ВЕСТНИК ТОМСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
2019 Управление, вычислительная техника и информатика № 48

ДИСКРЕТНЫЕ ФУНКЦИИ И АВТОМАТЫ

УДК 519.713
DOI: 10.17223/19988605/48/10

Е.В. Широкова, С.А. Прокопенко, Н.В. Шабалдина

К ПОСТРОЕНИЮ ПАРАЛЛЕЛЬНОЙ КОМПОЗИЦИИ РАСШИРЕННЫХ АВТОМАТОВ

Работа выполнена при поддержке Российского научного фонда, проект № 16-49-03012.

Рассматривается задача построения параллельной композиции двух взаимодействующих компонент, к одной из которых нет внешнего доступа. Поведение каждой компоненты описывается расширенным автоматом. Исследуются условия, при которых поведение взаимодействующих компонент можно описать расширенным автоматом без перехода к эквивалентным конечным автоматам.
Ключевые слова: расширенный автомат; конечный автомат; параллельная композиция расширенных автоматов.

Сложные системы, например веб-сервисы, обычно являются многоэлементными, т.е. представляются в виде композиции взаимодействующих компонент. Адекватной моделью, позволяющей описать поведение компонент веб-сервиса, является расширенный автомат [1]. Предполагается, что компоненты взаимодействуют в режиме диалога, т.е. в каждый момент времени активной является только одна компонента, а также соблюдается условие так называемой «медленной внешней среды», при котором очередной входной символ поступает на систему только после того, как она произвела внешний выходной символ в ответ на предыдущий внешний входной символ.

Для описания совместной работы взаимодействующих компонент обычно используется операция композиции расширенных автоматов, моделирующих поведение компонент. Однако задача построения композиции расширенных автоматов недостаточно изучена. Как правило, в случае, если области определения контекстных переменных и входных / выходных параметров расширенного автомата конечны, переходят к решению задачи композиции расширенных автоматов путем моделирования поведения последних. В этом случае поведение каждой компоненты описывается эквивалентным конечным автоматом [2], задача композиции которых хорошо изучена [3]. Однако если области определения контекстных переменных и входных / выходных параметров бесконечны либо слишком велики, то по заданному расширенному автомату нет возможности построить эквивалентный конечный автомат. В этом случае можно произвести упрощение исходного расширенного автомата, построив его срез. При построении срезов сохраняются некоторые нужные нам свойства расширенных автоматов.

Существуют различные виды срезов, например так называемый *SliceR*(*M*) [4] – срез, который сохраняет свойства достижимости исходного расширенного автомата. Для построения такого среза из исходного расширенного автомата удаляются все контекстные переменные, от которых не зависят предикаты переходов автомата. Уменьшение числа контекстных переменных дает возможность строить более простые конечные автоматы при моделировании поведения расширенного автомата на параметризированных входных последовательностях.

Срез расширенного автомата *SliceFSM*(*M*) (FSM, или конечноавтоматный срез) [4] строится для решения проблемы различимости состояний в расширенном автомате и не содержит контекстных переменных и выходных параметров. Для построения такого среза необходимо удалить переходы,

на которых предикат зависит от контекстных переменных. Необходимо отметить, что в общем случае FSM-срез не является конечным автоматом, поскольку некоторые переходы в нем могут зависеть от предикатов, которые, в свою очередь, зависят от входных параметров. Таким образом, такой срез является расширенным автоматов с пустым множеством контекстных переменных и выходных параметров.

Также для упрощения расширенного автомата можно построить так называемый *l*-эквивалент [5] путем моделирования поведения расширенного автомата на входных последовательностях длины не больше заданного числа *l*. Построенный *l*-эквивалент является конечным (в общем случае частичным) автоматом.

Методы построения композиции конечных автоматов хорошо изучены [3, 6]. Поэтому, построив для каждой из компонент конечно-автоматную абстракцию одним из вышеописанных способов, мы можем найти совместное поведение компонент в виде конечного автомата. Однако вопрос, каким образом вернуться от конечного автомата композиции к исходной модели, т.е. к расширенному автомату, остается открытым. Это, в частности, связано с тем, что непросто установить обратную связь между переходами в конечном автомате (например, *l*-эквиваленте) и соответствующими им переходами в расширенном автомате. В связи с этим хотелось бы рассмотреть возможность описания взаимодействия компонент композиции без перехода к эквивалентным конечным автоматам.

