Опубликовано в материалах 2-й межвузовской научной конференции по проблемам информатики СПИСОК-2011, с. 351-355.

К. В. Егоров, А. А. Шалыто
Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Применение генетического программирования для построения автоматов управления системами со сложным поведением на основе контрактов и тестовых примеров

Введение

Автоматное программирование – это парадигма программирования, в рамках которой программы предлагается проектировать в виде совокупности взаимодействующих автоматизированных объектов управления [1]. В автоматных программах выделяют три типа объектов: поставщики событий, система управления и объекты управления. Система управления представляет собой конечный автомат или систему взаимодействующих конечных автоматов. Поставщики событий генерируют события, а система управления по каждому событию может совершать переход, считывая значения входных переменных у объектов управления для проверки условия перехода.

Существуют различные способы построения автоматов управления со сложным поведением. Чаще всего такие системы строятся эвристически, но они могут содержать ошибки и требуют дополнительных проверок. В работе [2] был предложен способ построения автоматов с помощью генетического программирования на основе

тестовых примеров. Однако такой вариант построения конечных автоматов требовал дополнительной валидации и верификации, а в случае обнаружения ошибки, необходимо было изменять тестовые примеры и заново строить систему. В любом случае, при таком подходе нельзя гарантировать поведение построенной системы на входных данных отличных от тестовых. В работе [3] было предложено использовать верификацию при вычислении функции приспособленности, однако вклад результата верификации был дискретным – 0 или 1. В работе [4] было предложено строить систему совместно на основе обучающих примеров и темпоральных формул. Формулы записываются на языке логики линейного времени (*Linear Temporal Logic, LTL*) и позволяют утверждать, что построенная система соответствует заявленной спецификации. Результат верификации учитывается при мутации, скрещивании и при вычислении функции приспособленности, причем вклад каждой темпоральной формулы – значение на отрезке [0, 1]. При этом 0 – формула нарушается сразу же в стартовом состоянии, а значение 1 – формула выполнятся.

Однако построение конечных автоматов на основе LTL-формул приводило к замедлению процесса построения. Кроме того, запись утверждений на языке LTL оказалась сложной для неподготовленных пользователей. Допустив ошибку в формуле и не заметив ее, процесс построения автомата может никогда не завершиться, и сложно выявлять такие ошибки.

Автоматное программирование по контрактам

В настоящей работе предлагается вместо или совместно с LTL-формулами использовать контракты [5, 6]. Обычно выделяют три вида контрактов: предусловия, постусловия и инварианты. В классическом понимании

«программирования по контрактам» в объектно-ориентированном программировании: *предусловие* – ожидание метода объекта на входные параметры и состояние класса при его вызове, *постусловие* – обязательства метода при его завершении, *инвариант* – условие выполняющиеся на протяжении всего времени жизни экземпляра класса. При автоматном подходе контракты могут накладываться как на состояния, так и на переходы и группы состояний.

При автоматическом построении автомата управления заранее не известно о состояниях конечного автомата, поэтому не представляется возможным использование контрактов для состояний или групп состояний. Определим контракты через LTL-формулы. Язык LTL состоит из пропозициональных переменных и стандартных булевых и специальных темпоральных операторов [7]. Для настоящей работы важны два из них:

- **X** (ne**X**t) – «*Xp*» – в следующий момент выполнено p;
- **G** (**G**lobally in the future) – «*Gp*» – всегда в будущем выполняется p.

Определим предусловие как $G(Xp_1 \rightarrow p_2)$ – если на следующем шаге выполнено p_1, то выполнено p_2; постусловие как $G(p_1 \rightarrow Xp_2)$ – если выполнено p_1, то на следующем шаге выполнено p_2; инвариант как $G(p_1 \rightarrow p_2)$ – если выполнено p_1, то выполнено p_2. Например, предусловие на переход можно определить как $G((!e \;\&\&\; Xe) \rightarrow p)$ или как $G(Xe \rightarrow p)$, постусловие как $G((e \;\&\&\; X!e) \rightarrow Xp)$ или как $G(e \rightarrow Xp)$, инвариант как $G(e \rightarrow p)$, где *e* – «переход по событию *e*», *p* – предикат.

