слияния).

Построение начального автомата. Изначально автомат состоит из одного состояния. Затем по очереди в автомат добавляются все пути, соответствующие сценариям из множества *Sc*.

Будем строить автомат, последовательно обрабатывая сценарии. Будем хранить на каждом шаге указатель на текущее состояние в автомате v и номер i первого необработанного элемента сценария.

В начале процесса добавления v указывает на начальное состояние автомата, а $i = 1$. На каждом шаге проверяется существование исходящего из состояния v перехода по событию e_i и с логической формулой, совпадающей с f_i как с булевой формулой. Если такой переход не существует, то в автомате создается новое состояние u, и в него из v направляется переход по входному воздействию e_i, с охранным условием f_i и с последовательностью выходных воздействий A_i. После этого u становится текущим состоянием, а значение i увеличивается на единицу.

Если такой переход существует, то производится сравнение последовательности A_i и последовательности выходных воздействий A', которой помечен рассматриваемый переход. Если $A_i = A'$, то текущим состоянием становится состояние, в которое ведет рассматриваемый переход, а значение i увеличивается на единицу.

Если же указанные последовательности не совпадают, то заданное множество сценариев *Sc* является противоречивым, поэто-

му работа алгоритма прерывается, и пользователю выводится соответствующее сообщение.

После завершения обработки всех сценариев производится проверка охранных условий. Для каждого состояния перебираются все пары исходящих из него переходов. Если существует такая пара переходов, что они помечены одним и тем же событием, а их охранные условия имеют общий выполняющий набор значений входных переменных, то множество сценариев предполагает недетерминированное поведение. Поэтому работа алгоритма прерывается, и пользователю выводится соответствующее сообщение.

Заметим, что получившийся автомат не будет иметь циклов. На рис. 2 приведен пример построенного начального автомата.

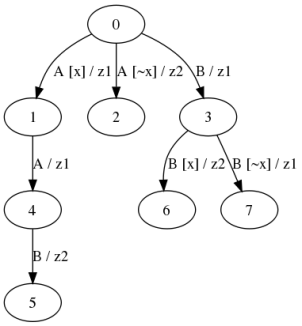

Рис. 2. Пример начального автомата.

Алгоритм слияния состояний. В предлагаемом алгоритме слияния происходят только между такими состояниями *s* и *t*, что из состояния *t* не ведет путей бесконечной длины. Слияние состояний *s* и *t* производится следующим образом. Каждый переход, ведущий в состояние *t*, перенаправляется в состояние *s*. После этого, для каждого перехода, ведущего из состояния *t*, ищется соответствующий переход из *s*, такой, что входные события и выходные воздействия совпадают, а охранные формулы на переходах совпадают как булевы формулы.

Если для рассматриваемого перехода из *t* такой переход, ведущий из *s*, не находится, то создается переход из вершины *s* в то же состояние, с теми же входными и выходными воздействиями и с тем же охранным условием, что и текущий переход из *t*.

Если для очередного перехода из *t* соответствующий переход из *s* находится, то алгоритм слияния запускается рекурсивно для состояний, в которые ведут рассматриваемые переходы из *s* и *t*.

Поскольку из состояния *t* не ведет путей бесконечной длины, алгоритм заканчивает свою работу за конечное время.

При слиянии некоторых вершин автомат может стать недетерминированным, либо перестать удовлетворять начальному множеству сценариев. Такие вершины в дальнейшем будем называть *несливаемыми*.

На рис. 3 приведен пример автомата, у которого состояния 0, 1 и 2 попарно не сливаемы из-за потери свойства детерминированности при слиянии, а состояния 3 и 4 являются сливаемыми.

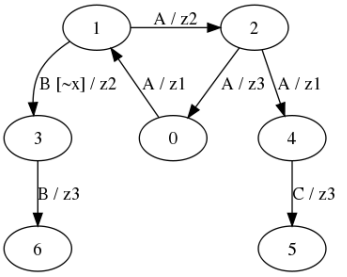

Рис. 3. Пример сливаемых и несливаемых состояний в автомате.

Стратегия слияния. В качестве стратегии слияния используется стратегия Blue-Fringe [8]. Данная стратегия была предложена на соревновании Abbadingo One, на котором алгоритм, использующий данную стратегию, занял первое место. Опишем данную стратегию.