В работе [7] приводится формальное описание композиции расширенных автоматов в алфавитах I_1, U, O_2, которая является частным случаем композиции, представленной на рис. 1. Также в [7] сформулированы ограничения, при которых такое описание возможно.

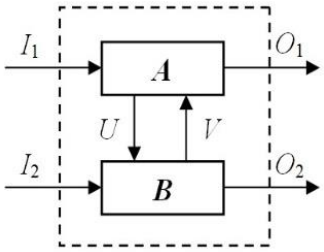


Рис. 1. Структура композиции
Fig. 1. Structure of the composition

Аналогичным образом хотелось бы построить расширенный автомат для параллельной композиции в алфавитах I_1, O_1, U, V и сформулировать условия существования такой композиции. В отличие от предыдущего вида композиции, где компоненты следуют одна за другой и отсутствует внутренний диалог между компонентами, в данном случае такой диалог возможен и компоненты взаимодействуют в режиме «медленной внешней среды» [3]. Такой режим работы композиции хорошо подходит для описания взаимодействия компонент веб-сервиса: клиентского и серверного приложений, так как взаимодействие между клиентским приложением и сервером осуществляется через последовательный обмен запросами и ответами. При этом доступ к серверам, предоставляемым сторонними разработчиками, может быть получен только посредством запросов от клиентского приложения, поэтому сервер рассматривается как встроенная компонента (не имеющая внешних входных и выходных символов).

1. Основные понятия и определения

Конечным автоматом [3] называется пятерка $S = (S, I, O, T_S, s_0)$, где S – непустое конечное множество состояний с выделенным начальным состоянием s_0, I – непустое множество входных символов, называемое входным алфавитом, O – непустое множество выходных символов, называемое выходным алфавитом, $T_S \in I \times S \times S \times O$ – отношение переходов.

Автомат S называется полностью определенным, если для каждой пары $(i, s) \in I \times S$ существует по крайней мере одна пара $(o, s') \in O \times S$, такая что $(i, s, s', o) \in T_S$. В противном случае автомат S называется частичным.

Автомат S называется детерминированным, если для любой пары $(i, s) \in I \times S$ существует не более одной пары $(o, s') \in O \times S$, такой что $(i, s, s', o) \in T_S$, в противном случае автомат называется недетерминированным.

Расширенный автомат [1] представляет собой пятерку $M = (S, I, O, V, T)$, где S – непустое конечное множество состояний автомата, I – непустое конечное множество входных символов, называемое входным алфавитом, O – непустое конечное множество выходных символов, называемое выходным алфавитом, V – конечное, возможно пустое, множество контекстных переменных, T – отношение переходов. Каждый переход t из T – это семерка (s, i, P, op, o, up, s'), где $s, s' \in S$ являются начальным и конечным состояниями перехода; $i \in I$ есть входной символ и $D_{inp\text{-}i}$ обозначает множество векторов, компонентами которых являются значения параметров, соответствующих входному символу i (далее входные параметры); $o \in O$ – выходной символ и $D_{out\text{-}o}$ обозначает множество векторов, компонентами которых являются значения параметров, соответствующих выходному символу o (далее выходные параметры); P, op и up – 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;

op: $D_{inp\text{-}i} \times D_V \rightarrow D_{out\text{-}o}$ – функция вычисления значений выходных параметров;

up: $D_{inp\text{-}i} \times D_V \rightarrow D_V$ – функция вычисления значений контекстных переменных.

Параметризованным входным (выходным) символом называется пара «входной символ i, вектор a из $D_{inp\text{-}i}$», т.е. пара (i, a) (пара «выходной символ o, вектор b из $D_{out\text{-}o}$», т.е. пара (o, b)).

Конфигурацией расширенного автомата M называется пара «состояние s, контекстный вектор v», т.е. (s, v).

Расширенный автомат называется полностью определенным, если в каждом состоянии s под действием каждого входного символа существует хотя бы один переход. Расширенный автомат M называется непротиворечивым, если из каждого состояния s для любого параметризованного входного символа и каждого значения контекстного вектора v существует не более одного перехода, предикат которого принимает значение «истина». Расширенный автомат M называется детерминированным, если в каждом состоянии s существует не более одного выполнимого перехода по любому параметризованному входному символу.

