ИСП Российская Академия наук
Институт Системного Программирования

ISSN 2079-8156 (Print)
ISSN 2220-6426 (Online)

Труды Института Системного Программирования РАН

Proceedings of the Institute for System Programming of the RAS

Том (Volume) 26

выпуск (issue) 6

Москва 2014

ИСП Российская Академия наук
Институт Системного Программирования

Труды Института Системного Программирования РАН

Proceedings of the Institute for System Programming of the RAS

Том (Volume) 26
Выпуск (issue) 6

Под редакцией
академика РАН В.П. Иванникова
Edited by
Academician V.P. Ivannikov

Москва 2014

УДК004.45

Труды Института системного программирования: Том 26, выпуск 6. /Под ред. Академика РАН В.П. Иванникова/ – М.: ИСП РАН, 2014.

Proceedings of the Institute for System Programming: Volume 26, issue 6. /Edited by Academician V.P. Ivannikov/ – M.: ISP RAS, 2014.

Шестой выпуск 26-го тома Трудов ИСП РАН содержит статьи, подготовленные по материалам, которые были представлены на совместном российско-французском семинаре, посвященном вопросам верификации, тестирования и оценки качества программного обеспечения.

Sixth issue of the Proceedings of the Institute for System Programming of the RAS consists of papers presented at the 1st Franco-Russian seminar 'Software Verification, Testing, and Quality Estimation'.

ISSN 2079-8156 (Print)
ISSN 2220-6426 (Online)

Содержание

П р е д и с л о в и е

24-25 ноября 2014 года в Париже в Институте Мин-Телеком (Institut Mines-Télécom) состоялся совместный российско-французский семинар, посвященный вопросам верификации, тестирования и оценки качества программного обеспечения (Software Verification, Testing, and Quality Estimation). Семинар был приурочен к открытию совместной исследовательской лаборатории по проблемам оценки качества в открытом веб-пространстве, которую организовали Института Мин-Телеком (Франция), Институт системного программирования РАН и Томский государственный университет. Одним из направлений исследований лаборатории будет оценка качества веб-приложений – бурного растущего сектора программных приложений. Ключевое слово этого направления Quality, послужило основой для названия совместной лаборатории – Qualipso.[1]

В открытии семинара приняли участие Д. Садек, директор по исследованиям Института Мин-Телеком, академик РАН В.П. Иванников, директор ИСП РАН и проректор ТГУ Д.В. Сухушин. Присутствующие официальные представители организаций-участников прокомментировали открытие совместной лаборатории.

Д. Садек, директор по исследованиям Института Мин-Телеком:

- Подписание настоящего соглашения в рамках развития международного сотрудничества позволяет нам консолидировать существующее совместное сотрудничество в образовательных и научно-исследовательских сферах, таким образом способствуя продвижению и усилению статуса Института Мин-Телеком в мире.

В.П. Иванников, директор Института системного программирования РАН:

- В настоящее время формальные методы активно используются на практике при разработке программного обеспечения. Исследователи Франции и России имеют богатый опыт в изучении формальных моделей и языков. Консолидация отношений между этими странами и учреждениями призвана стимулировать новые идеи для совместного франко-российского сотрудничества.

Д.В. Сухушин, проректор по Программам развития Томского государственного университета:

- Открытие лаборатории является итогом наших многолетних отношений и свидетельством взаимного доверия одновременно. Формат международной

[1] Проведение семинара и подготовка публикаций сотрудников и обучающихся ТГУ частично поддержаны проектом 739 госзадания МинОбрНауки РФ

сетевой лаборатории открывает для участников новые возможности по продвижению исследований в международном исследовательском сообществе. Интегрируя наши исследования, мы приобретаем конкурентное преимущество в стремлении к лидерству в приоритетном направлении исследований.

Первый доклад на семинаре сделал профессор А.К. Петренко. В своем докладе он рассмотрел современное состояние дел в области верификации операционных систем и представил стек инструментов и технологий ИСП РАН, при помощи которых ведется верификация различных операционных систем, начиная от микроядерных ОС реального времени и заканчивая модулями ядра ОС Linux и дистрибутив ОС Linux, включающих в себя до миллиона интерфейсов. Были представлены возможности различных техник верификации в контексте их применения к микроядрам размером около 10 тысяч строк на языке программирования, к модулям размером до 100 тысяч строк, к библиотекам в диапазоне размеров от тысячи до миллиона интерфейсов.

Профессор Ана Кавалли в своем докладе представила работы департамента вычислительных сетей университета Телеком Южный Париж в направлении тестирования и мониторинга веб-систем. Эта научная группа много лет работает в области использования формальных моделей для тестирования и оценки качества программного обеспечения и имеет большое количество интересных теоретических результатов и разработанных приложений. Одно из таких приложений, MMT, созданное совместно с предприятием Монтимаж (Франция), используется в различных компаниях для мониторинга безопасности компьютерных систем.

Профессор Н.В. Евтушенко представила сообщение на тему «Nondeterministic FSMs in Model-Based Testing», в котором было отмечено, что в настоящее время для описания дискретных систем все чаще используются модели с недетерминированным поведением. Недетерминизм появляется в силу различных причин, таких как опциональность в спецификациях, уровень абстракции, ограниченные управляемость и наблюдаемость и др. В докладе были представлены различные модели неисправности на основе недетерминированных автоматов для синтеза проверяющих тестов с гарантированной полнотой относительно перечисленных моделей.

В докладе Д. Ривера, Н. Кушик, К. Фьюензалида, А. Кавалли, Н. Евтушенко «Расширенные конечные автоматы с таймаутами для оценки удовлетворенности пользователя OTT сервисом» авторы предлагают использовать временной расширенный автомат для оценки качества предоставляемого медиа сервиса. Используя эту модель, провайдер может найти «узкие места» в правилах распространения сервиса, а именно, в бизнес модели, и соответственно повысить свою прибыль.

Стефан Мааг в своем докладе «Тестирование MANETs на основе формальной модели» отметил, что тестирование реализаций на соответствие спецификации (тестирование конформности) является одним из необходимых этапов при разработке программного обеспечения в системах передачи информации. Применение формальных моделей позволяет автоматизировать процесс генерации тестовых последовательностей, которые будут использоваться при тестировании реальных реализаций. Однако несмотря на большое разнообразие методов в этой области, остается много нерешенных вопросов при использовании формальных моделей для тестирования протоколов маршрутизации мобильной связи (беспроводные самоорганизующиеся сети). Для понижения числа неопределенных вердиктов при пассивном тестировании, которые встречаются достаточно часто, Стефан Мааг предлагает использовать технологию, основанную на самоподобии вершин сети.

Доклады С. Прокопенко «Поиск неисправной компоненты в сети расширенных автоматов» и М. Форостьяновой «Метод синтеза тестов для программных реализаций телекоммуникационных протоколов на основе древовидных автоматов» посвящены возможности использования *l*-эквивалентов при построении проверяющих и диагностических тестов по модели расширенного автомата. Иными словами, для расширенного автомата строится древовидный конечный автомат, поведение которого совпадает с поведением исходного расширенного автомата на всех последовательностях длины не больше *l*. В докладе С. Прокопенко *l*-эквивалент строится для композиций мутационных расширенных автоматов, каждая из которых соответствует неисправностям в определенной компоненте. Построив диагностический эксперимент для таких *l*-эквивалентов, по реакции композиции, предъявленной для тестирования, достаточно часто можно определить, какая из компонент является неисправной, при условии, что неисправности возможны только в одной компоненте. В докладе М. Форостьяновой предлагается метод построения проверяющего теста на основе *l*-эквивалента эталонного расширенного автомата. Компьютерные эксперименты, проведенные с JAVA реализацией протокола TCP (Windows), показали, что полнота тестов для 8-эквивалента относительно мутаций mJAVA, практически совпадает с полнотой тестов, построенных известными конечно автоматными методами, которые, как известно, доставляют тесты высокого качества для протокольных реализаций.

На семинаре были представлены две тесно связанных между собой статьи И. Б. Бурдонова и А. С. Косачева «Построение прямого и обратного остова коллективом автоматов» и И. Б. Бурдонова, А. С. Косачева и В. В. Кулямина «Параллельные вычисления на графе с помощью обмена сообщениями» , которые посвящены распределенным вычислениям на ориентированных графах на основе обмена сообщениями вдоль дуг между автоматами, находящимися в вершинах графа. Первая статья описывает

распределенный алгоритм извлечения структуры заранее неизвестного ориентированного графа в виде остовов для распространения сообщений от корневой вершины графа к любой другой его вершине и обратно. Во второй статье авторы предлагают алгоритм распределенного вычисления произвольной функции от мультимножества значений, которыми размечены вершины графа, использующий построенные первым алгоритмом остовы для эффективного обмена сообщениями.

В докладе Н.Г. Кушик были приведены результаты по способам понижения сложности задач, возникающих при верификации, тестировании и оценке качества программного обеспечения на основе конечных автоматов. Для понижения сложности использовались эвристики, масштабируемые представления и решение задач для определенных классов автоматов и полуавтоматов. Во второй части доклада были представлены результаты автора по понижению сложности задач, возникающих при оценке качества прикладного программного обеспечения, полученные совместно с Н. Евтушенко, Д.Парра, К. Фуензалида, Д. Ривера, Ж. Покхрел, В. Маллули.

В настоящее время активно развиваются методы синтеза тестов с гарантированной полнотой для систем, для которых важны временные аспекты, и доклады Александра Твардовский «К минимизации автоматов с таймаутами», (и М. Громова, Н. Евтушенко «К минимизации числа временных переменных во временных полуавтомата» и Г. Кидяровой «О сравнении скорости функционирования двух временных автоматов» были посвящены анализу автоматов и полуавтоматов с временными ограничениями. В докладе А. Твардовского предложен метод минимизации автоматов с входными и выходными таймаутами. Если в некотором состоянии автомата входной символ не поступает в течение определенного времени (входного таймаута), то автомат может спонтанно перейти в другое предписанное состояние. Выходной таймаут определяет время выполнения перехода, т.е. задержку выдачи выходного символа относительно поданного входного символа. Предложенный метод доставляет приведенный автомат, т.е. автомат, в котором никакие два состояния не эквивалентны, эквивалентный исходному автомату с таймаутами. Именно для приведенных временных автоматов существуют методы синтеза тестов с гарантированной полнотой. Показывается, что в отличие от классических автоматов, автомат с таймаутами может иметь несколько приведенных форм, которые попарно не изоморфны. В докладе М. Громова были предложены достаточные условия для уменьшения числа временных переменных во временном полуавтомате без изменения его поведения. В докладе Г. Кидяровой было предложено масштабируемое представление для сравнения скорости функционирования двух временных автоматов.

В докладе О.В. Кондратьевой, А. Кавалли и Н. Евтушенко «Решение автоматных уравнений для временных автоматов относительно параллельной

композиции» рассматривается параллельная композиция автоматов с таймаутами. Предлагается способ решения уравнения для такой композиции, и, в частности, формула для наибольшего (общего) решения. Отмечается, что для нахождения общего решения уравнения для автоматов с таймаутами понятие такого автомата необходимо расширить, допустив множество выходных таймаутов на переходах автомата. На основе решения уравнения можно частично управлять качеством синтезируемой компоненты в композиции веб сервисов, в частности, оптимизировать компоненту по времени обработки информации.

В докладе В. Х. Ла и А. Кавалли «Исследование отказоустойчивой маршрутизации в беспроводных сетях» отмечается, что в настоящий момент активно развиваются мобильные сенсорные сети. Экономия ресурсов в таких сетях ограничивает возможности безопасного проектирования и использование соответствующих, устойчивых к атакам, протоколов. С другой стороны, поскольку сенсорные датчики функционируют независимо и в достаточно сложных условиях, такие сети особенно склонны к отказам и уязвимы к преднамеренным атакам. Для надежного и безопасного функционирования таких сетей, необходимо исследовать и развивать устойчивость к потенциальным вторжениям. В докладе рассматриваются два таких устойчивых протокола маршрутизации, а именно, INSENS and ITSRP, анализируются их свойства. Моделирование и анализ быстродействия показали, что оба протокола достаточно хороши с практической точки зрения. В докладе К. Тоуми, М. Ауди, А. Кавалли «Моделирование и тестирование безопасности при взаимодействии сервисов» авторы рассмотрели известные подходы к моделированию и тестированию безопасности взаимодействия веб сервисов с использованием формальных моделей. Были предложены оценки различных технологий активного и пассивного тестирования. Полученные результаты были продемонстрированы для реальной системы с использованием методов пассивного тестирования.

В большинстве методов построения проверяющих тестов по автоматной модели рассматривается инициальный конечный автомат, и соответственно проверяющие тесты строятся как набор входных последовательностей, перед каждой из которых подается сигнал СБРОС, переводящий автомат в начальное состояние. Если последний является достаточно дорогим, то имеет смысл попытаться построить одну последовательность, которая позволяет выяснить, является ли проверяемая реализация конформной спецификации. В докладе А. Ермакова «Построение проверяющих последовательностей для недетерминированных автоматов» предлагается алгоритм синтеза адаптивной проверяющей последовательности, при условии, что исходный автомат обладает разделяющей последовательностью и детерминированным сильно связным подавтоматом, что гарантирует, что только автомат, изоморфный некоторому подавтомату спецификации с тем же числом состояний, может быть редукцией автомата-спецификации. Кроме того, предполагается, что

процесс тестирования является адаптивным, т.е. следующий входной символ определяется по выходной реакции тестируемой реализации на предыдущие входные символы. За счет адаптивного подхода к тестированию длина теста может оказаться более короткой, чем длина тестовых последовательностей, построенных заранее до начала процесса тестирования. Кроме того, отмечается, что, поскольку проверяющая последовательность подается адаптивно, условия для автомата-спецификации могут быть ослаблены; в частности, вместо детерминировано достижимых состояний можно использовать определенно-достижимые состояния и соответствующие преамбулы, а вместо разделяющей последовательности - различающий тестовый пример, если таковые существуют в автомате-спецификации.

В докладе Х. Лопега, С. Маага, Ж. Моралеса «Масштабируемый метод оценки для систем управления доверием на основе распределенных систем онлайн мониторинга» рассматриваются многокомпонентные (композитные) системы, которые становятся все более популярными, и требования к их взаимодействию постоянно повышаются. Соответственно, в таких системах можно разрешить выполнение только тех действий, которым можно доверять; причем решение, с кем и на каких условиях взаимодействовать, принимает каждая (под) система самостоятельно. В докладе авторы сфокусировались на вердиктах доверия, которые «выносятся» на основе поведения агентов и распределенного он-лайн мониторинга сети. Это дает возможность обеспечить управление доверием в системе с точки зрения опыта конкретной подсистемы, при условии, что анализируется так называемое «мягкое доверие». Предлагается масштабируемый метод для он-лайн оценки/анализа доверия с использованием дополнительной модели расширенного полуавтомата и методы понижения оценки времени на основе известных результатов.

В заключение семинара были обсуждены планы мероприятий, которые будут проводиться в рамках работ совместной исследовательской лаборатории Qualipso. Данный сборник содержит статьи, которые были написаны по результатам большинства представленных на семинаре докладов.

Заведующая кафедрой Информационных технологий
в исследовании дискретных структур ТГУ,
заведующая лабораторией Компьютерных наук,
профессор Н.В.Евтушенко

Заведующая отделом Сетевого программирования
института Телеком, Южный Париж, профессор Ана Р. Кавалли

Заведующий отделом Технологий
программирования ИСП РАН,
профессор А.К.Петренко

P r e f a c e

The 1st Franco-Russian seminar 'Software Verification, Testing, and Quality Estimation' has been held in Institut Mines-Télécom, Paris, France, on November, 24-25, 2014. Together with the seminar another important event took place: the joint Franco-Russian Research Laboratory that will focus on the use of formal methods for dealing with issues concerning service validity and reliability. The laboratory members are the Institut Mines-Télécom, France, the Institute for System Programming, of the Russian Academy of Sciences, Moscow, Russia, and the Tomsk State University, Tomsk, Russia. One of the laboratory research directions is the quality estimation of web applications and services that nowadays becomes one of the main areas of software development. The key word here is 'quality' and thus, the title of laboratory **Qualipso** has been established.[1]

The seminar has been opened by the key persons of the organizing institutions, namely, David Sadek, Research Director of Institut Mines-Télécom, Viktor P. Ivannikov, Director of the Institute for System Programming of Russian Academy of Sciences, and Dmitriy V. Sukhushin, Vice-Rector for Strategic Development of Tomsk State University.

According to David Sadek, Director for Research of Institut Mines-Télécom, "this kind of action, that comes within our international partnerships policy, allows us to consolidate our existing cooperation with flagship teams and institutions in our own research and education areas, thereby contributing to the worldwide influence of Institut Mines-Télécom."

For Viktor P. Ivannikov, Director of the Institute for System Programming of Russian Academy of Sciences, "On the one hand, formal methods have begun to be really used in Software Engineering practically; on the other hand, France and Russia have deep traditions on different formal models and languages. This gives hopes in new innovations for our French –Russian joint cooperation".

For Dmitriy V. Sukhushin, Vice-Rector for Strategic Development of Tomsk State University, "The opening of the joint laboratory is the result of our long collaboration and mutual trust. This international joint laboratory provides new opportunities for its members encouraging the integration of the research in the international scientific society. We believe that the integration of our studies will contribute to our mutual competitive advantage aiming at being leaders in the priority research areas."

The first presentation was made by Professor Alexander K. Petrenko (Institute for System Programming). The speaker discussed the state of the art of the verification of operating systems (OS) and presented the variety of instruments and technologies developed at the Institute for System Programming which are used for the

[1] The seminar and publications of TSU teachers and students are partially supported by project 739 Goszadanija MinObrNauki RF

verification of various operating systems starting from microkernel real time OS up to core modules of Linux and its distributives that contain millions of interfaces. The facilities of a number of verification techniques have been demonstrated in the context of their use for microkernels which have more than 10000 lines in programming languages, for modules with up to 100000 lines, for libraries having from 1000 up to one million interfaces.

Professor Ana Cavalli (Institut Mines-Télécom, Institute Télécom SudParis) presented a number of methods and techniques for web systems testing and monitoring developed at the department of Network Software. This scientific group has been working in the area of using formal models for testing and verification for many years, they have developed many novel methods and applications and it is recognized throughout the world for their scientific results. One of applications, MMT, developed together with the Montimage enterprise is now used in many companies for monitoring the safety of web systems.

Professor Nina Yevtushenko (Tomsk State University) presented the lecture 'Nondeterministic FSMs in Model-Based Testing' underlining that this time formal models with nondeterministic behavior are widely used. The nondeterminism appears due to the following reasons, such as the specification optionality, abstraction level, limited controllability or/and observability, etc. The presentation contained a number of fault models developed for nondeterministic Finite State Machines (FSMs) and some methods for deriving complete test suites w.r.t. these models.

In the presentation «A TEFSM-based Framework for QoE Evaluation of OTT Services» by Diego Rivera (the paper authored by D. Rivera, N. Kushik, C. Fuenzalida, A. Cavalli, N. Yevtushenko), a new approach has been presented for modelling multimedia OTT services, integrating not only the functional aspects of the service but also non-functional variables, which are classified into objective, subjective and business-related. The functional requirements are modeled using a Timed Extended Finite State Machine (TEFSM), which is augmented with context variables representing non-functional requirements related to the quality metrics QoS, QoE, and QoBiz. Using the proposed model a provider can detect bottlenecks for a developed service in order to increase the revenue.

In his presentation, «Model-Based Testing for MANETs», professor Stéphane Maag (Institut Mines-Télécom, Institute Télécom SudParis,) noted that the conformance testing in network engineering is a crucial phase in the development of complex communicating systems. Whilst many model-based techniques have been developed for testing, their application to test wireless routing ad-hoc protocols still raises many issues. Special attention has been given to presenting the node self-similarity reducing the number of inconclusive verdicts often met in traditional model based testing.

The presentations by Svetlana Prokopenko "Locating a faulty component of an EFSM composition" and Maria Forotyanova "Test derivation based on tree FSMs and tree automata" (Tomsk State University) have been devoted to deriving tests for

an Extended FSM (EFSM) by the use of their *l*-equivalents. Given an Extended FSM, a tree FSM is derived such that its behavior coincides with the EFSM behavior for each input sequence of length up to *l*. Svetlana Prokopenko proposed to derive such *l*-equivalents for mutation EFSMs of the EFSM composition under the assumption that only one component EFSM can be faulty. If there exists a distinguishing (adaptive) input sequence for the derived *l*-equivalents then based on the response of a composition under test to this input sequence a faulty component can be located. Maria Forostyanova proposed a method for deriving a test suite for an EFSM based on its *l*-equivalent and presented the experimental results for TCP implementation (Windows). A Java implementation of this protocol has been mutated by the tool mJava and the fault coverage of a number of test suites has been determined w.r.t. the set of obtained mutants. The experiments clearly show that the fault coverage of a test suite derived based of a 8-equivalent of the corresponding EFSM almost coincides with that of test suites returned using the known FSM methods which are widely used for deriving high quality tests for protocol implementations.

Two close related presentations of I. Burdonov, A. Kossatchev "Building direct and back spanning trees by automata on a graph» and I. Burdonov, A. Kossatchev, V. Kuliamin "Parallel calculations by automata on direct and back spanning tees of a graph" (Institute for System Programming) addressed distributed computations for oriented graphs based on the message exchange between automata which are situated in the graph nodes. The proposed algorithm builds two spanning trees of the graph: *the direct spanning tree*, which has the root node as the tree root, and *the back spanning tree*, directed to the root. The complexity of the algorithm is evaluated. In the second part, a parallel computation algorithm for calculating the arbitrary function value on a multiset of values distributed on oriented graph nodes has been presented. The key idea of the algorithm is to use structural information on the graph that can be extracted by its parallel exploration and encoded into structures of direct and back spanning trees of the graph, which were constructed by the algorithm mentioned in the former part of the presentation.

The presentation of Natalia Kushik addressed the complexity reducing when solving problems of test derivation and quality evaluation. For this purpose, different ways are used such as proper heuristics, scalable representations and solving techniques for proper automata and FSM classes. In the second part of the presentation, some results of the complexity reducing when evaluating the software quality have been mentioned. These results have been obtained in the collaboration with N. Yevtushenko, A. Cavalli, J. Parra, C Fuenzalida, D. Rivera, J. Pokherel, W. Mallouli.

Nowadays a lot of attention is paid for deriving high quality tests for systems for which time aspects are important and there were three presentations on this subject. M. Gromov proposed some sufficient conditions for reducing the number of time variables in finite automaton along with preserving its behavior while G. Kidjarova proposed a scalable representation for comparing the performance of two timed

FSMs. The presentation of Alexandre Tvardovskiy ‘On the minimization of timed Finite State Machines’ addressed the problem of minimizing an FSM augmented with input and output timeouts, since almost all methods for deriving complete test suites are developed for reduced (minimal) timed machines, i.e., FSMs where every two states are not equivalent. If at some state no input is applied until the corresponding (input) timeout expires then the FSM can spontaneously move to another prescribed state. An output timeout describes the time that is necessary for executing a transition that is the number of time instances needed for producing an output after an input has been applied. The author proposed a technique for minimizing such machines and has shown that differently from classical FSMs, an FSM with timeouts can have several minimal forms which can be not pair-wise isomorphic.

The presentation of O. Konratyeva “Web Service Composition Quality Management with Timed Finite State Machines” (the paper O. Kondratyeva, N. Yevtushenko, A. Cavalli ”Solving parallel equations for FSMs with timeouts”) was devoted to using compositions of FSMs with timeouts for optimizing web services’ compositions. The authors derive the general solution for a parallel equation over FSMs with timeouts and propose a way to extract from it a solution that can minimize the time of the message exchange in the composition of web services modeled as such FSMs.

In the presentation of Vin Hoa La (the paper Vin Hoa La, Ana Cavalli “Intrusion-tolerant Routing in Wireless Sensor Networks”), the wireless Sensor Networks (WSNs) have been considered. The resource constraint characteristics of WSNs limit the secure design and development of security protocols for them whilst, sensor nodes usually operating in unattended and even harsh environments are prone to failures and are vulnerable to malicious attacks. In the presentation, two intrusion-tolerant routing protocols for WSNs, namely INSENS and ITSRP, were analyzed and it was shown by using the simulation and performance analysis that both of them are practical. The presentation K. Toumi, M. Aouidi, A. Cavalli “Modeling and security testing of services interoperability” has been devoted to the modeling of services interoperability and the application of active testing and monitoring techniques, based on passive testing, to the validation of secure interoperability properties. The application of the proposed techniques was illustrated by an industrial case study.

Most FSM based methods for test derivation are developed for initialized FSMs and the latter means that a reliable reset is assumed in an implementation under test in order to glue test sequences together. If the reset is rather expensive then the number of test sequences has to be reduced and when it is reduced to a single sequence and this sequence is called a checking sequence. Anton Ermakov (the paper A. Ermakov “Deriving checking sequences for nondeterministic FSMs”) proposes a method for deriving an adaptive checking sequence when the specification FSM is nondeterministic and the conformance relation is the reduction relation. The latter means that the behavior of a conforming implementation should be contained in the

behavior of the specification. A method returns an adaptive checking sequence that detects each nonconforming implementation that has not more states than the specification FSM under the conditions that the specification has a distinguishing sequence and a deterministic strongly connected submachine. The author notes that these conditions can be weakened for the case when the specification has a distinguishing test case and each state of the specification is definitely reachable from another state. The testing process is adaptive, i.e., the next input is determined based on the outputs produced for the previous inputs. Such adaptive distinguishing sequence can be shorter than a preset checking sequence.

In his presentation, Jorge Lopez (the paper J. Lopez, S. Maag, G. Morales “Scalable Evaluation of Distributed On-line Network Monitoring for Behavioral Feedback in Trust Management”) addressed the trust problem in collaborative systems which are growing in use and popularity and thus, trustworthy interactions of the different systems become a priority. The decision regarding with whom and how to interact depends on each system itself. The authors focus on providing trust verdicts by evaluating the behaviors of different agents, using distributed on-line network monitoring. This will provide trust management systems information regarding a trustee experience for systems based on so-called "soft trust". The authors propose a scalable evaluation method for any on-line network monitoring system by using the model of an extended finite state automaton (EFSA), and the known methods for reducing the time complexity of the evaluation algorithm.

Moreover, at the seminar meetings, joint events have been discussed which will be organized by the **Qualipso** research laboratory. Most papers presented at the seminar are published in the journal issue.

The head of the Department of Computer Science of TSU,
Professor — Nina Yevtushenko

The head of the Department of Network Software of Institute Télécom
SudParis, Professor — Ana R. Cavalli

The head of the Software Engineering Department of ISP RAS,
Professor — Alexander K. Petrenko

A TEFSM-based Framework for QoE Evaluation of OTT Services

[1] D. Rivera <diego.rivera-villagra@it-sudparis.eu>
[2] N. Kushik <ngkushik@gmail.com>
[3] C. Fuenzalida <cami@niclabs.cl>
[1] A. Cavalli <ana.cavalli@it-sudparis.eu>
[2] N. Yevtushenko <yevtushenko@sibmail.com>
[1] Télécom SudParis, 9 Rue Charles Fourier, Evry, 91000, France
[2] Tomsk State University, 36 Lenin str., Tomsk, 634050, Russian Federation
[3] NIC Chile Research Labs, Blanco Encalada 1975, Santiago, 8370403, Chile

Abstract. Most of the work related to QoE maps objective variables (QoS-related) into a single one that predicts the quality the user perceives. In this paper we present a new approach to model multimedia OTT services, integrating not only the functional aspects of the service but also non-functional variables, which are classified into objective, subjective and business-related. Non-functional metrics might affect the Quality of Service (QoS), Quality of Experience (QoE) and Quality of Business (QoBiz) correspondingly. We also discuss how all these variables can be taken into account to predict the QoE of the service being studied. The functional requirements are modeled into a Timed Extended Finite State Machine (TEFSM), which is augmented with context variables (and their updating functions) representing non-functional requirements related to QoS, QoE and QoBiz. We use these parameters to evaluate the QoE using a trace model derived from the TEFSM previously built.

Keywords: quality of service (qos); quality of experience (qoe); quality of business (qobiz); ott service; formal model; time extended finite state machine (tefsm)

1. Introduction

Nowadays, well-known multimedia services like Netflix, Youtube, Spotify or beIN Sports Connect [1] are gaining popularity, since they provide an easy and often cheaper way to access movies and series, video on demand and music respectively. Moreover, other non-multimedia services like Whatsapp or Skype have started to take an important part of the market that has been traditionally occupied by telephony companies; these services are nowadays substitutes for SMSs and phone calls. All the examples presented above share some common characteristics, being the most important when using Internet as the delivery network.

Over-The-Top Services (OTT) is the concept that encloses all the given examples. They are characterized by being services offered over Internet, which makes its

distribution easier than the methods used by traditional service providers. For example, a movie service like Netflix avoids the installation of a physical network to deliver the content to the users, which lowers the costs of deployment and, therefore, lowering the subscription costs for the customer. However, even when the usage of Internet as the delivery network has the advantage of lowering the complexity and costs of deployment of the service, it also proposes challenges that need to be addressed.

Internet was conceived as a best-effort network in the sense that there is no way to ensure the delivery of a message; even when using TCP, we can be sure that the message will arrive its destination, but we cannot be sure how much time it will take. In other words, by using Internet we cannot ensure the quality of the services that rely on Internet as the delivery network. This raises the challenge to ensure quality for OTT services like the ones previously mentioned. In specific, in this work we aim to ensure three main types of quality: the Quality of Experience (QoE) to the user, granting him/her the best service experience; the Quality of Service (QoS), to the service providers avoiding network congestion; and the Quality of Business (QoBiz) to the stakeholders, maximizing the revenues. However, meeting all these objectives may not be an easy task due to the high correlation among the variables.

Wolter and Van Moorsel state that the user experience directly affects how much shopping the customer does, and therefore, how much money the business earns [2]. Moreover, most of the works surveyed by Schatz et al. state that there exists a direct relationship between QoS and QoE [3], but as Geerts et al. state, QoS is not the only factor that influences the QoE [4]. In this paper we propose a framework to estimate QoE of an OTT Service based on the formalism of Extended Finite States Machine with timeouts (TEFSM), representing the functional requirements of the service being modeled. The derived machine is then augmented with non-functional requirements – inserted in the machine as context variables – representing objective, subjective and business parameters. In addition to these parameters, we also provide their corresponding updating functions, showing how the values of these variables should be updated each time a 'functional' transition is executed. We also discuss how to derive the l-equivalent of the TEFSM model, which is used to predict the QoE of the service.

As a case of study, we apply these techniques to the beIN Sports Connect Service; a service conceived to stream sports events over Internet to French users. For this service, we derive the corresponding TEFSM and estimate the QoE using the l-equivalent model.

The rest of the paper is structured as follows. Section 2 presents the preliminary notions and defines the TEFSM concept used along this paper. Section 3 shows how to derive the model on the basis of a user-service interaction. Section 4 discusses the ways to augment the TEFSM with the quality parameters and how they are updated at each TEFSM transition. In this section, we also provide a method for calculating

the QoE of the OTT service. Finally, Section 5 has the conclusions and the future work of this research.

2. Preliminaries

2.1 Over-The-Top Services

An overview of the *Over-The-Top* concept was provided in Section 1, but Green et al. have provided a formal definition. They define *Over-The-Top Services* as "services carried over the networks, delivering value to the customers but without any carrier being involved in planning, selling, provisioning, or servicing them" [5]. As an example of such services, and as a case study, we consider the French OTT service called beIN Sports Connect. This service allows the users to access the beIN Sports channels – which are usually offered as a premium pack for cable TV subscribers – allowing them to watch sports content over Internet.

2.2 QoS/QoE/QoBiz

The *Quality of Service* (QoS) is the metrics that has been evolved during the last few years. Starting from Van Morsel, who defined it as "metrics as availability and performance" [6], it comes to the another definition given by ITU-T in the form of "totality of characteristics of a telecommunications service that bear on its ability to satisfy stated and implied needs of the user of the service" [7]. The authors of [8] refer to the QoS as "service features stemming to technical aspects". Typical examples of these features are packet loss, jitter, and delay; however, other parameters related to the service (codecs, compression levels, etc.) are also considered.

Despite the fact that there was an effort to give a standard definition of the QoS from the beginning, it was not clear how the user's expectations could be considered within these metrics. For example, Gozdecki et al. proposed the concept of *Perceived* QoS, which "reflects the customer's experience of using a particular service" [8]. This concept is what Le Callet et al. called *Quality of Experience* (QoE), defining it as the "degree of delight of the user of a service. In the context of communication services, it is influenced by content, network, device, application, user expectations and goals, and context of use" [9]. One may notice, that QoE is strictly related to user's expectations towards the service, which makes the metrics very subjective and, therefore, it is not easy to measure it compared with the QoS.

The quality metrics described above are in some way related to the user himself/herself. In a similar way, other metrics have been defined form the point of view of the service provider. The *Quality of Business* (QoBiz) is a concept introduced by Van Moorsel [6], in order to quantify the business revenue from a service offered over the Internet. By definition, QoBiz parameters are "all of those parameters that are expressed in monetary units" [10]. Some of them might include

(but are not limited to): transactions loss, dollars per transaction, revenue of web site or cost of servers.

2.3 Timed Extended Finite State Machines

As usual, objective, subjective, and business parameters values can be collected as the vector $V = (p_1, \dots, p_n)$. In this vector, three groups of parameters can be recognized depending on their origin: related to the QoS, related to the QoE, and related to the QoBiz. In order to efficiently evaluate the values of each parameter $p_i, i \in \{1 \dots n\}$ we propose to derive a Timed Extended Finite State Machine (TEFSM) that preserves the functional behavior of the OTT service, and, meanwhile, captures $p_1, \dots, p_n$ parameters as corresponding context variables.