По любой LTL-формуле можно построить автомат Бюхи. Алгоритм верификации основан на проверке пустоты языка пересечения допускаемого конечным автоматом модели и отрицанием LTL-формулы [7]. Можно показать, что для верификатора постусловие и предусловие

оказываются одинаковыми, так как их отрицание представляются «похожими» автоматами Бюхи. В принципе, контрактом можно назвать любую LTL-формулу, отрицание которой приведет к заранее заданной структуре автомата Бюхи. Под «заранее заданной структурой» будем понимать недетерминированный конечный автомат, который эквивалентен автомату контракта с точностью до пометок на переходах. На рисунке представлены автоматы для предусловия (а), постусловия (б) и инварианта (в) на переходы.

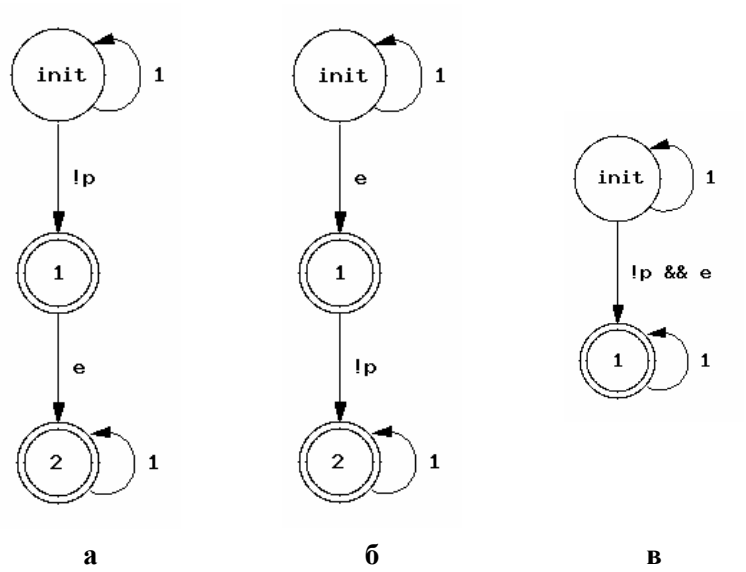

Рисунок. Автоматы Бюхи, построенные для отрицания LTL-формул
G(Xe → p) (а), G(e → Xp) (б), G(e → p) (в)

Применение контрактов для построения автоматов управления

При верификации произвольных темпоральных свойств, заранее не известна их семантика. Это приводит к

тому, что, обнаружив контрпример в автомате, невозможно определить, какой переход нарушает формулу. Когда конечный автомат строится на основе контрактов, точно известно, какие переходы в контрпримере нарушают его. Например, для инварианта последний переход нарушает его, а для предусловия и постусловия – последний и предпоследний. В результате такой априорной информации упрощается процесс модификации автомата с целью соблюдения контракта.

В настоящей работе предлагается генетический алгоритм построения конечных автоматов на основе тестовых примеров и контрактов. Особь в каждом поколении представляет собой конечный автомат, с событием на переходах и с указанием числа выходных воздействий [2]. Тестовые примеры позволяют расставлять выходные воздействия. Контракты позволяют проверять, что конечный автомат соответствует заявленной спецификации, и исключать из популяции особи, заранее ей не соответствующие, и модифицировать автомат так, чтобы контракт соблюдался. Для этого выполнимость контракта оценивается как значение на отрезке [0, 1], и полученный результат вносит вклад в функцию приспособленности. Переходы, нарушающие контракт, с большей вероятностью подвергаются мутации и/или не участвуют в скрещивании.