Рассмотрим композицию расширенных автоматов A и B на рис. 1, в которой:

– автомат A имеет входной алфавит $I_1 \cup V$, выходной алфавит $O_1 \cup U$ и отношение переходов T_A;

– автомат B имеет входной алфавит $I_2 \cup U$, выходной алфавит $O_2 \cup V$ и отношение переходов T_B;

– алфавиты I_1, I_2, O_1, O_2, V и U попарно не пересекаются.

Параллельная композиция расширенных автоматов A и B имеет:

– входной алфавит $I = I_1 \cup I_2$;

– выходной алфавит $O = O_1 \cup O_2$;

– под действием входного символа $i \in I_1 \cup I_2$ композиция вырабатывает внешний выходной символ $o \in O_1 \cup O_2$ или внутренний выходной символ, который является входным символом для другой компоненты;

– следующий входной символ может быть подан на композицию только после того, как компоненты закончили внутренний диалог.

2. Параллельная композиция расширенных автоматов

В настоящей работе для построения параллельной композиции расширенных автоматов хотелось бы использовать подходы к композиции классических конечных автоматов. Существует два

распространенных подхода: на основе перехода к полуавтоматам и путем построения дерева достижимости.

В работе [8] для построения композиции двух конечных автоматов предлагается для каждого автомата построить соответствующий полуавтомат, расширить алфавиты каждой компоненты до алфавитов другой компоненты и пересечь полученные полуавтоматы. Таким образом, получаем так называемый глобальный полуавтомат, который описывает совместное поведение взаимодействующих компонент. Далее для построения параллельной композиции необходимо ограничить глобальный полуавтомат на внешние алфавиты. Для того чтобы вернуться к модели конечного автомата, полученный полуавтомат должен удовлетворять следующему свойству: язык L этого полуавтомата должен содержаться в $(IO)^*$, где $I = I_1 \cup I_2$, $O = O_1 \cup O_2$. В случае, когда данное свойство не выполняется, необходимо сначала пересечь полученный полуавтомат с полуавтоматом, соответствующим языку $(IO)^*$, а затем вернуться к модели конечного автомата.

В работе [9] для получения параллельной композиции конечных автоматов предлагается строить дерево достижимости, которое описывает совместное поведение автоматов-компонент при подаче на вход композиции внешних входных символов из множества $I_1 \cup I_2$. Автомат композиции строится по дереву достижимости путем удаления всех внутренних переходов. Таким образом, каждый переход полученного автомата композиции помечен входным символом $i \in I_1 \cup I_2$ и внешним выходным символом $o \in O_1 \cup O_2$.

Первый подход (на основе перехода к полуавтоматам) затруднительно применить напрямую к расширенным автоматам, поскольку неясно, как выполнять расщепление перехода расширенного автомата на два перехода соответствующего полуавтомата, из-за наличия предикатов и функций обновления значений выходных параметров и контекстных переменных. Поэтому мы адаптируем второй подход (на основе дерева достижимости) для композиции расширенных автоматов и формулируем условия применимости данного подхода.

Мы рассматриваем частный случай параллельной композиции расширенных автоматов, а именно построение расширенного автомата для композиции в алфавитах I_1, O_1, U, V (т.е. компонента B является встроенной).

Пусть A и B – некоторые расширенные автоматы. Если параллельную композицию данных автоматов можно построить, не выполняя перехода к эквивалентным конечным автоматам, то необходимо построить дерево достижимости, которое описывает совместное поведение расширенных автоматов-компонент при подаче на вход композиции внешнего входного символа из алфавита I_1. Расширенный автомат композиции строится по дереву достижимости путем удаления всех внутренних переходов. Таким образом, каждый переход полученного расширенного автомата композиции помечен входным символом $i \in I_1$, значениями параметров, соответствующих входному символу i, предикатом, значениями контекстных переменных, внешним выходным символом $o \in O_1$, значениями параметров, соответствующих выходному символу o.