TEFSM is an extension of the Extended Finite State Machine model (EFSM) [11], which is an ordinary Finite State Machine (FSM) augmented with context variables, input/output parameters, predicates and update functions. More precisely, an extended finite state machine [12] A is a pair (S, T) of a set of states S and a set of transitions T between states from S, such that each transition $t \in T$ is a tuple $(s, i, o, P, v_p, o_p, s')$, where $s, s' \in S$ are the initial and final states of a transition; $i \in I$ is an input with the sets D_{inp-i} of possible input vectors of corresponding input parameter values, $o \in O$ is an output, where O is the set of outputs (with the sets D_{out-o} of possible output parameter values); P, v_p and o_p are functions, defined over input parameters, and context variables, namely: $P: D_{inp-i} \times D_V \rightarrow \{True, False\}$ is a predicate, where D_V is a set of context vectors; $o_p: D_{inp-i} \times D_V \rightarrow D_{out-o}$ is an output parameter update function; $v_p: D_{inp-i} \times D_V \rightarrow D_V$ is a context update function.

A Timed EFSM extends the usual EFSM with a corresponding timeout function $\Delta: S \rightarrow S \times \{\mathbb{N} \cup \{\infty\}\}$ where $\mathbb{N}$ denotes the set of nonnegative integers. As an example of a TEFSM, one may turn to Fig. 1, where the corresponding machine is presented and integers 1, 2,..., 8 mark the transitions of the machine. Consider the state 'waiting_for_personal_data'. In this figure there is a transition from this state to the state 'waiting_for_card_data', which triggers under the input *validate*, constrained by the value of the context variable *promo_flag* equals to 0. In this case, the output *user_payment_form* is produced. Also, in this figure, a timeout is used in the transition from 'stream_delivery' to 'check_state': if no input is applied until the timeout expires, the machine moves spontaneously to the state 'check_state'.

3. Specification of an OTT Service as a TEFSM

In [13], Andrews et al. propose a methodology to "derive hierarchies of Finite States Machines (FSMs) that model subsystems of the Web Applications". Based on this approach, we provide an outline of the methodology used in this work to generate a TEFSM of an OTT service: (1) develop a written description of the service from the point of view of the user; (2) identify the system states using the description. The authors in [13] suggest the partition of the service in order to identify its logical

blocks, which are easier to model; and (3) identify the transitions of the system, considering the user interaction with the system.

The first step of the methodology presented here is to build a written description of the service being studied. This description will be elaborated in such a way that it should state: (1) functional requirements, describing the behavior of the service from the point of view of the user; and (2) non-functional requirements, considering objective, subjective and business service requirements.

3.1 The beIN Sports Connect Service

In order to extract the functional requirements of the service, we turned to beIN Sports website [1] to describe the user's interaction. We can identify two main phases of the service's behavior: service registration (marked as (1), (2) and (3) in Fig. 1) and service utilization (marked as (4) in Fig. 1).

3.1.1. Service Registration

Before using the service, a new user must create an account in the site and associate a subscription plan into his/her account. The new customer must connect to the home page [1], which will show him/her the subscription options. In this page, the user can choose one of the three subscription types offered: two where the user has to pay a fee and another used to activate promotional codes to test the service (marked as (1) in Fig. 1).

Once the subscription type was chosen, the system shows the registration form and the payment options, the latter being showed only if the user selected a subscription that requires a payment (process (2) in Fig. 1). In the case of the user selected the subscription used to activate a promotional code, a field to introduce this code is shown instead of the payment options. The payment is processed in a third-party server, which requests the user's credit card information and stores it in order to automatically renew the subscription (marked as (3 Fig. 1). Finally, once the payment was successfully processed, the user is redirected to beIN Sports Connect Site, and an active subscription is associated with his/her account.

3.1.2. Service Utilization

A user who already has an account and a subscription associated can go to the home site, log into the site with his/her credentials (marked as (4) in Fig. 1). This action allows him/her to watch the stream online. The multimedia platform is designed to be able to adapt itself to the variable conditions of the network, being able to lower the quality of the video if the network technical variables require it. In fact, if it is not possible to stream an HD version of the video, a lower quality version (stressing less the network) will automatically be streamed to the user, avoiding any freeze of the video.

3.2 Generating TEFSM from service description

Once the description has been built, we need to generate the first model of the service. In this case, we will take into account that the user's queries are the TEFSM inputs and the system responses are the machine outputs. In addition, the introduction of timeouts allows us to model the ability of the service to periodically check the QoS values of the network. We reproduce this behavior in the following way: each *T* time units the system moves from a 'stream_delivery' state to a 'check_stream' state. In this state, the QoS parameter values are measured and the conclusion about the QoS is drawn. If the values of the QoS are not satisfactory, the system moves to a 'stop_stream' state, otherwise the system continues the streaming adapting it to a lower or higher QoS value if it is necessary. The final derived machine for the beIN Sports Connect Service is shown in Fig. 1.

Within this model, we can identify the four main processes a new customer needs to accomplish to start using the service, namely: (1) connect to the site and select a subscription plan, (2) provide his/her personal information, (3) provide his/her payment information, and (4) use the service with an active subscription.

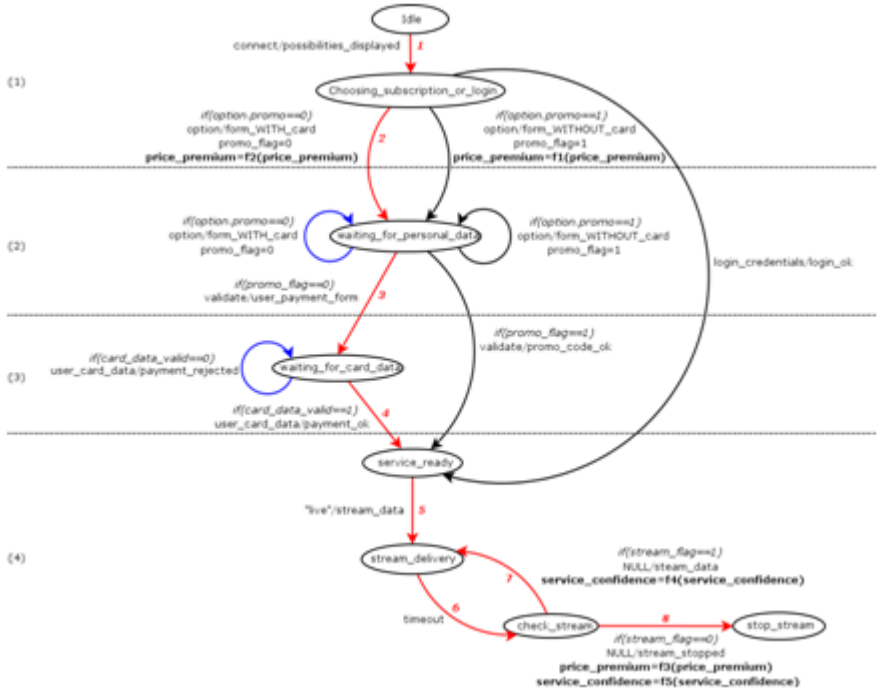


Fig. 1. TEFSM describing beIN Sports Connect.

4. Augmenting TEFSM transitions with quality parameters

The model previously built captures the functional requirements of the service, but we still need to introduce the non-functional requirements. In order to include non-functional parameters into the FSM, we will introduce the corresponding context variables into the FSM model. Each one of them will represent a single parameter, and its updating function will indicate how and in which transitions these parameters must be updated.

4.1 Integrating Objective and Subjective Parameters

In this work we aim to consider not only QoS-related parameters to the QoE evaluation, thus the classical QoS-QoE relationships – mostly surveyed by Schatz et al. in [3] – cannot be used.

4.1.1. Objective Parameters

As we stated in Section 3.2, the system has the ability to check the QoS parameters automatically. In our example, we model the decision of the service in a single QoS-related parameter called ‘stream_delivery’. This flag will be set to 1 if the minimal network requirements are met to continue with the streaming and 0 otherwise.

We note in Fig. 1 that the machine spontaneously moves to state ‘check_stream’ each *T* time units, leaving the ‘stream_delivery’ state and therefore interrupting the streaming while checking the network status. Any user would not like to see a service interruption to check the technical network conditions, thus the machine needs to be patched to include the ability of checking the network in a parallel way. In this work we do not focus in the TEFSM that allows parallel execution, however, this issue can be solved by introducing a ‘stream_and_check’ state where the two processes are executed simultaneously. Due this formalism is out of the scope of this work, we leave this analysis for future work.

4.1.2. Subjective Parameters

As Laghari and Connelly state, any subjective factor that influences the QoE should include the user's perception of the service [14]. Given that, we consider a subjective variable called ‘service_confidence’, which represents the ‘trust’ of a user towards the service. This parameter will be measured using a 3-value scale: 0, 1 and 2, representing, respectively, a user dissatisfied, neutral and satisfied.

In addition, considering the meaning of the variable introduced here, the value of the experienced ‘trust’ of the user will be updated each time the system makes a decision with the stream. In other words, once the execution has reached the state ‘check_stream’, the system makes a decision whether to continue with the service, executing transition 7 or 8 in Fig. 1. If the system ‘decides’ to continue with the stream (transition 7), the customer will increase his/her satisfaction with the service,

otherwise (transition 8), the customer will low his/her satisfaction. We reflect this behavior in the specification of the updating functions f_4 and f_5, shown in Table. 1.

4.2 Integrating Business Parameters

One of the main contributions presented in this work is the inclusion of business parameters into the OTT model. Regarding business metrics, Aaker proposes the *Brand Equity Ten* as a set of ten variables to measure the value of a brand [15]. From this set, we selected the Price premium measurement, defined as the "amount a customer will pay for the brand in comparison with another brand (or set of comparison brands) offering similar benefits". In the same work, Aaker suggests the segmentation of the market in three types of buyers of a brand: non-customers, brand switchers and loyal customers [15]. We consider these points by introducing a context variable 'price_premium' into the model using a 3-value scale to measure it: 0, representing a non-customer user; 1, used to exemplify a brand switcher; and 2, symbolizing a loyal customer of the brand. Moreover, following the definition of this variable, we will update its value considering the type of the contract the user selected: a customer who selected a long-commitment subscription – a one year engagement – will be considered as a loyal customer, otherwise, he/she will be considering a brand switcher.

We model this decision of the user in the TEFSM by updating correspondingly the context variable 'price_premium' once he/she has selected the subscription type. This occurs in both transitions that take the user out of the state 'choosing_subscription_or_login' to the state 'waiting_for_personal_data'. In addition, we will also update this parameter if the streaming is stopped due low technical network conditions, lowering the loyalty value in this case. We reflect all these observations in the corresponding updating functions f_1, f_2 and f_3, which are specified in Table. 2.

Table. 1 Function modifying 'service_confidence'.

service_confidence	f_4(service_confidence)	f_5(service_confidence)
0	1	0
1	2	0
2	2	1

Table. 2 Function modifying 'price_premium'.

price_premium	f_1(price_premium)	f_2(price_premium)	f_3(price_premium)
0	1	1	0
1	1	2	0
2	2	2	1

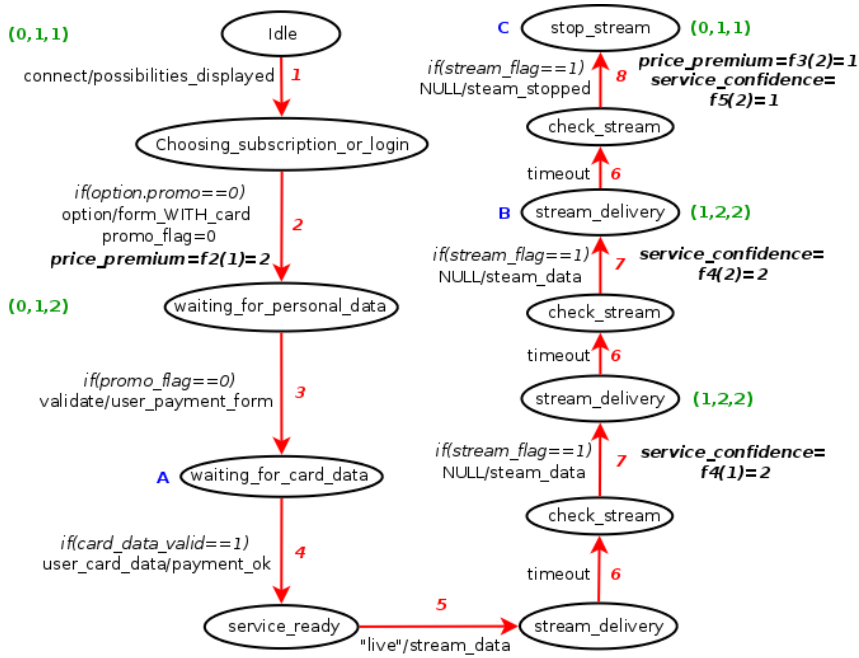


Fig. 2 Path representing a typical user interaction.

4.3 QoE Evaluation

At this point, we have built a TEFSM that has been augmented with quality parameters inserted as context variables of the machine. However, before using this model to predict the QoE of the OTT service, we must derive an *l*-equivalent model from the given TEFSM.

4.3.1. Deriving the l-equivalent of the TEFSM

The model we have built up to this point captures the functional requirements as well as objective, subjective and business parameters. However, it is not possible to clearly identify the final states of the executions paths. To achieve this last goal, we need to unroll the model into its *l*-equivalent tree-shaped form. This new equivalent TEFSM can be calculated using a process for which we provide the outlines: (1) insert constrains to the loops of the model and fix a desired length *l* for the height of the tree that will be analyzed; (2) create new paths starting from the root node, considering the constrains introduced; and (3) draw the final *l*-equivalent model.

In Fig. 2 we provide a single path extracted from the *11*-equivalent[1] model derived from Fig. 1. The transitions are remarked in both diagrams using integers from 1 to 8 to enumerate them. This path represents the actions taken by a new customer subscribing to the service: (1) the user connects to the service; (2) he/she chooses a paid subscription and pays for it without making any mistake when introducing the required data (thus loops marked in blue in Fig. 1 are not executed); (3) the user starts to use the service immediately; and (4) the system checks the stream three times, deciding to cut the streaming in the latter due the bad quality of the network at that time.

4.3.2. Evaluating the QoE value in terminal states

With the *l*-equivalent mode we can proceed to evaluate the QoE. This is done in three steps: (1) define the initial values for the context variables; (2) update their values using the updating functions, simulating the behavior of the user; and (3) calculate the QoE at the leaves of the tree by applying a QoE model that integrates the values of the variables involved. The process presented here does not require the usage of a specific QoE model, making the proposed framework a generic approach to predict the QoE.

In our case study, we used a simple linear model proposed by Sandoval et al. [16]. In this model, fixed values for the weights will be used, taking into account the following observations: (1) due an OTT uses the network as a delivery method, they do not have control over QoS parameters, thus a weight of 0.5 will be used for the QoS-related parameter; (2) due a user who recommends the service generated higher revenues for the company, the service provider would like to keep its customers with high levels of 'service_confidence', assigning a weight of 1 to this variable; and (3) the OTT provider aims to maximize their revenues, giving a high importance to the 'price_premium' variable, thus a value of 1 will be used.

With the linear model and the weights used, we can evaluate the QoE in the three stages A, B and C marked in Fig. 2. The initial vector of values (0,1,1) was constructed with the characteristics of an average new customer: he/she has not used the service yet (stream_flag=0), he/she is indifferent with the service (service_confidence=1) and he/she is a brand switcher (price_premium=1). The results are shown in Table. 3.

Table. 3 Parameter values and the QoE for the l-equivalent

Stage	Length	Parameters	QoE
A	4	(0,1,2)	3.0
B	9	(1,2,2)	4.5
C	11	(0,1,1)	2.0

[1] The length of a tree has been calculated by counting the number of transitions executed from the root node.

5. Conclusions

This work presented a framework based on TEFSM to estimate the QoE of a given OTT Service. Unlike other frameworks, it does not only involve QoS-related service parameters, but also considers the inclusion of subjective and business-related parameters, which can impact in the QoE of the service. The proposed framework starts with the construction of a TEFSM capturing the functional requirements observed from the point of view of the user. The model is then augmented with the parameters in analysis, inserting them as context variables in combination with their updating functions. In order to use this model to evaluate the QoE of the OTT service, an *l*-equivalent model is derived, showing all the possible combinations of queries/responses of a fixed length *l* or less. Finally, a QoE model is applied in the leaves of this tree, obtaining a predicted QoE value in the terminal states of the execution. In addition, with this framework, we also presented an example of its utilization with beIN Sports Connect service, analyzing the QoE for a completely new customer of the service. As a future work, we are going to perform more experimentation in order to estimate an optimal value for the length *l* of the *l*-equivalent model.

References

[1]. Société beIN Sports France, beIN Sports Connect, https://beinsportsconnect.fr/index.fr. 2014, (Accessed August 2014).

[2]. Wolter K., Van Moorsel A. The relationship between Quality of Service and Business Metrics: monitoring, notification and optimization. Hewlett-Packard Labs Technical Report HPL-2001-96, 2001.

[3]. Schatz R., Hoßfeld T., Janowski L., Egger S. From packets to people: Quality of experience as a new measurement challenge, LNCS, 2013, vol. 7754, pp. 219-263.

[4]. Geerts D., De Moor K., Ketykó I., Jacobs A., Van den Bergh J., Joseph W., Martens L., De Marez L. Linking an integrated framework with appropriate methods for measuring QoE. Second International Workshop on Quality of Multimedia Experience, 2010, pp. 158-163.

[5]. Green W., Lancaster B., Sladek J. Over-the-top Services. http://www.pipelinepub.com/1207/pdf/Article_3.pdf, 2006 (Accessed August 2014).

[6]. Van Moorsel A. Metrics for the Internet age: Quality of Experience and Quality of Business. Fifth International Workshop on Performability Modeling of Computer and Communication Systems, 2001, pp. 26-31, Universität Erlangen-Nürnberg, Germany.

[7]. ITU. Vocabulary for performance and quality of service. Recommendation P.10/G.100, July 2006.

[8]. Gozdecki J., Jajszczyk A., Stankiewicz R. Quality of service terminology in IP networks, IEEE Communications Magazine, 2003, vol. 41, pp. 153-159.

[9]. Le Callet P., Möller S., Perkis A. Qualinet whitepaper on definitions of quality of experience. Seminar report, European Network on Quality of Experience in Multimedia Systems and Services, June 2012.

[10]. Stojanovic M., Factors affecting service provider's quality of business in NGN environment. Center for Quality, 2012.

[11]. Cheng K. T., Krishnakumar A. S. Automatic functional test generation using the extended finite state machine model, in Proceedings of the 30th International Design Automation Conference, 1993, pp. 86-91.
[12]. Petrenko A., Boroday S., Groz R. Confirming configurations in EFSM testing. IEEE Transactions on Software Engineering, 2004, vol. 30, pp. 29-42.
[13]. Andrews A. A., Offutt J., Alexander R. T. Testing web applications by modeling with FSMs. Software & Systems Modeling, 2005, vol. 4, no. 3, pp. 326-345.
[14]. Laghari K., Connelly K. Toward total quality of experience: A QoE model in a communication ecosystem. IEEE Communications Magazine, 2012, vol. 50, pp. 58-65.
[15]. Aaker D. A. Measuring brand equity across products and markets. California management review, 1996, vol. 38, no. 3, pp. 102-120.
[16]. Sandoval J., Ehijo A., Casals A., Estevez C. New model and open tools for real testing of QoE in mobile broadband services and the transport protocol impact: The operator's approach. IEEE Latin-America Conference on Communications (LATINCOM), 2013, pp. 1-5.

Расширенные конечные автоматы с таймаутами для оценки удовлетворенности пользователя OTT сервисом

[1] *Д. Ривера <diego.rivera-villagra@it-sudparis.eu>*
[2] *Н, Кушик <ngkushik@gmail.com>*
[3] *К. Фьюензалида <cami@niclabs.cl>*
[1] *А. Кавалли <ana.cavalli@it-sudparis.eu>*
[2] *Н.Евтушенко <yevtushenko@sibmail.com>*
[1] *Télécom SudParis, 9 Rue Charles Fourier, Evry, 91000, France*
[2] *НИ ТГУ, 634050, Россия, г. Томск, пр. Ленина, дом 36*
[3] *NIC Chile Research Labs, Blanco Encalada 1975, Santiago, 8370403, Chile*

Аннотация. В последние годы технология OTT (over-the-top) предоставления пользователям доступа к медиа контенту в сети интернет становится все более популярной, и, соответственно, актуальной задачей является оценка качества предоставляемого сервиса. Большинство методов оценки качества сервиса конечным пользователем основаны на сопоставлении наборов значений объективных параметров оценки качества сервиса некоторому единому значению, предсказывающему уровень удовлетворенности пользователя. В данной статье рассматривается новый подход к моделированию мультимедиа OTT сервисов, направленный на интегрирование в модели как функциональных, так и нефункциональных аспектов поведения сервиса. Рассматривается три класса нефункциональных параметров: объективные, субъективные и бизнес-параметры, которые, соответственно, влияют на оценку качества сервиса (QoS), качества удовлетворенности пользователя (QoE) и качества бизнеса (QoBiz). Все три класса параметров необходимо учитывать для вычисления/предсказания оценки удовлетворенности пользователя выбранным сервисом. Функциональные требования к поведению сервиса формализованы в виде модели расширенного конечного автомата с таймаутами, который дополнен контекстными переменными, представляющими нефункциональные требования к QoS, QoE и QoBiz. Для предсказания удовлетворенности пользователя, по построенному расширенному автомату сервиса с учетом возможных сценариев поведения пользователя строится l-эквивалентный древовидный автомат, поведение которого совпадает с поведением исходного расширенного автомата на всех входных последовательностях длины не больше l; значения соответствующих контекстных переменных в финальных состояниях автомата представляют собой оценку удовлетворенности пользователя. Предложенный метод иллюстрируется на примере сервиса beIN Sports Connect.

Ключевые слова: качество сервиса (qos); качество удовлетворенности пользователя (qoe); качество бизнеса (qobiz); отт сервис; формальная модель; расширенный конечный автомат с таймаутами

Список литературы

[1]. Société beIN Sports France, beIN Sports Connect, https://beinsportsconnect.fr/index.fr. 2014, (Accessed August 2014).

[2]. Wolter K., Van Moorsel A. The relationship between Quality of Service and Business Metrics: monitoring, notification and optimization. Hewlett-Packard Labs Technical Report HPL-2001-96, 2001.

[3]. Schatz R., Hoßfeld T., Janowski L., Egger S. From packets to people: Quality of experience as a new measurement challenge, LNCS, 2013, vol. 7754, pp. 219-263.

[4]. Geerts D., De Moor K., Ketykó I., Jacobs A., Van den Bergh J., Joseph W., Martens L., De Marez L. Linking an integrated framework with appropriate methods for measuring QoE. Second International Workshop on Quality of Multimedia Experience, 2010, pp. 158-163.

[5]. Green W., Lancaster B., Sladek J. Over-the-top Services. http://www.pipelinepub.com/1207/pdf/Article_3.pdf, 2006 (Accessed August 2014).

[6]. Van Moorsel A. Metrics for the Internet age: Quality of Experience and Quality of Business. Fifth International Workshop on Performability Modeling of Computer and Communication Systems, 2001, pp. 26-31, Universität Erlangen-Nürnberg, Germany.

[7]. ITU. Vocabulary for performance and quality of service. Recommendation P.10/G.100, July 2006.

[8]. Gozdecki J., Jajszczyk A., Stankiewicz R. Quality of service terminology in IP networks, IEEE Communications Magazine, 2003, vol. 41, pp. 153-159.

[9]. Le Callet P., Möller S., Perkis A. Qualinet whitepaper on definitions of quality of experience. Seminar report, European Network on Quality of Experience in Multimedia Systems and Services, June 2012.

[10]. Stojanovic M., Factors affecting service provider's quality of business in NGN environment. Center for Quality, 2012.

[11]. Cheng K. T., Krishnakumar A. S. Automatic functional test generation using the extended finite state machine model, in Proceedings of the 30th International Design Automation Conference, 1993, pp. 86-91.

[12]. Petrenko A., Boroday S., Groz R. Confirming configurations in EFSM testing. IEEE Transactions on Software Engineering, 2004, vol. 30, pp. 29-42.

[13]. Andrews A. A., Offutt J., Alexander R. T. Testing web applications by modeling with FSMs. Software & Systems Modeling, 2005, vol. 4, no. 3, pp. 326-345.

[14]. Laghari K., Connelly K. Toward total quality of experience: A QoE model in a communication ecosystem. IEEE Communications Magazine, 2012, vol. 50, pp. 58-65.

[15]. Aaker D. A. Measuring brand equity across products and markets. California management review, 1996, vol. 38, no. 3, pp. 102-120.

[16]. Sandoval J., Ehijo A., Casals A., Estevez C. New model and open tools for real testing of QoE in mobile broadband services and the transport protocol impact: The operator's approach. IEEE Latin-America Conference on Communications (LATINCOM), 2013, pp. 1-5.

Model-Based Testing for MANETs

S. Maag <Stephane.Maag@telecom-sudparis.eu>
Institut Mines-Telecom/Telecom SudParis, CNRS UMR 5157,
9 rue Charles Fourier, 91011 Evry Cedex, France

Abstract. Conformance testing in network engineering is a crucial phase in the development of complex communicating systems. Model-based testing allows to automatize the testing process by generating test suites from a formal specification and to execute them on a real IUT. While many techniques have been developed, their application to test wireless routing ad-hoc protocols still raises many issues. The paper objective paper is to present the node self-similarity reducing the number of inconclusive verdicts often met in traditional MBT.

Keywords: model based testing; manet; nodes self-similarity; routing protocols

1. Introduction

Conformance testing in network engineering is a crucial phase in the development of complex communicating systems. Among the different testing steps, the development and execution of test cases based on a formal model is an important issue for testing communication protocols and other reactive systems. The purpose of these tests is to determine whether a protocol implementation conforms to its specification. Usually a conforming implementation is required to have the same input/output behavior as defined by the specification. In various application domains, such as telecommunication systems, communication protocols and other reactive systems, the specification can be represented in the form of an extended finite state machine (EFSM). In particular, EFSMs are the underlying models for formal description techniques, such as SDL and SysML. Several model based testing (MBT) techniques have been proposed to generate and execute test cases from formal specification [21]. While most of these specification techniques allow to efficiently design these above mentioned systems, there are currently new environments that bring their own inherent constraints. This is notably the case when considering wireless Mobile ad-hoc Networks (MANET).

In Wireless Mobile Ad Hoc Networks (MANETs) there are no predefined infrastructures, no administrative node and each node participates in the provision of reliable operations in the network. The nodes may move continuously leading to a volatile network topology with interconnections between nodes that are often modified. As a consequence of this infrastructureless environment, each node communicates using their radio range with open transmission medium and some of

them behave as routers to establish multi-hop connections. Besides, conformance testing for ad hoc routing protocols is crucial to the reliability of those networks. The major techniques used by the ad hoc network experts to design and ensure the quality of their routing protocols essentially rely on descriptions for simulations and/or emulations. Andel et al. [5] tries to illustrate some comparisons between NS-2 [1], OpNet [2] and Glomosim [4]. Results providing by the simulation testing is sometimes far from the one obtained in a real case study.

Formal description techniques and their testing tools are rarely applied in such kind of networks. The main reasons are the difficulty to take into account the inherent MANET protocol characteristics and the mobility of nodes in the test sequences generation and their execution. Our work focuses on a testing technique based on algorithms to check the conformance of ad hoc routing protocols. Nevertheless, the execution of these test sequences is currently an issue. Indeed there is often a gap between the dynamic topology designed in a specification and the one of a real case study. We illustrate the concept of node self similarity in order to generate test sequences and execute them on a real wireless ad hoc routing protocol taking into account the network topologies.

The remainder of the paper is as it follows. In Section 2, we present some related works. In Section 3, basics on MBT are presented. Then, we describe the node self-similarity and its application on an emulated testbed in Section 4 and 5 respectively.

2. Related work

Verisim [6] is a model combining NS-2 and the trace verification component provided by the Monitoring and Checking system namely MAC [7]. The goal is to generate a NS-2 trace *T* and to verify if the expected properties are included in the implementation I according to a scenario *S*. The authors have shown that the AODV implementation in NS-2 was false regarding some properties. This first work was very interesting, disturbing and raised several issues regarding the efficiency/reliability of the simulation/emulation. These works have known recently a certain resonance by the publication of [23] in which MANET simulations results still reveal pitfalls. The authors describe "design space" of MANET routing in terms of its basic dimensions and corresponding parameters.

While Verisim performs a validation through invariants in a simulator, [8] proposes a formal methodology to specify and analyze a MANET routing protocol. It is based on the Relay Node Set (RNS) concept. A RNS is a set allowing to reach all nodes in the network. However, it does not allow studying functional properties neither the nodes' interactions.

From a need of specification, it came another work. In order to study the interest and the performances of their new routing protocols (LTLS, Logical Topology based Location Service), the authors [9] developed a formal model namely Distribued Abstract State Machines (DASM). While this model allows verifying the behavior of a node in a functional way, unfortunately the model is non-executable and does not allow to observe the nodes' interoperability.

[10] is another approach to formalize the routing in a MANET by applying game theory concepts. The game theory is based on the ‘‘income’’ calculus. In our case, the income means for instance the convergence when the topology is modified or the induced overhead. Despite an innovative approach, many issues like the required knowledge of the network, are still present.

In our work, we propose a new testing approach relying on well-known formal methods, especially a nodes self similarity concept by considering the eventual topology modifications. Our main goal is to facilitate the execution of generated test sequences and to reduce the inconclusive verdicts when checking the conformity of an implementation in relation with its specification.

corresponding context variables.

3. Conformance testing and formal models

3.1 Basics

The conformance testing usually relies on the comparison between the behavior of an implementation and the formal specification of a given protocol. The conformance testing procedure follows these steps:

Step 1. Define a testing architecture with respect to the characteristics of the system under test and its possible implementations. This step could impact on each following step and has to be defined according to the context.

Step 2. Make some assumptions that are sometimes required to enable the test.

Step 3. Design a precise formal specification of the system to be tested. This specification takes into account the system functionalities as well as the data specific to the test environment (test architecture, test interface, etc.).

Step 4. Select the appropriate tests. This step is the definition of the test purposes.

Step 5. Generate the test sequences. The test purposes are used as a guide by an algorithm based on simulation to produce the test sequences from the specification.

Step 6. Format the test sequences i.e. to produce test sequences in some accepted formalism as Test Description Language (TDL) [13] or in Testing and Test Control Notation (TTCN3), the ITU-TS standard language used for test specification.

As above mentioned, our approach is based on formal model. In our work we define and use as a formal specification the Extended Finite State Machine (EFSM).

Definition 1. An EFSM M is defined as: $M = (I, O, S, x, T)$ with I, O, S, x and T, respectively, a set of input symbols, a set of output symbols, a set of states, a vector of variables and a set of transitions. Each transition $t \in T$ is a 6-tuple defined as:

$t = (s_t, q_t, i_t, o_t, P_t, A_t)$ where s_t is the current state, q_t is the next state, i_t is an input symbol, o_t is an output symbol, $P_t(x)$ a predicate on the values of the variables, $A_t(x)$ an action on the variables.

Our protocols are specified using an EFSM based language, named Specification Description Language (SDL) standardized by ITU-T [22]. This is a widely used language to specify communicating systems and protocols, based on the semantic

model of EFSM. Its goal is to specify the behavior of a system from the representation of its functional aspects. It allows describing the architecture of the system i.e. the connection and organization of the elements (blocks, processes, etc.) with the environment and between them.

3.2 Test case generation

There exist several techniques to generate test suites from a formal specification and especially from an EFSM [3]. In this work, we used the TESTGEN-SDL approach. For a given EFSM, $M = (I, O, S, x, T)$, each combination of a state in S and variable values of x consists of a configuration. The initial state $s(0)$ with the initial variable values $x(0)$ forms an initial configuration. On the other hand, we are only interested in the configurations that are reachable from the initial configuration, and they can be represented by a reachability graph as follows. It is costly to take the system to the initial configuration, whereas the test sequence length makes little difference. Specifically, a test is a path from $s(0)$ in the reachability graph. Assume that we are given a desired fault coverage, i.e., a set of system entities that we want to cover, such as transitions and variable values. We assign a distinct color to each such entity and we have a set C of $k=|C|$ distinct colors. Each state and transition is associated with a subset of colors from C, denoted by $c(v) \subseteq C$ and $c(u, v) \subseteq C$, respectively. The function $c(.)$ designates the required coverage associated with a state or transition. For instance, if a transition t is assigned a distinct color c_t, which is to be covered. Then each edge (u, v), which is from the transition t, has $c_t \in c(u, v)$.

We are interested in a complete test set that covers all the colors. That is, the union of the color sets of the nodes and edges on the paths (tests) is the set of all colors C. Note that they are not necessarily the conventional covering paths that cover all the edges. Further, more details on TESTGEN-SDL are provided in [11].

3.3 Test case execution

In order to verify the conformance of an implementation, active testing requires a set of test sequences. A test sequence is a succession of inputs provoking outputs obtaining from the formal model. Once we obtain these sequences, they are injected into the Implementation Under Test (IUT) and the outputs are collected and compared with the ones expected by the model to finally provide a verdict.

In order to interact with the IUT, a testing architecture is needed and is composed of Points of Control and Observation (PCO) that are connected with the Upper and Lower Testers (UT and LT) controlled by the Test Coordination Procedure (TCP). Each time one of the testers observes a packet, the TCP checks if it is the one expected regarding the specification.

