Статья опубликована в сборнике докладов "Международной конференции по мягким вычислениям и измерениям" (SCM`2011). СПб ГЭТУ "ЛЭТИ". 2011. Т. 2, с. 69 - 75

ПРИМЕНЕНИЕ МЕТОДОВ РЕШЕНИЯ ЗАДАЧИ О ВЫПОЛНИМОСТИ БУЛЕВОЙ ФОРМУЛЫ ДЛЯ ПОСТРОЕНИЯ УПРАВЛЯЮЩИХ КОНЕЧНЫХ АВТОМАТОВ ПО СЦЕНАРИЯМ РАБОТЫ

В.И. Ульянцев, Ф.Н. Царев, А.А. Шалыто

Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Abstract. In the paper we describe the method of extended finite-state machine induction using SAT-solvers. Input data for the induction algorithm is a set of so called test scenarios. The algorithm consists of several steps: scenario tree construction, compatibility graph construction, Boolean formula construction, finite-state machine construction from satisfying assignment. These extended finite-state machines can be used in automata-based programming, in context which programs are designed using so called automated controlled objects. Each controlled object contains a finite-state machine and a controlled object. 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 все в более широком кругу задач начинает применяться автоматное программирование, в рамках которого поведение программ описывается с помощью детерминированных конечных автоматов [1].

Для многих задач автоматы удается строить эвристически, однако существуют задачи, для которых такое построение автоматов затруднительно. К задачам этого класса относятся, в частности, задачи об «Умном муравье» [2–4], об управлении моделью беспилотного летательного аппарата [5].

Для построения автоматов в задачах такого типа успешно применяются генетические алгоритмы [6], в том числе на основе обучающих примеров [7]. Недостатком генетических алгоритмов является то, что время их работы весьма велико и его достаточно трудно оценить аналитически.

Целью настоящей работы является разработка метода построения управляющего конечного автомата, лишенного указанных недостатков.

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 не допускающие.

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

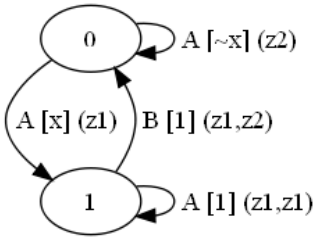

Рис. 1. Пример управляющего автомата

Для данного автомата множество входных событий равно {*A*, *B*}, охранные условия зависят от единственной логической входной переменной *x*, множество выходных воздействий равно {*z*1, *z*2}. Далее, состояние автомата с номером 0 будем считать начальным.

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 $T_i = \langle e_i, f_i, A_i \rangle$, 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* исходит 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 работе [8] разработан метод построения автомата-распознавателя по заданному набору слов. Данный метод основан на сведении поставленной задачи к задаче о выполнимости булевой формулы (boolean satisfiability problem, SAT) – по набору слов строится логическая формула. Выполняющая подстановка для нее ищется с помощью сторонней программы. Затем, на основании полученных значений переменных логической формулы строится искомый автомат-распознаватель. Результаты вычислительных экспериментов показывают более высокую скорость работы этого метода по сравнению с методами, основанными на объединении состояний [9, 10].

В настоящей работе решается задача построения управляющего конечного автомата с заданным числом состояний *C* по заданному множеству сценариев работы *Sc*, которым автомат должен удовлетворять.

Этапы предлагаемого метода. Построение управляющего автомата осуществляется в пять этапов:

1. Построение дерева сценариев.
2. Построение графа совместимости вершин дерева сценариев.
3. Построение булевой КНФ-формулы, задающей требования к раскраске построенного графа и выражающей непротиворечивость системы переходов результирующего автомата.
4. Запуск сторонней программы, решающей задачу о выполнимости булевой КНФ-формулы.
5. Построение автомата по найденному выполняющему набору значений переменных.

Построение дерева сценариев. *Деревом сценариев* назовем дерево, каждый переход которого помечен событием, булевой формулой и последовательностью выходных воздействий. Опишем алгоритм построения дерева сценариев по заданному множеству сценариев *Sc*.

Изначально дерево сценариев состоит из единственной вершины – корня дерева. Затем по очереди добавим в дерево все сценарии работы из *Sc*.

Для каждого из сценариев будем добавлять его элементы в дерево в порядке возрастания их номеров начиная с первого. При этом будем хранить указатель на текущую вершину дерева *v* и номер *i* первого необработанного элемента сценария.