Пример 1. Рассмотрим пример использования модели расширенного автомата и параллельной композиции таких автоматов для описания взаимодействия клиентской и серверной частей протокола POP3. POP3 (Post Office Protocol Version 3) [10] – протокол, с помощью которого клиенты электронной почты могут получать почту с удаленных почтовых серверов по TCP соединению. Стандартный порт POP3 сервера – 110. Изначально сервер прослушивает порт 110, ожидая TCP соединения. Когда клиент желает воспользоваться сервисом POP3, он должен установить соединение с сервером. После установки соединения сервер посылает клиенту приветствие. Далее должна быть выполнена авторизация пользователя. При успешной авторизации клиент и сервер начинают обмениваться командами и ответами до тех пор, пока соединение не будет закрыто или прервано. После команды клиента QUIT и ответа от сервера +OK сервер освобождает все ресурсы, задействованные в состоянии передачи данных. Ответ сервера всегда начинается с одного из двух идентификаторов: +OK, что означает успешное выполнение команды, поступившей от клиента, или –ERR, что означает, что команда не была выполнена.

Для того чтобы при построении параллельной композиции расширенных автоматов избежать возникновения цепочек внутренних переходов с разными предикатами и функциями вычисления значений переменных, предикаты переходов расширенного автомата, описывающего поведение сервера, были перенесены в расширенный автомат композиции. После авторизации сервер должен передать клиенту число писем n в почтовом ящике, где n – контекстная переменная для всех расширенных автоматов в примере. Массив *message* длины n является также контекстной переменной расширенных автоматов, описывающих поведение клиентского приложения и взаимодействие клиента и сервера. Изначально все элементы массива *message* равны 1. При удалении i-го письма элемент *message*(i) = 0. При восстановлении писем $\forall\ i \le n$ *message*(i) = 1.

На рис. 2 представлен фрагмент расширенного автомата, описывающего поведение клиентской части в состоянии транзакции (передачи данных). Переходы данного автомата перечислены ниже:

1) «Запрос письма (i)» ($i \le n$ и *message*(i) $\neq$ 0) / RETR (i);

2) +OK (*weight*) octets The (i) message / «Письмо (i), (*weight*) октетов»;

3) «Запрос письма (i)» ($i > n$ или *message*(i) = 0) / «Такого письма нет»;

4) «Удалить письмо (i)» ($i \le n$) / DELE (i);

5) +OK message (i) deleted / *message*(i) = 0, «Письмо (i) удалено»;

6) «Удалить письмо (i)» ($i > n$) / «Такого письма нет»;

7) «Восстановить письма» / RSET;

8) +OK maildrop has n messages ((*weight*) octets) / Для всех $i \le n$ *message*(i) = 1, «Письма восстановлены, в почтовом ящике n писем, (*weight*) октетов».

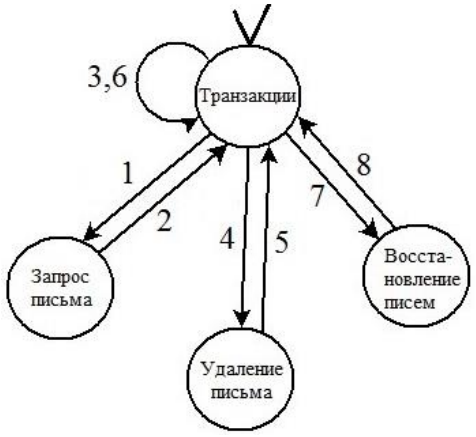


Рис. 2. Расширенный автомат, описывающий поведение клиентского приложения в состоянии обмена командами и ответами с сервером
Fig. 2. An extended automaton describing the behavior of the client application in the state of exchanging commands and responses with the server

Для простоты мы рассматриваем расширенный автомат, описывающий поведение серверной части протокола POP3 только в состоянии транзакции. В этом состоянии рассматриваются только следующие входные действия: получить i-е письмо – команда RETR (i), удалить i-е письмо – команда DELE (i), восстановить все письма, помеченные как удаленные, – команда RSET. Переходы автомата имеют вид:

1) RETR (i) / +OK (*weight*) octets The (i) message;

2) DELE (i) / +OK message (i) deleted;

3) RSET / +OK maildrop has n messages ((*weight*) octets).