The International Organization for Standardization proposes in one of its standard [12] different conformance testing architectures. The main differences between the presented architectures rely on the position, communication and synchronization of the components.

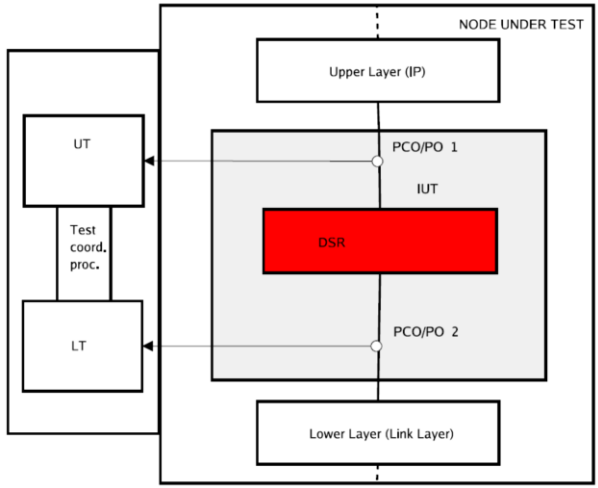


Fig. 1. Our testing architecture

In a wired environment, the choice of the testing architecture is not a difficulty especially because of the fixed node, the reliable communications, as well the realist assumption that all received packets will be processed. In the MANETs, these problems have another dimension. The communications are not so reliable, subject to distortion, multiple retransmissions, routes modifications, delays, collisions, etc. These aspects raise several issues about the dynamicity/reactivity to the observed events of the testing architecture depending on the radio interfaces.

Generated test suites from SDL models have been applied on several implementations under test specifically in wired networks. However, interesting results have also been obtained for wireless routing protocols rising at the same time novel issues [14]. Indeed, we performed applications of these above model based testing techniques for protocols in Mobile ad-hoc Networks (MANET). Many inconclusive verdicts were obtained compared to the PASS ones. The reasons were that the SDL specification was unable to consider topological changes due to packet losses, radio disconnections, etc. We therefore defined a novel approach based on nodes self similarity. This is what we describe in the next section.

4. Nodes Self Similarity

In opposition to wired and fixed networks, when a test sequence has been generated and has to be executed on a MANET routing protocol, if the network is not controlled (as mentioned before) it becomes very difficult to know how to execute it and how to interpret the verdict provided by the testers. We therefore need to provide test sequences enforceable on any real network topology. Nevertheless, due to the volatility of these kind of networks, their topology may be quite different that

the one represented by the specification. In order to answer to those issues, we present in the following the nodes' self similarity concept.

4.1 Self similarity notions

The self similarity notion is presented and applied in [17] especially to lead a real wired network topology to another one that is smaller and equivalent regarding the testing process. Indeed, two nodes in a network being self similar may be composed to become a single node. This technique allows, under some conditions, to map some topologies from the same equivalence class to a simpler topology from the same class. In other words, the self similarity means that if a packet is received by a node *A* and sent to *B* and forwarded to another one, then *A* and *B* has the same behavior according to this packet and it is possible to compose *A* and *B* to *A*0 that represents the two nodes with the encapsulated internal communications. Formally, we define the combination of two nodes and then the self similarity notion.

Definition 2. Nodes combination.

Let $\{N_i\}_{i\in E}$ where $E \in [1\ldots n]$ and $n \in N$ be a collection of models that can be described as EFSMs. We note $N_1 \circ \ldots \circ N_n$ the combination of all N_i defined as:

$O(N) = \bigcup_{i\in E} O(N_i)$

$I(N) = \bigcup_{i\in E} I(N_i) - \bigcup_{i\in E} O(N_i)$

$S(N) = \prod_{i\in E} S(N_i)$

$x(N) = \prod_{i\in E} x(N_i)$

$T(N) = (s, s', e, o, P_i(x), A_i(x))$

if $(s_i, s'_i, e, o, P_i(x), A_i(x)) \in T(N_i)$ where $P_i(x) \equiv P_i(x_i)$, $A_i(x) \equiv A_i(x_i)$, $(e, o) \in I(N_i) \times O(N_i)$.

Let $\Phi \subset O(N)$, we define $ActHide_\Phi(N)$ as the obtained EFSM from N where each action of Φ becomes an internal one. This application transforms the communications between the different components of N into non-observable actions. Thus, we may define the self similarity of two nodes as:

Definition 3. Nodes self similarity.

Let two possible actions for a node be *send*(*Message*, *n*, *m*) and *receive*(*Message*, *n*', *m*') where *n* (respectively *m*') is the observed node, *m* (respectively *n*') the *destination* of the packet (respectively *sender*), and *Message* is the whole possible contents of a packet. Let *N* be a node specification. We note *Tr*(*N*) the set of observable traces, a trace being an input/output sequence. Besides, *Tr*(*N*) is a finite set, indeed the variable domains of the EFSM are discrete and finite (as most of the communication protocols).

Some $N_{i\in I}$ are self similar if:

$\boldsymbol{Tr(ActHide_\Phi(N_1 \circ N_2)) \subseteq Tr(N)}$, where $\Phi = \{send(Message, N_1, N_2), send(Message, N_2, N_1), receive(Message, N_1, N_2), receive(Message, N_2, N_1)\}$.

Due to the inherent constraints, we use the self similarity considering:

1. The self similarity is applied from the viewpoint of a single node, the IUT.

2. The self similarity is applied each time a packet of the test sequences is received or sent in order to simplify the possible topologies known by the IUT.

3. The self similarity is applied only for a specific communication on a defined route between the IUT and another node.

4.2 Nodes self similarity through MANETs

As the links in those kinds of networks are unreliable and unpredictable, it is needed to consider a communication failure. Therefore, a path from a source S to a destination D is divided in two parts. The first one is composed by the nodes following the source S that succeeded the packet forward, and the second one that contains D and all nodes that did not receive the packet.

In the case of conformance testing where the network is perceived only from the IUT viewpoint, three kinds of nodes are noted: the source, the destination and the other path nodes N_i. We also assume that except S and D, all other N_i have the same functional behavior. A route is defined as a succession of S, N_i, $i \in [1 \ldots n]$, and D.

We consider the nodes in the route from the viewpoint of S which is the IUT. Two possible cases arise during a communication between nodes on a particular route: either the communication between two successive nodes N_i and N_{i+1} succeeds, or it fails. We consider a communication as a success if a packet received by N_i is forwarded to N_{i+1} and forwarded after to N_{i+2} without provoking a RteError regardless of the meaning used for the acknowledgment.

The process of nodes self similarity may be illustrated as follows:

- Transmission success: If a transmission between N_i and N_{i+1} succeeds, we combine these two nodes in a new node $N_\circ$. The communications between N_i and N_{i+1} are considered as $N_\circ$ internal actions. If the communication between $N_\circ$ and N_{i+2} succeeds, we iterate the process and so on. Thus, in case that the packet from S reaches D without causing a RteError, we may combine all the intermediate nodes as illustrated in Fig. 2.

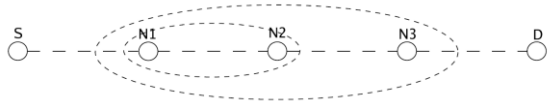


Fig. 2. Combination by self similarity when all communications succeed.

- Transmission failure: If a communication fails between N_i and N_{i+1}, it means that all the previous communications have succeeded. So the nodes between N_1 and N_i are combined. Finally, all the nodes after N_{i+1}, including D have the same behavior for an observer placed on the IUT. We therefore combine all the nodes from N_{i+1} to D into a new node D (Fig. 3).

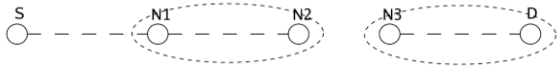


Fig. 3. Combination by self similarity when a communication fails.

With these definitions, the IUT behavior is not impacted by the path length when the test sequences are executed (except for the test sequence selection). However, even if the self similarity may reduce the specification from which test sequences are generated, their executions on the implementation and the relationship with this specification have to be defined. Due to the lack of space, we herein do not detail this relationship but the interested reader may have a look to [15].

The node self-similarity enables to represent a large class of topologies with a small number of nodes and to execute test sequences regardless of the number of intermediate nodes. Thereby we can reduce the number of nodes used in our specification in order to generate test scenarios.

5. An experimental study

In order to generate more accurate and shorter test scenario, we minimize our specification. Due to node self-similarity, we can decide to keep only the smallest number of nodes required to generate a test sequence according to specific test objectives. To test functional properties of the DSR protocol (Dynamic Source Routing), we did not find, from the requirements, test objectives requiring more than 2 routes into the network. Then, our specification is reduced to 4 nodes, *S*, *N*0, *N*1 and *D* which compose 2 routes [*S*,*N*0,*D*] and [*S*,*N*1,*D*] as represented in Fig. 4.

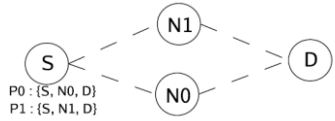


Fig. 4. Specification topology

This topology could represent a large class of real networks by node self-similarity and using two sets of routes during the execution of the test. Our main idea here is to create a relation between the specification and the implementation defined as:

Let S_{spec} and D_{spec} be respectively the representation of S and D in the specification $Spec$ and S_{imp}, D_{imp} their representation in the implementation Imp. Let $(p_n(x) \mid \{x, n\} \in \mathbb{N})$ be the n^{th} route chosen by S_{imp} to reach D_{imp} and composed by x nodes.

- In $Spec$:
  - $P0_{spec} = \{(S_{spec}, N0, D_{spec})\}$
  - $P1_{spec} = \{(S_{spec}, N1, D_{spec})\}$
- In $Imp : p_n(x) \in P(n \bmod 2)_{imp}$

All along the test execution, a *Test Coordination Procedure* (*TCP*) will preserve a relation between $P0_{spec}$ and $P0_{imp}$, and also between $P1_{spec}$ and $P1_{imp}$ assuming it as an elected strategy. Both sets save the theoretical RouteCache in the *TCP*. With respect to *Spec*, $P0_{imp}$ and $P1_{imp}$ match possible routes described in the specification. For instance, if a test sequence implies that *P*0*spec* disappears: the *TCP* will detect the RouteError packet as an input, will erase the first element of $P0_{imp}$, $p0(x)$ and will select $p1(y) \in P1_{imp}$ as the new route that *IUT* must use.

5.1. Dynamic Source Routing

Dynamic Source Routing (DSR) is a reactive protocol that discovers and maintains routes between nodes on demand [16]. It relies on two main mechanisms, Route Discovery and Route Maintenance. In order to discover a route between two nodes, DSR floods the network with a Route Request packet. This packet is forwarded only once by each node after concatenating its own address to the path. When the targeted node receives the Route Request, it piggybacks a Route Reply to the sender and a route is established. Each time a packet follows an established route, each node has to ensure that the link is reliable between itself and the next node. DSR provides three successive steps to perform this maintenance: link layer acknowledgment, passive acknowledgment and network layer acknowledgment.

5.2. Experiments

Once the *RouteDiscovery* has been performed for *S*, the IUT has a representation of the topology. From the viewpoint of S, the selected route can be simplified by self-similarity for each subsequent I/O as above explained. In order to insert the test scenario in the IUT, we use a UP and a LT linked to the TCP. In our example, a test scenario containing two objectives is used:

- After having sent a RReq packet, the IUT waits for at least one RteRep packet, before sending a SrcRte packet by the shortest route to the destination in its cache.
- If a RteError is received, the IUT will use the next shortest route to the destination in its cache to send a SrcRte containing the original message.

We detail hereafter the different steps of the execution in the network shown in Fig. 5. We decided to apply a direct emulation technique based on: (i) a simulator: it manages the nodes' mobility into the network and the communications with each other. A great majority of simulators currently integrates an emulator process, (ii) a focal machine: this machine hosts the network simulation whose the different components may be simulated or emulated (according to the options available in the simulators), and (iii) virtual machines: virtual nodes (simulated ones) communicating with the focal machines.

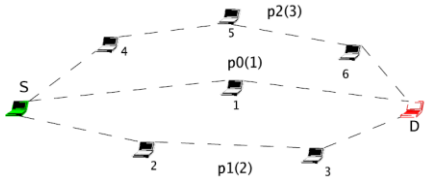


Fig. 5. An example of network.

A discrete events simulator such as NS-2 is sufficient and we will thus use NS-2e (NS-2 patched as emulator) [18]. Four virtual machines are executed on the focal machine that also runs the simulation side. Our approach is applied on experimentation through the DSR-UU implementation [19]. The test sequences are provided by one of our tools TESTGEN-SDL and some test purposes. Direct emulation is used. It allows using a real implementation of a protocol stack with a simulator to represent the mobility and to manage the communications. The direct emulation is performed on a focal machine with the following characteristics: Pentium M 1,6 GHz, 512 Mo Ram, Fedora-2.6.15 kernel with skas patch, TUN/TAP interfaces activated. We use User Mode Linux [20] to create virtual machines with existing prepared kernel and file system. DSR-UU was added in the kernel. NS-2 patched for emulation was performed to manage mobility and wireless communication between the virtual machines. If we want a large collection of nodes, it is necessary to distribute the virtual machines on more than one focal computer. The proposed emulation and testing architecture are depicted in Fig. 6.

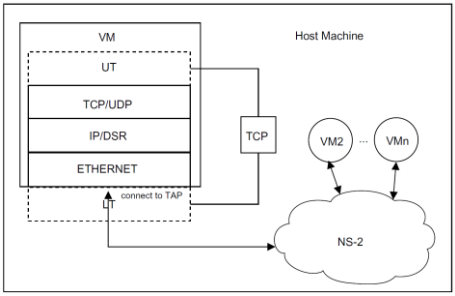


Fig. 6. Direct emulation and testing architecture.

We give next an example of the application of our approach through a testbed. This example illustrates the execution of a test scenario obtained from a formal specification on a real network taking into account the dynamic topologies. While we obtained ~95% of inconclusive testing verdicts without the nodes self similarity, we here reduce this rate to 5%.

1- UT injects a Packet in the IUT to the destination node D – pass.
2- LT observes an output RReq(S,D) – pass.
3- LT observes an input RReq(S,D) identical as the one sent where N0 has added its own address –pass.
4- LT observes an input RReq(S,D) identical as the one sent where N1 has added its own address –pass.
5- LT observes an input RRep(S,1,D) and TCP stores this route – pass.
6- LT observes an input RRep(S,2,3,D) and TCP stores this route – pass.
7- LT observes an input RRep(S,3,4,5,D) and TCP stores this route – inconclusive this packet is not expected but do not invalidate the conformance. TCP sorts the different routes by generating p0(1) = (S, 1,D), p1(2) = (S, 2, 3,D) and p2(3) = (S,4,5,6,D) which are put in the sets P0imp et P1imp as defined by the ‘‘shortest path’’ strategy. A route pointer defines which route the TCP expects the IUT to use. This pointer indicates the first element of P0imp, p0(1) = (S,1,D). The sets P0imp and P1imp in Imp are equivalent to P0spec and P1spec for the test scenario.
8- LT observes an output SrcR(p0(1)) and p0(1) $\in$ P0imp – pass because the route chosen in Imp is equivalent to the one chosen in Spec. From this step we can use Node self similarity to represent the route from the point of view of the IUT. We assume the link between node 1 and node D to be broken.
9- LT observes an inputRErr(1,D), TCP checks P0 et P1 and removes p0(1) from P0. The pointer moves on P1(1) = p1(2) = (S, 2, 3,D) – pass.
10- LT observes an outputSrcR(p1(2)) and p1(2) $\in$ P1 – pass because the route chosen in Imp is equivalent to the one chosen in Spec.
11- TCP, as the oracle, gives the final verdict PASS, each expected I/O has been observed.

6. Conclusion

We have presented the node self similarity (NSS) approach adapted and applied in a MANET. This approach allows to reduce the formal specification of the protocol by considering functional similarity of the nodes in the network. By the same way, we reduced the number of test cases but particularly, we did face to the changing topologies and the dynamicity of certain nodes. This approach has been applied to an implementation of the DSR protocol. Compared to experiments without our NSS TCP strategy, we note that we reduced the number of inconclusive verdicts obtained from the execution of test suites.

References

[1]. NS2, The network simulator. Available from: <http://www.isi.edu/nsnam/ns>, 2004.

[2]. OPNet, The opnet modeler. <http://www.opnet.com/products/modeler/home.html>, 2005.

[3]. Rita Dorofeeva, Khaled El-Fakih, Stephane Maag, Ana R. Cavalli, Nina Yevtushenko, FSM-based Conformance Testing Methods: a Survey annotated with Experimental Evaluation, in Elsevier Information and Software Technology, Vol. 52, p.1286-1297, 2010.

[4]. L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, M. Gerla, Glomosim: a scalable network simulation environment, Technical Report 990027, UCLA Computer Science Department, May 1999.

[5]. Andel, Todd R., and Alec Yasinsac. On the credibility of manet simulations. Computer 39 (7) (2006): 48-54.

[6]. K. Bhargavan, C. Gunter, I. Lee, O. Sokolsky, M. Kim, D. Obradovic, M. Viswanathan, Verisim: formal analysis of network simulations, IEEE Transactions on Software Engineering, 28 (2) (2002) 129‑145.

[7]. M. Kim, M. Viswanathan, H. Ben-Abdallah, S. Kannan, I. Lee, O. Sokolsky, Formally specified monitoring of temporal properties, in: Euromicro Conference on Real-Time Systems, 1999.

[8]. T. Lin, S.F. Midkiff, J.S. Park, A framework for wireless ad hoc routing protocols, Wireless Communications and Networking, 2, 2003 1162‑1167.

[9]. U. Glasser, Q.-P. Gu, Formal description and analysis of a distributed location service for mobile ad hoc networks, Theoretical Computer Science, 2003

[10]. I. Zakkuidin, T. Hawkins, N. Moffat, Towards a game theoretic understanding of ad hoc routing, Electronic Notes in Theoretical Computer Science, 2005, 119.

[11]. Besse, Cédric, Ana Cavalli, and David Lee. An automatic and optimized test generation technique applying to TCP/IP protocol. 14th IEEE International Conference on Automated Software Engineering, 1999.

[12]. Willcock, Colin, et al. Multi Component TTCN-3. An Introduction to TTCN-3, Second Edition, 77-98, 2011.

[13]. Yu, Wenjing, et al. TDL: a transformation description language from feature model to use case for automated use case derivation. Proceedings of the 18th International Software Product Line Conference-Volume 1. ACM, 2014.

[14]. Koceilah Merouane, Cyril Grepet, Stephane Maag, A Methodology for Interoperability Testing of a MANET Routing Protocol , The Third IEEE International Conference on Wireless and Mobile Communications ICWMC 2007.

[15]. S. Maag, C. Grepet and A. Cavalli, A formal validation methodology for MANET routing protocols based on nodes' self similarity, Computer Communications Journal, Vol.31:4, pp. 827-841, 2008

[16]. D. Johnson, D. Maltz, Y.-C. Hu, The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR) ‑ Experimental RFC, IETF MANET Working Group. July 2004.

[17]. Djouvas, C., Griffeth, N. D., & Lynch, N. A. Testing Self-Similar Networks. Electronic Notes in Theoretical Computer Science, 164 (4), 67-82, 2006.

[18]. NS-2 emulator. Available from: <http://www.isi.edu/nsnam/ns/ns-emulation.html>.

[19]. E. Nordstrom, Dsr-uu v0.1. Available from: <http://core.it.uu.se/core/index.php/DSR-UU>, Uppsala University

[20]. J. Dike, user-mode-linux. Available from: <http://user-mode-linux.sourceforge.net/>.

[21]. R.Hierons et al., Using formal specifications to support testing. ACM Computing Surveys, page 41(2):176, 2009.
[22]. ITU-T, Recommendation Z.100: CCITT Specification and Description Language (SDL), Technical Report ITU-T, 1999.
[23]. Daniel Hiranandani, Katia Obraczkaand J.J Garcia-Luna-Aceves, Manet protocol simulations considered harmful: The case for benchmarking. IEEE Wireless Communications, vol. 20, no 4, 2013.

Тестирование в беспроводных самоорганизующихся сетях на основе формальных моделей

Стефан Мааг <Stephane.Maag@telecom-sudparis.eu>
Institut Mines-Telecom/Telecom SudParis, CNRS UMR 5157,
9 rue Charles Fourier, 91011 Evry Cedex, France

Аннотация. Тестирование является одним из ключевых этапов разработки сложных взаимодействующих систем. Использование формальных моделей при тестировании позволяет автоматизировать процесс генерации тестовых последовательностей по формальному описанию спецификации и дальнейшего тестирования реальной системы. Несмотря на то, что тестирование на основе формальных моделей достаточно хорошо развито, использование этого подхода при тестировании протоколов маршрутизации в беспроводных самоорганизующихся сетях (ad-hoc) требует решения ряда специальных возникающих проблем, поскольку отличительной чертой беспроводных мобильных самоорганизующихся сетей (MANET) является отсутствие предопределенной инфраструктуры и отсутствие управляющих узлов. В данной работе, для формального описания спецификации используется модель расширенного конечного автомата, описанная в терминах языка SDL. Для уменьшения числа неопределенных вердиктов, часто возникающих в традиционном тестировании на основе моделей, рассматривается самоподобие узлов сети. Практическая значимость предложенного подхода иллюстрируется эмуляцией тестирования протокола DSR (Dynamic Source Routing).

Ключевые слова: тестирование на основе формальных моделей; мобильные самоорганизующиеся сети; самоподобие узлов; протоколы маршрутизации.

Список литературы

[1]. NS2, The network simulator. Available from: <http://www.isi.edu/nsnam/ns>, 2004.
[2]. OPNet, The opnet modeler. <http://www.opnet.com/products/modeler/home.html>, 2005.
[3]. Rita Dorofeeva, Khaled El-Fakih, Stephane Maag, Ana R. Cavalli, Nina Yevtushenko, FSM-based Conformance Testing Methods: a Survey annotated with Experimental Evaluation, in Elsevier Information and Software Technology, Vol. 52, p.1286-1297, 2010.
[4]. L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, M. Gerla, Glomosim: a scalable network simulation environment, Technical Report 990027, UCLA Computer Science Department, May 1999.
[5]. Andel, Todd R., and Alec Yasinsac. On the credibility of manet simulations. Computer 39 (7) (2006): 48-54.
[6]. K. Bhargavan, C. Gunter, I. Lee, O. Sokolsky, M. Kim, D. Obradovic, M. Viswanathan, Verisim: formal analysis of network simulations, IEEE Transactions on Software Engineering, 28 (2) (2002) 129‑145.

[7]. M. Kim, M. Viswanathan, H. Ben-Abdallah, S. Kannan, I. Lee, O. Sokolsky, Formally specified monitoring of temporal properties, in: Euromicro Conference on Real-Time Systems, 1999.
[8]. T. Lin, S.F. Midkiff, J.S. Park, A framework for wireless ad hoc routing protocols, Wireless Communications and Networking, 2, 2003 1162‑1167.
[9]. U. Glasser, Q.-P. Gu, Formal description and analysis of a distributed location service for mobile ad hoc networks, Theoretical Computer Science, 2003
[10]. I. Zakkuidin, T. Hawkins, N. Moffat, Towards a game theoretic understanding of ad hoc routing, Electronic Notes in Theoretical Computer Science, 2005, 119.
[11]. Besse, Cédric, Ana Cavalli, and David Lee. An automatic and optimized test generation technique applying to TCP/IP protocol. 14th IEEE International Conference on Automated Software Engineering, 1999.
[12]. Willcock, Colin, et al. Multi Component TTCN-3. An Introduction to TTCN-3, Second Edition, 77-98, 2011.
[13]. Yu, Wenjing, et al. TDL: a transformation description language from feature model to use case for automated use case derivation. Proceedings of the 18th International Software Product Line Conference-Volume 1. ACM, 2014.
[14]. Koceilah Merouane, Cyril Grepet, Stephane Maag, A Methodology for Interoperability Testing of a MANET Routing Protocol , The Third IEEE International Conference on Wireless and Mobile Communications ICWMC 2007.
[15]. S. Maag, C. Grepet and A. Cavalli, A formal validation methodology for MANET routing protocols based on nodes' self similarity, Computer Communications Journal, Vol.31:4, pp. 827-841, 2008
[16]. D. Johnson, D. Maltz, Y.-C. Hu, The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR) ‑ Experimental RFC, IETF MANET Working Group. July 2004.
[17]. Djouvas, C., Griffeth, N. D., & Lynch, N. A. Testing Self-Similar Networks. Electronic Notes in Theoretical Computer Science, 164 (4), 67-82, 2006.
[18]. NS-2 emulator. Available from: <http://www.isi.edu/nsnam/ns/ns-emulation.html>.
[19]. E. Nordstrom, Dsr-uu v0.1. Available from: <http://core.it.uu.se/core/index.php/DSR-UU>, Uppsala University
[20]. J. Dike, user-mode-linux. Available from: <http://user-mode-linux.sourceforge.net/>.
[21]. [21] R.Hierons et al., Using formal specifications to support testing. ACM Computing Surveys, page 41(2):176, 2009.
[22]. ITU-T, Recommendation Z.100: CCITT Specification and Description Language (SDL), Technical Report ITU-T, 1999.
[23]. Daniel Hiranandani, Katia Obraczkaand J.J Garcia-Luna-Aceves, Manet protocol simulations considered harmful: The case for benchmarking. IEEE Wireless Communications, vol. 20, no 4, 2013

Locating a faulty component of an EFSM composition

Svetlana Prokopenko <s.prokopenko@sibmail.com>
Tomsk State University,
634050, 36 Lenin av., Tomsk,Russia

Abstract. When a component of a discrete event system is faulty there is a problem how to locate a faulty component. In this paper, we consider the composition of two Extended Finite State Machines and propose an approach for locating a faulty component using preset and adaptive experiments with Finite State Machines.

Keywords: Extended Finite State Machine; Finite State Machine; l-equivalent; transfer and output faults.

1. Introduction

Telecommunication systems are multi-component systems. When some of components are faulty the behavior of the whole system can be different from that of the specification system. The problem arises how to locate faulty components. In this paper, we propose an approach for locating a faulty component in sequential composition of two Extended Finite State Machines (EFSM). The joint behavior of these components is presented as a composed EFSM. We assume that only one component can have transfer or output faults. Faults of each type are described using a Mutation Machine. Since there are no formal methods for distinguishing two mutation EFSMs, we unfold those EFSMs as classical Finite State Machines (FSM) and use preset and adaptive experiments with FSM for distinguishing two EFSMs.

2. Preliminaries

A Finite State Machine (FSM) [1] A is a 5-tuple (S, I, O, h, s_0) where S is the non-empty finite set of states with the initial state s_0, I and O are input and output alphabets, $h \subseteq S \times I \times O \times S$ is a transition relation. An FSM is called *complete* if for any pair $(s, i) \in S \times I$ there exists a transition $(s, i, o, s') \in h$; otherwise the FSM is *partial*. If for every pair $(s, i) \in S \times I$ there exists at most one transition $(s, i, o, s') \in h$ then the FSM is called *deterministic*; otherwise the FSM is *non-deterministic*. If for any triple $(s, i, o) \in S \times I \times O$ there exists at most one state

$s' \in S$ such that a transition $(s, i, o, s') \in h$ then the FSM is called *observable*, otherwise, the FSM is *non-observable*.

An Extended FSM (EFSM) [2] M is a pair (S, T) where S is the set of states and T is the set of transitions between states of the set S, such that each transition $t \in T$ is a 7-tuple (s, i, P, op, up, o, s'), where s and s' are the initial and final states of the transition t; $i \in I$ is an input with the set $D_{inp\text{-}i}$ of *input parameter vectors*; $o \in O$ is an output with the set $D_{out\text{-}o}$ of *output parameter vectors*; $P: D_{inp\text{-}i} \times D_V \rightarrow \{\text{True, False}\}$ is a *predicate* where D_V is the set of *context vectors*; $op: D_{inp\text{-}i} \times D_V \rightarrow D_{out\text{-}o}$ is an output parameter *update* function; $vp: D_{inp\text{-}i} \times D_V \rightarrow D_V$ is a context *update* function.

A *configuration* of an EFSM M is a pair $(s, \mathbf{v})$ where s is a state of the EFSM and $\mathbf{v}$ is a context vector. A pair $(i, \boldsymbol{\rho})$ is a *parameterized* input symbol where $\boldsymbol{\rho} \in D_{inp\text{-}i.}$ An input sequence of parameterized inputs is a *parameterized input sequence*; however some of inputs may be non-parameterized.

An EFSM is *complete* if for each state s with an appropriate context vector $\mathbf{v}$ and any parameterized input $(i, \boldsymbol{\rho})$ there exists at least one transition (s, i, P, op, up, o, s') with the predicate that is true for given $(s, \mathbf{v})$ and $(i, \boldsymbol{\rho})$, otherwise the EFSM is *partial*. If for each state s with an appropriate context vector $\mathbf{v}$ and any parameterized input $(i, \boldsymbol{\rho})$ there exists at most one transition (s, i, P, op, up, o, s') for which the predicate is true for given $(s, \mathbf{v})$ and $(i, \boldsymbol{\rho})$ then the EFSM is called *deterministic*; otherwise the EFSM is *non-deterministic*.

A transition $t = (s, i, P, op, up, o, s')$ of an EFSM M has a *transfer* fault if the final state s'' of an implementation transition $t' = (s, i, P, op, up, o, s'')$ is different from the final state s' of the transition t. A transition $t = (s, i, P, op, up, o, s')$ has an *output* fault if an output symbol o' of a transition $t' = (s, i, P, op, up, o', s')$ is different from that of the transition t. An output fault of a transition t means that the output of the transition t has been changed when the specification EFSM was implemented. Transfer and output faults model many other faults in an EFSM.

Given an EFSM, its behavior can be described by a corresponding FSM. States of an FSM are configurations of an EFSM, inputs and outputs correspond to parameterized inputs and outputs of the EFSM. The FSM at current state $(s, \mathbf{v})$ under a parameterized input $(i, \boldsymbol{\rho})$ verifies which transition is valid for current values of context vector $\mathbf{v}$ and input parameter vector $\boldsymbol{\rho}$, calculates new value $\mathbf{v}'$ of context vector and output parameter vector $\boldsymbol{\mu}$, produces a parameterized output $(o, \boldsymbol{\mu})$ and moves to a final state s' of the corresponding transition. The corresponding transition of the FSM has the initial state $(s, \mathbf{v})$, input $(i, \boldsymbol{\rho})$, output $(o, \boldsymbol{\mu})$ and final state $(s', \mathbf{v}')$.

If a corresponding FSM is big enough the behavior can be described over all input sequences of length l [3], i.e., an FSM that is an l-equivalent of the initial EFSM can be derived. One can simulate a behavior of the FSM under all input sequences of length l accepted at the initial state of the FSM. In general case, the l-equivalent is a

partial (possible non-observable) FSM and its behavior coincides with that of the original FSM under all input sequences of length *l*.

Given two FSMs over the same input and output alphabets, the FSMs can be distinguished by a preset or adaptive distinguishing experiment. A preset distinguishing test case is an input sequence such that the output responses to this sequence of the two FSMs do not intersect. Since a corresponding FSM for a given EFSM can be partial and non-observable, an approach proposed in [4] can be applied. Sometimes when two FSMs cannot be distinguished by a preset distinguishing experiment they still can be distinguished by an adaptive distinguishing experiment.

An adaptive distinguishing test case is a single-input output-complete connected initialized FSM that has a finite number of traces. In other words, at each intermediate state of the test case only one input with all possible outputs is defined. Such a test case first has been derived for two states of an observable FSM [5, 6] and then was extended for any number of states of a non-observable FSM [7]. The test case is a distinguishing test case for the set of states of a given FSM if each trace of the test case from the initial to a deadlock state is a trace at most at one state of the given set of states.

Consider an observable FSM *S* with two initial states s_0 and s_1 in Fig. 1. Let *S*/1 denote the FSM *S* with initial state s_0 while *S*/2 denotes the FSM *S* with initial state s_1.

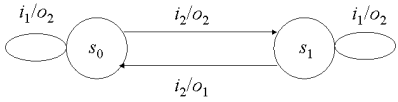


Fig. 1. FSM S with two initial states s_0 and s_1.

A test case *P* over alphabets $I = \{i_1, i_2\}$ and $O = \{o_1, o_2\}$ is an adaptive distinguishing test case of *S*/1 and *S*/2 and is shown in Fig. 2.We also notice that the machines can be separated with a single input, namely, the input i_2.

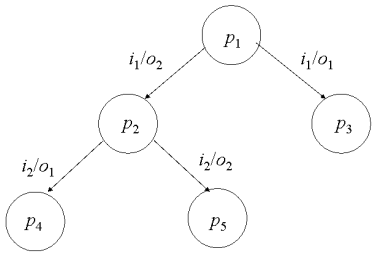


Fig. 2. A test case P.

3. Fault detection in sequential composition of two EFSMs

Consider a sequential composition N of two components A_1 and A_2 in Fig. 3.

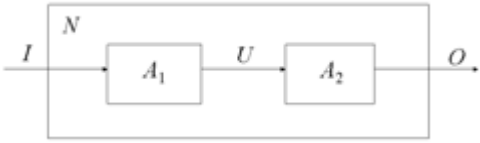


Fig. 3. A sequential composition of two EFSMs A_1 and A_2.

Let $A_1 = (C, T_1)$ and $A_2 = (Q, T_2)$ be completely specified and deterministic EFSMs. We suppose that the set of outputs of A_1 coincides with the set of inputs of A_2 which are not parameterized, i.e., predicates of the EFSM A_2 depend only on context variables. Each transition of an EFSM A_1 is a 6-tuple $(c, i, P_1, up_1, u_{out1}, c')$ and every transition of an EFSM A_2 is a 7-tuple $(q, u_{inp2}, P_2, op_2, up_2, o, q')$.