Для того, что бы вклад контракта в функцию приспособленности не был дискретным (0 или 1), предлагается в процессе верификации помечать те переходы, которые верны и соответствуют спецификации. Алгоритм верификации LTL-формул основан на двойном обходе в глубину [7], поэтому в тот момент, когда при первом обходе алгоритм покидает состояние, все исходящие переходы можно считать соответствующими спецификации. Таким образом, в качестве вклада

контракта в функцию приспособленности в простейшем случае можно взять отношение числа проверенных переходов к общему числу достижимых. Тем самым, чем больше число проверенных переходов, тем больше функция приспособленности.

Аналогично вычислению функции приспособленности, помеченные переходы можно учитывать при скрещивании двух особей: переходы, проверенные верификатором и на которых соблюдаются контракты, можно сохранить в новой особи без изменений. Таким образом, часть переходов в автомате уже будет соответствовать спецификации. Такое скрещивание позволяет сохранить ту часть автомата, на которой соблюдаются контракты, и приводит к росту функции приспособленности новой особи.

Мутация переходов может быть следующих типов: изменение входного воздействия, изменение числа выходных воздействий, изменение конечного состояния или удаление перехода. Как было отмечено выше, контракт нарушают два последних перехода в контрпримере, или же последний переход, в случае инварианта. Таким образом, увеличение вероятности мутации таких переходов, приводит к росту функции приспособленности.

Экспериментальные исследования

Были проведены экспериментальные исследования на примере автомата управления дверьми лифта из работы [4]. Было проведено 1000 построений для каждого из перечисленных выше методов. В процессе экспериментов популяция состояла из 2000 особей и использовалась стратегия элитизма, когда в новое поколение переходило 10% лучших особей. В качестве оцениваемого параметра использовалось число вычислений функции приспособленности.

При использовании только тестовых примеров построенный автомат менее чем в 1% случаев соответствовал спецификации, при использовании совместно верификации и тестовых примеров среднее число вычислений функции приспособленности оказалось равным – 827857. При совместном использовании тестов и контрактов – 710882.

Заключение

В работе исследована возможность применения контрактов для автоматического построения автоматов управления систем со сложным поведением, предложен метод мутации и скрещивания на основе выполнимости или не выполнимости контракта, проведено экспериментальное исследование метода при построении автомата управления дверьми лифта.

В результате проведенных экспериментов было показано, что применение контрактов приводит к ускорению построения автомата управления, и запись утверждений становится проще. Однако заметим, что контрактами не всегда можно полностью описать поведение системы, так что целесообразно использовать их совместно с LTL-формулами.

Литература

1. *Поликарпова Н. И., Шалыто А. А.* Автоматное программирование. СПб.: Питер, 2010.
2. *Царев Ф. Н.* Метод построения автоматов управления системами со сложным поведением на основе тестов с помощью генетического программирования / Материалы Международной научной конференции «Компьютерные науки и информационные технологии». Саратов: СГУ. 2009, с. 216 – 219.

3. *Johnson C.* Genetic Programming with Fitness based on Model Checking. Lecture Notes in Computer Science. Springer Berlin/Heidelberg. 2007. V. 4445, pp. 114 – 124.
4. *Егоров К. В., Царев Ф. Н., Шалыто А. А.* Применение генетического программирования для построения автоматов управления системами со сложным поведением на основе обучающих примеров и спецификации // Научно-технический вестник СПбГУ ИТМО. 2010. № 69, с. 81–85.
5. *Мейер Б.* Объектно-ориентированное конструирование программных систем. М.: Интернет-университет информационных технологий, 2005.
6. *Борисенко А., Федотов П., Степанов О., Шалыто А.* Разработка надежного программного обеспечения со сложным поведением / Сборник трудов конференции 5th Central and Eastern European Software Engineering Conference in Russia. М.: 2009, с. 125 – 128.
7. *Кларк Э., Грамберг О., Пелед Д.* Верификация моделей программ: Model Checking. М.: МЦНМО, 2002.