Автомат композиции клиентской и серверной частей имеет одно состояние и следующие переходы:

1) «Запрос письма (i)» ($i \le n$ и *message*(i) $\neq$ 0) / «Письмо (i), (*weight*) октетов»;

2) «Запрос письма (i)» ($i > n$ или *message*(i) = 0) / «Такого письма нет»;

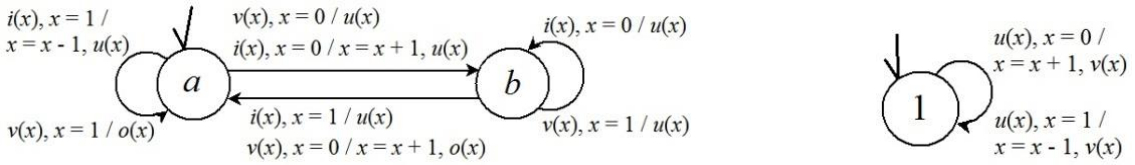
3) «Удалить письмо (*i*)» (*i* ≤ *n*) / *message*(*i*) = 0, «Письмо (*i*) удалено»;

4) «Удалить письмо (*i*)» (*i* > *n*) / «Такого письма нет»;

5) «Восстановить письма» / Для всех *i* ≤ *n* *message*(*i*) = 1, «Письма восстановлены, в почтовом ящике *n* писем, (*weight*) октетов».

Данный пример иллюстрирует, что в частном случае, когда возможен перенос предикатов компоненты в расширенный автомат композиции, можно формально описать параллельную композицию расширенных автоматов без использования моделирования.

Пример 2. Пусть *A* и *B* – расширенные автоматы (рис. 3, *a, b*). Предикаты переходов компоненты *A* зависят от выходных параметров компоненты *B*; предикаты переходов компоненты *B* зависят от выходных параметров компоненты *A*.



a *b*

Рис. 3. Расширенный автомат *A* (*a*); Расширенный автомат *B* (*b*)
Fig. 3. Advanced automaton *A* (*a*); Advanced machine *B* (*b*)

В данном случае, при построении параллельной композиции расширенных автоматов *A* и *B*, возникают цепочки внутренних переходов с разными предикатами и функциями вычисления значений переменной *x*. В настоящий момент нам не известно, каким образом описать переход между устойчивыми состояниями в композиции, сформировав по цепочке внутренних переходов предикат и функцию вычисления значений переменной *x*, т.е. описать параллельную композицию расширенных автоматов формально.

Построим дерево достижимости. Начальное состояние системы *a*1. При подаче входного символа *i*(*x*) со значением параметра *x* = 0 система переходит из состояния *a*1 в состояние *b*1 с выдачей выходного символа *u*(1). При подаче входного символа *i*(*x*) со значением параметра *x* = 1 система остается в состоянии *a*1 с выдачей выходного символа *u*(0) и т.д. Строим дерево достижимости до тех пор, пока в каждой ветви не получим внешний выходной символ. Дерево достижимости для расширенных автоматов *A* и *B* представлено на рис/ 4.

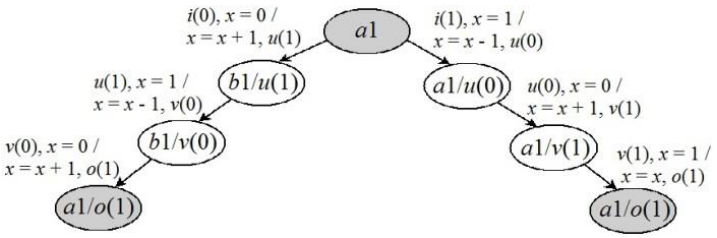


Рис. 4. Дерево достижимости
Fig. 4. Reach tree

Полученное дерево достижимости показывает, что при построении параллельной композиции расширенных автоматов *A* и *B* возникают цепочки внутренних переходов с разными предикатами и разными функциями вычисления значения переменной *x*. Если мы хотим перейти к расширенному автомату композиции во внешних алфавитах, т.е. «свернуть» внутренние цепочки переходов, то возникает вопрос, как сформировать предикат перехода и функции вычисления значений параметров.

Таким образом, можно сформулировать следующие утверждения.