The composition of A_1 and A_2 is an EFSM $N = (Z, R)$ where $Z = C \times Q$ and $R = \{(z, i, P_1 \,\&\, P_2, op_2, \{up_1, up_2\}, o, z') \mid \exists\, (c, i, P_1, up_1, u_{out1}, c') \in T_1 \;\exists\, (q, u_{inp2}, P_2, op_2, up_2, o, q') \in T_2\, (u_{out1} = u_{inp2}),\; z = (c, q),\; z' = (c', q')\}$ [8].

As a formal model, we consider a composition of two Extended Finite State Machines and suppose that only one of two components can be faulty. Moreover, we assume that the faulty component has transfer and/or output faults. In this paper, we describe such faults using a special EFSM called a Mutation Machine *MM* [9].

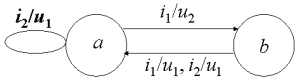
Let the component A_1 have transfer or output faults and EFSMs MM_{Tr1} and MM_{Out1} describe these faults in the component A_1. We note that mutation machines can become non-deterministic. EFSMs $MM_{Tr1} \,@\, A_2$ and $MM_{Out1} \,@\, A_2$ correspond to a behavior of faulty composition N. Let the component A_2 have transfer or output faults and EFSMs MM_{Tr2} and MM_{Out2} describe these faults in A_2. Let EFSMs $A_1 \,@\, MM_{Tr2}$ and $A_1 \,@\, MM_{Out2}$ correspond to the behavior of a faulty composition N.

To detect transfer or output faults in the composition N one can use a strategy proposed in [4]. An input sequence α is called a *separating* sequence for states s and s' of an FSM if it is defined at states s and s' and sets of outputs to this sequence at both states do not intersect. An input sequence α is called a separating sequence for two FSMs if it separates initial states of FSMs.

If two FSMs that correspond to $A_1 \,@\, MM_{Tr2}$ and $MM_{Tr1} \,@\, A_2$ (or $A_1 \,@\, MM_{Out2}$ and $MM_{Out1} \,@\, A_2$) without traces of the specification composed FSM can be distinguished by a preset or adaptive experiment, i.e., there exists a separating sequence or an adaptive test case such that after applying this test case, then we could learn which component is faulty after applying such a test case. If there is no chance to construct FSMs corresponded to $A_1 \,@\, MM_{Tr2}$ and $MM_{Tr1} \,@\, A_2$ (or $A_1 \,@\, MM_{Out2}$ and $MM_{Out1} \,@\, A_2$) due to their complexity then an approach based on l-equivalents can be used. For each mutation machine input sequences and

corresponding traces are derived for both machines one by one and each two traces are checked for the distinguishability. In this case, it is unnecessary to construct the composed EFSM; it is sufficient to model the behavior of the compositions on input sequences of length *l*. Experiments with telecommunication protocols [10] show that more than 80 % of transfer and output faults in protocol implementations can be detected using 5-equivalents of corresponding EFSMs.

An example of utilizing a proposed approach for sequential composition of two FSMs is given below. Consider component FSMs in Figs 3a and 3b and assume that a transition shown in bold can have any transition or output fault.



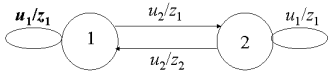


Fig. 3a. The head component FSM. *Fig. 3b. The tail component FSM.*

After deleting the specification traces from the component FSMs the mutation machines shown in Figs 4a and 4b are obtained.

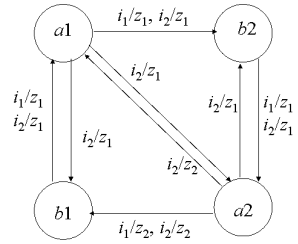
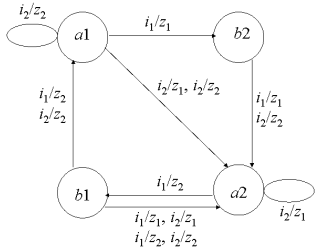


Fig. 4a. The mutation machine for the head component FSM



.Fig. 4b. The mutation machine for the tail component FSM.

By direct inspection, one can assure that the machines in Figs 4a and 4b can be distinguished by the input sequence $i_2i_2i_1$. If the output sequence to i_2 is z_2 then only the tail component can be faulty. If the output sequence to i_2 is z_1 then the next input i_2 is applied. If the output to the next input i_2 is z_1 then only the head component can

be faulty. If the output sequence to i_2 is z_2 then the next input i_1 is applied. Output sequences to i_1 are different and both mutation machines are distinguished.

In case when a separating sequence does not exist, one can try to construct an adaptive distinguishing test case [7]. If an adaptive distinguishing test case exists then based on the output response of the composition to this adaptive test case one can conclude which component FSM is faulty.

4. Conclusion

In the paper, we have discussed how to locate a faulty component in sequential composition of two EFSMs. The corresponding *l*-equivalents of mutation EFSMs which describe component faults can be derived based on the composed EFSM as well as by simulating the composition behavior under input sequences of length *l*. Moreover, the use of *l*-equivalents simplifies the problem of deleting the specification traces from the mutation machine and the approach allows eliminating the state explosion problem of an EFSM unfolding when *l* is not very big. Additional research is necessary in order to evaluate the integer *l* for appropriate cases, for example, for locating faults in protocol implementations as well as for considering other kinds of faults such as predicate faults and faults in the implementations of the update functions.

Acknowledgment

This research is partially supported by project 739 (Goszadanie RF).

References

[1]. T. Villa, N. Yevtushenko, A. Mishenko, R. K.Brayton, A. Petrenko, A. Sangiovanni-Vincentelli The unknown component problem: theory and applications. – Berlin: Springer, 2012. 311 p.

[2]. A. Petrenko, S. Boroday, and R. Groz. Confirming Configurations in EFSM Testing, IEEE Trans. Software Eng. 30(1), 2004. pp. 29-42.

[3]. V. Karibskiy, P. Parhomenko, E. Sogomonyan, V. Halchev. Basics of technical diagnostics, M.: Energya, 1976 (in Russian).

[4]. N. Kushik, N. Yevtushenko, A. Cavalli. On testing against partial non-observable specifications. QUATIC 2014 : 9th International Conference on Quality of Information and Communication Technology, Sept. 23-26, 2014, Portugal.

[5]. R. Alur, C. Courcoubetis, M. Yannakakis. Distinguishing tests for nondeterministic and probabilistic machines. In Proc. of the 27th ACM Symposium on Theory of Computing, 1995. pp. 363-372.

[6]. A. Petrenko, N. Yevtushenko, G. v. Bochmann. Testing Deterministic Implementations from their Nondeterministic Specifications. In Proc. of the IFIP Ninth International Workshop on Testing of Communicating Systems, 1996. pp. 125-140.

[7]. N. Kushik, K. El-Fakih, N. Yevtushenko, A. R. Cavalli: On adaptive experiments for nondeterministic finite state machines. Software Tools for Technology Transfer, Springer, DOI 10.1007/s10009-014-0357-7 (2014) (in press).

[8]. Fedoseev A.O. Kompozicija rasshirennyx avtomatov [Composition of Extended Finite State Machines]. Diplomnaja rabota [Diploma project], Tomsk, 2005 (in Russian).
[9]. Kolomeets A.V. Algoritmy sinteza proverjajuwix testov dlja upravljajuwix sistem na osnove rasshirennyx avtomatov [Diagnostic test derivation methods for telecommunication systems based on an EFSM model]. Dissertacija na soiskanie uchenoj stepeni kandidata texnicheskix nauk [PhD thesis], Tomsk, 2010. 129 p.
[10]. N. Kushik, M. Forostyanova, S. Prokopenko, and 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, ISBN: 978-1-63248-051-4, DOI 10.15224/ 978-1-63248-051-4-94.

Локализация неисправной компоненты в композиции расширенных автоматов

Светлана Прокопенко <s.prokopenko@sibmail.com>
НИ ТГУ, 634050, Россия, г. Томск, пр. Ленина, дом 36

Аннотация. Проблема локализации неисправной компоненты в автоматной сети хорошо известна, и в данной работе мы решаем эту проблему для бинарной сети из расширенных автоматов. Для каждой из компонент строится мутационный расширенный автомат, описывающий наиболее вероятные неисправности компоненты. Для композиции мутационного автомата со спецификацией другой компоненты посредством моделирования определяется древовидный конечный автомат, поведение которого совпадает с поведением исходного расширенного автомата на всех последовательностях длины не больше l (l-эквивалент). Из l-эквивалентов удаляются вход-выходные последовательности, принадлежащие композиции-спецификации, и для полученных мутационных l-эквивалентов строится диагностический эксперимент. Если такой диагностический эксперимент существует, то по реакции композиции, предъявленной для тестирования, достаточно часто можно определить, какая из компонент является неисправной, при условии, что неисправности возможны только в одной компоненте.

Ключевые слова: расширенный автомат, конечный автомат, l-эквивалент, ошибки переходов и выходов

Список литературы

[1]. T. Villa, N. Yevtushenko, A. Mishenko, R. K.Brayton, A. Petrenko, A. Sangiovanni-Vincentelli The unknown component problem: theory and applications. – Berlin: Springer, 2012. 311 p.
[2]. A. Petrenko, S. Boroday, and R. Groz. Confirming Configurations in EFSM Testing, IEEE Trans. Software Eng. 30(1), 2004. pp. 29-42.
[3]. V. Karibskiy, P. Parhomenko, E. Sogomonyan, V. Halchev. Basics of technical diagnostics, M.: Energya, 1976 (in Russian).
[4]. N. Kushik, N. Yevtushenko, A. Cavalli. On testing against partial non-observable specifications. QUATIC 2014 : 9th International Conference on Quality of Information and Communication Technology, Sept. 23-26, 2014, Portugal.
[5]. R. Alur, C. Courcoubetis, M. Yannakakis. Distinguishing tests for nondeterministic and probabilistic machines. In Proc. of the 27th ACM Symposium on Theory of Computing, 1995. pp. 363-372.

[6]. A. Petrenko, N. Yevtushenko, G. v. Bochmann. Testing Deterministic Implementations from their Nondeterministic Specifications. In Proc. of the IFIP Ninth International Workshop on Testing of Communicating Systems, 1996. pp. 125-140.

[7]. N. Kushik, K. El-Fakih, N. Yevtushenko, A. R. Cavalli: On adaptive experiments for nondeterministic finite state machines. Software Tools for Technology Transfer, Springer, DOI 10.1007/s10009-014-0357-7 (2014) (in press).

[8]. Fedoseev A.O. Kompozicija rasshirennyx avtomatov [Composition of Extended Finite State Machines]. Diplomnaja rabota [Diploma project], Tomsk, 2005 (in Russian).

[9]. Kolomeets A.V. Algoritmy sinteza proverjajuwix testov dlja upravljajuwix sistem na osnove rasshirennyx avtomatov [Diagnostic test derivation methods for telecommunication systems based on an EFSM model]. Dissertacija na soiskanie uchenoj stepeni kandidata texnicheskix nauk [PhD thesis], Tomsk, 2010. 129 p.

[10]. N. Kushik, M. Forostyanova, S. Prokopenko, and 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, ISBN: 978-1-63248-051-4, DOI 10.15224/ 978-1-63248-051-4-94.

Building direct and back spanning trees by automata on a graph

Igor Burdonov <igor@ispras.ru>
Alexander Kossatchev <kos@ispras.ru>
Institute for System Programming Russian Academy of Sciences,
109004, A. Solzhenitsina, 25, Moscow, Russia

Abstract. The paper presents a parallel graph exploration algorithm. Automaton on a graph is an analogue of the Turing machine — tape cells correspond to graph vertices, where the automaton can store some data, and moves along the tape correspond to moves along graph arcs. This system can be considered also as an aggregate of finite automatons located in graph vertices and interacting by message sending. Each automaton changes its state according to the data stored in the corresponding vertex, and moves along graph arcs are replaced with messages sent by the automaton of the arc's starting vertex to the one of the ending vertex. The suggested parallel graph exploration algorithm has worst case working time bound O(n/k+D), where n is the number of vertices, and D is the graph diameter, the maximum length of simple path (non-self intersecting path). As a result the algorithm builds two spanning trees of the graph: the direct spanning tree, which has the root vertex as its tree root and is directed from the root, and the back spanning tree, directed to the root.

Keywords: directed graphs; graph exploration, group of automata; graph spanning tree

1. Introduction

A task of graph exploration with a goal to uncover a structure of unknown graph by moving along its arcs can be met in many domains. In this paper we regard as possible applications exploration of networks and exploration of Web-application structure. In many cases such an exploration can be considered as being performed by agents working in graph vertices and sending each other messages along graph arcs.

Graph exploration starts from some specified vertex, called root vertex. Directed graph exploration isn't a trivial task. In the worst case it takes time of an order *O(mn)*, where *n* — the number of graph vertices, *m* — the number of graph arcs. This holds for various algorithms of graph exploration based on breadth-first or depth-first traversal [[1],[2],[3]].

In 1966 M. O. Rabin posed the problem of directed graph exploration with a finite automaton [[4]]. Automaton on a graph is an analogue of the Turing machine — tape cells correspond to graph vertices, where the automaton can store some data,

and moves along the tape correspond to moves along graph arcs. This system can be considered also as an aggregate of finite automatons located in graph vertices and interacting by message sending. Each automaton changes its state according to the data stored in the corresponding vertex, and moves along graph arcs are replaced with messages sent by the automaton of the arc's starting vertex to the one of the ending vertex.

Messages are both input and output symbols of vertex automata. If message size and number of each automaton states are bounded globally, all automata are just finite state machines. Usual graph exploration corresponds to possibility for a single message to have a size linear on the number of vertices. The most efficient known algorithm of graph exploration with a single finite automaton is suggested in [5] and has worst case working time $O(nm+n^2 loglogn)$. If the traversal is repeated by message interchange between automata located in vertices known after the first exploration worst case working time becomes $O(nm+n^2 l(n))$, where $l(n)$ is the number of times logarithm calculation is repeated until $1 \le log(log...(n)...) < 2$ holds [[6]].

In this paper we consider parallel exploration of a graph — many messages can walk through its arcs in parallel. Working time estimation in this case depends on *the capacity* of an arc k, which means the maximum number of messages that can be transmitted through a single arc simultaneously.

The suggested parallel graph exploration algorithm has worst case working time bound $O(n/k+D)$, where n is the number of vertices, and D is the graph diameter, the maximum length of simple path (non-self intersecting path). As a result the algorithm builds two spanning trees of the graph: *the direct spanning tree*, which has the root vertex as its tree root and is directed from the root, and *the back spanning tree*, directed to the root. The detailed description and proofs of all statements can be found in [[7]].

2. Spanning trees building algorithm

Sending a message an automaton working in some vertex should specify the arc, by which this message should be delivered. We count that arcs starting in some vertex are numbered and an arc is specified by its number. We also suppose that operation time of an automaton is negligibly small and the time of message transport through an arc is bounded by some constant.

Let us denote s the maximum number of arcs starting in the same vertex of the graph, then $m \le ns$. We suggest the algorithm of spanning trees building with the following features:

- vertex automaton memory is bounded by $\boldsymbol{O}(nDlog\ s)$,
- message size is bounded by $\boldsymbol{O}(Dlog\ s)$,
- arc capacity is k,
- algorithm worst case working time is $\boldsymbol{O}(n/k + D)$.

Below we call *direct arcs* the arcs belonging to the direct spanning tree constructed by the algorithm, *chords* — all other arcs, *back arcs* — arcs belonging to the back spanning tree. A back arc can be a chord or a direct arc. *Path vector* is the list of arc numbers along the path. *Vertex vector* is the vector of the simple path. leading to this vertex from the root along the direct spanning tree. The root has empty vertex vector ε. The size of simple path vector or vector of a simple cycle is ***O****(Dlogs)*. A message in the algorithm below consists of a constant number of simple path vectors, or ***O****(Dlogs)* bits.

The algorithm is partitioned in four parts. The first part describes building of the back spanning tree, the second part performs the check that back spanning tree is completely constructed, the third part depicts classification of arcs starting in one vertex, the fourth one describes maintenance of incoming back arc counters in vertices.

The first part uses messages of four kinds: **Start**, **Root search**, **Direct,** and **Back**. **Start** message is sent by the root automaton to automata of all other vertices, it contains the vertex vector and initiates vertex automaton operation, which is started by sending **Root search** messages. **Root search** messages are sent through all outgoing arcs in such a way that they pass some simple path to the root vertex and report the root automaton the vector of this simple path. In response root vertex automaton sends **Direct** message, which reaches the **Root search** initiator and reports it the back simple path vector, computed by root search. The initiator then sends **Back** message, which sets some tags along the back simple path.

The second part has the task to determine that the back spanning tree is completely constructed. It uses arc counting — when the arc counter in the root vertex becomes zero, the tree building is complete. For each arc $a \rightarrow b$ the root automaton gets first the message from a, where "+1" is set for the arc $a \rightarrow b$, and then it gets the message from b, where "-1" is set for $a \rightarrow b$. The second message comes from b later than messages from the same b, setting "+1" for all arcs starting in b. For this goal the modified **Root search** and two additional message kinds — **Finish** and **Minus** — are used. **Root search** contains the number of arcs starting from its initiator. When the root automaton gets **Root search** message, it adds this number to the global counter of outgoing arcs. **Finish** message is sent from a vertex after getting **Direct** message. After sending **Finish** message and setting back arc in the vertex, the **Minus** message is send. The root automaton decreases its arc counter by 1 after getting **Minus** message.

During the third part of the algorithm the outgoing arcs from each vertex are marked as either direct arcs or chords. At first they all are considered as chords. Then each arc, along which **Direct** message is sent, is marked as a direct arc.

During the fourth part the incoming back arc counters are set in all vertices. Two message kinds — **Start counting** and **End counting** — are used for that. **Start counting** messages move from the root along direct arcs to all other vertices, **End counting** message move from each vertex along back arcs and each vertex counts such messages created in the start vertices of back arcs ending in this vertex.

3. Conclusion

At the end of algorithm work each vertex automaton stores type of each outgoing arc and the number of incoming back arcs. These data can further be used for parallel computation of some functions of values stored in graph vertices.

References

[1]. Steven S. Skiena. The Algorithm Design Manual. Springer-Verlag, New York, 1997.
[2]. I. B. Burdonov, A. S. Kossatchev, V. V. Kuliamin. Irredundant Algorithms for Traversing Directed Graphs: The Deterministic Case. Programming and Computer Software, 29(5):245-258, 2003.
[3]. I. B. Burdonov, A. S. Kossatchev, V. V. Kuliamin. Irredundant Algorithms for Traversing Directed Graphs: The Nondeterministic Case. Programming and Computer Software, 30(1):2-17, 2004.
[4]. M.O. Rabin. Maze Threading Automata. An unpublished lecture presented at MIT and UC, Berkeley, 1967.
[5]. I. B. Burdonov. Traversal of an unknown directed graph by a finite automaton. Programming and Computer Software, 30(4): 11-34, 2004.
[6]. I. B. Burdonov. Backtracking on a tree in traversal of an unknown directed graph by a finite automaton. Programming and Computer Software, 30(6): 6-29, 2004.
[7]. I. B. Burdonov, A. S. Kossatchev, V. V. Kuliamin. Parallel computations on graphs. Programming and computer Software, 41(1): 1-13, 2015.

Построение прямого и обратного остовов автоматами на графе

Игорь Бурдонов <igor@ispras.ru>
Александр Косачев <kos@ispras.ru>
Институт Системного Программирования РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.

Аннотация. В работе представлен алгоритм параллельного исследования графа. Автомат на графе является аналогом машины Тьюринга – ячейки ленты соответствуют вершинам графа, где автомат может сохранять некоторую информацию, а движение по ленте соответствует движениям по дугам графа. Такая система может рассматриваться также как совокупность автоматов, размещенных в вершинах графа и взаимодействующих путем посылки сообщений по дугам. Каждый автомат изменяет свое состояние в соответствии с данными, сохраняемыми в вершине, а движение по дугам заменяется посылкой сообщений. Время работы предлагаемого алгоритма параллельного исследования графа ограничено сверху O(n/k+D), где n – число вершин графа, D – диаметр графа, максимальная длина простого пути (пути без самопересечений). В результате работы алгоритма строится два остова графа: прямой остов, корнем которого является корневая вершина графа, ориентированный от корня, и обратный остов, ориентированный к корню.

Keywords: directed graphs; graph exploration, group of automata; graph spanning tree

Список литературы

[1]. Steven S. Skiena. The Algorithm Design Manual. Springer-Verlag, New York, 1997.
[2]. И.Б. Бурдонов, А.С. Косачев, В.В. Кулямин. "Неизбыточные алгоритмы обхода ориентированных графов. Детерминированный случай" // Программирование, 2003 г., №5, с. 59-69.
[3]. И.Б. Бурдонов, А.С. Косачев, В.В. Кулямин. "Неизбыточные алгоритмы обхода ориентированных графов. Недетерминированный случай" // Программирование, 2004 г., №1, с. 2-17.
[4]. M.O. Rabin. Maze Threading Automata. An unpublished lecture presented at MIT and UC, Berkeley, 1967.
[5]. И.Б. Бурдонов. "Обход неизвестного ориентированного графа конечным роботом" // Программирование, 2004 г., № 4, с. 11-34.
[6]. И.Б. Бурдонов. "Проблема отката по дереву при обходе неизвестного ориентированного графа конечным роботом" // Программирование, 2004 г., № 6, с. 6-29.
[7]. .Б. Бурдонов, А.С. Косачев, В.В. Кулямин. "Параллельные вычисления на графе"// Программирование, 2015 г., № 1, с. 3-20.

Parallel calculations by automata on direct and back spanning trees of a graph

Igor Burdonov <igor@ispras.ru>
Alexander Kossatchev <kos@ispras.ru>
Victor Kuliamin <kuliamin@ispras.ru>
Institute for System Programming Russian Academy of Sciences,
109004, A. Solzhenitsina, 25, Moscow, Russia

Abstract. The paper presents a parallel computation algorithm of an arbitrary function value on a multiset of values distributed on directed graph vertices. The computation is performed by message passing executed by automata distributed on the graph vertices. The key idea of the algorithm is to use a structural information on the graph that can be extracted by its parallel exploration and encoded into structures of direct and back spanning trees of the graph, which require only finite number of bits in each graph vertex, and to represent the function calculated as a composition of so called aggregate function and another one. Aggregate functions are characterized by possibility to calculate their value on a union of multisets by aggregating their values on separate multisets, that makes them easy for parallel computation.

Keywords: parallel computation; directed graphs; group of automata; graph spanning tree

1. Introduction

A task of graph exploration with a goal to uncover a structure of unknown graph by moving along its arcs can be met in many domains. In this paper we regard as possible applications exploration of networks and exploration of Web-application structure. In many cases such an exploration can be considered as being performed by agents working in graph vertices and sending each other messages along graph arcs.

Sometimes graph exploration is performed to make possible calculation of some function of values stored in graph vertices. Since different vertices can store equal values, such a function is a function of a multiset of values. We consider situation where calculation is initiated by an external stimulus processes by the automaton located in some root vertex of the graph, and the value calculated is sent by this root automaton to the requester.

Graph processing algorithms and their efficiency highly depend on the known information on the graph, in the worst case in the initial state the graph can be completely unknown and an algorithm should gather information on its structure

itself by walking through arcs. Here we suppose that vertex automata are in such states, that contains the following information on the graph structure. Graph has specified direct spanning tree, having the tree root coinciding with the root vertex of the graph and directed from the root, and back spanning tree directed to the root. Arcs not belonging to the direct spanning tree are called chords. Arcs of the direct spanning tree are called direct arcs, of the back one — back arcs. A back arc can be a direct one or a chord. Each vertex automaton stores arc kind (direct, chord, back + direct, back + chord) for each arc outgoing from its vertex. Also vertex automaton stores the number of back arcs incoming to its vertex. Such data can be gathered during previous exploration of the graph described in our paper "Building direct and back spanning trees by automata on a graph".

Function calculation by vertex automata is performed with pulse algorithm. Its idea is first to send request message from the root vertex to all other vertices, then to send response message from each vertex to the root one. It allow to calculate in parallel any function of multiset of values stored in vertex automata (we also sometimes say that they are stored in graph vertices). The worst case working time of the algorithm is $O(D)$, where D is the graph diameter, the maximum length of non-self intersecting path. The details of the algorithm and proofs of all statements can be found in [[1]].

2. Aggregate Functions and Aggregate Extension

Pulse algorithm computes so-called aggregate functions, for which function value on a union of multisets can be computed using function values on each of multisets in the union. Here we give definition of an aggregate function and aggregate extension of any given function $f(x)$, which helps to represent it as $h(g(x))$, where g is aggregate. For each f there exists single (up to isomorphism) minimal aggregate extension, which provide a minimum information sufficient to compute f. Aggregate functions theory presented here is a modification of inductive functions theory given in [[2]].

Below we consider functions on finite multisets, which elements belong to some base set X. The set of all finite multisets of elements of X we denote as X^{-}. Note that union of multisets gives a multiset having all the elements of united ones with multiplicity of any element being a sum of its multiplicities in united multisets.

An aggregate function $g:X^{-}\to A$ – is such a function on multisets that

$$\exists e:A\times A\to A\ \ \forall a,b\in X^{-}\ \ g(a\cup b) = e(g(a),g(b)).$$

An aggregate extension of function $f:X^{-}\to A$ is an aggregate function $g:X^{-}\to B$, such that $\exists h : B\to A\ \ \forall a\in X^{-}\ \ f(a) = h(g(a))$.

An aggregate extension g of f is such an aggregate function, that using its values one can compute f. Of course, some extensions are not useful, e.g., taking an identity

function g on X^{-}, and taking f itself as h, one doesn't simplify calculations. To avoid such useless solutions, we use a minimal aggregate extension — intuitively, an aggregate function, which give a minimum information sufficient to calculate f.

An aggregate extension $g : X^{-} \to B$ of $f : X^{-} \to A$ is called *minimal*, if $g(X^{-}) = B$ and $\forall g` : X^{-} \to C$, aggregate extension of $f$, $\exists i : C \to B\ g = ig`$.

Aggregate extension of $f : X^{-} \to A$ exists and is unique up to one-to-one mapping.

3. Pulse Algorithm

Pulse algorithm is intended to calculate a value of f on a multiset $x \in X^{-}$ of values stored in the graph vertices. We suppose that some value is stored in any vertex with multiplicity 1. Pulse algorithm uses the graph structure information stored in vertex automata as it is described in the Introduction.

Pulse algorithm uses two message kinds: **Request** and **Response**. At first the root vertex automaton gets **Request** from some external source, this message provides three functions: h, e, and g. This message is then transmitted along direct arcs to all other vertices, and all vertex automata store e and g. Each vertex automaton then calculates the value of g on values stored in the vertices of subtree with the root coinciding with this vertex of back spanning tree, and sends the value obtained as a parameter of **Response** message along back arcs. The root vertex automaton calculates $g(x)$, and then sends **Response** to the request initiator with the value $f(a)=h(g(x))$.

Pulse algorithm worst case working time is $\boldsymbol{O}(D)$.

References

[1]. I. B. Burdonov, A. S. Kossatchev, V. V. Kuliamin. Parallel computations on graphs. Programming and computer Software, 41(1): 1-13, 2015.

[2]. Kushnirenko A. G., Lebedev G. V. Programming for mathematicians. Nauka, Moscow, 1988. (in Russian).

Параллельные вычисления автоматами на прямом и обратном остовах графа

Игорь Бурдонов <igor@ispras.ru>
Александр Косачев <kos@ispras.ru>
Виктор Кулямин <kuliamin@ispras.ru>
Институт Системного Программирования РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.

Аннотация. В статье представлен алгоритм параллельного вычисления произвольной функции от мультимножества значений, находящихся в вершинах графа. Вычисления выполняются автоматами, размещенными в вершинах графа и обменивающихся сообщениями по дугам графа. В основе алгоритма лежит использование информации о структуре графа, извлеченной в результате параллельного исследования графа и представляющей собой прямой и обратный остовы графа. Для хранения этой информации требуется конечное число бит в каждой вершине графа. Вычисление значения той или иной функции основано на алгоритме пульсации: сначала от автомата начальной вершины по всему графу распространяются сообщения-вопросы, которые должны достигнуть каждой вершины, а затем от каждой вершины «в обратную сторону» к начальной вершине двигаются сообщения-ответы. Алгоритм пульсации, по сути, вычисляет агрегатные функции, для которых значение функции от объединения мультимножеств вычисляется по значениям функции от этих мультимножеств. Показано, что любая функция f(x) имеет агрегатное расширение, то есть может быть вычислена как h(f`(x)), где f` агрегатная функция.

Keywords: параллельные вычисления; ориентированный граф; коллектив автоматов; остов

Список литературы

[1]. .Б. Бурдонов, А.С. Косачев, В.В. Кулямин. "Параллельные вычисления на графе"// Программирование, 2015 г., № 1, с. 3-20.
[2]. Кушнеренко А.Г., Лебедев Г.В. "Программирование для математиков", Наука, Главная редакция физико-математической литературы, Москва, 1988.

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.

К минимизации автоматов с таймаутами

Твардовский Александр Сергеевич <tvardal@mail.ru>
Национальный исследовательский Томский Государственный университет, 634050, проспект Ленина, 36, Томск, Россия

Аннотация. В данной статье предлагается алгоритм минимизации конечного полностью определенного автомата с таймаутами. Показывается, что в отличие от классических конечных автоматов, для таких автоматов минимальная форма не является единственной с точностью до изоморфизма.

Ключевые слова: конечные автоматы, минимизация, таймауты, приведённая форма.

1. Введение

Модель конечного автомата широко используются при анализе и синтезе дискретных систем. Однако в ряде случаев при описании поведения реальных систем необходимо учитывать временные аспекты, и соответственно, понятие конечного автомата необходимо расширить. Известно несколько способов введения временной переменной в конечный автомат [1, 2]. Поскольку сложность решения многих задач в теории автоматов существенно зависит от числа состояний рассматриваемого автомата, большое внимание уделяется задаче минимизации временного автомата, т. е. построению эквивалентного автомата с наименьшим числом состояний. В данной работе мы предлагаем алгоритм минимизации конечного автомата с входными и выходными таймаутами.

2. Основные определения и обозначения

Автоматом с таймаутами или в данной работе *временным автоматом* называется пятёрка $S = (S, I, O, h_S, \Delta_S)$, где I – *входной* алфавит, O – *выходной* алфавит, S – конечное непустое множество *состояний*, $h_S \subseteq (S \times I \times O \times S \times Z)$ – *отношение переходов*, Z – множество целых неотрицательных чисел, определяющих число единиц времени между поступлением входного символа и выдачей выходного, Δ_S: $S \to S \times (\mathrm{N} \cup \{\infty\})$ – функция *таймаута*, определяющая для каждого состояния максимальное время ожидания входного символа, N – множество натуральных чисел. Иными словами, если в некотором состоянии автомата входной сигнал не поступает в течение определенного времени (таймаута), то автомат может изменить своё

состояние. Например, если $\Delta_S(s) = (s', T)$ и в состоянии s в течение T единиц времени на автомат не было подано ни одного входного символа, то автомат переходит в состояние s'. После перехода в состояние s' отсчет времени начинается с 0. Если $\Delta_S(s) = (s, \infty)$, то автомат может ожидать входной символ в состоянии s бесконечно долго. Если в автомате есть переход (s, i, o_1, s_1, d) и входной символ будет подан в состоянии s менее чем через T единиц времени, то автомат перейдет в состояние s_1 и через d единиц времени выдаст выходной символ. Автомат с таймаутами называется *детерминированным*, если в нём не может существовать двух кортежей вида (s, i, o_1, s_1, d_1), $(s, i, o_2, s_2, d_2) \in h_s$, в которых $o_1 \neq o_2$, $d_1 \neq d_2$ или $s_1 \neq s_2$. Автомат с таймаутами называется *полностью определённым*, если для любой пары $(s, i) \in S \times I$, то есть для любого входного символа i, поступающего на вход автомата в состоянии s, существует кортеж $(s, i, o, s', d) \in h_s$.

Временным входным символом называется пара (i, t), где i – символ входного алфавита, t – время поступления входного символа после перехода автомата в текущее состояние. *Временным выходным символом* называется пара (o, d), где o – символ выходного алфавита, d – число единиц времени между подачей входного символа и выдачей выходного символа. Последовательность временных входных символов (i_1, t_1), (i_2, t_2) … (i_n, t_n) называется *временной входной последовательностью*; последовательность временных выходных символов (o_1, d_1), (o_2, d_2) … (o_n, d_n) называется *временной выходной последовательностью*. Для каждого состояния s автомата с таймаутами введём функцию $time(s, t) = s'$, которая определяет, в каком состоянии s' находится автомат через t тактов времени после перехода в состояние s, при условии, что входной символ не был подан. Временная выходная последовательность, соответствующая временной входной последовательности α, поступившей на автомат в состоянии s, называется (*выходной*) *реакцией* автомата в состоянии s на последовательность α. В настоящей работе рассматриваются только детерминированные полностью определённые временные автоматы, т. е. в каждом состоянии автомата каждой временной входной последовательности соответствует единственная выходная последовательность.

Состояния s и p двух временных автоматов *эквивалентны*, если выходные реакции автоматов в этих состояниях на каждую временную входную последовательность совпадают. Полностью определенный детерминированный автомат называется *приведенным*, если любые два состояния в автомате не являются эквивалентными. Два автомата *эквивалентны*, если для каждого состояния одного автомата существует эквивалентное состояние другого автомата, и наоборот.