На каждом шаге алгоритма состояния автомата разделены на три множества, каждому множеству условно соответствует цвет. Множеству необработанных состояний соответствует *белый* цвет. Множество *красных* состояний – попарно не сливаемые состояния, которые являются частью результирующего автомата. Все состояния, не являющиеся красными, в которые ведут переходы из красных состояний, являются *синими*. Пример автомата после нескольких шагов работы алгоритма приведен на рис. 4. На рисунке символом «R» помечены красные вершины, символом «B» – синие, а непомеченные вершины – белые.

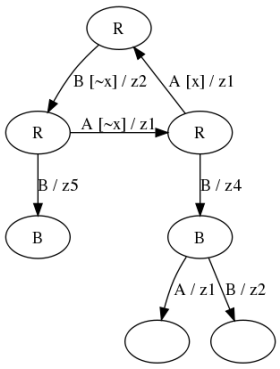

Рис. 4. Пример рассматриваемого автомата после нескольких шагов работы алгоритма.

На каждом шаге алгоритма все синие состояния, которые нельзя слить ни с одним из красных, перекрашиваются в красный цвет. Затем выбирается пара состояний красного и синего цвета с наибольшим значением *функции, оценивающей слияния*. После этого производится слияние выбранной пары состояний. В процессе работы алгоритма производится перекраска белых вершин, которые стали потомками красных, в синий цвет.

В предлагаемом алгоритме функция, оценивающая слияния, принимает на вход красное состояние r и синее состояние b и возвращает суммарное число различных состояний на непротиворечивых путях из r и b, если вершины можно слить. Если же вершины несливаемы, функция возвращает $-\infty$.

Результаты. Экспериментальное исследование проводилось на двух задачах. Первая задача – задача построения автомата управления часами с будильником [6]. Вторая задача – задача построения автомата управления банкоматом [9].

Для задачи построения автомата управления часами с будильником было задано 38 сценариев, аналогичных тестам, приведенным в работе [7]. Дерево сценариев, построенное на данном наборе сценариев, содержало 119 вершин. На основе этих сценариев был построен автомат, изоморфный автомату, построенному вручную в работе [6].

Для задачи построения автомата управления банкоматом было задано 162 сценария. Дерево сценариев, построенное на данном наборе сценариев, содержало 474 вершины. Построение автомата на компьютере с процессором Intel Core 2 Duo происходит за несколько секунд. В результате был получен автомат, состоящий из двух состояний, удовлетворяющий всем сценариям.

Заключение. В настоящей работе предложен метод построения управляющих конечных автоматов по сценариям работы. Этот метод основан на методе слияния состояний EDSM. Работоспособность метода проверена на задаче построения автомата управления часами с будильником и задаче построения автомата управления банкоматом. На этих задачах соответствующий управляющий автомат был построен корректно, а время работы алгоритма составляло меньше секунды на персональном компьютере с процессором Intel Core 2 Duo.

Литература

1. Поликарпова Н.И., Шалыто А.А., Автоматное программирование. СПб: Питер, 2009.
2. Angeline P.J., Pollack J. Evolutionary Module Acquisition // Proceedings of the Second Annual Conference on Evolutionary Programming. 1993.
3. Jefferson D., Collins R., Cooper C., Dyer M., Flowers M., Korf R., Taylor C., Wang A. The Genesys System. 1992.
4. Chambers L. Practical Handbook of Genetic Algorithms. Complex Coding Systems. Volume III. CRC Press, 1999.
5. Царев Ф.Н., Шалыто А.А. Гибридное управление беспилотными летательными объектами на основе автоматного программирования / 1-я Российская мультиконференция по проблемам управления. Сборник докладов четвертой научной конференции «Управление и информационные технологии». СПб ГЭТИ «ЛЭТИ». 2006, с. 138-144.
6. Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы. М.: Физматлит, 2006.
7. Царев Ф. Н. Метод построения управляющих конечных автоматов на основе тестовых примеров с помощью генетического программирования // Информационно-управляющие системы. 2010. № 5, с. 31–36.
8. Lang K., Pearlmutter B., Price R. Results of the abbadingo one dfa learning competition and a new evidence-driven state merging algorithm // Grammatical Inference / Ed. by V. Honavar, G. Slutz- ki. Springer Berlin / Heidelberg, 1998. Vol. 1433 of Lecture Notes in Computer Science. Pp. 1–12.
9. Первушин Е.В., Шалыто А.А. Моделирование банкомата. Проектная документация. СПб, 2003. http://is.ifmo.ru/download/bankomat.pdf.