Утверждение 1.1. Пусть *A* и *B* – некоторые расширенные автоматы. Параллельную композицию данных автоматов можно построить, не выполняя перехода к эквивалентным конечным автоматам,

если на любой внутренний входной символ компонента *A* вырабатывает внешний выходной символ и предикаты компоненты *A* не зависят от выходных параметров компоненты *B*.

Доказательство. Поскольку на всякий внутренний входной символ компонента *A* вырабатывает внешний выходной символ, то внутренний диалог состоит только из обмена одной входо-выходной парой. Так как предикат перехода автомата *A* не зависит от выходных параметров компоненты *B*, то не возникает конфликта при вычислении значения предиката перехода при таком внутреннем диалоге. Таким образом, для данного случая можно построить параллельную композицию расширенных автоматов *A* и *B*, не выполняя перехода к эквивалентным конечным автоматам.

Утверждение 1.2. Пусть *A* и *B* – некоторые расширенные автоматы. Параллельную композицию данных автоматов можно построить, не выполняя перехода к эквивалентным конечным автоматам, если во внутреннем диалоге компонент содержится не более одного перехода, помеченного предикатом.

Доказательство. Если во внутреннем диалоге автоматов *A* и *B* нет переходов, помеченных предикатами, то при построении дерева достижимости не возникает цепочек внутренних переходов с разными функциями вычисления значений параметров и контекстных переменных. Таким образом, можно построить параллельную композицию расширенных автоматов *A* и *B*, не выполняя перехода к эквивалентным конечным автоматам.

Если во внутреннем диалоге данных автоматов содержится только один переход, помеченный предикатом, то при построении дерева достижимости для автоматов *A* и *B* возникает один внутренний переход с разными функциями вычисления значений параметров и контекстных переменных. В этом случае возможен перенос предиката в переход расширенного автомата композиции, так как мы можем отследить по полученному дереву достижимости, какое именно внешнее входное воздействие с каким параметром привело к возникновению такого внутреннего перехода и выдаче соответствующего внешнего выходного символа. То есть для данного случая можно построить параллельную композицию расширенных автоматов *A* и *B*, не выполняя перехода к эквивалентным конечным автоматам.

Заключение

Таким образом, в данной работе мы рассмотрели возможность переноса методов теории конечных автоматов для построения параллельной композиции на случай взаимодействующих расширенных автоматов. Сформулированы условия, при которых данный перенос возможен, т.е. можно построить композицию расширенных автоматов без их моделирования. В дальнейшем будет рассмотрена композиция, в которой обе компоненты взаимодействуют с внешней средой.

ЛИТЕРАТУРА

1. Petrenko A., Boroday S., Groz R. Confirming Configurations in EFSM Testing // IEEE Trans. Software Eng. 2004. V. 30 (1). P. 29–42.
2. Гилл А. Введение в теорию конечных автоматов. М. : Наука, 1966. 272 с.
3. Евтушенко Н.В., Рекун М.В., Тихомирова С.В. Недетерминированные автоматы: анализ и синтез : учеб. пособие. Томск : Том. гос. ун-т, 2009. Ч. 2: Решение автоматных уравнений. 111 с.
4. Коломеец А.В. Алгоритмы синтеза проверяющих тестов для управляющих систем на основе расширенных автоматов : дис. … канд. техн. наук. Томск, 2010. 129 с.
5. Карибский В.В., Пархоменко П.П., Согомонян Е.С., Халчев В.Ф. Основы технической диагностики. М. : Энергия, 1976. 464 с.
6. Villa T., Yevtushenko N., Mishenko A., Brayton R.K., Petrenko A., Sangiovanni-Vincentelli A. The unknown component problem: theory and applications. Berlin : Springer, 2012. 311 p.
7. Prokopenko S. Locating a faulty component of an EFSM composition // The Proc. of ISP RAS. 2014. V. 26, is. 6. P. 47–55.
8. Castagnetti G., Piccolo M., Villa T., Yevtushenko N., Mishchenko A., Brayton R.K. Solving Parallel Equations with BALM-II. Technical Report No. UCB/EECS2012-181 / Electrical Engineering and Computer Sciences University of California at Berkeley. Berkeley, CA, 2012. URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-181.pdf (accessed: 27.05.2018).

