Метод синтеза тестов для программных реализаций телекоммуникационных протоколов на основе древовидных автоматов

[1] *М.С. Форостьянова <mariafors@mail.ru>*
[1] *Томский государственный университет,*
634050, Россия, г. Томск, пр-т Ленина, д. 36

Аннотация. В статье предложен подход к тестированию программных реализаций телекоммуникационных протоколов на основе древовидных автоматов. Эффективность предложенного подхода иллюстрируется на примере протокола TCP (Windows).

1. Введение

Гарантированная полнота тестирования разрабатываемого программного обеспечения возможна только при синтезе тестов с использованием некоторой формальной модели, т.е. некоторой спецификации программного продукта. Однако извлечение такой модели из требований к программному продукту является достаточно трудоемким. Соответственно, актуальным является вопрос синтеза качественных тестов для программного обеспечения без непосредственного построения формальной модели или на основе не слишком сложной модели, при условии, что тесты, построенные по этой модели, будут достаточно качественными.

В настоящей работе для синтеза тестовых последовательностей предлагается использовать древовидные автоматы. Эффективность предложенного метода иллюстрируется на примере телекоммуникационного протокола TCP (Windows).

Структура статьи следующая. В разделе 2 кратко описываются методы синтеза тестов для конечных и расширенных автоматов. Предлагается метод синтеза тестовых последовательностей для древовидного автомата, соответствующего тестируемой программной реализации. В разделе 3 приводятся результаты компьютерных экспериментов, иллюстрирующих на примере протокола TCP (Windows), что тесты, построенные предложенным методом, сравнимы по полноте с классическими конечно автоматными тестами. В заключении обсуждаются перспективы дальнейших исследований.

2. Методы синтеза тестов на основе моделей с конечным числом переходов

Методы тестирования на основе модели детерминированного конечного автомата включают в себя этап построения как можно более компактного набора входных (тестовых) последовательностей, позволяющих обнаружить любую программную реализацию из заданного класса, не эквивалентную спецификации, т.е. программную реализацию, которая имеет отличное от спецификации поведение на некоторой входной последовательности.

2.1 Методы синтеза тестов на основе конечного автомата

В данной работе под *автоматом* понимается детерминированный конечный инициальный *автомат S*, определяемый как пятерка $S = (S, I, O, T, s_0)$, где S – непустое конечное множество состояний с выделенным начальным состоянием s_0, I – входной алфавит, O – выходной алфавит, T – *отношение переходов* (отношение *поведения*), $T \subseteq S \times I \times O \times S$ [1].

Под *тестом* для конечного автомата S, как обычно, понимается конечное множество конечных входных последовательностей этого автомата. Тест, построенный на основе *обхода графа переходов* автомата, для каждого состояния s содержит входную последовательность α, переводящую автомат из начального состояния в состояние s, продолженную всеми допустимыми в состоянии s входными символами. Такой тест фактически «проходит» по каждому ребру графа переходов автомата и обнаруживает все выходные ошибки в реализации. Как показывают многочисленные эксперименты [2], тесты, построенные обходом графа переходов эталонного автомата, обнаруживают и ряд других несоответствий, в частности, при использовании таких тестов для тестирования протокольных реализаций.

Как отмечается в работе [3], основой теста с гарантированной полнотой, построенного практически любым методом на основе конечного автомата, является обход графа переходов автомата; кроме того, после «покрытия» каждого перехода соответствующая тестовая последовательность дополняется различающими последовательностями различными способами. В работе Василевского [4] предложенный им *W*-метод использует различающее множество *W*, которое существует для каждого минимального детерминированного полностью определенного автомата. Другие методы синтеза тестов [5] на основе конечного автомата используют семейство гармонизированных идентификаторов (HSI-метод), диагностическую последовательность (DS-метод) и т.д. Описание практически всех известных методов и сравнение длин тестов, построенных этими методами, можно найти в статье [6]. Длина таких тестов существенно зависит от числа переходов автомата, описывающего поведение эталонной системы, и основными проблемами при использовании конечно автоматных методов построения тестов являются сложность извлечения этой модели из требований к

программному продукту и ее громоздкость для достаточно сложных программ.

2.2 Методы синтеза тестов на основе расширенного автомата

Определение «конечный автомат» ограничено в том смысле, что оно «явно» не охватывает дополнительных структур, таких как выделение части информации о состоянии в отдельные переменные или введение параметров переходов и реакций, что позволяет сократить количество переходов в автомате. Автоматы, в которых помимо состояний и переходов есть конечное множество внутренних переменных, способных принимать различные значения, называются расширенными автоматами [7].