Для нахождения разбиения на эквивалентные состояния для полностью определенного детерминированного автомата с таймаутами построим соответствующий конечный автомат [1]. Для автомата на рис. 1 соответствующий конечный автомат приведен на рис. 2. Таким образом, если

таймаут в состоянии s равен T, то в классический автомат вводятся дополнительные состояния $(s, 1)$, …, $(s, T-1)$. Подобно [4], временной выходной символ (o, d) соответствует тому, что выходной символ выдается через d единиц времени после поступления входного символа. Таким образом имеет место следующее утверждение.

Утверждение 1. Для временного автомата с таймаутами число состояний соответствующего конечного автомата равно сумме числа состояний, в которых таймаут равен ∞, с суммой таймаутов в состояниях, в которых таймаут не равен ∞.

В нашем примере соответствующий конечный автомат (рис. 2) имеет 9 состояний. Как показано в [1], конечные автоматы, построенные для автоматов с таймаутами, эквивалентны, если и только если эквивалентны исходные временные автоматы. Для конечного автомата, соответствующего автомату с таймаутами, можем построить разбиение на эквивалентные состояния [5].

Теорема 2. Состояния s_1 и s_2 исходного автомата с таймаутами S эквивалентны, если и только если в соответствующем конечном автомате эквивалентными являются состояния $(s_1, 0)$ и $(s_2, 0)$.

Доказательство. Пусть состояния s_1 и s_2 автомата с таймаутами S эквивалентны. Допустим, что состояния $(s_1, 0)$ и $(s_2, 0)$ соответствующего конечного автомата не являются эквивалентными, т.е. для них существует различающая последовательность. Тогда по правилам построения найдётся различающая последовательность и для состояний s_1 и s_2, что противоречит утверждению, что состояния s_1 и s_2 эквивалентны.

Пусть состояния $(s_1, 0)$ и $(s_2, 0)$ эквивалентны в построенном конечном автомате. Тогда по правилам построения, для любого $t = 1, \ldots, n, \ldots$ состояния $time(s_1, t)$ и $time(s_2, t)$ являются эквивалентными; последнее означает эквивалентность состояний s_1 и s_2. ♦

Пример. Построим разбиение на эквивалентные состояния для автомата на рис. 1.

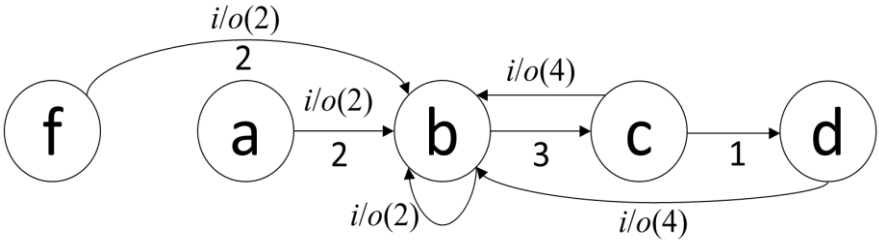


Рис 1. Автомат с таймаутами S.

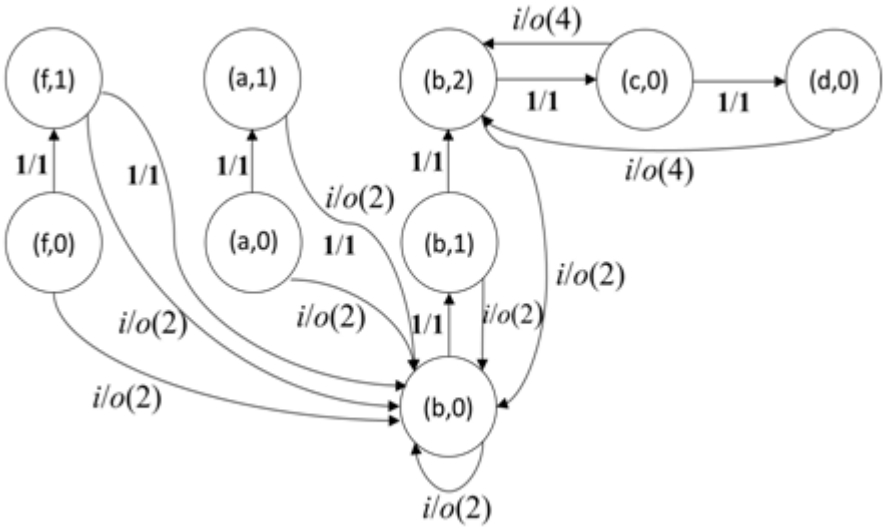


Рис 2. Классический конечный автомат FSM(S), соответствующий автомату с таймаутами на рис. 1.

Для этого рассмотрим соответствующий конечный автомат на рис. 2. Непосредственной проверкой можно убедиться, что для автомата на рис. 2 разбиение на эквивалентные состояния имеет следующий вид: {(*a*, 0), (*f*, 0)}, {(*a*, 1), (*f*, 1)}, {(*b*, 0)}, {(*b*, 1)}, {(*b*, 2)}, {(*c*, 0), (*d*, 0)}. Таким образом, эквивалентными состояниями в автомате с таймаутами на рис. 1 являются состояния *a* и *f*, *c* и *d*.

3. Построение приведённой формы автомата с таймаутами

Алгоритм построения приведенной формы B полностью определённого детерминированного автомата с таймаутами S включает следующие шаги.

Шаг 1: По исходному автомату с таймаутами S строится соответствующий классический конечный автомат FSM(S), для которого строится разбиение τ множества состояний на эквивалентные состояния.

Шаг 2: Строим разбиение π состояний автомата S на эквивалентные состояния следующим образом: состояния s_1 и s_2. автомата S принадлежат одному блоку разбиения π, если и только если состояния (s_1, 0) и (s_2, 0) принадлежат одному блоку разбиения τ.

Шаг 3: Строится приведенная форма B автомата S**.** Входной и выходной алфавиты автомата B совпадают с таковыми для автомата S, состояния b_1, b_2, …, b_l соответствуют блокам B_1, B_2, …, B_l разбиения π. Множество переходов

h_B автомата B формируем следующим образом: для состояний b_i и b_j существует кортеж $(b_i, i, o, b_j, d) \in h_B$, тогда и только тогда, когда существуют $s_i \in B_i$ и $s_j \in B_j$, такие что кортеж $(s_i, i, o, s_j, d) \in h_S$. Для функции таймаута $\Delta_B(b_i) = (b_j, T)$ тогда и только тогда, когда $\Delta_S(s_i) = (s_j, T)$, $s_i \in B_i$ и $s_j \in B_j$ и $i \neq j$.

Утверждение 3. Пусть B – автомат, построенный для автомата S по выше описанному алгоритму. Автомат B является детерминированным полностью определенным автоматом, и состояние b_i автомата B эквивалентно состоянию s автомата S, если и только если $s \in B_i$.

Доказательство. Автомат B является полностью и определенным автоматом, поскольку автомат S обладает этими свойствами. Детерминированность автомата B следует из того факта, что любые два состояния s, s', принадлежащие одному блоку B_i, являются эквивалентными (теорема 2).

Покажем, что состояния $s \in B$ и b, соответствующее блоку B разбиения π эквивалентны. Для любого кортежа $(s_1, i, o_1, s_2, d_1) \in h_s$, существует кортеж $(b_1, i, o_1, b_2, d_1) \in h_B$, т. е. реакции автоматов B и S в состояниях b и s на любой входной символ совпадают. Поскольку $s_2 \in B_2$, то для состояний s_2 и b_2, и для всех последующих их преемников по любому входному символу i можно провести аналогичные рассуждения. Кроме того, для любого $\Delta_S(s_1) = (s_3, T)$ существует $\Delta_B(b_1) = (b_3, T)$, где $s_3 \in B_3$, т.е реакции автоматов B и S в состояниях b и s на любую временную входную последовательность совпадают. Покажем теперь, что состояния $s_j \in B_j$ и b_i, где $i \neq j$, не эквивалентны. Состояние $s_i \in B_i$ эквивалентно состоянию b_i по первой части доказательства. Состояния s_i и s_j не эквивалентны, так как находятся в различных блоках разбиения π. Таким образом, состояния s_j и b_i не эквивалентны. ♦

Следствие 1. Автомат B, построенный на Шаге 3, есть приведённый автомат.

Следствие 2. Автомат B, построенный на Шаге 3, эквивалентен автомату S.

Таким образом, на основании следствий 1 и 2 справедлива следующая теорема.

Теорема 4. Пусть S – полностью определённый детерминированный автомат с таймаутами. Полностью определённый детерминированный автомат B, построенный по выше описанному алгоритму, является приведённой формой автомата S.

4. Свойства приведённой формы автомата с таймаутами

В работе [4] показывается, что для полностью определенных детерминированных автоматов с временными ограничениями приведенная форма единственна с точностью до изоморфизма. К сожалению, это свойство не выполняется для автоматов с таймаутами. В [3] было показано, что для инициального временного автомата с таймаутами могут существовать несколько попарно неизоморфных приведенных форм. Приведенные формы

могут оказаться неизоморфными, поскольку некоторые цепочки переходов по таймаутам могут быть объединены в один переход. Изменяя немного пример из работы [3], можно сделать такое же заключение и для неинициальных автоматов (рис 3). В частности, в состоянии *p* отсутствует переход по таймауту 2, который есть в состоянии *a* другой приведенной формы этого же автомата.

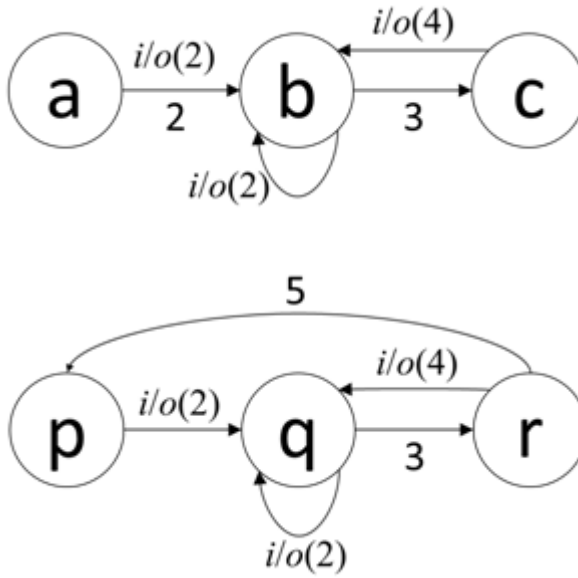


Рис.3. Два эквивалентных приведённых, но неизоморфных автомата с таймаутами.

Тем не менее, в отличие от инициальных автоматов, для неинициальных автоматов с таймаутами все приведенные формы автомата имеют одно и то же число состояний.

Утверждение 5. Число состояний двух эквивалентных приведённых автоматов с таймаутами совпадает.

Доказательство утверждения следует из того факта, что каждому состоянию одного автомата соответствует эквивалентное состояние другого автомата, и два любых состояния в одном автомате не являются эквивалентными. ♦

5. Заключение

В настоящей статье была рассмотрена задача минимизации полностью определённых детерминированных автоматов с таймаутами и предложен алгоритм построения приведенной формы для такого автомата. Показано, что в отличие от автоматов с временными ограничениями для автомата с

таймаутами может существовать несколько попарно неизоморфных приведенных форм. Вопрос о необходимых и достаточных условиях существования единственной (с точностью до изоморфизма) приведенной формы, так же, как и установление соответствия между приведенными формами автоматов с таймаутами и временными ограничениями, требует дальнейших исследований. Полученные результаты могут быть использованы при генерации тестов с гарантированной полнотой для систем, поведение которых описывается автоматами с таймаутами.

В заключение автор выражает благодарность профессору Н. В. Евтушенко за интерес к работе и полезные дискуссии.

Список литературы

[1]. *Bresolin D., El-Fakih K., Villa T., Yevtushenko N.* Timed Finite State Machines: Equivalence checking and expressive power. Intern Conf. CANDALF'2014, pp. 203-216.

[2]. *Кондратьева О.В.* Разработка методов синтеза и анализа композиций временных автоматов: магистерская диссертация на соискание степени магистра радиофизики. Томск, 2012. 72 с.

[3]. *Кондратьева. О.В.* Минимизация временных автоматов с таймаутами // Материалы конференции "Новые информационные технологии в исследовании сложных структур". – Томск : Издательский дом Томского государственного университета, 2014. – 132 с.

[4]. *Твардовский А., Евтушенко Н.,* К минимизации автоматов с временными ограничениями. "Вестник ТГУ. Управление, вычислительная техника и информатика". – Томск : Издательский дом Томского государственного университета, № 4, 2014. 77-83 с.

[5]. *Gill A.* Introduction to the Theory of Finite-State Machines. M.: Nauka, 1966. 272 p.

On the minimization of timed Finite State Machines

Alexandre Tvardovskii <tvardal@mail.ru>
Tomsk State University, 634050, Russia, Tomsk, av. Lenina, 36

Abstract. This paper addresses the problem of minimizing a Finite State Machine (FSM) augmented with input and output timeouts, since almost all methods for deriving complete test suites are developed for reduced (minimal) timed machines, i.e., FSMs where every two states are not equivalent. If at some state no input is applied until the corresponding (input) timeout expires then the FSM can spontaneously move to another prescribed state. An output timeout describes the time that is necessary for executing a transition that is the number of time instances needed for producing an output after an input has been applied. A technique for minimizing such machines based on corresponding classical FSMs is proposed; it is also shown that differently from classical FSMs, an FSM with timeouts can have minimal forms which are not pair-wise isomorphic.

Keywords: FSM minimization, input and output timeouts, minimal (reduced) form.

References

[1]. Bresolin D., El-Fakih K., Villa T., Yevtushenko N. Timed Finite State Machines: Equivalence checking and expressive power. Intern Conf. CANDALF'2014, pp. 203-216.
[2]. Kondratyeva O. Razrabotka metodov sinteza i analiza kompozitsiy vremennykh avtomatov : magisterskaya dissertatsiya na soiskanie stepeni magistra radiofiziki. Tomsk, 2012. 72 p.
[3]. Kondratyeva O. Minimizatsiya vremennykh avtomatov s taymautami // Materialy konferentsii " Novye informatsionnye tekhnologii v issledovanii slozhnykh struktur ". – Tomsk: Izdatel'skiy dom Tomskogo gosudarstvennogo universiteta , 2014. 132 p.
[4]. Tvardovskiy A., Yevtushenko N. Minimizing timed Finite State Machines. Tomsk State University Journal of control and computer science. – Tomsk: Izdatel'skiy dom Tomskogo gosudarstvennogo universiteta , № 4, 2014, pp. 77-83.
[5]. Gill A. Introduction to the Theory of Finite-State Machines. M.: Nauka, 1966,. 272 p.

Решение автоматных уравнений для временных автоматов относительно параллельной композиции

[1,2] *О.В. Кондратьева <olga.kondratyeva@telecom-sudparis.eu>*
[1] *Н.В. Евтушенко <yevtushenko@sibmail.com>*
[2] *А.Р. Кавалли <ana.cavalli@telecom-sudparis.eu>*
[1] *НИ ТГУ, 634050, Россия, г. Томск, пр. Ленина, дом 36*
[2] *Телеком Южный Париж, 91000, Франция, г. Эври, ул. Чарльза Фурье, дом 9*

Аннотация. Рассматривается задача синтеза неизвестной компоненты, которая при совместной работе с известной частью системы удовлетворяет заданной спецификации, в случае, когда поведение компонент и спецификация системы в явном виде зависят от времени. Результаты, известные для решения параллельного автоматного уравнения для классических конечных автоматов, обобщаются на случай автоматов с таймаутами. Задача синтеза неизвестной компоненты возникает во многих приложениях и в статье иллюстрируется ее применение для реализации композиции сервисов с заданными ограничениями на временные параметры качества.

Ключевые слова: временной автомат, параллельная композиция, автоматное уравнение, наибольшее решение, ограничения внешней среды.

1. Введение

При синтезе и анализе многокомпонентных дискретных систем нередко возникает задача синтеза неизвестной компоненты, которая при совместной работе с известной частью системы удовлетворяет заданной спецификации. Как показано в [1, 2] данная задача часто может быть формализована в виде решения соответствующего языкового или автоматного уравнения, однако в разработанных алгоритмах явно не учитываются временные аспекты поведения систем. В данной работе мы рассматриваем модель автомата с таймаутами, далее временной автомат [3, 4], расширенную на случай возможного недетерминизма по времени задержек обработки входных символов. В [5] уже было показано, что такое расширение оказывается необходимым при описании совместного поведения компонент, допускающих недетерминизм по входных символам.

Реальные системы всегда функционируют в некоторой среде, которая может воздействовать на систему и в которой возможно наблюдение за реакциями системы. Свойства среды влияют как на управляемость и наблюдаемость

системы, так и могут накладывать дополнительные ограничения на поведение системы. Использование таких ограничений внешней среды, как частичная управляемость, инерционность (конечность обработки сигналов средой и системой), конечность времени ожидания ответа от используемого приложения, может оказаться полезным для повышения эффективности разработки функциональных и качественных требований, для оптимизации многокомпонентных систем, а также при построении тестов. Поэтому мы рассматриваем построение композиции и решение уравнений для временных автоматов с учетом дополнительных ограничений, накладываемых на среду и/или компоненты системы.

Статья организована следующим образом: в разделе 2 вводятся необходимые определения и обозначения для автомата с таймаутами и рассматриваются ограничения для внешней среды через поведение максимального автомата. Раздел 3 посвящен построению параллельной композиции, для которой в разделе 4 приводится формула для нахождения наибольшего решения при различных ограничениях для внешней среды. В разделе 5 обсуждаются перспективные направления для дальнейших исследований.

2. Временной автомат с таймаутами

Функционирование автомата всегда осуществляется в некоторой *внешней среде*, предполагаемой или заданной явно. Свойства среды влияют как на возможность или невозможность подачи и наблюдения определенных символов (управляемость и наблюдаемость системы), так и могут накладывать дополнительные ограничения на поведение системы.

В общем случае, без дополнительных ограничений, внешняя среда, в которой функционирует временной автомат, удовлетворяет следующим предположениям и требованиям: 1) со стороны внешней среды на автомат могут поступить только символы из некоторого входного алфавита I, и со стороны автомата во внешнюю среду поступают символы из некоторого выходного алфавита O; 2) выходной символ может быть получен внешней средой только после подачи на автомат некоторого входного символа, т.е. входной символ всегда предшествует выходному символу; 3) после получения автоматом входного символа, возможно, по истечении некоторого времени, будет произведен некоторый выходной символ; 4) внешняя среда является достаточно «медленной», т.е.следующий входной символ может быть подан на систему, поведение которой описано временным автоматом, не ранее, чем был получен выходной символ на предыдущее входное воздействие. Описанная подобным образом среда может рассматриваться как наиболее общая для функционирования автоматов с заданными входным и выходным алфавитами, так как выполнение данных требований не накладывает никаких дополнительных ограничений на поведение и структуру автоматов.

Как обычно, **N** далее обозначает множество натуральных чисел.

Автоматом с таймаутами [3, 5] или в данной статье просто *временным автоматом* называется семёрка $S = (S, I, O, s_0, \lambda_S, \Delta_S, \sigma_S)$, где пятёрка $(S, I, O, s_0, \lambda_S)$ есть классический конечный автомат, дополненный *функцией задержки входного символа* $\Delta_S: S \to S \times (\mathbf{N} \cup \{\infty\})$ и *функцией задержки выходного символа* $\sigma_S: \lambda_S \to 2^{(\mathbf{N} \cup \{0\})} \setminus \varnothing$, где S – конечное непустое множество состояний с выделенным начальным состоянием s_0, I и O – конечные входной и выходной алфавиты соответственно, $\lambda_S \subseteq S \times I \times S \times O$ – отношение переходов. Функция Δ_S (далее также называемая функцией таймаута) определяет для каждого состояния максимальное время ожидания входного символа (таймаут) и состояние, в которое автомат перейдёт по истечении таймаута в случае, если ни один входной символ не поступил. Функция σ_S определяет для каждого перехода непустое, возможно бесконечное множество допустимых интервалов времени, необходимых для выполнения соответствующего перехода и получения выходного символа в ответ на поступившее входное воздействие (выходной таймаут). Аналогично [5], далее мы предполагаем, что множество значений σ_S для каждого перехода может быть представлено множеством значений конечного набора целочисленных линейных функций вида $\{b + k \cdot t \mid b, k \in \{0\} \cup \mathbf{N}\}$.

Свойства детерминированности и полной определенности формулируются для временных автоматов аналогично конечным автоматам. Автомат называется *полностью определенным*, если для любого $s_1 \in S$ и любого $i \in I$ существует, по крайней мере, одна пара $(s_2, o) \in S \times O$, такая, что $(s_1, i, s_2, o) \in \lambda_S$, в противном случае автомат называется *частичным*. Временной автомат называется *i-детерминированным*, или *детерминированным по входным символам*, если для любых состояния $s_1 \in S$ и входного символа $i \in I$ определено не более одной пары $(s_2, o) \in S \times O$ такой, что $(s_1, i, s_2, o) \in \lambda_S$, в противном случае, автомат называется *i-недетерминированным*. Временной автомат называется *t-детерминированным*, или *детерминированным по времени*, если для каждого перехода $(s_1, i, s_2, o) \in \lambda_S$ определено единственное значение допустимой задержки выхода, т.е. $|\sigma_S((s_1, i, s_2, o))| = 1$, в противном случае, автомат называется *t-недетерминированным*. Временной автомат называется *детерминированным*, если он *i-* и *t-*детерминированный, в противном случае – *недетерминированным*.

Поведение временного автомата существенно зависит от временных параметров, поэтому для его описания вводятся понятия временных входного и выходного символов [3]. *Временной входной символ* $(i, t) \in I \times (\{0\} \cup \mathbf{N})$, поданный на автомат в текущем состоянии s, означает, что входной символ i подается на автомат по истечении t единиц времени с момента достижения автоматом состояния s или по истечении t единиц времени с момента наблюдения предыдущего выходного символа. *Временной выходной символ* $(o, k) \in O \times (\{0\} \cup \mathbf{N})$ показывает, что символ o выдается автоматом через k единиц времени после получения входного символа. Для вычисления реакции автомата на временной входной символ (i, t), поданный в состоянии s, на

основе таймаутов вычисляется состояние, в которое автомат перейдет из состояния sпо истечении времени t [3]: входной символ iпоступит на автомат в состоянии s_p, в котором оканчивается последовательность переходов по таймаутам $\Delta_S(s) = (s_1, T_1)$, $\Delta_S(s_1) = (s_2, T_2)$, …, $\Delta_S(s_{p-1}) = (s_p, T_p)$ такая, что $T_1 + T_2 + \ldots + T_{p-1} < t$, но $T_1 + T_2 + \ldots + T_p \geq t$. Отметим, что если $\Delta_S(s) = (s, \infty)$, то $s_p = s$ при любом t. Таким образом, в результате подачи символа (i, t) в состоянии s автомат может выдать выходной символ o через k единиц времени, если и только если существует переход $(s_p, i, s', o) \in \lambda_S$ и $k \in \sigma_S((s_p, i, s', o))$, и в этом случае временной выходной символ (o, k) есть реакция автомата на временной входной символ (i, t) в состоянии s.

Рассмотрим временную входную последовательность $\alpha = (i_1, t_1)(i_2, t_2)\ldots(i_n, t_n)$ и временную выходную последовательность $\beta = (o_1, k_1)(o_2, k_2)\ldots(o_n, k_n)$. Если в автомате S существует такая последовательность переходов, что выходная последовательность β может быть реакцией автомата в состоянии s на входную последовательности α, то пара $\alpha\beta$ называется *временной вход-выходной последовательностью* или *временной трассой* автомата S в состоянии s. Множество всех временных трасс автомата S в состоянии s обозначим $trace_S(s)$.

Два состояния s и p временных автоматов S и P называются *эквивалентными*, если множества трасс в этих состояниях совпадают, то есть $trace_S(s) = trace_P(p)$. Автоматы S и P называются *эквивалентными*, если их начальные состояния эквивалентны. Автомат S есть *редукция* автомата P, обозначение $S \leq P$, если $trace_S(s_0) \subseteq trace_P(p_0)$.

Множество всех временных трасс автомата S может быть представлено соответствующим конечным полуавтоматом $Aut(S)$ с выделенным специальным символом $1 \notin I \cup O$ для обозначения ожидания в течение единицы времени, такого, что полуавтомат $Aut(S)$ распознает последовательность $\alpha = 1^{t_1} i_1 1^{k_1} o_1 \ldots 1^{t_m} i_m 1^{k_m} o_m$, если и только если $(i_1, t_1).(o_1, k_1) \ \ldots \ (i_m, t_m).(o_m, k_m)$ есть временная трасса автоматаS[5]. Конструктивно, соответствующий полуавтомат может быть построен заменой каждого перехода временного автомата на цепочку переходов, в которой переходы по входному и выходному символу разделены требуемым числом переходов по символу 1. Переходы по таймауту представляются цепочкой переходов по символу 1 через «копии» начального состояния цепочки.

Утверждение 1[5]. Временные автоматы S и P эквивалентны, если и только если эквивалентны соответствующие им полуавтоматы $Aut(S)$ и$Aut(P)$. Автомат S есть редукция автомата P, если и только если $Aut(S)$ есть редукция$Aut(P)$.

Язык, распознаваемый полуавтоматом $Aut(S)$, будем называть (*развернутым по времени*) *языком временного автомата*S и обозначать $L(S)$.

Максимальным временным автоматом в алфавитах I и O будем называть такой временной автомат $M_{I,O}$, который содержит в себе любой временной автомат S как редукцию. Язык максимального временного автомата обозначим $L(M_{I,O}) = [(1)^*I(1)^*O]^*(1)^*$. По определению, язык $L(S)$ некоторого временного автомата является $[(1)^* \cup (1)^*I(1)^*O]$-префикс-замкнутым подмножеством языка $L(M_{I,O})$, то есть таким, что для любых последовательностей $\alpha \in L(M_{I,O})$ и $b \in [(1)^* \cup (1)^*I(1)^*O]$ справедливо: если $\alpha b \in L(S)$, то и $\alpha \in L(S)$. Отметим, что язык полуавтомата, соответствующего временному автомату, распознает не только трассы временного автомата, но и все их продолжения с суффиксами из $(1)^*$. Данные суффиксы отражают тот факт, что после выдачи последнего выходного символа автомат не прекращает свое функционирование и не осуществляет сброс в начальное состояние (если не сказано иное), и может продолжать изменять состояние под действием таймаутов.

Максимальным временным автоматом в заданной среде будем называть максимальный подавтомат $M_{I,O}$, удовлетворяющий ограничениям, связанным в частности с дополнительными ограничениями внешней среды, в которой функционирует временной автомат.

Полезными на практике оказываются следующие ограничения и их комбинации:

1. *Инерционность внешней среды*: после получения от автомата выходного символа, среде требуется некоторое минимальное время на генерацию следующего входного символа. Язык максимального автомата в инерционной среде будет иметь вид $[(1)^*I(1)^*O(1)^n]^*(1)^*$, где $n \in \mathbf{N}$ представляет минимальное время реакции среды.

2. *Инерционность системы*: любой системе требуется некоторое минимальное отличное от нуля время, чтобы произвести выходную реакцию на полученное входное воздействие. Язык максимального автомата для инерционных систем будет иметь вид $[(1)^*I(1)^m(1)^*O]^*(1)^*$, где $m \in \mathbf{N}$ представляет минимальное время реакции системы.

3. *Ограниченность времени ожидания*: среда может ожидать выходную реакцию системы на поданное входное воздействие не дольше некоторого заранее заданного интервала времени. Язык

максимального автомата для систем в такой среде будет иметь вид $[(1)^* I (1)^{\le k} O]^* (1)^*$, где $k \in \mathbf{N}$ представляет максимальное время ожидания реакции системы. Использование данного предположения на практике требует, во-первых, явного описания поведения внешней среды в случаях нарушения времени ожидания (например, перезагрузка системы, прерывание текущей операции, и т.п.), во-вторых, интерпретации отсутствия выходного символа по истечении заданного времени ожидания.

4. *Запрет на подачу входной последовательности* $\alpha = (i_1, t_1) \ldots (i_n, t_n)$: на временной автомат не может быть подана входная последовательность α в силу свойств внешней среды или если данная последовательность приводит к недопустимому поведению системы. Данное требование полностью описывается рассмотрением редукций соответствующего частичного максимального автомата, имеющего язык вида $[(1)^* I (1)^* O]^* (1)^* / [(1)^{t_1} i_1 (1)^* O] \ldots [(1)^{t_n} i_n (1)^* O] (1)^*$.

Можно рассмотреть более сложные ограничения для внешней среды, однако далее мы предполагаем, что для любого рассматриваемого ограничения на внешнюю среду, если временной автомат *M* удовлетворяет предъявленному ограничению, то и любая редукция автомата *M* также ему удовлетворяет. В таком случае ограничения среды могут быть представлены в виде соответствующего максимального временного автомата, то есть автомата, описывающего максимально возможное поведение при соблюдении допустимых ограничений.

3. Параллельная композиция временных автоматов

Для описания взаимодействия временных автоматов в режиме диалога вводится операция параллельной композиции. Схема бинарной параллельной композиции приведена на рис. 1. Без ограничения общности, далее мы полагаем, что алфавиты I_1, I_2, O_1, O_2, U, V попарно не пересекаются. Обозначим через $I = I_1 \cup I_2$ и $O = O_1 \cup O_2$ внешние входной и выходной алфавиты композиции.

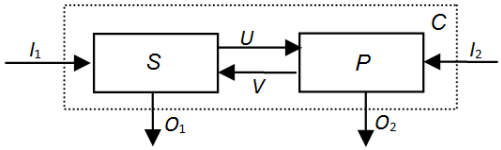


Рис. 1. Схема параллельной композиции

Аналогично композиции для конечных автоматов [1], параллельная композиция временных автоматов может быть построена по соответствующим полуавтоматам. *Параллельной композицией* [1, 2] полуавтоматов $A = (A, X \cup Y, a_0, \delta_A, F_A)$ и $B = (B, Y \cup Z, b_0, \delta_B, F_B)$ называется полуавтомат $C = (C, X \cup Z, c_0, \delta_C, F_C)$, такой, что$C = [A_{\Uparrow Z/X} \cap B_{\Uparrow X/Z}]_{\Downarrow X \cup Z}$ (обозначение: $C = A \Diamond B$), где операции пересечения ($\cap$), расширения ($\Uparrow$) и ограничения ($\Downarrow$) полуавтоматов определены известным образом на основе соответствующих операций над регулярными языками [1, 2].

Пусть заданы временные автоматы $S = (S, I_1 \cup V, O_1 \cup U, s_0, \lambda_S, \Delta_S, \sigma_S)$ и $P = (P, I_2 \cup U, O_2 \cup V, p_0, \lambda_P, \Delta_P, \sigma_P)$ и ограничения внешней среды для композиции выражены автоматом $M_C \leq M_{I,O}$. Тогда временной автомат C, которому соответствует полуавтомат $Aut(C) = [Aut(S) \Diamond Aut(P)] \cap Aut(M_C)$, есть *параллельная композиция временных автоматов* S и P*при ограничении* M_C. В том случае, если дополнительные ограничения для внешней среды не заданы, то есть $M_C = M_{I,O}$, то определение параллельной композиции совпадает с таковым из [5].

Утверждение 2. Если $S_1 \leq S_2$ и $P_1 \leq P_2$, то $S_1 \Diamond P_1 \leq S_2 \Diamond P_2$.

Доказательство следует из свойств операций (расширение, ограничение, пересечение) над языками, используемых для определения операции композиции.

Следствие. Максимальный автомат композиции любых компонент S и P, удовлетворяющих ограничениям, выраженным максимальными автоматами M_S и M_P соответственно, есть автомат $M_C = M_S \Diamond M_P$.В частности, композиция инерционных автоматов (с ненулевой минимальной задержкой выхода) также будет инерционным автоматом.

4. Наибольшее решение уравнения для временных автоматов

Рассмотрим автоматное уравнение $A \Diamond X \sim C$, коэффициенты A и C которого описываются временными автоматами, $\Diamond$ есть оператор параллельной композиции, а в качестве отношения конформности $\sim$ выступает редукция (в этом случае можно говорить об автоматном неравенстве). Тогда по определению параллельной композиции, неравенство для временных автоматов вида

$$A \lozenge X \leq C$$

можно решать как неравенство над соответствующими полуавтоматами

$$Aut(A) \lozenge Aut(X) \cap Aut(M_C) \leq Aut(C),$$

которое сводится к следующему неравенству над языками полуавтоматов

$$L(A) \lozenge L(X) \cap L(M_C) \subseteq L(C).$$

Далее, применяя известное преобразование $A \cup B = C \Leftrightarrow A = C \cap \overline{B}$ из теории множеств, получим языковое неравенство вида

$$L(A) \lozenge L(X) \subseteq L(C) \cup \overline{L(M_C)}.$$

Используя известные результаты для наибольшего решения языкового неравенства [1], найдем наибольшее решение: $L_{Solution} = \overline{L(A) \lozenge \left(\overline{L(C)} \cap L(M_C)\right)}$. В общем случае, язык $L_{Solution}$ не является языком некоторого временного автомата, но являясь наибольшим решением, содержит в себе в том числе и все решения, соответствующие временным автоматам и являющиеся решениями исходного автоматного уравнения. Наибольшее решение $L_{Solution\text{-}TFSM}$, являющееся языком временного автомата и, следовательно, решением автоматного уравнения, есть наибольшее префикс-замкнутое подмножество языка $L_{Solution} \cap L(M_X)$, где M_Xесть максимальный автомат для неизвестной компоненты. Для краткости, обозначим I_X и O_X входной и выходной алфавиты неизвестной компоненты, соответственно.