В начале процесса добавления *v* указывает на корень дерева сценариев, а $i = 1$. На каждом шаге проверяется существование исходящего из вершины *v* ребра, помеченного событием e_i и логической формулой, совпадающей с f_i как

булевой функции. Если такое ребро не существует, то создается новая вершина дерева u, и в нее направляется ребро, помеченное тройкой $<e_i, f_i, A_i>$. После этого u становится текущей вершиной, а значение i увеличивается на единицу.

Если такое ребро существует, то производится сравнение последовательности A_i и последовательности выходных воздействий A', которой помечено рассматриваемое ребро. Если $A_i = A'$, то текущей становится вершина, в которую ведет рассматриваемое ребро дерева, а значение i увеличивается на единицу.

Если же указанные последовательности не совпадают, то заданное множество сценариев Sc является противоречивым, поэтому работа алгоритма прерывается, и пользователю выводится соответствующее сообщение.

После завершения добавления всех сценариев в дерево производится проверка охранных условий. Для каждой вершины перебираются все пары исходящих из нее ребер. Если существует такая пара ребер, 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.

На рис. 2 приведен пример дерева сценариев. Сценариям данного дерева удовлетворяет автомат, приведенный на рис. 1.

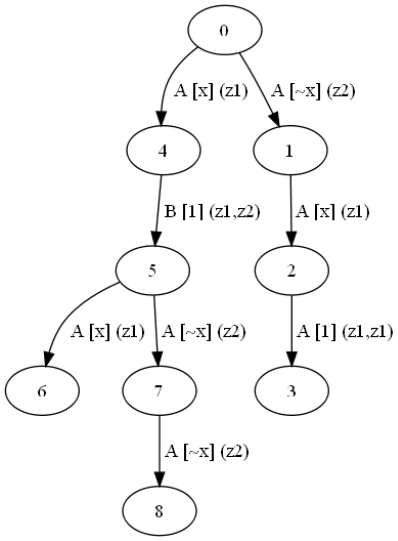

Рис. 2. Пример дерева сценариев

Построение графа совместимости вершин дерева сценариев. Для построения управляющего автомата необходимо «раскрасить» вершины дерева сценариев в заданное число цветов (равное числу состояний, которое задается в качестве одного из параметров алгоритма). При этом вершины одного цвета будут объединены в одно состояние результирующего автомата, а множество исходящих из состояния переходов будет строиться из объединения множеств ребер исходящих из вершин заданного цвета.

Для задания ограничений на раскраску построим так называемый *граф совместимости* вершин дерева сценариев. Множество вершин этого графа совпадает с множеством вершин дерева сценариев, поэтому в дальнейшем вершины графа и дерева различаться не будут. Ребра графа определяются следующим образом.

Вершины графа совместимости u и v соединены ребром (далее такие вершины будем называть *несовместимыми*), если существует последовательность пар $<e_1, values_1>\ldots<e_k, values_k>$ событий и наборов значений входных переменных, которая различает соответствующие вершины дерева. Будем говорить, что указанная последовательность *различает вершины* u и v, если выполняется совокупность следующих условий:

- из вершины u существует путь P_u, ребра которого помечены соответственно событиями $e_1 \ldots e_k$ и такими охранными условиями $f_1 \ldots f_k$, что наборы значений входных переменных $values_1 \ldots values_k$ являются, соответственно, их выполняющими подстановками;
- аналогичный путь P_v существует из вершины v;
- для последних ребер путей P_u и P_v верно хотя бы одно из двух условий:
  - пометки этих ребер различаются в части выходных воздействий;
  - у охранных условий этих ребер есть общий выполняющий набор значений входных переменных, но они не совпадают как булевы функции.

Опишем алгоритм построения графа совместимости. Напомним, что множество вершин этого графа совпадает с множеством вершин дерева сценариев. Основной идеей данного алгоритма является метод динамического программирования [11].

Для каждой вершины дерева сценариев v найдем все несовместимые с ней вершины. Обозначим как $S(v)$ множество вершин, несовместимых с v. Будем вычислять значения функции $S(v)$ начиная с листьев дерева сценариев. Для каждого из листьев u множество $S(u)$ пусто по определению несовместимых вершин.