Под *расширенным автоматом* понимается пятёрка $M = (S, I, O, V, T)$, где S – непустое конечное множество состояний автомата, I – непустой входной алфавит, O – непустой выходной алфавит, V – конечное, возможно пустое множество контекстных переменных, T – множество переходов между состояниями из S. Каждый переход t из T есть семёрка (s, x, P, op, y, up, s'), где $s, s' \in S$ являются *начальным* и *конечным* состояниями перехода соответственно. Символ $i \in I$ есть входной символ и $D_{inp\text{-}i}$ обозначает множество векторов, компонентами которых являются значения параметров, соответствующих входному символу i (*входные* параметры). Соответственно, $o \in O$ – выходной символ и $D_{out\text{-}o}$ обозначает множество векторов, компонентами которых являются значения параметров, соответствующих выходному символу o (*выходные* параметры). Символами P, op и up обозначаются функции, определенные над входными параметрами и контекстными переменными из V:

- P: $D_{inp\text{-}i} \times D_V \rightarrow$ {Истина, Ложь} – предикат, где D_V - множество контекстных векторов, то есть векторов, компонентами которых являются значения контекстных переменных

- op: $D_{inp\text{-}i} \times D_V \rightarrow D_{out}o$ – функция вычисления выходного параметра

- up: $D_{inp\text{-}i} \times D_V \rightarrow D_V$ – функция вычисления значения контекстной переменной

Входной последовательностью расширенного автомата называется последовательность параметризированных входных символов, т.е. пар (i, ρ), $\rho \in D_{inp\text{-}i}$, и возможно, непараметризированных входных символов. *Конфигурацией* в расширенном автомате называется пара (s, σ), где σ – вектор значений контекстных переменных. Если тест, синтезированный по расширенному автомату, направлен на проверку передачи данных, то в качестве таких тестов часто используют тестовые последовательности, которые удовлетворяют критериям all-use [8], используемым при тестировании программного обеспечения. Последнее означает, что любой

простой путь в графе переходов данного расширенного автомата покрывается некоторой тестовой последовательностью. Основной проблемой при синтезе тестовых последовательностей на основе расширенного автомата является то, что необходимо использовать только так называемые *выполнимые* (executable) [9] входные последовательности. Для выполнимой входной последовательности в каждой конфигурации (s, σ), достижимой по начальному отрезку последовательности, следующий параметризированный входной символ (i, ρ) должен быть таким, что в расширенном автомате при переходе из состояния s под действием входного символа i соответствующий предикат будет истинным для пары (σ, ρ). Обычно задача построения таких последовательностей решается с использованием анализа достижимости [10]. Одним из методов синтеза теста с гарантированной полнотой на основе расширенного автомата является построение эквивалентного конечного автомата [11], который не всегда существует, если область определения некоторых контекстных переменных и/или входных параметров бесконечна. К сожалению, даже если для расширенного автомата существует эквивалентный конечный автомат, то для реальных систем такой конечный автомат является настолько большим, что построить такой автомат и/или проверяющие тесты с гарантированной полнотой по такому автомату практически невозможно. Поэтому широко используется частичное моделирование расширенного автомата с указанием допустимого числа конфигураций и/или длины входной последовательности. Тесты, построенные по конечному автомату с ограниченным числом конфигураций, покрывают широкий класс ошибок, однако в большинстве случаев полнота тестирования остаётся неизвестной. В данной работе мы рассматриваем так называемый l-эквивалент расширенного автомата, т.е. моделируем поведение расширенного автомата на входных последовательностях длины не больше l. Как иллюстрируется в работе [12], тесты, построенные по l-эквиваленту даже с небольшим значением l, обнаруживают достаточно много несоответствий в протокольных реализациях. Проверяющий тест, проходящий по каждому переходу расширенного автомата, строится на основе l-эквивалента, и в разделе 3 мы иллюстрируем, что в ряде случаев такой тест сравним по полноте с классическими тестами, построенными по конечному автомату, моделирующему исходный расширенный автомат.