Утверждение 3. Наибольшее решение автоматного уравнения (неравенства) $A \lozenge X \leq C$ для временных автоматов есть временной автоматSтакой, что $L(S)$ есть наибольшее $[(1)^* \cup (1)^* I_X (1)^* O_X]$-префикс-замкнутое подмножество языка $L_{Solution} \cap L(M_X)$, где $L_{Solution} = \overline{L(A) \lozenge \left(\overline{L(C)} \cap L(M_C)\right)}$ и M_Xесть максимальный автомат для неизвестной компоненты. Если $L_{Solution} = \varnothing$, то уравнение неразрешимо. Если $L_{Solution} \neq \varnothing$, но $L(S) = \varnothing$, то уравнение не имеет автоматного решения.

Доказательство. Пересечение с языком $L(M_X)$ обеспечивает, что решение, если существует, содержится в максимально возможном допустимом поведении автомата для неизвестной компоненты. Алгоритм построения наибольшего префикс-замкнутого подмножества заданного регулярного языка известен [1]. Следовательно, $L(S)$ есть максимальный язык временного автомата, содержащийся в $L_{Solution}$ и являющийся решением автоматного уравнения (неравенства).

Временной автомат, соответствующий языку $L(S)$, в общем случае представляет собой недетерминированный, возможно частичный, автомат, описывающий совокупность всех возможных допустимых реализаций неизвестной компоненты. Стоит заметить, однако, что частичность решения

требует особого внимания: каждая реализация на практике должна быть полностью определенной, поэтому особый интерес представляет вопрос решения уравнений и неравенств для полностью определенных автоматов, а также вопросы интерпретации и адекватного до определения частичных и недетерминированных решений.

Заметим, что в силу того, что языковые уравнения решаются одинаковым образом для любых коэффициентов-языков, описанный процесс построения наибольшего решения уравнения для временных автоматов является универсальным при любых ограничениях внешней среды для композиции, если данные ограничения могут быть выражены через поведение максимального временного автомата в заданной среде.

5. Пример

В качестве примера для иллюстрации решения уравнений для временных автоматов рассмотрим задачу реализации композиции двух сервисов. При разработке сервиса в виде композиции других сервисов возникают следующие задачи: 1) проверить, предоставляет ли взаимодействие заданных компонент требуемый сервис, и 2) осуществить выбор компонент, которые бы во взаимодействии предоставляли желаемый сервис. В том случае, если проектируемый сервис и сервисы-компоненты описаны временными автоматами с таймаутами, данные задачи могут быть решены, в том числе, и с учетом требований к временным параметрам качества композиции сервисов.

Рассмотрим в качестве примера простую композицию для заказа товара в пункт самовывоза: покупатель осуществляет заказ (request)и по истечении оговоренного времени доставки может забрать заказ из пункта самовывоза (deliver).Время доставки для такого сервиса является одним из наиболее важных параметров качества сервиса. Потребуем, чтобы время доставки не превышало 7 дней. Тогда временной автомат, приведенный на рис. 2 (слева), описывает заданную композицию с указанным требований к качеству сервиса (ограничением максимального времени доставки).

Рис. 2. Временной автомат для сервиса заказа с самовывозом с ограничением времени доставки (слева) и без ограничения времени доставки (справа)

Заданная композиция может быть реализована композицией двух сервисов: сервиса *Продавец* и сервиса *Пункт самовывоза*. Опишем взаимодействие между сервисами: покупатель заказывает некоторый товар у *Продавца* (request) и *Продавец* отправляет запрос на проверку наличия товара в *Пунтке самовывоза* (check). Если товар есть в наличии (product), то покупателю отправляется уведомление о возможности получить заказ (deliver), иначе отправляется запрос на поставку товара со склада (ship). После поставки

товара в *Пункт самовывоза* (get_product), после проверки (check)и подтверждения, что был поставлен необходимый товар (product), покупатель получает уведомление о доставке, либо в случае ошибки (no_product) *Продавец* повторно заказывает доставку товара. На рис. 3 приведен временной автомат, описывающий работу *Продавца*.

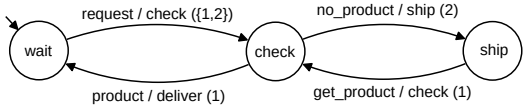


Рис. 3. Временной автомат для компоненты Продавец

Найдем решение уравнения для неизвестной компоненты *Пункт самовывоза* без ограничения времени доставки, но при условии, что никакое действие не может быть осуществлено моментально и минимальная задержка выходного сигнала составляет 1 день (предположение об инерционности системы).Наибольшее решение представлено на рис. 4. Решение существует, следовательно, описанная композиция может быть реализована.На рис. 4символ * использован для обозначения любого произвольного поведения (т.е. множества всех возможных входо-выходных пар автомата с любой возможной задержкой выхода), и соответствует части поведения компоненты, которая не может быть достигнута из заданного контекста; соответственно, эта часть не влияет на результат композиции и потому может быть определена (реализована) любым оптимальным образом. Символ Nобозначает множество натуральных чисел.

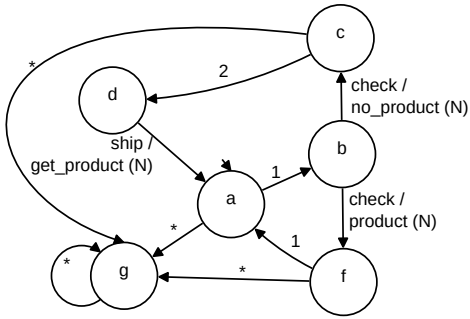


Рис. 4. Наибольшее решение для компоненты Пункт самовывоза

Оптимизируем работу *Пункта самовывоза* таким образом, чтобы минимизировать время доставки товара, и проверим, при каких условиях возможно гарантировать выполнение требования о доставке в течение 7 дней. Решение (полностью определенное) с минимальным временем задержки приведено на рис. 5 (слева) и композиция данного частного решения с контекстом приведена на рис. 5 (справа). Из построенной композиции видно, что при заданном контексте (*Продавец*)выбранное решение для компоненты

Пункт самовывоза не может гарантировать соблюдение ограничения по максимальному времени доставки в силу возможности возникновения осцилляции (о чем свидетельствует наличие периода 5*t*в множестве задержек выхода композиции без ограничения верхнего значения). Дальнейший более детальный анализ композиции может показать, что композиция с ограничением времени доставки в 7 дней будет гарантировано выполняться только в том случае, если запрашиваемый продукт всегда будет находиться в *Пункте самовывоза* (т.е. соответствующая компонента в ответ на запрос check всегда будет отвечать product). В иных ситуациях требуется либо пересмотр заданных требований к качеству, либо дальнейшая оптимизация компонент композиции, включая дальнейшую оптимизацию работы компоненты *Продавец*.

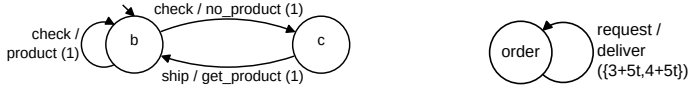


Рис. 5. Оптимальное по времени поведение компоненты Пункт самовывоза (слева) и ее композиция с компонентой Продавец (справа)

6. Заключение

В статье рассмотрено построение параллельной композиции временных автоматов и синтеза неизвестной компоненты, которая при взаимодействии с известной частью системы удовлетворяет заданной спецификации композиции. На примере простой композиции для заказа товара с доставкой в пункт самовывоза проиллюстрированы перспективы использования введенных операций для оценки и управления качеством композиции сервисов.

Отметим, что значения функций таймаутов и задержек выхода в рассматриваемой модели временного автомата могут принимать только целочисленные значения. При рассмотрении реальных систем (в частности, сервисов) подобное ограничение оказывается достаточно жестким: для задержек выхода, более реалистичной интерпретацией является определение интервала времени, в течение которого автомат может выдать выходной символ. Так как в композиции задержки выходного символа одной компоненты определяют моменты времени поступления внутренних символов на вход другой компоненты, то расширение области значений функции задержки выходного символа на произвольные интервалы, в общем случае, требует дальнейших исследований. Однако все результаты, полученные при целочисленных задержках выходного символа, сохраняются при использовании открытых справа интервалов, то есть при замене каждой целочисленной задержки *n*на интервал $[n, n + 1)$. Это достигается за счет того, что при целочисленных таймаутах любой входной символ, поступивший на автомат в некоторый момент времени в интервале $[n, n + 1)$, обрабатывается автоматом так же, как входной символ, поступивший точно в момент времени

n. Подобные рассуждения согласуются с выводами работы [4] при сравнении выразительности моделей временного автомата с таймаутами и интервалами.

Использование параллельной композиции и решения соответствующих уравнений для временных автоматов представляется перспективным для решения таких задач, как тестирование в контексте, и анализ свойств композиции взаимозаменяемых компонент, например, при оценке качества динамической композиции сервисов. Также интересной для дальнейшего исследования проблемой является вопрос разрешимости автоматного уравнения при различных ограничениях внешней среды.

Список литературы

[1]. T. Villa, N. Yevtushenko, R.K. Brayton, A. Mishchenko, A. Petrenko, A.L. Sangiovanni Vincentelli. The unknown component problem: theory and applications. Springer, 2012. 311 p.

[2]. Н.В. Евтушенко, А.Ф. Петренко, М.В. Ветрова. Недетерминированные автоматы: анализ и синтез. Ч.1. Отношения и операции: Учебное пособие. Томск: Томский государственный университет, 2006. 142 с.

[3]. M. Zhigulin, N. Yevtushenko, S. Maag, A.R. Cavalli. FSM-based test derivation strategies for systems with timeouts. Proceedings of the international conference QSIC 2011. P. 141-149.

[4]. D. Bresolin, K. El-Fakih, T. Villa, N. Yevtushenko. Deterministic Timed Finite State Machines: Equivalence Checking and Expressive Power. Proceedings GandALF 2014. EPTCS 161, 2014. P. 203-216.

[5]. О.В. Кондратьева, Н.В. Евтушенко, А.Р. Кавалли. Параллельная композиция конечных автоматов с таймаутами. Вестн. Том. гос. ун-та. Управление, вычислительная техника и информатика. № 2(27). 2014. стр. 73–81.

Solving parallel equations for Finite State Machines with Timeouts

[1, 2] *O. Kondratyeva <olga.kondratyeva@telecom-sudparis.eu>*
[1] *N. Yevtushenko <yevtushenko@sibmail.com>*
[2] *A. Cavalli <ana.cavalli@telecom-sudparis.eu>*
[1] *Tomsk State University, 634050, Russia, Tomsk, av. Lenina, 36*
[2] *Telecom SudParis, 91000, France, Evry, rue Charles Fourier 9*

Abstract. The problem of designing an unknown component that combined with the known part of a discrete event system satisfies the given overall specification arises in a number of applications. In this paper, we extend the known results for classical Finite State Machines (FSM) to Finite State Machines with Timeouts (TFSM).
A TFSM is an FSM augmented with an input and an output timeout functions, prescribing the change of current state if no input is applied until a specified timeout expires and system delays needed to response to applied input, respectively. We represent the behavior of a TFSM by the corresponding regular language, so that L(S) has a word α = 1t1 i1 1k1 o1 … 1tm im 1km om if and only if the given TFSM S has a timed trace (i1, t1).(o1, k1) … (im, tm).(om, km). The parallel composition of two TFSMs C = S ◊ P is defined via composition of languages L(C) = [L(S) ◊ L(P)] ∩ L(MC), where MC is a maximal TFSM over input and output alphabets of the composition.
The unknown component X is then designed as a solution to the equation A ◊ X ≤ C with TFSM coefficients A (context) and C (specification), where ≤ is the reduction relation which specifies that the behavior of a system behavior to be designed is contained in that of the specification. Similar to classical FSMs, the equation is transformed to a language equation $L(A) \Diamond L(X) \subseteq L(C) \cup \overline{L(M_C)}$ with the largest solution $L_{Solution} = \overline{L(A) \Diamond \left(\overline{L(C)} \cap L(M_C)\right)}$, the largest TFSM subset of which provides the largest solution to the solvable TFSM equation. After the largest solution is derived, a corresponding reduction can be extracted in order to provide the component with required properties.
The application areas vary from testing in context to quality optimization of service compositions. Future work includes studying equation solvability criteria and properties of nondeterministic and partial specifications and solutions.

Keywords: finite state machine with timeouts, parallel composition, automata equations, the largest solution, environment restrictions.

References

[1]. T. Villa, N. Yevtushenko, R.K. Brayton, A. Mishchenko, A. Petrenko, A.L. Sangiovanni Vincentelli. The unknown component problem: theory and applications. Springer, 2012. 311 p.

[2]. N.V. Yevtushenko, A.F. Petrenko, M.V. Vetrova. Nedeterminirovannie avtomaty: analiz i sintez. Ch.1. Otnosheniya i operazii [Nondeterministic finite state machines: analysis and synthesis. Part 1. Relations and operations]: Tutorial, Tomsk: Tomsk State University, 2006. 142 p. (in Russian).

[3]. M. Zhigulin, N. Yevtushenko, S. Maag, A.R. Cavalli. FSM-based test derivation strategies for systems with timeouts. Proceedings of the international conference QSIC 2011. P. 141-149.

[4]. D. Bresolin, K. El-Fakih, T. Villa, N. Yevtushenko. Deterministic Timed Finite State Machines: Equivalence Checking and Expressive Power. Proceedings GandALF 2014. EPTCS 161, 2014. P. 203-216.

[5]. O.V. Kondratyeva, N.V. Yevtushenko, A.R. Cavalli. Parallel composition of nondeterministic Finite State Machines with Timeouts // Tomsk State University Journal of Control and Computer Science. 2014. № 2(27). P. 73–81. (in Russian).

A study of Intrusion-tolerant routing in Wireless Sensor Networks

V. H. La <vinh_hoa.la@telecom-sudparis.eu>
A. R. Cavalli <ana.cavalli@telecom-sudparis.eu>
Telecom SudParis
9 rue Charles Fourier, 91011 Evry, France

Abstract. Wireless Sensor Networks (WSNs) emerge recently as one of the most attractive research subjects. The resource- constraint characteristics of WSNs limit the secure design and development of security protocols for them. Whilst, sensor nodes those usually operate in unattended and even harsh environments, are prone to failures and are vulnerable to malicious attacks. For reliable and secure communications in WSNs, intrusion-tolerant routing becomes a critical attribute that should be integrated into WSNs. In this paper, we study two intrusion- tolerant routing protocols for WSNs, namely INSENS and ITSRP, as well as analyze the intrusion-tolerant properties gained from these two propositions. Simulation and performance analysis have proved that both of them are practical.

Keywords: intrusion-tolerance; wireless sensor networks; routing; attack-tolerance

1. Introduction

In the current era, WSNs are rapidly emerging as an important area in both the research community and the public, due to the unique features (limited energy lifetime, slow embedded processors, severely constrained memory and low-bandwidth radios). For example, Waspmote [1], the modern open source sensor device distributed by Libelium, contains simply a 14 MHz micro-processor, 3.3 V-4.2 V battery voltage, 8 KB SRAM, 128 KB flash memory and 4 KB EEPROM to save sensed data, run and operating system and application programs. These resource constraints limit the degree of encryption, decryption, and authentication, in addition to physical security risks of being deployed in inaccessible terrains or unattended and even hostile environment, thus, the concept security and WSNs were likely contradictory. The integration of an intrusion detection system seems to be too expensive in terms of resource. WSNs are exposed to a variety of security threats in addition to the ones normally observed in traditional wired and wireless networks. Therefore, it is essential to take intrusion tolerant concept into consideration to sustain the sensor network functionalities without interruption despite malicious attacks and sensor node failures.

Our paper aims to present an analysis on two existing intrusion-tolerant routing protocols proposed for WSNs, namely INSENS and ITSRP. They are, in our opinion, two best intrusion-tolerant routing protocols so far. We attempt to briefly describe them and mainly focused our analysis on intrusion-tolerance properties. Simulations and performance analysis will be also discussed.

The rest of this paper is organized as follows: Section 2 is devoted to describe and assess INSENS and ITSRP. Section 3 contains the simulation results and performance analysis. Finally, we summarize our study as well as propose the future work in Section 4.

2. Intrusion-tolerant Routing Protocols in WSNs

Intrusion tolerance [5], [6], [7] is generally understood as the capability to continue to function properly with minimal degradation of performance, despite intrusions or malicious attacks. A great deal of work has been done to address the sensor network security problems recently so that the WSNs can tolerate and/or prevent intrusions [3], [4]. In the following subsections, we attempt to identify two current approaches used for achieving intrusion-tolerant routing- one of the most critical features in WSNs.

2.1 INSENS- INtrusion-tolerant routing protocol for wireless SEnsor NetworkS

INSENS [15], [16], [17] could be subdivided into 2 phases: ***Route Discovery*** phase and ***Data Forwarding*** phase. The goal of the first phase is to collect topology knowledge and to construct appropriate forwarding tables at every node. Whilst, the second phase simply enables forwarding of data from each sensor node to the base station and vice versa. It is worth mentioning that every communication between nodes is one-way forwarded (unicast) via base station. The Route Discovery phase is composed of three rounds: ***Route Request, Route Feedback, and Computing and Propagating Multi-path Routing Tables.***

In the beginning (or when the topology may have changed substantially because of nodes' mobility), the base station floods (limited flooding) a *request message* to all the reachable sensor nodes in the network. After receiving a *request message* for the first time, a sensor node *x* broadcasts in turn another *request message* that includes a *path* from the base station to *x* and also the *identity* of *x*. Whenever receiving duplicate request messages, it records the *identity* of the sender as a neighbor, but stop re-broadcasting the duplicate request. The base station authenticates the *feedback messages* received from sensor nodes (authentication manner will be further discussed in the follow parts of this paper). After that, it constructs a topological picture of the network from the authenticated neighborhood information, computes the forwarding tables for each sensor node, and sends the

tables respectively to nodes using a *routing update message*. To address the influences of compromised nodes, INSENS builds redundant multi-path routing tables containing disjoint paths. Therefore, even if a single node or path is taken down by an intruder, secondary paths will substitute.

The main idea of this approach is to add *OWS* (one-way sequences) field and the *MAC* (Message Authentication Code) field *(MACR* and *MACF)*, that support the intrusion-tolerant properties of INSENS, into message format. First of all, the base station uses one-way sequences (OWS) proposed by *μTESLA* protocol [8] F to generate a sequence of numbers K0, K1... Kn, such that Ki = F(Ki+1), where 0 < i < n and F satisfies the condition that it is computationally infeasible to compute Ki+1 in a limited time by only knowing F and Ki. Initially, every node is pre-configured to know K0 and F. In the first Route Discovery phase, the base station includes K1 in the request message that it broadcasts. Similarly in general, the base station uses Ki in the ith Route Discovery phase. After receiving a request message, a node verifies if the sequence number did indeed originate from the base station by checking whether K0 = Fi (Ki). A malicious node would be computationally impossible to guess the next OWS. As a result, a compromised node cannot spoof the base station by generating new OWS. On the other hand, a sensor node will save the most up-to-date or freshest OWSfresh that it has just seen from the base station. This fact resists an intruder to disrupt the network by using old OWS to flooding old request messages.

In fact, a malicious node is still possible to flood a modified request message in using the current OWS from a valid request message that it has just received from the base station. Such an attack is discussed as ***rushing attack*** in [2]. However, nodes in the tree (Fig. 1), that are closer to the base station than the malicious node m, will receive the valid request message first. These nodes will drop the intruder's spurious request messages received later because, as mentioned above, nodes do not rebroadcast duplicate request messages (contain the same OWS). Even when neighboring nodes of m accept to forward the fake request message created by m, they forward only once. DOS attack, thus, is no longer in our concerns. Nevertheless, an unsolved issue in primary INSENS is that attacker could pack a fake path into its spurious request message or drop the request message instead of forwarding it. Some nodes can be harmed (not getting a request message or not being able to forward their feedback message to the base station in the second round) but, as shown in Fig. 1, the damage is locally confined to the nodes nearest to and downstream from the intruder. This conclusion seems to be logical but remains intuitive and needs further evaluation to know whether such damage can still seriously disrupt the network.

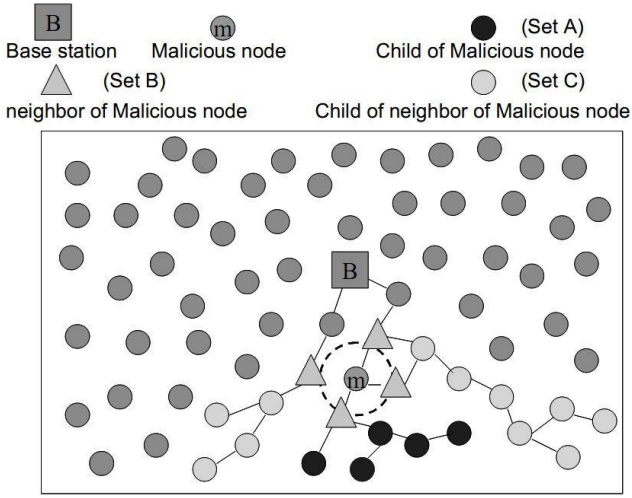


Fig. 1. The damage inflicted by a malicious node m is confined to a localized portion of the sensor network, i.e., nodes downstream from m and downstream from neighbors of m

In addition to OWS, keyed MAC (Message Authentication Code) algorithm is another factor that provides intrusion- tolerant properties for INSENS. Each sensor node is initially configured with a separate *secret key* that is shared only with the base station. When a node *x* receives a request message for the first time, before forwarding it, *x* appends its identity to the path list, and then generates a MAC of the complete new path with its key: $MACR_x = MAC\ (size|path|OWS|type,\ key_x)$ where "|" denotes concatenation. The value of MACR is also appended to the request message that is then forwarded downstream. This MACR field will eventually be exploited by the base station to verify the integrity of the path contained in the packet. Additionally, even if a node is compromised, only its secret key will be revealed, so an intruder cannot compromise the entire network.

In the second round (*Route* Discovery-*Route Feedback*), keyed MAC is applied one more time to protect the integrity of feedback messages. List of neighbors *nbr_info* and the path *path_info* to a node x are protected by the following keyed $MACF_x$: $MACF_x = MAC\ (path\ info\ |\ nbr\ info\ |\ OWS\ |\ type,\ key_x)$.

In feedback messages, *parent info* field determines a child's upstream neighbors and takes part in forwarding the feedback message to the base station. Using only identity I_p would be an obvious vulnerability because it does not require the casual intruder to have either any knowledge of the local topology or of the current state of the topology discovery process. To address this concern, INSENS requires a child node to put its parent's $MACR_p$ that is included in the parent's original request message into the *parent_info* field. This $MACR_p$ is tightly linked with the current state of the OWS request-feedback cycle, as well as to the path to the child node. In

other words, the $MACR_p$ plays a role as not only an addressing function but also a security function. A casual attacker, that only knows *node id*, would be unable to forward a spurious feedback message.

2.2. ITSRP- Intrusion Tolerant Secure Routing Protocol

ITSRP [19] is a novel secure routing protocol which focuses on the design of some fields to emphasize the security accounting to the key exchange, but not result in the complexity of the protocol. The main target of the ITSRP is to tolerate damage caused by an intruder who has compromised deployed sensor nodes and is intent on injecting, modifying, or blocking packets [20], but in a reasonable price regarding energy factor. As any other routing protocol, ITSRP consists of following steps: *path discovery* (sink node informs to other nodes that it is in need of network topology), *path reverse* (nodes send back topology information to help sink node build up routing tables) and *data transfer* (simply based on routing tables).

The general idea to achieve intrusion-tolerance in ITSRP is derived from the procedure to establish a secret session key *SK* whenever a source node N_0 wants to send a confidential message *M* to the sink node N_n but it lacks neither an available *route path* nor a *shared session key* with N_n. It is worth mentioning that each node is initially issued a Distributed Key (*DK*) that is shared only between itself and the sink node (e.g., the base station) and that each node store a *local route table* (LRT) containing entries whose format is demonstrated as follow:

Table 1. Format of an entry in LRT

Tag	Ancestor	Successor	Energy	Lifetime

In the first phase, N_0 must establish a *route path* and a *session key SK* uniquely shared with N_n through intermediate nodes. The source node N_0 first generates the unique *Tag* for this route and realizes these following steps:

(1) - Select randomly a secret session key SK_0, compute the energy consumption E_0 for transmitting the message M.

(2) - Set $M_0 = [Tag|N_0|N_n|SK_0]$ (“|” depicts the concatenation) and encrypt M0 in using the DK0: C0 = Edk (M0).

(3) - Encapsulate the packet [Tag|Nn|C0|E0] and broadcast it to all nodes within its wireless transmission range.

(4) - Store the entry (Tag, 0, ?, E0, T0) to its LRT0 where T0 is the timer for the route and starts when the entry is added. Each node close to N_0 node receives packet $[Tag|Nn|C_0|E_0]$, it checks and drops if this packet has been already received before. Otherwise, it broadcasts this packet within its range and store $(Tag|N_0|?|E_0|T_0)$ in its LRT_1. Without loss of generality, assuming that packet passes the intermediate nodes N_1, N_2... N_{n-1} and reaches the sink N_n. Node N_i stores $(Tag|N_{i-1}|?|E_0|T_i)$ in its LRT_i.

In the second phase, the sink node N_n must send back to the source node N_0 the reverse path. N_n works as follows:

(1) - Knowing DK_0, the sink node N_n is able to decrypt C_0 to get M_0, and then get SK_0.

(2) - Make $M_n = [Tag|N_0|N_n|SK_0]$ and use DK_0 to encrypt M_n as C_n.

(3) - Look up its ancestor node N_{n-1} according to Tag in its LRT_n, then use DK_{n-1} to encrypt Tag as C_{n-1} and send $[C_n|C_{n-1}]$ to N_{n-1}.

When the node N_{n-1} receives $[C_n|C_{n-1}]$ from N_n, it works as follows:
(1) - Use DK_{n-1} to recover Tag.

(2) - Look up in its LRT_{n-1} according this Tag and update the entry saved from the first phase: $(Tag|N_{n-2}|N_n|E_0|T_{n-1})$.

Similarly to the end, N_0 gains the entry (Tag, 0, N_1, E_0, T_0) in its LRT_0. Not only has the route from N_0 to N_n been discovered but also a shared session key SK between of them has been established.

Indeed, ITSRP is intrusion tolerant firstly because of the redundancy in possible paths. For a route, there are multiple paths possibly used to reduce the failure rate caused by intruder. Furthermore, ITSRP provides significant protection against a variety of malicious attacks during the routing set up phase as well as during the data forwarding phase, especially in comparison with the naive Directed Diffusion routing protocol.

In fact, different salient attacks in WSNs routing have been described in literature [2], including Sinkhole attacks, Wormhole attacks and Sybil attacks that induce incorrect routing information to provoke incorrectly forwarding messages. In a sinkhole attack, a malicious node presents itself having the shortest path to a well-known destination, e.g., a sink node. In networks that apply a routing scheme allowing nodes to select their routing path based on neighborhood routing information, a sinkhole attack can lead to incorrect routing paths towards the malicious node. The malicious node, thus, can disturb the routing activities as well as collect illegitimately data from networks. However, the distributed key management and the mechanism to build a route path of ITSRP protect itself from sinkhole attacks. A malicious node has no possibility to pretend a sink node, as well as pretend to belong to the shortest path without knowing the distributed key. In a Sybil attack, a malicious node forges multiple fake identities and then deceives other sensor nodes using those fake identities. This attack is eliminated from ITSRP because of the distributed key management. Each node is not only identified by its ID but also by its DK that is shared secretly between itself and sink node. The forgery of sink node is also not feasible without knowledge about DKs. Rushing attacks (that was formerly discussed above in INSENS's section) are also avoided

out of our concerns because each message forwarded by a node is encrypted with a distributed key (in path discovery phase and path reserve phase) or with a session key (in data transfer phase). For HELLO flood attacks where an adversary with a powerful transmitter reaches every node in the network, and pretends to be a neighbor, ITSRP assure each pair of neighboring nodes to establish a secure communication channel by agreeing on a unique secret key DK between two neighboring nodes, which is bidirectional. Each legitimate node keeps only an ancestor and a successor in the routing map (LRT), so each node only accepts and forwards one copy of the same packet, otherwise, drop them. This fact defenses against any attacker contriving HELLO flood attack.

In [19], the authors affirm that their proposition is immune to Wormhole attacks in which two malicious nodes exchange their routing information using a fast and secure channel or tunnel, and then trap or warp the routing paths of their neighbor nodes. However, in our opinion, a wormhole attack can still harm ITSRP, even though the probability of this event can be very low. In fact, in the path discovery phase, each node after receiving a path discovery packet will broadcast it within its wireless communication range. This packet can be logically duplicated but will be actually drop very fast because one node will allow to receive this packet only one time and drop whenever detect the duplication. The shortest path will be found quickly due to the quick rate of wireless communication (in fact that depends on which mean of wireless communication used in the network, e.g., Wifi, GPRS, Bluetooth, ZigBee, etc.). If somehow two malicious nodes can exchange this packet faster than legitimate nodes, they are still able to set up a tunnel and thus, generate an incorrect route path. After that, they can disrupt this tunnel or do whatever they want because they have gained a very powerful role in routing activities.

3. Analysis and performance evaluation

Firstly, by theoretically analyzing aforementioned protocols, we affirmed that they, indeed, are able to tolerate some critical attacks and kinds of intrusion. However, they have themselves shortcomings that should be accomplished by further researches. Table 2 summaries our analysis.

Table 2. Intrusion-tolerant properties and shortcomings of INSENS and ITSRP

Routing protocol	Intrusion/Attack tolerance	Attack “intolerance”
INSENS	Redundancy Battery drain attack Memory exhaustion attack DOS/ DDOS attack Rushing attack	Rushing attacks are not completely solved. Although the influences are restricted in a local partition of the network but further theoretical and practical evaluations deserve more research and consideration

ITSRP	Redundancy (multiple paths) Sink hole attack Sybil attack HELLO flood attack Rushing attack	Wormhole attacks are still able to occur despite the low probability

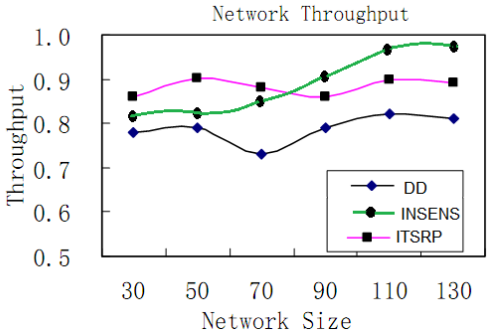


Fig. 2. Average network throughput of ITSRP and Directed Diffusion

Secondly, to evaluate the performance of INSENS and ITSRP, we performed a simulation by **OMNeT++** integrated Castalia plug-in. We simulated a rectangular region of area 150m x 150m in which the wireless sensor nodes were deployed and 25 percent of nodes misbehave. In Fig. 2, we witness that throughput of three metrics decreases more or less because of nodes misbehave. However, the network experiences a considerable decrease in DD (Directed Diffusion) protocol, but only a slight change in INSENS or ITSRP. This result proves one more time the intrusion-tolerance properties of ITSRP and declares that INSENS is better than ITSRP if the number of sensor nodes enormously increases.

4. Conclusion and perspectives

In this paper, we have presented an analysis on INSENS and ITSRP that are, from our point of view, two best intrusion- tolerant routing protocols insofar. We have briefly described these two protocols and mainly focused our analysis on intrusion-tolerance properties. The Table 2 summarizes the main intrusion-tolerant properties assured by INSENS and ITSRP as well as missing issues that could not be thoroughly solved. We have also analyzed the performance to evaluate the practicality of both two protocols. We believe that our paper is useful for WSNs researchers as a study on the state of the art and for developers in implementing a secure WSN. In future, we would like to implement further experiments to

compare the performance of INSENS with ITSRP as well as simulate some attack scripts to verify the resilience and stability. The missing issues mentioned in Table 2 are also future works that we would like to accomplish.

References

[1]. Libelium Comunicaciones Distribuidas S.L. “Waspmote Datasheet”, 2014

[2]. Akyildiz I, SuW, et al. “Wireless sensor networks: a survey”, in Computer Networks, pp. 393-422, 2002.

[3]. Ruiping Ma, Liudong Xing, Howard E. Michel. “Fault-Intrusion Tolerant Techniques in Wireless Sensor Networks”, in 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing, pp. 85- 94, 2006.

[4]. Liang-min Wang, Jian-feng Ma, Chao Wang, A.C. Kot. “Fault and intrusion tolerance of wireless sensor networks”, in 20th International Parallel and Distributed Processing Symposium, 2006.

[5]. Yves Deswarte, David Powell, “Intrusion-tolerance on the Internet (Tolrance aux intrusions sur Internet)”, in Presentation slide, LAAS- CNRS, Toulouse, France, 2005.

[6]. B.B. Madan and K.S. Trivedi, “Security modeling and quantification of intrusion tolerant systems”, in ISSRE, 2002.

[7]. Feiyi Wang, Raghavendra Uppalli, Charles Killian, “Analysis of tech- niques for building intrusion tolerant server systems”, in Military Communications Conference MILCOM ’03 IEEE, pp. 729 - 734 Vol.2, 2003.

[8]. Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, J. D. Tygar, “SPINS: Security Protocols for Sensor Networks”, in Journal of Wireless Networks, Volume 8, Issue 5, pp. 521- 534, Sept. 2002.

[9]. Chris karlof, Naveen Sastry, David Wagner. “TinySec: A Link Layer Security Architecture for Wireless Sensor Networks”, in Proceedings of the 2nd international conference on Embedded networked sensor networks, pp. 162-175, 2004.

[10]. Soumya Basu, M.Pushpalatha. “Analysis of energy efficient ECC and TinySec based security schemes in Wireless Sensor Networks”, in IEEE International Conference on Advanced Networks and Telecommunication Systems (ANTS), pp. 1 - 6, 2013.