Покажем, как вычислить значение $S(v)$, если оно уже вычислено для всех значений «детей» вершины v. Переберем все вершины дерева – вершина u входит в множество $S(v)$, если существует пара ребер ux (помечено событием e, формулой f_1 и последовательностью действий A_1) и vy (помечено также событием e, формулой f_2 и последовательностью действий A_2) такая, что выполняется одно из трех:

- формулы f_1 и f_2 имеют общий выполняющий набор значений входных переменных, но не совпадают, как булевы функции. Тогда $<e, values>$, где как $values$ обозначена выполняющая подстановка f_1, – последовательность, различающая u и v;
- формулы f_1 и f_2 совпадают, как булевы функции, а последовательности A_1 и A_2 не совпадают. Тогда вершины u и v различает такая же последовательность;
- формулы f_1 и f_2 совпадают, как булевы функции, и вершина x входит во множество $S(y)$, посчитанное заранее. Тогда существует последовательность $<e_1, values_1> \ldots <e_k, values_k>$, различающая вершины x и y, а вершины v и u различает последовательность $<e, values> <e_1, values_1> \ldots <e_k, values_k>$.

Время работы этого алгоритма составляет $O(n^2)$ (где за n обозначено число вершин в дереве сценариев), так как каждая пара ребер дерева сценариев в процессе работы алгоритма будет рассмотрена не более одного раза. При этом такое время работы достижимо, если заранее для каждой пары формул вычислено, равны ли они как булевы функции и имеют ли общий выполняющий набор значений входных переменных.

В худшем случае время работы этапа обработки формул составляет $O(2^{2m}n^2)$, где за m обозначено максимальное число входных переменных, использующихся в одном охранном условии. На практике число m не превышает четырех.

Построение булевой КНФ-формулы. Опишем алгоритм построения булевой КНФ-формулы, задающей требования к раскраске построенного графа и выражающей непротиворечивость системы переходов результирующего автомата. Данное построение аналогично построению КНФ-формулы, используемой в работе [8] для построения автомата-распознавателя.

Напомним, что в настоящей работе на вход алгоритму подается число C состояний результирующего автомата.

В данной формуле будут использоваться следующие логические переменные:

- $x_{v,i}$ (для каждой вершины дерева сценариев v и цвета вершины i – числа от

1 до C) – верно ли, что вершина v имеет цвет i. Напомним, что вершины одного цвета будут объединены в одно состояние результирующего автомата;

- $y_{a,b,e,f}$ (для каждой пары состояний результирующего автомата a и b, каждого события e, каждой формулы f, встречающейся в сценариях) – верно ли, что в результирующем автомате существует переход из состояния a в состояние b, помеченный событием e и формулой f.

КНФ-формула состоит из следующих дизъюнктов:

- $(x_{v,1} \vee \ldots \vee x_{v,C})$ (для каждой вершины v дерева сценариев) – накладываем условие существования цвета вершины v;
- $(\sim x_{v,i} \vee \sim x_{v,j})$ (для каждой вершины v дерева сценариев, цвета i и цвета j, где $i < j$) – накладываем условие, что вершина v не покрашена одновременно в цвета i и j;
- $(\sim x_{v,i} \vee \sim x_{u,i})$ (для каждой пары несовместимых вершин u и v дерева сценариев и цвета i) – накладываем ограничения на раскраску, задаваемые графом совместимости;
- $(\sim y_{a,b,e,f} \vee \sim y_{a,d,e,f})$ (для каждой тройки состояний результирующего автомата a, b и d ($b < d$), каждого события e, каждой формулы f) – из состояния a выходит не более одного ребра, помеченного событием e и формулой f;
- $(y_{a,b,e,f} \vee \sim x_{v,a} \vee \sim x_{u,b})$ (для каждого ребра vu дерева сценариев) – если выполняется:

  ─ ребро из вершины дерева v в вершину u помечено событием e и формулой f;
  ─ вершина v покрашена в цвет a;
  ─ вершина u покрашена в цвет b;

  то в результирующем автомате существует переход из состояния a в состояние b, помеченный событием e и формулой f;

- $(\sim y_{a,b,e,f} \vee \sim x_{v,a} \vee x_{u,b})$ (для каждого ребра vu дерева сценариев) – если выполняется:

  ─ ребро из вершины дерева v в вершину дерева u помечено событием e и формулой f;
  ─ вершина v покрашена в цвет a;
  ─ в результирующем автомате существует переход из состояния a в состояние b, помеченный событием e и формулой f;

  то вершина u покрашена в цвет b.

Запуск сторонней программы, решающей задачу о выполнимости булевой КНФ-формулы. Для того чтобы найти выполняющий набор для построенной КНФ-формулы, воспользуемся сторонней программой (SAT-solver), решающей задачу SAT.

В результате анализа результатов соревнования SAT-Race 2010 [12] был произведен выбор программы cryptominisat [13]. Эта программа признана победителем в указанном соревновании.