2.3 Синтез тестов на основе древовидного автомата

Древовидные структуры активно используются для описания большого числа спецификаций из различных областей: например, протоколы, базы данных, индексация Yahoo!, Open Directory Project и т.д. Если эталонный расширенный автомат является древовидным автоматом, то последовательность, покрывающая каждую ветвь соответствующего дерева, является выполнимой, и тест строится как множество последовательностей, покрывающих все ветви соответствующего дерева. Если исходный расширенный автомат не является

древовидным, то строится его *l*-эквивалент. В нашей работе значение *l* выбиралось равным 8, хотя, как показывают результаты других экспериментов [11], достаточно качественные тесты получаются даже при *l* = 2. Предлагаемый в статье метод основан на покрытии всех переходов расширенного эталонного автомата, которые присутствуют в соответствующем *l*-эквиваленте, и добавлении такой критической последовательности в тестовое множество. Преимущество использования древовидной структуры заключается в том, что при построении критической последовательности для заданного перехода рассматривается не весь автомат, который может быть очень большим, а только одна из ветвей древовидной структуры.

Рассмотрим расширенный, возможно частичный автомат $S = (S, I, O, V, T, s_0)$, описывающий поведение эталонной системы. Если автомат не является древовидным автоматом, то зададим значение $l > 1$ для неявного построения его *l*-эквивалента. Под *критической* последовательностью для перехода $t = (s, i, P, op, o, up, s')$ расширенного автомата *S* мы понимаем некоторую входную последовательность α, которая позволяет покрыть этот переход в *l*-эквиваленте, для чего строится соответствующая ветвь *l*-эквивалента. После покрытия перехода (если возможно покрыть переход входной последовательностью γ длины не больше *l*), последовательность γ удлиняется до последовательности длины *l*, входящей в *l*-эквивалент, которая и включается в тестовое множество.

Выбор очередного входного символа, приписываемого к последовательности γ, может осуществляться с учетом древовидной структуры автомата или *l*-эквивалента. В частности, для выбора очередного перехода расширенного автомата, а затем для выбора следующего входного символа можно использовать различные методы обхода дерева [13], например, левый обход дерева в глубину.

При описанном выше подходе к покрытию переходов эталонного расширенного автомата довольно сложно оценить полноту построенных тестов. Однако, как показывают проведенные компьютерные эксперименты, описанные в следующем разделе, в ряде случаев полнота построенного теста оказывается достаточно высокой, в частности, полнота построенных тестов оказывается сравнимой с полнотой тестов, которые построены известными конечно автоматными методами.

3 Сравнение предложенного метода с методами синтеза тестов на основе конечных автоматов (экспериментальные результаты)

Для того чтобы оценить качество теста, построенного на основе покрытия переходов эталонного расширенного автомата с использованием *l*-эквивалента мы провели компьютерные эксперименты по тестированию программных

реализаций серверной части протокола TCP [14]. На основании имеющейся спецификации был построен расширенный автомат; по расширенному автомату посредством моделирования был построен конечный автомат, который получился частичным. В качестве программной реализации с открытым кодом мы использовали разработанную нами реализацию серверной части протокола TCP на языке Java.

Были построены наборы тестовых последовательностей описанным в предыдущем разделе методом (на основе *l*-эквивалента для $l = 8$) и различными методами по построенному конечному автомату. При помощи инструмента MuJava для автоматического внесения ошибок в программу на языке Java был сгенерирован набор всех возможных мутантов для тестируемой программной реализации. На следующем шаге осуществлялась проверка, какое количество мутантов можно обнаружить тестами, построенными различными методами, в том числе с учетом и без учета так называемых эквивалентных мутантов. Ниже представлена таблица, демонстрирующая полученные результаты.

Для автомата, представляющего работу серверной части протокола TCP, синтез тестов на основе конечного автомата осуществлялся в помощью пакета прикладных программ «FMSTest-1.0» [15]. В частности, для обхода графа переходов в конечном автомате потребовалось пять тестовых последовательностей. При использовании других конечно автоматных методов построенный автомат доопределялся до полностью определенного, проверяющий тест строился по полностью определенному автомату, и при достижении начальным отрезком тестовой последовательности состояния автомата, в котором следующий входной символ является неопределенным, тестовая последовательность на этом символе «обрезалась». Размеры построенных тестов следующие: для W- и Wp-методов тест содержит 53 и 55 тестовых последовательностей, для H- и HSI-методов - 76 и 89 тестовых последовательностей, соответственно. Такая разница в длине тестовых последовательностей напрямую связана с особенностями метода построения тестов по частичному автомату.