[11]. Ajay Mahimkar, Theodore S. Rappaport. “SecureDAV: a secure data aggregation and verification protocol for sensor networks”, in IEEE Global Telecommunications Conference (GLOBECOM), pp. 2175 – 2179, Vol.4, 2004.

[12]. Leonardo B. Oliveira, Hao Chi Wong, Antonio A. Loureiro. “LHA- SP: secure protocols for hierarchical wireless sensor networks”, in 9th IFIP/IEEE International Symposium on Integrated Network Manage- ment, pp. 31 - 44, 2005.

[13]. D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar. “Next century challenges: Scalable coordination in sensor networks ”, in Mobile Computing and Networking , pp. 263 – 270, 1999.

[14]. Maha Sliti, Mohamed Hamdi, Noureddine Boudriga. “Intrusion-tolerant framework for heterogeneous Wireless Sensor Networks”, in The 7th IEEE/ACS International Conference on Computer Systems and Applications, AICCSA 2009, pp. 633- 636, May 10-13, 2009.

[15]. J. Deng, R. Han, and S. Mishra. “INSENS: Intrusion-Tolerant Routing in Wireless Sensor Networks”, Poster paper in 23rd IEEE International Conference on Distributed Computing Systems, 2003

[16]. J. Deng, R. Han, and S. Mishra. “INSENS: Secure and Intrusion Tolerant Routing for Wireless Sensor Networks”, Technical Report CU-CS 939-02. Department of Computer Science. University of Colorado, Boulder, CO. November 2002.

[17]. J. Deng, R. Han, and S. Mishra. “A Performance Evaluation of Intrusion-Tolerant Routing in Wireless Sensor Networks”, in 2nd IEEE International Workshop on Information Processing in Sensor Networks, 2003

[18]. Sapon Tanachaiwiwat, Pinalkumar Dave, Rohan Bhindwale, Ahmed Helmy. “Secure Locations: Routing on Trust and Isolating Compromised Sensors in Location-Aware Sensor Networks”, in SenSys’08, Los Angeles, California, USA, ACM 1-58813-707-9/03/0011, 2008

[19]. Jiliang Zhou, Caixia Li , Qiying Cao and Yu Shen, “An intrusion-tolerant secure routing protocol with key exchange for wireless sensor network”, in International Conference on Information and Automation-ICIA 2008, pp. 1547 - 1552, 2008.

[20]. L.M. Feeney. “An energy consumption model for performance analysis of routing protocols for mobile ad hoc networks”, in Mobile Networks and Applications, pp. 239 - 249, 2001.

Исследование отказоустойчивой маршрутизации в беспроводных сенсорных сетях

В. Х. Ла <vinh_hoa.la@telecom-sudparis.eu>
А. Кавалли <ana.cavalli@telecom-sudparis.eu>
Telecom SudParis, 9 rue Charles Fourier, 91011 Evry, France

Аннотация. Развитие технологий и снижение стоимости беспроводных сенсорных систем привело к открытию новых областей для их применения и повышению требований к обеспечению безопасности и надежности сетей. Ограничения ресурсов узлов беспроводных сенсорных сетей приводят к жестким рамкам для реализации безопасных протоколов их взаимодействия. Так как сенсорные узлы работают, как правило, в неконтролируемой или даже враждебной среде, они подвержены отказам и уязвимы для атак. Для обеспечения надежности и безопасности взаимодействия сенсоров в сети отказоустойчивая маршрутизация становится ключевым элементом, который должен быть реализован в беспроводных сенсорных сетях. В данной работе исследуются два отказоустойчивых протокола маршрутизации, INSENS и ITSRP, для которых проводится анализ параметров их устойчивости к атакам. Моделирование и анализ быстродействия показали, что оба протокола достаточно хороши с практической точки зрения.

Ключевые слова: отказоустойчивость; беспроводные сенсорные сети; маршрутизация; устойчивость к атакам.

Список литературы

[1]. Libelium Comunicaciones Distribuidas S.L. “Waspmote Datasheet”, 2014
[2]. Akyildiz I, SuW, et al. “Wireless sensor networks: a survey”, in *Computer Networks*, pp. 393-422, 2002.
[3]. Ruiping Ma, Liudong Xing, Howard E. Michel. “Fault-Intrusion Tolerant Techniques in Wireless Sensor Networks”, in *2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, pp. 85- 94, 2006.
[4]. Liang-min Wang, Jian-feng Ma, Chao Wang, A.C. Kot. “Fault and intrusion tolerance of wireless sensor networks”, in *20th International Parallel and Distributed Processing Symposium*, 2006.
[5]. Yves Deswarte, David Powell, “Intrusion-tolerance on the Internet (Tolrance aux intrusions sur Internet)”, in *Presentation slide*, LAAS- CNRS, Toulouse, France, 2005.

[6]. B.B. Madan and K.S. Trivedi, “Security modeling and quantification of intrusion tolerant systems”, in *ISSRE*, 2002.
[7]. Feiyi Wang, Raghavendra Uppalli, Charles Killian, “Analysis of techniques for building intrusion tolerant server systems”, in *Military Communications Conference MILCOM ’03 IEEE*, pp. 729 - 734 Vol.2, 2003.
[8]. Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, J. D. Tygar, “SPINS: Security Protocols for Sensor Networks”, in *Journal of Wireless Networks*, Volume 8, Issue 5, pp. 521- 534, Sept. 2002.
[9]. Chris karlof, Naveen Sastry, David Wagner. “TinySec: A Link Layer Security Architecture for Wireless Sensor Networks”, in *Proceedings of the 2nd international conference on Embedded networked sensor networks*, pp. 162-175, 2004.
[10]. Soumya Basu, M.Pushpalatha. “Analysis of energy efficient ECC and TinySec based security schemes in Wireless Sensor Networks”, in *IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*,pp. 1 - 6, 2013.
[11]. Ajay Mahimkar, Theodore S. Rappaport. “SecureDAV: a secure data aggregation and verification protocol for sensor networks”, in *IEEE Global Telecommunications Conference (GLOBECOM)*,pp. 2175 - 2179 Vol.4, 2004.
[12]. Leonardo B. Oliveira, Hao Chi Wong, Antonio A. Loureiro. “LHA- SP: secure protocols for hierarchical wireless sensor networks”, in *9th IFIP/IEEE International Symposium on Integrated Network Management*, pp. 31 - 44, 2005.
[13]. D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar. “Next century challenges: Scalable coordination in sensor networks, in *Mobile Computing and Networking* , pp. 263270, 1999.
[14]. Maha Sliti, Mohamed Hamdi, Noureddine Boudriga. “Intrusion-tolerant framework for heterogeneous Wireless Sensor Networks”, in *The 7th IEEE/ACS International Conference on Computer Systems and Applications*, AICCSA 2009, pp. 633- 636, May 10-13, 2009.
[15]. J. Deng, R. Han, and S. Mishra. “INSENS: Intrusion-Tolerant Routing in Wireless Sensor Networks”, Poster paper in *23rd IEEE International Conference on Distributed Computing Systems*, 2003
[16]. J. Deng, R. Han, and S. Mishra. “INSENS: Secure and Intrusion Tolerant Routing for Wireless Sensor Networks”, Technical Report CU-CS 939-02. Department of Computer Science. University of Colorado, Boulder, CO. November 2002.
[17]. J. Deng, R. Han, and S. Mishra. “A Performance Evaluation of Intrusion-Tolerant Routing in Wireless Sensor Networks”, in *2nd IEEE International Workshop on Information Processing in Sensor Networks*, 2003
[18]. Sapon Tanachaiwiwat, Pinalkumar Dave, Rohan Bhindwale, Ahmed Helmy. “Secure Locations: Routing on Trust and Isolating Compromised Sensors in Location-Aware Sensor Networks”, in *SenSys’08, Los Angeles, California, USA, ACM 1-58813-707-9/03/0011.*, 2008
[19]. Jiliang Zhou, Caixia Li , Qiying Cao and Yu Shen, “An intrusion-tolerant secure routing protocol with key exchange for wireless sensor network”, in International Conference on Information and Automation-ICIA 2008, pp. 1547 - 1552, 2008.
[20]. L.M. Feeney. “An energy consumption model for performance analysis of routing protocols for mobile ad hoc networks”, in Mobile Networks and Applications, pp. 239 - 249, 2001.

Синтез проверяющих последовательностей для недетерминированных автоматов относительно редукции

Антон Ермаков <antonermak@inbox.ru>
Национальный исследовательский Томский государственный университет, 634050, пр. Ленина, 36, Томск, Россия

Аннотация. В данной статье предлагается алгоритм построения адаптивной проверяющей последовательности для недетерминированного конечного полностью определенного автомата относительно редукции.

Ключевые слова: синтез тестов; недетерминированный автомат; отношение редукции;

1. Введение

В большинстве методов построения проверяющих тестов по автоматной модели рассматривается инициальный конечный автомат, и соответственно проверяющие тесты строятся как набор входных последовательностей, перед каждой из которых подается сигнал СБРОС, переводящий автомат в начальное состояние. Если последний является достаточно дорогим, то имеет смысл попытаться построить одну последовательность, которая позволяет выяснить, является ли проверяемая реализация конформной спецификации.

Кроме того, обычно под тестом понимается множество заранее построенных входных последовательностей. В том случае, когда проверяющий тест строится по недетерминированному автомату-спецификации, а автомат-реализация является детерминированным, имеет смысл говорить об адаптивном тестировании. В этом случае следующий входной символ определятся по выходной реакции тестируемой реализации на предыдущие входные символы. За счет адаптивности длина теста может оказаться более короткой, чем длина тестовых последовательностей, построенных заранее до начала процесса тестирования [1].

В данной работе мы предполагаем, что автомат-спецификация обладает разделяющей последовательностью и детерминированным сильно связным подавтоматом. Эти условия гарантируют, что только автомат, изоморфный

некоторому подавтомату спецификации с тем же числом состояний, может быть редукцией автомата-спецификации [2]. Соответственно для проверки, является ли проверяемый автомат редукцией спецификации, необходимо пройти по всем переходам проверяемого автомата; последовательность, покрывающую переход, необходимо продолжить разделяющей последовательностью для проверки финального состояния перехода. Как обычно, тестирование осуществляется в два шага. На первом шаге в каждом состоянии проверяемого автомата определяется реакция на разделяющую последовательность. После этого осуществляется обычная проверка переходов в проверяемом автомате посредством покрытия соответствующего перехода и последующей подачей разделяющей последовательности.

В заключение мы отмечаем, что, поскольку проверяющая последовательность будет прикладываться адаптивно, вместо д-передаточных последовательностей можно использовать определенно-достижимые состояния и соответствующие преамбулы [1]. Кроме того, вместо разделяющей последовательности можно использовать различающий тестовый пример [3], если таковые существуют в автомате-спецификации.

Структура работы следующая. Раздел 2 содержит основные определения и обозначения, используемые в работе, а также пример автомата для иллюстрации предложенного алгоритма. В разделе 3 предложен алгоритм построения адаптивной проверяющей последовательности и доказаны соответствующие утверждения. В разделе 4 предложенный алгоритм иллюстрируется на простом примере.

2. Основные определения и обозначения

Конечным автоматом называется пятерка $S = (S, I, O, h_S, s_0)$, где S – конечное непустое множество состояний с выделенным начальным состоянием s_0, I – входной алфавит, O – выходной алфавит, и $h_S \subseteq S \times I \times O \times S$ – отношение переходов. При этом четверка $(s, i, o, s') \in h_S$ называется *переходом* в автомате A из состояния s в s'. Инициальные автоматы используются для описания реактивных систем, в которых присутствует сигнал СБРОС, переводящий систему из любого состояния в некоторое фиксированное (начальное) состояние. Предполагается, что этот сигнал реализован правильно в любой реализации, но является достаточно дорогим, чтобы использовать его в течение тестирования несколько раз (например, включение/выключение компьютера).

Автомат S называется *детерминированным*, если для любой пары $(s, i) \in S \times I$ существует не более одной пары $(o, s') \in O \times S$, такой, что $(s, i, o, s') \in h_S$. В противном случае автомат называется *недетерминированным*. При этом, если для каждой пары $(s, i) \in S \times I$ существует хотя бы один переход, то автомат называется *полностью определенным*. Кроме того, если для любой тройки $(s, i,$

$o) \in S \times I \times O$ существует не более одного состояния $s' \in S$ такого, что $(s, i, o, s') \in h_S$, то автомат называется *наблюдаемым*.

Далее в работе для обозначения множества выходных символов $o \in O$ автомата в состоянии $s \in S$ под воздействием входного символа $i \in I$ будем использовать обозначение $out_S(s, i)$.

Пусть $s, s' \in S$, $\alpha = i_1 i_2 \ldots i_k \in I^*$, и $\beta = o_1 o_2 \ldots o_k \in O^*$. Тогда $(s, \alpha, \beta, s') \in h_S$, если существует последовательность состояний $s = s_1, s_2, \ldots, s_k = s'$ таких, что $(s_j, i_j, o_j, s_{j+1}) \in h_S$, $j = 1, \ldots, k - 1$; в этом случае α/β называется *вход-выходной последовательностью* или *траекторией* автомата S в состоянии s. Множество выходных последовательностей β, таких что $\alpha/\beta \in Tr_S(s)$, обозначается $out_S(s, \alpha)$. Множество всех вход-выходных последовательностей автомата S в состоянии s будем обозначать $Tr_S(s)$. Если s – начальное состояние, то множество всех траекторий обозначается Tr_S. При этом автомат S называется *сильно связным*, если каждое состояние достижимо из любого другого состояния по некоторой траектории $\alpha/\beta \in Tr_S$.

В нашей работе мы будем рассматривать отношения конформности между двумя автоматами, один из которых будет описывать эталонное поведение системы, т.е. является *автоматом-спецификацией* (обозначение: S), а второй описывает поведение некоторой реализации системы, т.е. является *автоматом-реализацией* (обозначение: P). В настоящей работе автоматы P и S являются полностью определенными автоматами, причем S является наблюдаемым, а P - детерминированным автоматом. Автомат P называется *конформным* спецификации, если P есть редукция автомата S.

Формально полностью определенный автомат P является *редукцией* полностью определенного автомата S, если для любой входной последовательности справедливо: $out_P(p_0, \alpha) \subseteq out_S(s_0, \alpha)$.
Автомат P является *подавтоматом* автомата S, если $P \subseteq S$, $p_0 = s_0$ и $h_P \subseteq h_S$.

Пересечением автоматов $S = (S, I, O, h_s, s_0)$ и $P = (P, I, O, h_P, p_0)$ называется максимальный связный подавтомат Q автомата $(S \times P, I, O, h, s_0p_0)$, в котором $(sp, i, o, s'p') \in h_Q$, если и только если $(s, i, o, s') \in h_S$ и $(p, i, o, p') \in h_P$.

Утверждение 1 [1]. Полностью определенный детерминированный автомат P является редукцией полностью определенного наблюдаемого автомата S, если и только если пересечение этих автоматов есть полностью определенный автомат.

Состояния автомата S называются *разделимыми*, если существует входная последовательность α, такая, что множества выходных последовательностей в любых двух состояниях s и p автомата S на последовательность α не пересекаются, т.е. $out_S(s, \alpha) \cap out_P(p, \alpha) = \varnothing$ (обозначение $s \nsim_\alpha p$). Последовательность α называется *разделяющей* для автомата S, если эта

последовательность является разделяющей для любых двух различных состояний автомата.

Состояние s' автомата S называется *детерминировано достижимым* или *д-достижимым* из состояния s, если существует входная последовательность α, такая что для любой последовательности $\beta \in out_S(s, \alpha)$, последовательность α/β переводит автомат из состояния s в состояние s'. Последовательность α в этом случае называется *д-передаточной* последовательностью из состояния s в состояние s' (обозначение: $\alpha_{ss'}$).

В данной работе мы рассматриваем построение проверяющих тестов относительно модели неисправности $<S, \leq, \mathfrak{R}>$, в которой S – полностью определенный наблюдаемый автомат-спецификация, называемый часто *эталонным* автоматом, $\leq$ - отношение редукции, $\mathfrak{R}$ - область неисправности, т.е. конечное множество полностью определенных сильно связных детерминированных автоматов, которые описывают поведение систем со всеми возможными неисправностями, т.е. мы предполагаем, что поведение тестируемой реализации описывается некоторым автоматом из множества $\mathfrak{R}$. Под *проверяющей последовательностью* относительно модели неисправности $<S, \leq, \mathfrak{R}>$ понимается входная последовательность α, такая что при адаптивной подаче последовательности на любой автомат $P \in \mathfrak{R}$, который не является редукцией автомата S, наблюдаемая выходная реакция не содержится в множестве выходных реакций автомата S.

В настоящей работе мы полагаем, что автомат-спецификация обладает разделяющей последовательностью и сильно связным детерминированным подавтоматом с таким же числом состояний, т.е. для любой пары состояний s и s' существует d-передаточная последовательность $\alpha_{ss'}$. Эти условия гарантируют, что только автомат, изоморфный некоторому подавтомату спецификации с тем же числом состояний, может быть редукцией автомата-спецификации [2].

Утверждение 2 [2]. Полностью определенный детерминированный автомат P является редукцией полностью определенного наблюдаемого автомата S с n состояниями, который обладает разделяющей последовательностью и сильно связным детерминированным подавтоматом с таким же числом состояний, если и только если автомат P имеет n состояний и изоморфен подавтомату автомата S.

В качестве примера рассмотрим автомат на рис. 1.

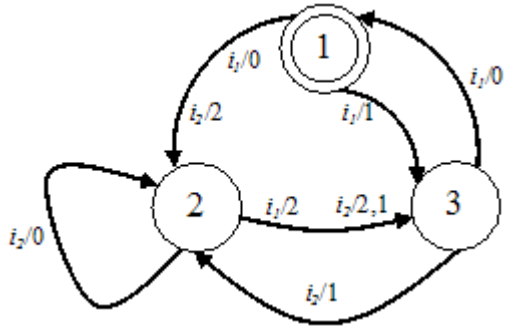


Рис. 1 Автомат-спецификация S.

Автомат на рис. 1 обладает разделяющей последовательностью i_2i_1. Реакции на эту последовательность для каждого состояния автомата приведены в таблице 1.

Табл. 1 Выходные реакции на разделяющую последовательность.

Состояние	Выходные реакции на входную последовательность i_2i_1
1	22
2	02, 20, 10
3	12

Кроме того, автомат обладает сильно связным детерминированным подавтоматом: $\alpha_{12} = i_2$, $\alpha_{31} = i_1$ и $\alpha_{23} = i_1$.

В следующем разделе мы предлагаем метод построения адаптивной проверяющей последовательности относительно модели неисправности $<S, \leq, \mathfrak{R}>$, в которой S – полностью определенный наблюдаемый автомат-спецификация с n состояниями, обладающий сильно связным детерминированным подавтоматом с таким же числом состояний и разделяющей последовательностью δ, $\leq$ - отношение редукции, $\mathfrak{R}$ - множество всех полностью определенных детерминированных автоматов с числом состояний не более n.

3. Алгоритм построения адаптивной проверяющей последовательности

Вход. Автомат $S = (S, I, O, h_S, s_0)$ с n состояниями, разделяющая последовательность δ для автомата S, д-передаточные последовательности $\alpha_{ss'}$ для каждой пары состояний s и s', полностью определенный детерминированный автомат P с числом состояний не более n

Выход. Вердикт *pass*, если *P* есть редукция *S*, или вердикт *fail*, если *P* не есть редукция *S*; проверяющая последовательность σ.

Как обычно, алгоритм проверки, является ли предъявленный автомат редукцией автомата-спецификации, содержит две части. Данное разбиение изображено в виде блочной схемы на рис. 2.

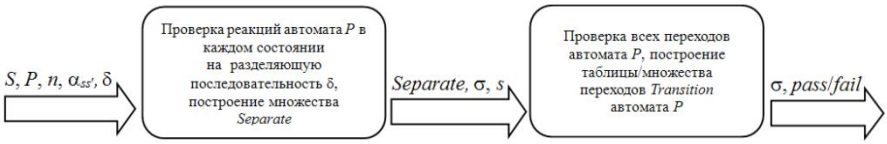


Рис. 2 Алгоритм построения адаптивной проверяющей последовательности

В первом блоке алгоритма идентифицируются состояния проверяемого автомата. Если автомат-реализация *P* реагирует на эту часть проверяющей последовательности ожидаемым образом, то в этом блоке посредством разделяющей последовательности устанавливается взаимно однозначное соответствие между состояниями проверяемого автомата и автомата-спецификации. Во втором блоке алгоритма устанавливается взаимно однозначное соответствие между переходами проверяемого автомата и автомата-спецификации.

3.1. Построение множества Separate

Множество *Separate*, формируемое в первом блоке алгоритма, содержит тройки (s, ρ, s'), где s – текущее состояние автомата P, $\rho = out_P(s, \delta)$, т.е. выходная реакция автомата P в состоянии s на разделяющую последовательность δ, s' - состояние, в которое автомат P при этом перешел.

Алгоритм 1 идентификации состояний проверяемого автомата

Вход. Автомат-спецификация *S*, разделяющая последовательность δ и д-передаточные последовательности $\alpha_{ss'}$, проверяемый автомат *P*, текущая проверяющая последовательность σ;
Выход: Множество *Separate* или сообщение, что предъявленный автомат не является редукцией автомата *S*;

σ: = ε;
Separate: = ∅;
s: = s_0;
flag: = 0;

Шаг 1.

Пока *flag* = 0 выполнить:

{

Подать на автомат *P* последовательность δ;

σ**:** = σδ;

Если реакция ρ на δ автомата *P* не содержится в множестве реакций автомата *S* на последовательность δ, **то** выдать вердикт '*fail*' и сообщение, что предъявленный автомат не является редукцией автомата *S*, **КОНЕЦ** алгоритма;

Иначе определить преемник *s*′ состояния *s* по траектории δ/ρ;

Если в множестве *Separate* есть тройка (*s*, ρ, *s*′), то *flag* : = 1;

/*произошло зацикливание по разделяющей последовательности;*/

Иначе занести тройку (*s*, ρ, *s*′) в множество *Separate*;

s: = *s*′;

}

Шаг 2.

Если мощность множества *Separate* равна *n*, **то КОНЕЦ** алгоритма;

/*установлено взаимно однозначное соответствие между состояниями автоматов *P* и *S* */

Иначе определить состояние *s*′, для которого не проверена выходная реакция на последовательность δ, т.е. в множестве *Separate* нет тройки (*s*′, *, *);

{

Подать $\alpha_{ss'}$ на автомат *P*

σ**:** = σ $\alpha_{ss'}$;

}

Если реакция автомата *P* не содержится в множестве реакций автомата *S* на последовательность $\alpha_{ss'}$, **то** выдать вердикт '*fail*' и сообщение, что предъявленный автомат не является редукцией автомата *S*, **КОНЕЦ** алгоритма;

Иначе Шаг 1.

3.2. Построение множества Transition

Под множеством *Transition* понимается множество четверок (*s*, *i*, *o*, *s*′), т.е. практически множество переходов в автомате-реализации *P*.

Алгоритм 2 проверки всех переходов предъявленного автомата *P*

Вход. Автомат-спецификация *S*, разделяющая последовательность δ и текущее состояние автомата после подачи первой части проверяющей последовательности на проверяемый автомат *P*, множество *Separate*;
Выход. Вердикт *pass*, если *P* есть редукция *S*, или вердикт *fail*, если *P* не есть редукция *S*; множество переходов автомата *P*;

Transition: = ∅;

Пока мощность множества *Transition* не равна произведению *n* на число входных символов выполнить:
{
Если в текущем состоянии *s* есть переход по входному символу *i*, не входящий в множество *Transition*, **то**
Подать на автомат *P* последовательность *i* δ;
σ**:** = σ *i* δ;
Если реакция *o*ρ автомата *P* на последовательность *i* δ не содержится в множестве реакций автомата *S* на эту последовательность, **то** выдать вердикт '*fail*' и сообщение, что предъявленный автомат не является редукцией автомата *S*, **КОНЕЦ** алгоритма;
Иначе по множеству *Separate* определить тройку (*s*′, ρ, *s*′′) ∈ *Separate*;
Занести четверку (*s*, *i*, *o*, *s*′) в множество *Transition*;
s: = *s*′′;
Иначе по множеству *Transition* найти входную последовательность γ для перехода из текущего состояния *s* в состояние *s*′, в котором есть переход по входному символу *i*, не входящий в множество *Transition*;
/* Последовательность существует, т.к. по свойствам автомата-спецификации *S* редукциями *S* могут быть только автоматы с тем же числом состояний*/
Подать на проверяемый автомат последовательность γ;
σ**:** = σ γ;
s: = *s*′;
}

Выдать вердикт 'pass' и сообщение, что предъявленный автомат является редукцией автомата *S*, **КОНЕЦ** алгоритма;

Сформулируем ряд утверждений, из которых следует, что во второй части алгоритма выдается вердикт *pass*, если *P* есть редукция *S*, и вердикт *fail*, если *P* не есть редукция *S*.

Во-первых, по свойствам разделяющей последовательности, справедливо следующее утверждение.

Утверждение 3. Если перед переходом ко второй части алгоритма не было выдано сообщение, что предъявленный автомат не является редукцией автомата *S*, то можно сделать следующие выводы о предъявленном автомате *P*:

1) Автомат *P* имеет в точности *n* состояний;

2) Множество троек *Separate*, в котором тройка (*s*, ρ, *s*′) означает, что предъявленный автомат *P* под действием разделяющей последовательности δ из состояния, соответствующего состоянию *s* в автомате *S*, выдает выходную последовательность ρ и переходит в состояние, соответствующее в автомате *S* состоянию *s*′.

Утверждение 4. Если во второй части алгоритма мощность множества *Transition* не равна произведению *n* на число входных символов и переходы из текущего состояния по всем входным символам присутствуют в множестве *Transition*, то в множестве *Transition* существует последовательность переходов с входной последовательностью γ для перехода из текущего состояния *s* в состояние *s*′, в котором есть переход по входному символу *i*, не входящий в множество *Transition*.

Утверждение доказывается методом от противного. Если это не так, то построен подавтомат автомата *P* с числом состояний меньше *n*, который есть редукция автомата *S*, что невозможно в силу свойств автомата *S* (утверждение 2).

Теорема 1. Алгоритм выдает вердикт '*pass*', если и только если предъявленный автомат является редукцией спецификации.

Доказательство. Если реакция на построенную входную последовательность автомата *P* содержится в множестве реакций автомата *S* на эту последовательность (т.е. в результате работы алгоритма выдан вердикт '*pass*'), то после выполнения первой части алгоритма установлено взаимно однозначное соответствие между состояниями автоматов *P* и *S*. Во второй части алгоритма устанавливается такое соответствие между переходами автоматов. Таким образом, в пересечении *P* и *S* присутствуют только пары *ps* состояний, такие что выходная реакция автомата в состоянии *p* на каждый входной символ содержится в множестве выходных реакций автомата *S* в состоянии *s* на этот входной символ. Соответственно, если в процессе тестирования не было выдано сообщения, что автомат *P* не является редукцией автомата *S*, которое выдается только в случае, когда реакция автомата *P* на некоторую входную последовательность не содержится в множестве выходных реакций автомата *S* на эту входную последовательность, то, согласно утверждению 1, предъявленный автомат есть редукция автомата *S*.

4. Иллюстрация алгоритма на примере

Рассмотрим работу алгоритма построения проверяющей последовательности на примере, приведенном на рисунке 1. Для данного автомата разделяющая последовательность δ: $i_2 i_1$. В качестве реализации рассмотрим автомат на рис. 3

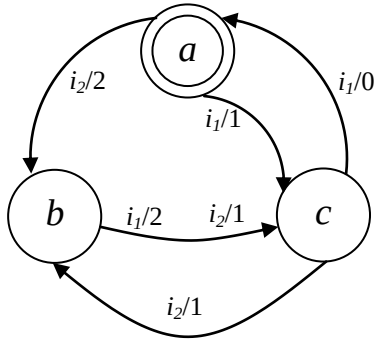


Рис. 3 Проверяемый автомат P, рассматриваемый как реализация автомата на рис. 1

Согласно алгоритму, в первой части алгоритма на автомат *P* подаем входную последовательность δ, пока не перейдем в уже рассмотренное состояние. В данном случае получаем в ответ на i_2i_1 выходную последовательность 22, что соответствует спецификации. Далее, заносим тройку (1, 22, 3) в множество *Separate*. При дальнейшей подаче $i_2i_1i_2i_1$ выясняем, что произошло зацикливание, т.к. в множестве *Separate* уже есть тройка (3, 12, 3). Согласно шагу 2 определяем состояние, для которого нет соответствующей строки в множестве *Separate* – это состояние 2. Поэтому подаем $\alpha_{32}=i_2$ для перехода из состояния 3 в состояние 2 и возвращаемся на шаг 1.

Подаем i_2i_1 и получаем новую тройку в множество *Separate*: (2, 10, 1). При повторной подаче i_2i_1 выясняем, что тройка (1, 22, 3) уже есть в *Separate*.

Таким образом, на выходе первого алгоритма множество *Separate* имеет вид, представленный в таб. 2

Таблица 2.

s	**ρ**	***s′***
1	22	3
2	10	1
3	12	3

Кроме того, проверяющая последовательность σ имеет вид: $i_2i_1\ i_2i_1i_2i_1\ i_2\ i_2i_1i_2i_1$ и текущим состоянием является состояние 3.

Далее необходимо проверить множество переходов. В соответствии с алгоритмом проверки переходов определяем, есть ли в текущем состоянии 3 переход, которого нет в *Transition*. Такой переход есть – например, переход по входному символу i_1. После подачи i_1 на выходе автомата *P* получаем 0, далее подаем δ и получаем на выходе 22, что соответствует третьей строке множества *Separate* и означает, что после подачи i_1 автомат *P* перешел,

соответствующее состоянию 3. По множеству *Separate* определяем, что после подачи δ мы возвращаемся в состояние 3. В данном состоянии есть непроверенный по входному символу i_2. После подачи этого символа, на выходе автомата *P* получаем выходной символ 1, и после подачи δ по полученной реакции 10 можно установить, что проверяемый автомат из состояния 3 под действием входного символа i_2 перешел в состояние, соответствующее состоянию 2 в автомате-спецификации. На данном этапе $\sigma = i_2i_1i_2i_1i_2i_1i_2i_2i_1i_2i_1i_1i_2i_1i_1i_2i_1$ и т.д. Аналогичным образом проверяются остальные переходы автомата *P*.

По завершению построения проверяющей последовательности на выходе будет получен вердикт *pass*, и действительно автомат на рис. 3 является редукцией автомата на рис. 1. При этом $\sigma = i_2i_1i_2i_1i_2i_1i_2i_2i_1i_2i_1 + i_1\ i_2i_1\ i_2i_2i_1i_1i_2i_1i_1i_2i_2i_1i_2i_1i_2i_1i_2i_2i_2i_1$.

5. Заключение

В данной работе представлен предложенный нами алгоритм построения адаптивной проверяющей последовательности для недетерминированного автомата относительно редукции**.** Данный алгоритм позволяет определить, является ли предъявленный для тестирования детерминированный автомат редукцией автомата-спецификации.

Необходимо отметить, что в данной работе мы не рассматривали оптимизацию строящейся проверяющей последовательности; задачи оптимизации проверяющей последовательности требуют дальнейших исследований.

В заключение мы отмечаем, что поскольку проверяющая последовательность будет прикладываться адаптивно, вместо д-передаточных последовательностей можно использовать определенно-достижимые состояния и соответствующие преамбулы [1]. Кроме того, вместо разделяющей последовательности можно использовать различающий тестовый пример [3], если таковой существует в автомате-спецификации. Использование таких адаптивных установочных и различающих тестовых примеров может существенно сократить длину проверяющей последовательности.

Благодарность. Работа частично поддержана проектом 2.739 госзадания МинОбрНауки РФ. В заключение хочу выразить благодарность профессору Евтушенко Н.В. за интересные дискуссии при подготовке статьи.

Список литературы

[1]. Petrenko A. Adaptive Testing of Deterministic Implementations Specified by Nondeterministic FSMs / A. Petrenko, N. Yevtushenko // ICTSS, 2011. P. 162-178.
[2]. М.В. Ветрова. Разработка алгоритмов синтеза и тестирования конечно автоматных компенсаторов. Дисс. На соискание степени канд. техн. наук, Томский госуниверситет, 2004..
[3]. Н.Г. Кушик. Методы синтеза установочных и различающих экспериментов с недетерминированными автоматами. Дисс. На соискание степени канд. физ.-мат. наук, Томский госуниверситет, 2013.

Deriving checking sequences for nondeterministic Finite State Machines with respect to the reduction relation

Anton Ermakov <antonermak@inbox.ru>
National Research Tomsk State University,
634050, Lenina ave., 36, Tomsk, Russia

Abstract. Most FSM based methods for test derivation are developed for initialized Finite State Machines (FSM) and the latter means that a reliable reset is assumed in an implementation under test in order to glue test sequences together. If the reset is rather expensive then the number of test sequences has to be reduced and when it is reduced to a single sequence, this sequence is called a checking sequence. In this paper, a methods is proposed for deriving an adaptive checking sequence when the specification FSM is nondeterministic and the conformance relation is the reduction relation. The latter means that the behavior of a conforming implementation should be contained in the behavior of the specification. A method returns an adaptive checking sequence that detects each nonconforming implementation that has not more states than the specification FSM under the conditions that the specification has a distinguishing sequence and a deterministic strongly connected submachine. These conditions can be weakened for the case when the specification has a distinguishing test case and each state of the specification is definitely reachable from another state. The testing process is adaptive, i.e., the next input is determined based on the outputs produced for the previous inputs. Such adaptive distinguishing sequences can be shorter than preset checking sequences.