Подадим на вход выбранной программе построенную КНФ-формулу, записанную в формате DIAMAX (http://www.satlib.org/ubcsat/satformat.pdf). Если программа не обнаружила выполняющий набор значений переменных, то будем считать, что по данному набору сценариев невозможно построить управляющий автомат с заданным числом состояний.

Если же программа обнаружила выполняющий набор, то построим искомый автомат. Для этого на основании полученных значений переменных $x_{v,i}$ определим цвет каждой вершины дерева сценариев. На рис. 3 приведен пример раскраски дерева сценариев, приведенного на рис. 2.

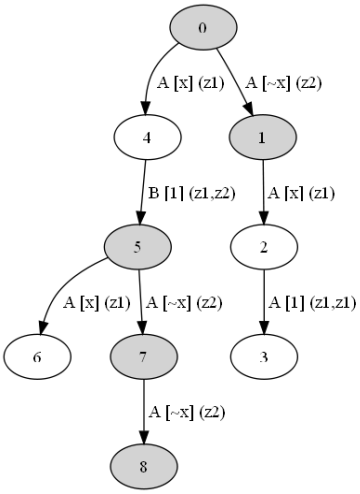

Рис. 3. Пример раскраски дерева сценариев в два цвета

После этого объединим все вершины одного цвета в одно состояние автомата, а начальным состоянием положим состояние, соответствующее цвету корня дерева сценариев. Множество исходящих из состояния переходов построим из объединения множеств ребер, исходящих из вершин заданного цвета.

Например, после объединения вершин дерева, приведенного на рис. 3 получим автомат, изображенный на рис. 4. Заметим, что данный автомат изоморфен автомату, приведенному на рис. 1. Отметим, что не исключено существование нескольких автоматов, удовлетворяющих сценариям множества *Sc*.

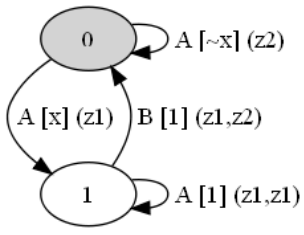

Рис. 4. Автомат, полученный после объединения вершин дерева

Экспериментальное исследование. Экспериментальное исследование проводилось на задаче построения автомата управления часами с будильником [1]. Было задано 38 сценариев, аналогичных тестам, приведенным в работе [7]. На основе этих сценариев был построен автомат, изоморфный автомату, построенному вручную в работе [1]. Его построение заняло менее секунды на персональном компьютере с процессором Intel Core 2 Quad Q9400, что позволяет говорить о достаточно высокой производительности разработанного метода.

Заключение. В настоящей работе предложен метод построения управляющих конечных автоматов по сценариям работы. Этот метод основан на сведении указанной задачи к задаче о выполнимости булевой формулы. Работоспособность метода проверена на задаче построения автомата управления часами с будильником. На этой задаче соответствующий управляющий автомат был построен корректно, а время работы алгоритма составляло меньше секунды на персональном компьютере с процессором Intel Core 2 Quad Q9400.

Исследование выполнено в рамках Федеральной целевой программы «Научные и научно-педагогические кадры инновационной России на 2009-2013 годы».

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. http://www.demo.cs.brandeis.edu/papers/ep93.pdf
3. Jefferson D., Collins R., Cooper C., Dyer M., Flowers M., Korf R., Taylor C., Wang A. The Genesys System. 1992. www.cs.ucla.edu/~dyer/Papers/AlifeTracker/Alife91Jefferson.html
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. Heule M., Verwer S. Exact DFA Identification Using SAT Solvers // Grammatical Inference: Theoretical Results and Applications 10th International Colloquium, ICGI 2010, pp. 66-79. Lecture Notes in Computer Science 6339, Springer.
9. Oncina J., Garcia P. Inferring regular languages in polynomial update time / Pattern Recognition and Image Analysis. Series in Machine Perception and Artificial Intelligence, vol. 1, pp. 49–61. World Scientific, Singapore (1992)
10. Oliveira A.L., Marques-Silva J.P. Efficient search techniques for the inference of minimum sized finite state machines // Proseeding of 5th Symposium on String Processing and Informarion Retrieval, pp. 81–89. 1998.
11. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы. Построение и анализ. М.: Вильямс. 2010.
12. Presentation of SAT-Race results at the SAT'10 conference http://baldur.iti.uka.de/sat-race-2010/downloads/SAT-Race-2010-Presentation.pdf
13. Soos M. CryptoMiniSat 2.5.0 http://baldur.iti.uka.de/sat-race-2010/downloads/SAT-Race-2010-Presentation.pdf