Табл. 1 Сравнение конечно-автоматных методов синтеза тестов

Метод	Кол-во всех мутантов	Кол-во мутантов, обнаруженных тестом	Кол-во эквивалентных мутантов	Кол-во обнаруженных мутантов/ кол-во всех мутантов (%)	Кол-во обнаруженных мутантов/кол-во всех мутантов минус кол-во эквив. (%)
Обход графа переходов	115	102	10	87%	97 %
W-метод	115	102	10	88,6%	97,1 %
Wp-метод	115	104	10	90,4 %	99 %

H-метод	115	106	10	88,6%	97,1 %
HSI-метод	115	103	10	89,6 %	98 %
Метод синтеза тестов на основе 8-эквивалента	121	104	11	85,9 %	94,5 %

Число тестовых последовательностей для серверной части протокола TCP, построенных по 8-эквиваленту, равно 23. Как видно из таблицы 1, после исключения эквивалентных мутантов полнота теста, построенного по 8-эквиваленту, для тестируемой программной реализации протокола TCP составила примерно 95 %, что практически совпадает с полнотой тестов, построенных классическими конечно автоматными методами.

4. Заключение

В статье предложен подход к тестированию программных реализаций на основе расширенных древовидных автоматов. Проведенные компьютерные эксперименты с протоколом передачи данных TCP показали, что полнота тестов, построенных предложенным методом на основе древовидного автомата, по своей полноте не уступает тестам, построенным классическими конечно автоматными методами. В дальнейшем предполагается включение в тест критических последовательностей, различающих *l*-эквиваленты эталонного расширенного автомата и соответствующего мутанта (мутационное тестирование [16]). Полнота построенных тестов будет оцениваться экспериментально для различных протокольных реализаций.

Автор выражает благодарность канд. физ.-мат. наук Кушик Н.Г. и профессору Евтушенко Н.В. за внимание к работе и полезные дискуссии.

Список литературы

[1]. А. Гилл, Введение в теорию конечных автоматов. М., Наука. 1966, 272 с.
[2]. А.В. Коломеец, Алгоритмы синтеза проверяющих тестов для управляющих систем на основе расширенных автоматов: дис. … канд. техн. наук. Томский гос. ун-т. Томск, 2010, 129 с.
[3]. T. S. Chow. Testing software design modelled by finite state machines. IEEE Transactions on Software Engineering. 1978, pp. 178-187.
[4]. М.П Василевский. О распознавании неисправностей автоматов. Кибернетика, 9(4). 1973, с. 93-108.
[5]. R. Dorofeeva, K. El-Fakih, S. Maag, A.R. Cavalli, N. Yevtushenko. Experimental evaluation of FSM-based testing methods. In Proc. of the IEEE International Conference on Software Engineering and Formal Methods (SEFM05). 2005, pp. 23-32.
[6]. K. El-Fakih, S. Prokopenko, N. Yevtushenko, G. Bochmann. Fault Diagnosis in Extended Finite State Machines. Lecture Notes in Computer Science, V. 2644. 2003, pp. 197-210.

[7]. А. В. Коломеец, С. А. Прокопенко. Метод синтеза диагностических тестов для расширенных конечных автоматов. Вестник ТГУ. Приложение №6.2003, с. 174-177.
[8]. El-Fakih, A. Kolomeez, S. Prokopenko, N. Yevtushenko. Extended Finite State Machine Based Test Derivation Driving By User Defined Faults. International Conference ICST. 2008, pp.308-317.
[9]. H.Ural. Test sequence selection based on static data flow analysis. Computer communications, V. 10, № 5. 1987, pp. 234 – 242.
[10]. H. Chen. Test sequence generation from the protocol data portion based on the selecting Chinese Postman algorithm. W.- Information Processing Letters, V. 65. 1998, pp. 261-268.
[11]. A. Cavalli, D. Lee, C. Rinderknecht, F. Zaidi. Hit-or-Jump: An algorithm for embedded testing with applications to IN services.. FORTE XII and PSTV XIX, China,1999, pp. 41-58.
[12]. N. Kushik, M. Forostyanova, S. Prokopenko, N. Yevtushenko. Studying the optimal height of the EFSM equivalent for testing telecommunication protocols. Proc. of the Second Intl. Conf. on Advances In Computing, Communication and Information Technology – CCIT. 2014, pp. 159-163.
[13]. И.Б. Бурдонов, А.С. Косачев, В.В. Кулямин Неизбыточные алгоритмы обхода ориентированных графов. Детерминированный случай. Программирование, № 5, 2003, с 59-69.
[14]. К. Хант. TCP/IP — Сетевое администрирование. Изд-во Cumbo. 2008, 816 с.
[15]. ППП. Свид-во о государственной регистрации программы для ЭВМ № 2014661807 "Программа синтеза тестов конечно-автоматными методами", Национальный исследовательский государственный университет.
[16]. A. J. Offutt, R. H. Untch. Mutation 2000: Uniting the Orthogonal. In Proceedings of the 1st Workshop on Mutation Analysis (MUTATION’00). Mutation Testing for the New Century. California, 2001, pp. 34–44.