Keywords: test derivation, checking sequence, nondeterministic Finite State Machine, reduction relation

References

[1]. Petrenko A. Adaptive Testing of Deterministic Implementations Specified by Nondeterministic FSMs / A. Petrenko, N. Yevtushenko // ICTSS, 2011. P. 162-178.
[2]. M.V. Vetrova. Razrabotka algoritmov sinteza i testirovaniya konechno avtomatnih kompensatorov. Diss. na soiskanie stepeni kand. techn. nauk, Tomskiy gosuniversitet [Tomsk State University], 2004.
[3]. N.G. Kushik. Metodi sinteza ustanovochnih i razlichaushih eksperimentov s nedeterminirovannimi avtomatami. Diss. na soiskanie stepeni kand. phis.-mat. nauk, Tomskiy gosuniversitet [Tomsk State University], 2013.

Scalable Evaluation of Distributed On-line Network Monitoring for Behavioral Feedback in Trust Management

[1] *Jorge López <jorge.eleazar.lopez_coronado@telecom-sudparis.eu>*
[1] *Stephane Maag <stephane.maag@telecom-sudparis.eu>*
[2] *Gerardo Morales <gmorales@galileo.edu>*
[1] *Institut Mines Telecom, Telecom SudParis CNRS UMR 5157,*
9 rue Charles Fourier, Évry, Île-de-France, 91000, France
[2] *RLICT Universidad Galileo,*
7a. Av. Final, Calle Dr. Eduardo Suger Cofiño, Zona 10, Guatemala, Guatemala

Abstract. Collaborative systems are growing in use and popularity. The need to boost the methods concerning the interoperability is growing as well; therefore, trustworthy interactions of the different systems are a priority. The decision regarding with whom and how to interact with other users or applications depends on each system. We focus on providing trust verdicts by evaluating the behaviors of different agents, using distributed on-line network monitoring. This will provide trust management systems information regarding a trustee experience, for those trust management systems based on "soft trust". In this work, we propose a scalable evaluation method for any on-line network monitoring system, by using an auxiliary model, an extended finite state automaton (EFSA), and as well as other known methods to reduce the time complexity of the evaluation algorithm.

Keywords: trust management; on-line network monitoring; scalable evaluation;

1. Introduction

Internet applications have become one of the most popular ways to socially interact, make commerce, and create collaborative work; making them a daily part of our living. With time, the collaborative aspects supported by Internet have evolved bringing new tools, methodologies and concepts. These systems keep growing in use and in popularity. The need to boost the interoperability methods related to them is growing as well; making thus trustworthy interactions of the different systems a priority.

These concepts of trust have been brought to computer science. The systems need to interact with users and with other applications. The decision regarding with whom and how to interact with other users or applications depend on each application or system. There are many definitions of trust in the literature, but the one we adopt

here is the one commonly applied and defined in [1], “the firm belief in the competence of an entity to act dependably, securely, and reliably within a specified context”. From this definition different types of trust management engines have been created.

Some trust management systems use security policies and authentication in order to provide the concept of trust. In these types of systems, to determine the entity called trustee, it implies there is a related authentication mechanism. The policy languages are used to express the actions allowed for each trustee. This is called ``hard trust'' because the actions can only be permitted or denied. For example, pioneering systems like PolicyMaker [2], KeyNote [3], REFEREE [4] and SD3 [5], have presented trust management systems based on security policies. More flexible and recent, hard trust management systems have been created, one among them is the work tool TrustBuilder2 [6].

“Soft trust” management systems, on the other hand are trust management systems that are based on concepts like experience, reputation and other dynamic evaluation parameters. For this purpose, the observations of the trustee behaviors are added to evaluate the trustee experience. Most of the works dedicated to trust estimations in different kinds of systems are based on local observations through monitored entities. One example of such systems can be found in [7].

One crucial point is that soft trust management engines assume the evaluation of the behavior is always granted and available for them. Additionally, to the best of our knowledge, no generic methods to check behaviors are found in the trust literature. As well, no formal approaches have been defined considering several points of observation. We propose to use distributed network monitoring techniques to analyze the packets exchanged between entities, in order to prove the interactions are trustworthy based on the observation of network messages in different points of observation. One important characteristic, which is always desirable in trust systems, is to have the trust information, as fast as possible. Using our proposed mechanism, we are able to provide the behavioral feedback of the systems on-line. Our aim is to provide trust information in a generic manner such that, any generic framework can use the information about these behaviors and incorporate it into the trust estimation algorithm. It is our point of view, that trust management systems will benefit from different inputs using different techniques; that is the reason why we do not aim to provide another approach how to assess trust, but rather providing existing trust management systems with behavioral evaluation of interactions.

On-line network monitoring cannot be directly applied by performing formal verification or model checking [8] techniques. The reason is that, a model of the system under test has to be derived in advance, and furthermore a set of properties can be verified for corresponding violations. Typically, the system description is omitted when performing on-line monitoring/passive testing, and therefore, this issue is left out of the scope of the paper. On the other hand, a formal specification of the system under test can be obtained by observing input/output traces and applying machine learning techniques [9]. However, when performing machine

learning techniques, well known problems are encountered, such as statistics gathering, explosion problems (especially, for state transition models), etc. Therefore, in this paper, we discuss how a number of properties can be still verified for a system under test when the formal system specification is absent; and especially how to perform this process in a scalable way.

2. Distributed On-line Monitoring for Behavioral Evaluation

2.1. Approach

Our main objectives are: i) to be able to detect untrustworthy behaviors of entities where all other approaches fail to achieve it, providing feedback as fast as possible; ii) to provide a generic method to describe these untrustworthy behaviors and finally, iii) to test those described behaviors in a scalable manner.

To tackle the first point, the distributed network monitoring approach was proposed. With the use of distributed network monitoring, we can see behaviors that cannot be seen when using a single point of observation. Let us present a possible case scenario, a client computer is sending request to perform operations to a server. Both, the client and the server have a trust management engine, and they have allowed actions and replies from each other. The client computer sends a message of type “A” to the server and at the server and the server receives a type “B” message. The message type “B” is an allowed message type and the server performs the action. If the network traffic for both points of observation could be obtained and compared, an untrustworthy behavior can be detected. This example is illustrated in Fig. 1. Without correlating both points of observation, the untrustworthy behavior cannot be detected, even if having trust management engines incorporated, the systems will consider the interaction trustworthy.

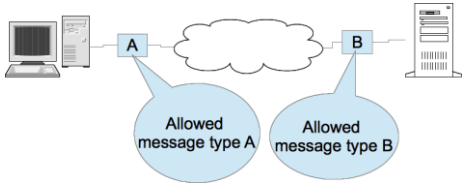


Fig. 1. Different allowed actions at communication ends.

We do not consider this simply a security issue. In fact, due to the trust definition we do not focus if the untrustworthy answer was due to an attack, a software failure (bug), system failure, misconfiguration, etc. The relevant fact is that the interaction was not proper and reporting the untrustworthy interaction as soon as possible is our goal.

2.2. Definitions and Assumptions

In order to accurately understand how our proposed approach solves the stated issues, first we need to introduce some preliminary concepts. A network packet (packet for short) is the abstraction of the transmitted bit-streams in a computer network; this abstraction allows us to interpret a packet as a formatted data unit. A packet is interpreted as a "Message" from a telecommunication protocol, for example a DNS query, a DNS response, etc. Analyzing a packet is to access the data inside that packet to search for particular values; these values have a defined meaning depending on the network protocol. Finally, network monitoring is the technique of analyzing the packets transmitted over a computer network. Several works like, [10], [11] and [12], proposed monitoring approaches considering local observations.

In our paper, we assume that the network packets are being forwarded from the different sources of interest to a monitoring server. Each of these sources contains network entities monitored through network interfaces called points of observation (P.O). We also assume that if the network entity has many interfaces, all the forwarded packets from the same network entity will be considered at the same point of observation.

The sequence of packets from a point of observation is called a network trace. A network trace (trace for short) is potentially infinite. When we have different traces from the points of observation, we can analyze the packets from one trace and create a relationship to another trace, defining the concept of distributed network monitoring.

In order to provide evaluation about behaviors, relationships between packets from different POs are created. The relationships are created with the packets' fields and conditions that hold over those fields in regards of other packets. Basically, the relationships are made performing comparisons. We can compare the values of these observations with constant values or variable values. The variable values are extracted from other packets (previously observed packets). These comparisons are defined formally in our work [13], by the definition of *atoms* and we also note that for the time relationships, we assume the network traces are synchronized using the NTP protocol [14]. Since there are multiple network traces from multiple POs, the comparisons can be done from: i) a specific network trace, that is using a specific point of observation, ii) any network trace, except a specific one or iii) any network trace, that is, at any point of observation, i.e., not specifying a point of observation.

The packet relationships and comparing the values will result in a composition. This composition is formally defined as a conjunction of atoms, which we call a *prototype*. A *prototype* is an abstract model of all the necessary and sufficient conditions a network packet should meet, including all its dependencies. For example, to describe a DNS query for an IP address, a packet prototype will be expressed in the formal language as: $p.flags.response = 0 \wedge p.queries.type = \text{'A'}$.

A *prototype* is a part of the formal definition of formulae. One formula is a formal representation of what we will call a trust property. Many trust properties can be described and formalized in order to describe trust on an environment or context. Once the desired trust properties are checked on the network traces, we can give a verdict regarding the checked trust property. The possible verdicts are *pass* and *fail* if the statement is present. If the trust property does not reach a verdict, the result will be temporarily assigned as an inconclusive verdict. If many trust properties are described, then, different trust verdicts can be obtained.

The motivation and a method behind our approach were presented previously. Now, in order to test the proposed trust properties using distributed network monitoring we need to be able to express those properties. It is not sufficient to express the trust properties, in fact, we need to accurately express them, not leaving room for any ambiguity. Considering that, we need to employ a formalism. A formalism is not only useful to unambiguously express the properties, but, also for the software tools to be able to provide accurate verdicts. Without a doubt, our approach has a higher value, when verdicts can be automatized with a software program. Further, when providing a formalism, more researchers related to the field can generate trust properties to test. Because of those reasons we have created the necessary formal approach.

We decided to use our own approach rather than using other existing ones, the reason is that with the use of our formalization, we can describe the packets finely parameterized and at a granular level. Thus, we can make more complex and detailed relationships between packets. Another reason is that new application protocols rely heavily on the data parts and their semantics, for this reason they require a more data-oriented checking which the other approaches are not able to provide. Even old protocols have semantics that if the packets are treated as bit-streams some data values can be inaccurately obtained. For example, in the DNS protocol, the DNS notation and data compression method allow to specify a pointer to previously used data in the packet to avoid duplication of data (see [15]).

The formalism basic and most important principles are: the representation of a protocol message (packet) and the formal language lexical, syntactical and semantical properties; nevertheless, for the scope of this paper, only knowing the concepts regarding the language, namely, *atoms* and *prototypes* in particular are enough, and that is the reason why the interested reader might look for the formal language definition in our previous work [13]. Therefore, we present only the basic concepts regarding the packet hierarchical representation next.

A communication protocol message can be represented as a hierarchical set of label-value pairs. The representation of the packet will have the form defined by a message representation:

Definition 1: A message representation $\mathcal{M}$ is defined by the set of pairs $\mathcal{M} = \{(l, v) | l \in \mathcal{L} \wedge v \in \mathcal{S} \cup \Re \cup \mathcal{M}'\}$, where $\mathcal{L}$ is a predefined set of string labels, $\mathcal{S}$ represents the set of string values, $\Re$ represents the set of real numbers, and $\mathcal{M}'$ is a message representation sub-set.

For a given network protocol P, an associated message representation $\mathcal{M}$ can generally be defined by the set of labels and data values derived from the message format defined in the protocol specification. A message of a protocol P is any element $m \in \mathcal{M}$. For each $m \in \mathcal{M}$, we add two fields: a real number $t_m \in \Re^+$, which represents the time when the message m is received or sent by the monitored entity, and a PO string label which represents the point of observation from which the message *m* is collected.

Example of a message representation: a possible message for the DNS protocol [15]; specified using the previous definition could be:

$\mathcal{M}$={(time,154.576889000),(PO,"ADS"),(query_id,58921),(flags,{(response,0),(op code,std_query),(truncated,0),

(recursion_desired,1),(reserved,0),(non_auth_data_acceptable,0)}),(questions,1),(answers,0),(authorityRRs,0),

(additionalRRs,0),(queries,{(name,"telecom-sudparis.eu"),(type,"A"),(class,"IN")})}

Representing a DNS query for the IP address of the associated domain name telecom-sudparis.eu.

For any given network protocol we have a mapping function between the bit-stream and the message representation.

Definition 2: The mapping function is the function $\mathcal{F}: \mathcal{B} \mapsto \mathcal{M}$, where $\mathcal{B}$ is the bit-stream of the network protocol and $\mathcal{M}$ is a message hierarchical representation as presented in Definition 1.

Once having the representation of the network messages, and the concepts of *prototypes* and *atoms*, some important constrains of on-line network monitoring systems need to be mentioned: i) a prototype has a set of conditions which can involve the packet itself or previously stored packets (dependencies); ii) for each packet, all prototypes must be tested, since, each packet could be observed at any given state during the execution time.

3. Scalable Evaluation of On-line Network Monitoring Systems

The evaluation process in an on-line monitoring system consists in evaluating if each packet satisfies the desired trust properties we need to check. Therefore, a scalable way for the evaluation algorithm is perhaps the biggest requirement. The trust properties have a set of conditions (*atoms*) that packet's data need to match against constant values or against the values of previously stored packets, as explained before. After matching the packet's conditions, checking if the matched packet completed a trust property is necessary. In order to provide verdicts

regarding the trust properties we developed a first approach using the algorithm presented in our work in [16]. The worst-case analysis of the time work performed by the previously mentioned algorithm is expressed by the following equation:

$$T(eval\_prots) = 3N_p + \sum_{i=1}^{N_p} NPA_i + \sum_{i=1}^{N_p} 2NPD_i + \sum_{i=1}^{N_p} (N_p - i)QL_i + \sum_{i=1}^{N_p} (N_p - i)(QL_i * NDA_i)$$

Where N_p is the number of prototypes in the formulae, NPA_i is number of atoms that require no dependencies of the *ith* prototype, NPD_i is the number of dependencies of the *ith* prototype, QL_i is the length of the queue of stored packets of the *ith* prototype, and NDA_i is the number of atoms that require dependencies of the *ith* prototype.

The experimental results achieved with the first algorithm are good. However, due to the on-line monitoring constraints, we are required to create the most scalable algorithm for the evaluation of trust properties. Based on the time complexity analysis of our algorithm, we note that the term that dominates the equation is the term, $\sum_{i=1}^{N_p}(N_p - i)(QL_i * NDA_i)$; from this term we can observe that *atoms* (conditions) need to be checked against the stored packet queues and this is being repeated up to $(N_p - i)$ times. In order to create a scalable algorithm, we need to avoid repeating any checks for all packets.

In order to improve the algorithm, known techniques are applied. First, we propose to make use of a data structure that will aid avoiding repeated checks. In addition to that, we propose to keep a track of previously visited packets in the stored queue to avoid re-visiting packets, which did not match previous tests, and therefore, not to check stored packets that do not meet all the necessary conditions.

We have chosen to use a tree-structured (single rooted) extended finite state automaton (EFSA) as the structure for the scalable evaluation. The reason is that, this structure fits the desired purpose of the algorithm. We propose evaluating the packets by doing the atomic test once and to keep track of the already tested atoms (a transition model based on predicates) and then, when a packet is found to match a *prototype* (at some accept state), execute some actions (updating functions), for instance, storing the packet on a queue or reporting a property verdict. These types of models have become popular to achieve scalable algorithms, for example, several works like, [17,18] use different types of automata, finite, non-deterministic, hybrid and extended to evaluate a regular expression language to achieve a scalable deep packet inspection.

Our target is to generate the EFSA from the necessary *prototypes*. The strategy in order to avoid repeating atomic tests is to generate transitions from the root state, adding predicates of the atomic tests which are more common at the beginning and

creating related atomic tests (atomic tests which are part of the same *prototype*) along the same path; for the next prototypes, uncommon atoms will be branches added at the current state after following the common transitions. Therefore, our algorithm to generate the EFSA relies on three principal actions: i) comparing each atom and add a count of how many times it appears in the formulae; ii) sort the prototypes putting first the ones containing the most common atoms, then, do a nested-sorting according the second most common atom, and so on; iii) finally, going along the path of the EFSA creating new nodes branching with its respective transitions based on the atoms or just following the already existing ones (starting from the root) and adding the proper updating functions. The algorithm to generate our EFSA based on the formulae prototypes can be found in the Algorithm 1.

```
Input: Prototypes 𝒫 , State root
Output: EFSA
foreach atom ∈ 𝒫 do
    repeated[atom] ← count(atom, 𝒫);
end
maxRepeated ← max(repeated);
sortedPrototypes ← 𝒫;
repeat
    atom ← atomRepeated(maxRepeated, 𝒫);
    sortedPrototypes ← sortWRT(sortedPrototypes, atom)          /* nested-sorting */
    repeated[atom] ← -repeated[atom];
    maxRepeated ← max(repeated);
until maxRepeated>1;
current ← root;
foreach P ∈ sortedPrototypes do
    while (atom ← higerCountAtom(P)) > 0 do
        if atom ∉ transitions(current) then
            addTransition(current, atom);
        end
        delete(atom,P);
        doTransition(current, atom);
    end
    addUpdatingFuncs(current, selectFuncs(P))        /* appropriate for the prototype */
end
```

Algorithm 1. EFSA generation algorithm.

Example of an EFSA generation: Let us consider the trust property: "For all responses from an authoritative DNS server, all future responses from other points of observation are the same replies from the authoritative DNS server if the queries are the same".

Using our approach we express this trust property by having the following prototypes:

$p1 \leftarrow p.dns.flags.R = 0 \wedge p.PO = 'ADS'$

$p2 \leftarrow p.dns.flags.R = 1 \wedge p.PO = 'ADS' \wedge p.dns.ID = p1.dns.ID$

$p3 \leftarrow p.dns.flags.R = 0 \wedge p.PO \neq 'ADS' \wedge p.dns.queries = p1.dns.queries$

$$p4 \leftarrow p.dns.flags.R = 1 \wedge p.PO \neq' ADS' \wedge p.dns.ID = p3.dns.ID \wedge p.dns.answers = p2.dns.answers$$

The atom count for these prototypes is:

$p.dns.flags.R = 0 \leftarrow 2$
$p.PO = 'ADS' \leftarrow 2$
$p.dns.flags.R = 1 \leftarrow 2$
$p.dns.ID = p1.dns.ID \leftarrow 1$
$p.PO \neq' ADS' \leftarrow 2$
$p.dns.queries = p1.dns.queries \leftarrow 1$
$p.dns.ID = p3.dns.ID \leftarrow 1$
$p.dns.answers = p2.dns.answers \leftarrow 1$

For this particular example, the order of the prototypes is not altered when sorting them. Finally, the generated EFSA by our algorithm (Algorithm 1) is represented in the Fig. 2.

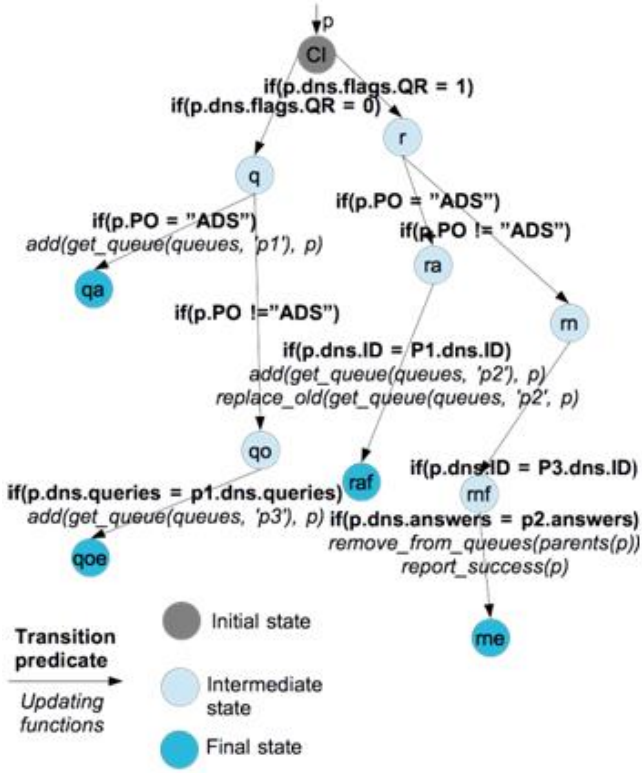


Fig. 2. Generated EFSA example

Once having the generated EFSA, we can introduce the proposed algorithm that is used to evaluate the packets using the auxiliary data structure we generated. The algorithm is shown as Algorithm 2:

```
Input: EFSA A, StoredPackets sps, Packet p_i
Output: Property verdicts
repeat
    transitioned ← false;
    foreach predicate in getTransitions(getCurrentState(A)) do
        if hasDependencies(predicate) then
            partialResult ← false;
            tempStoredPackets ← createStoredPackets();
            foreach storedPacket in getStoredPackets(sps, predicate) do
                if evalDependency(predicate, p_i, storedPacket) then
                    partialResult ← true;
                    add(tempStoredPackets, storedPacket)     /* avoid unnecessary checks */
                end
            end
            setStoredPackets(sps, predicate, tempStoredPackets);
            transitioned ← partialResult;
        else
            transitioned ← evalIndependent(predicate, p_i);
        end
        if transitioned then
            foreach updatingFunc in getUpdatingFuncs(A, predicate) do
                execute(updatingFunc, p_i);
            end
            setCurrentState(A, getStateTransition(A, predicate));
            goto end_loop;
        end
    end
    end_loop :
until transitioned;
```

Algorithm 2. Evaluation algorithm using EFSA.

Finally, we can proceed to calculate the complexity of the evaluation algorithm using the auxiliary EFSA (as shown in Algorithm 2). The work of the algorithm can be expressed by:

$$T(eval\_states) = \sum_{i=1}^{|S|} (\Theta(1) + T(eval\_trans_i)),$$

where $|S|$ is the cardinality of the set of states in the EFSA. Respectively, the work of $T(eval\_trans_i)$:

$$T(eval\_trans_i) = \sum_{j=1}^{|TA_i|} (\Theta(1) + \Theta(1) + \Theta(1) + T(eval\_sp_j) + \Theta(1) + \Theta(1)) + \Theta(1) + \sum_{j=1}^{|U_i|} (\Theta(1)) + \Theta(1),$$

where $|TA_i|$ is the cardinality of the set of transitions of the *ith* element of the state set, $|U_i|$ is the cardinality of the set of updating functions for the *ith* state executed transition. Subsequently, The work of $T(eval\_sp_j)$:

$$T(eval\_sp_j) = \sum_{k=1}^{|Q_j|} (\Theta(1) + \Theta(1) + \Theta(1)),$$

where $|Q_j|$ is the length of the queue of the *ith* prototype stored packets queue.

Substituting and simplifying the equations we get that (we omit the algebraic operations):

$$T(eval\_states) = 3|S| + 5\sum_{i=1}^{|S|} |TA_i| + \sum_{i=1}^{|S|} |U_i| + 3\sum_{i=1}^{|S|} \sum_{j=1}^{|TA_i|} |Q_i|$$

We note that counting from all states each transition is the equivalent to count all transitions, i.e., $|T|$, the cardinality of all transitions. Similarly, counting from all states each updating function is the equivalent to count all updating functions, i.e., $|U|$ is the cardinality of all updating functions. After this substitution in the previous equation we get:

$$T(eval\_states) = 3|S| + 5|T| + |U| + 3\sum_{i=1}^{|T|} |Q_i|$$

The complexity of our algorithm results in an improved linear complexity, $O(|T|) = \sum_{i=1}^{|T|} |Q_i|$. We also note that any algorithm that runs in linear time can only modify a linear amount of memory cells and therefore, the space complexity of the algorithm yields a linear space complexity. It is also important to remark that the complexity of the algorithm (both in time and space) highly depends on the length of the stored queues of packets. In our previous works we have proposed having a continuous parallel process that given a timeout threshold, will remove from the packet queues unused packets. We do this in order to avoid resource starvation in the monitoring system.

4. Conclusions

In this paper, we have presented a scalable approach to evaluate on-line network monitoring systems. Furthermore, we have introduced an algorithm that regardless of the language used to express the monitoring properties is capable of generating an auxiliary model to evaluate them; the only requirements are the basic concepts and constraints that any on-line network monitoring system has. The proposed method after creating the data-structure uses a second algorithm that we presented in order to evaluate the packets and provide verdicts regarding them in a linear time.

Our contribution focuses on providing verdicts in a scalable manner and as stated in Section 3, the algorithm highly depends on the length of the queues of previously stored packets. Therefore, our future work includes proposing a complete model that takes timeouts into account and also to extend our current language to be able to express variable timeouts for each *prototype* individually. Naturally, developing a tool that incorporates the proposed approach is included into our perspectives.

References

[1]. T. Grandison, M. Sloman. A survey of trust in internet applications. IEEE Communications Surveys and Tutorials, 2000, vol. 3, no. 4, pp. 2-16.

[2]. M. Blaze, J. Feigenbaum, J. Lacy. Decentralized trust management. Proc. the IEEE Symposium on Security and Privacy, 1996. pp. 164–173. Oakland, CA, USA.

[3]. M. Blaze, J. Feigenbaum, A.D. Keromytis. Keynote: Trust management for public-key infrastructures. Proc. the Springer 6th International Workshop of Security Protocols, 1999. pp. 59–63. Cambridge, UK.

[4]. Y.-H. Chu, J. Feigenbaum, B. Lamacchia, P. Resnick, M. Strauss. Referee: Trust management for web applications. O'Reilly World Wide Web Journal, 1997, vol. 2, no. 3, pp. 127-139.

[5]. T. Jim. Sd3: A trust management system with certified evaluation. Proc the IEEE Symposium on Security and Privacy, 2001. pp. 106–115. Oakland, California, USA.

[6]. A. J. Lee, M. Winslett, K. J. Perano. Trustbuilder2: A reconfigurable framework for trust negotiation. Proc. the Third IFIP WG 11.11 International Conference, 2009. pp. 176–195. West Lafayette, IN, USA.

[7]. Z. Movahedi, M. Nogueira, G. Pujolle. An autonomic knowledge monitoring scheme for trust management on mobile ad hoc networks. Proc. the IEEE Wireless Communications and Networking Conference, 2012. pp. 1898–1903. Paris, France.

[8]. G. Holzmann. The spin model checker : primer and reference manual. Addison-Wesley Professional. 2003, pp. 1-596.

[9]. M.-N. Irfan, C. Oriat, R. Groz. Model inference and testing. Elsevier Advances in Computers, 2013, vol. 89, pp. 89-139.

[10]. D. Lee, R. Miller. Network protocol system monitoring-a formal approach with passive testing. IEEE/ACM Transactions on Networking, 2006, vol. 14, no. 2, pp. 424-437.

[11]. A. R. Cavalli, S. Maag, E. M. de Oca. A passive conformance testing approach for a manet routing protocol. Proc. the ACM Symposium on Applied Computing (SAC), 2009. pp. 207–2011. Honolulu, Hawaii, USA.

[12]. X. Che, F. Lalanne, S. Maag. A logic-based passive testing approach for the validation of communicating protocols. Proc. the 7th International Conference on Evaluation of

Novel Approaches to Software Engineering (ENASE), 2012. pp. 53–64. Wroclaw, Poland.

[13]. X. Che, J. Lopez, S. Maag, G. Morales. Testing trust properties using a formal distributed network monitoring approach. Springer Annals of telecommunications - Annales des télécommunications, 2014. pp. 1-11. doi: 10.1007/s12243-014-0454-3.

[14]. D. L. Mills. Internet time synchronization: the network time protocol. IEEE Transactions on Communications, 1991, vol. 39, no. 10, pp. 1482-1493.

[15]. P. V. Mockapetris, RFC 1035 Domain names — implementation and specification. Internet Engineering Task Force, 1987.

[16]. J. López, X. Che, S. Maag. An online passive testing approach for communication protocols. Proc. the 9th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE), 2014. pp. 136–143. Lisbon, Portugal.

[17]. R. Smith, C. Estan, S. Jha, S. Kong .Deflating the big bang: Fast and scalable deep packet inspection with extended finite automata. Proc. Conference on Data Communication, SIGCOMM '08, 2008. pp. 207–218. New York, NY, USA.

[18]. M. Becchi, C. Wiseman, P. Crowley. Evaluating regular expression matching engines on network and general purpose processors. Proc. The 5th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), 2009. pp. 30–39. New York, NY, USA.

Масштабируемый метод оценки управления доверием на основе распределенных систем онлайн мониторинга

[1] *Х. Лопез <jorge.eleazar.lopez_coronado@telecom-sudparis.eu>*
[1] *С. Мааг <stephane.maag@telecom-sudparis.eu>*
[2] *Ж. Моралес <gmorales@galileo.edu>*
[1] *Institut Mines Telecom, Telecom SudParis CNRS UMR 5157,*
9 rue Charles Fourier, Évry, Île-de-France, 91000, France
[2] *RLICT Universidad Galileo,*
7a. Av. Final, Calle Dr. Eduardo Suger Cofiño, Zona 10, Guatemala, Guatemala

Аннотация. Корпоративные системы для организации и поддержания совместной работы становятся все более популярными. В условиях роста использования таких систем разработка методов, обеспечивающих надежное доверительное взаимодействие вовлеченных агентов, становится одной из приоритетных задач. Решение о том, с какими агентами (другими пользователями или приложениями) и каким образом осуществлять взаимодействие, может быть различным для различных систем. В данной работе мы акцентируем внимание на предоставлении вердикта о степени доверия на основе оценки поведения различных агентов с использованием распределенного сетевого он-лайн мониторинга. Предложенная оценка предоставляет системам управления, основанным на «мягком доверии» информацию об опыте доверителя. В данной работе мы предлагаем масштабируемый метод оценки для любого он-лайн мониторинга с использованием вспомогательной модели расширенного конечного полуавтомата и известных методов для уменьшения временной сложности алгоритма оценки.

Ключевые слова: управление доверием; онлайн сетевой мониторинг; масштабируемая оценка.

Список литературы

[1]. T. Grandison, M. Sloman. A survey of trust in internet applications. IEEE Communications Surveys and Tutorials, 2000, vol. 3, no. 4, pp. 2-16.
[2]. M. Blaze, J. Feigenbaum, J. Lacy. Decentralized trust management. Proc. the IEEE Symposium on Security and Privacy, 1996. pp. 164–173. Oakland, CA, USA.
[3]. M. Blaze, J. Feigenbaum, A.D. Keromytis. Keynote: Trust management for public-key infrastructures. Proc. the Springer 6th International Workshop of Security Protocols, 1999. pp. 59–63. Cambridge, UK.
[4]. Y.-H. Chu, J. Feigenbaum, B. Lamacchia, P. Resnick, M. Strauss. Referee: Trust management for web applications. O'Reilly World Wide Web Journal, 1997, vol. 2, no. 3, pp. 127-139.

[5]. T. Jim. Sd3: A trust management system with certified evaluation. Proc the IEEE Symposium on Security and Privacy, 2001. pp. 106–115. Oakland, California, USA.
[6]. A. J. Lee, M. Winslett, K. J. Perano. Trustbuilder2: A reconfigurable framework for trust negotiation. Proc. the Third IFIP WG 11.11 International Conference, 2009. pp. 176–195. West Lafayette, IN, USA.
[7]. Z. Movahedi, M. Nogueira, G. Pujolle. An autonomic knowledge monitoring scheme for trust management on mobile ad hoc networks. Proc. the IEEE Wireless Communications and Networking Conference, 2012. pp. 1898–1903. Paris, France.
[8]. G. Holzmann. The spin model checker : primer and reference manual. Addison-Wesley Professional. 2003, pp. 1-596.
[9]. M.-N. Irfan, C. Oriat, R. Groz. Model inference and testing. Elsevier Advances in Computers, 2013, vol. 89, pp. 89-139.
[10]. D. Lee, R. Miller. Network protocol system monitoring-a formal approach with passive testing. IEEE/ACM Transactions on Networking, 2006, vol. 14, no. 2, pp. 424-437.
[11]. A. R. Cavalli, S. Maag, E. M. de Oca. A passive conformance testing approach for a manet routing protocol. Proc. the ACM Symposium on Applied Computing (SAC), 2009. pp. 207–2011. Honolulu, Hawaii, USA.
[12]. X. Che, F. Lalanne, S. Maag. A logic-based passive testing approach for the validation of communicating protocols. Proc. the 7th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE), 2012. pp. 53–64. Wroclaw, Poland.
[13]. X. Che, J. Lopez, S. Maag, G. Morales. Testing trust properties using a formal distributed network monitoring approach. Springer Annals of telecommunications - Annales des télécommunications, 2014. pp. 1-11. doi: 10.1007/s12243-014-0454-3.
[14]. D. L. Mills. Internet time synchronization: the network time protocol. IEEE Transactions on Communications, 1991, vol. 39, no. 10, pp. 1482-1493.
[15]. P. V. Mockapetris, RFC 1035 Domain names — implementation and specification. Internet Engineering Task Force, 1987.
[16]. J. López, X. Che, S. Maag. An online passive testing approach for communication protocols. Proc. the 9th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE), 2014. pp. 136–143. Lisbon, Portugal.
[17]. R. Smith, C. Estan, S. Jha, S. Kong .Deflating the big bang: Fast and scalable deep packet inspection with extended finite automata. Proc. Conference on Data Communication, SIGCOMM '08, 2008. pp. 207–218. New York, NY, USA.
[18]. M. Becchi, C. Wiseman, P. Crowley. Evaluating regular expression matching engines on network and general purpose processors. Proc. The 5th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), 2009. pp. 30–39. New York, NY, USA.