Tree automata based test derivation method for telecommunication protocol implementations

[1] *M.S. Forostyanova <mariafors@mail.ru>*
[1] *Tomsk State University, 26*
Lenina av-u, Tomsk, 634050, Russia

Abstract. In this paper, an approach for testing software implementations of telecommunication protocols based on tree finite state machines (FSM) is proposed. The first step is the extraction of the specification Extended FSM from an informal protocol description. The next step is to derive a corresponding EFSM l-equivalent that is a tree FSM. Based on the set of considered faults corresponding sequences of the l-equivalent are included into a test suite. The proposed approach is illustrated by protocol TCP (Windows).

Keywords: telecommunication protocols, extended finite state machine, tcp, l-equivalent

References

[1]. A. Gill, Introduction to the Theory of Finite-state Machines. M. Science. 1966, 272 p.
[2]. A.V. Kolomeets, Algoritmy sinteza proveryayushchikh testov dlya upravlyayushchikh sistem na osnove rasshirennykh avtomatov: dis. ... kand. tekhn. nauk. Tomskii gosudrastvennyi universitet, [PhD dissertation, Tomsk state university] 2010, 129 s.
[3]. T. S. Chow. Testing software design modelled by finite state machines. IEEE Transactions on Software Engineering. 1978, pp. 178-187.
[4]. M.P. Vasilevskii. O raspoznovanii neispravnostei avtomatov. [On the recognition of failure] Kibernetika, [Cybernetic] 9(4). 1973, pp. 93-108.
[5]. R. Dorofeeva, K. El-Fakih, S. Maag, A.R. Cavalli, N. Yevtushenko. Experimental evaluation of FSM-based testing methods. In Proc. of the IEEE International Conference on Software Engineering and Formal Methods (SEFM05). 2005, pp. 23-32.
[6]. K. El-Fakih, S. Prokopenko, N. Yevtushenko, G. Bochmann. Fault Diagnosis in Extended Finite State Machines. Lecture Notes in Computer Science, V. 2644. 2003, pp. 197-210.
[7]. A.V. Kolomeets, S.A. Prokopenko. Metod sinteza diagnosticheskikh testov dlya rasshirennykh konechnykh avtomatov. [Extended finite state machine based test derivation strategies] Vestnik TGU. Prilozhenie #6. [Vestnik of the Tomsk State University, part #6] 2003, pp. 174-177.
[8]. K. El-Fakih, A. Kolomeez, S. Prokopenko, N. Yevtushenko. Extended Finite State Machine Based Test Derivation Driving By User Defined Faults. International Conference ICST. 2008, pp. 308-317.

[9]. H. Ural. Test sequence selection based on static data flow analysis. Computer communications, V. 10, № 5. 1987, pp. 234 – 242.
[10]. H. Chen. Test sequence generation from the protocol data portion based on the selecting Chinese Postman algorithm. Information Processing Letters, V. 65. 1998, pp. 261-268
[11]. A. Cavalli, D. Lee, C. Rinderknecht, F. Zaidi. Hit-or-Jump: An algorithm for embedded testing with applications to IN services. FORTE XII and PSTV XIX, China,1999, pp. 41-58.
[12]. N. Kushik, M. Forostyanova, S. Prokopenko, N. Yevtushenko. Studying the optimal height of the EFSM equivalent for testing telecommunication protocols. Proc. of the Second Intl. Conf. on Advances In Computing, Communication and Information Technology – CCIT. 2014, pp. 159-163.
[13]. I. Burdonov, A.S. Kosachev, V.V. Kuljamin. Neizbytochye algoritmy obxoda orientirovannyx grafov. Determinirovannyj sluchaj.[Irredundant algorithms for traversing oriented graphs. deterministic case], Trudy ISP RAN [The Proceedings of ISP RAS], 2003, pp. 59-69.
[14]. K. Hant. TCP/IP Network Administration. Cumbo. 2008, 816 p.
[15]. The application package. The certificate of state registration of the computer program №2014661807 "Programma sinteza testov konechno-avtomatnymi metodami" [The program of synthesis tests based on finite state machine], Tomsk State University
[16]. A. J. Offutt, R. H. Untch. Mutation 2000: Uniting the Orthogonal. In Proceedings of the 1st Workshop on Mutation Analysis (MUTATION'00). Mutation Testing for the New Century. California, 2001, pp. 34–44.