9. El-Fakih Kh., Trenkaev V., Spitsyna N., Yevtushenko N. FSM Based Interoperability Testing Methods // Lecture notes in Computer Science. 2004. V. 2978. P. 60–75.
10. RFC 1939 – Протокол POP3. URL: http://www.rfc2.ru/1939.rfc (дата обращения: 27.05.2018).

Поступила в редакцию 20 июня 2018 г.

Shirokova E.V., Prokopenko S.A., Shabaldina N.V. (2019) ON DERIVING THE PARALLEL COMPOSITION OF EXTENDED FINITE STATE MACHINES. *Vestnik Tomskogo gosudarstvennogo universiteta. Upravlenie vychislitelnaja tehnika i informatika* [Tomsk State University Journal of Control and Computer Science]. 48. pp. 83–91

DOI: 10.17223/19988605/48/10

In this paper we consider a problem of communicating components of telecommunication systems. The components interact not only with each other but also with an environment. The interaction is in a dialog mode and the so-called “slow environment” condition takes place. As usual, each component of the system is a software. An Extended Finite State Machine (EFSM) allows to describe the software. In order to verify and test the communicating components we need a formal description of the system. It is well known how to get such description for the case when each component is described as a classical Finite State Machine (FSM). However, it is not always possible to derive an equivalent FSM for the appropriate EFSM. The first reason is that domains of input / output parameters and context variables can be infinite. The second one is that even domains are finite we can face to so-called state explosion problem. So in this paper we discuss approaches how to describe the composition of EFSMs in case when one of components is embedded. There are two approaches to derive the composition of FSMs. One of them is based on transformation of each FSM into corresponding automaton. These automata are composed and then the composition is transformed into an FSM. Unfortunately, we can’t use this approach because we don’t know how to transform the EFSM into automaton. The second approach is based on derivation of reachability tree. In this case, the composition at a stable state accepts an external input, moves through intermediate states (an internal dialog between components), reaches new stable state and produces an external output. The composition has only stable states and transitions between these states are marked by external inputs and outputs. And we adopted this approach for the case of EFSM’s composition. We have formulated two conditions when this approach is valid for EFSMs. If for each internal input the Context produces an external output and predicates of Context don’t depend on output parameters of Embedded component, then it’s possible to construct a composition of EFSMs without derivation equivalent FSMs. The second condition is that the approach is valid if there is not more than one transition with predicate in internal dialog between components.

Keywords: extended finite state machine; finite state machine; parallel composition of extended finite state machines.

SHIROKOVA Ekaterina Vladimirovna (Post-graduate Student, National Research Tomsk State University, Tomsk, Russian Federation).
E-mail: darusenkova@gmail.com

PROKOPENKO Svetlana Anatolievna (Candidate of Technical Sciences, Associate Professor, National Research Tomsk State University, Tomsk, Russian Federation).
E-mail: s.prokopenko@sibmail.com

SHABALDINA Natalia Vladimirovna (Candidate of Technical Sciences, Associate Professor, National Research Tomsk State University, Tomsk, Russian Federation).
E-mail: nataliamailbox@mail.ru

REFERENCES

1. Petrenko, A., Boroday, S. & Groz, R. (2004) Confirming Configurations in EFSM Testing. *IEEE Trans. Software Eng*. 30(1). pp. 29–42.
2. Gill, A. (1966) *Vvedenie v teoriyu konechnykh avtomatov* [Introduction to the theory of finite state machines]. Translated from English. Moscow: Nauka.
3. Evtushenko, N.V., Rekun, M.V. & Tikhomirova, S.V. (2009) *Nedeterminirovannye avtomaty: analiz i sintez* [Nondeterministic finite state machines: analysis and synthesis]. Part 2. Tomsk: Tomsk State University.
4. Kolomeets, A.V. (2010) *Algoritmy sinteza proveryayushchikh testov dlya upravlyayushchikh sistem na osnove rasshirennykh avtomatov* [Diagnostic test derivation methods for telecommunication systems based on an EFSM model]. Engineering Cand. Diss. Tomsk.
5. Karibskiy, V., Parkhomenko, P., Sogomonyan, E. & Khalchev, V. (1976) *Osnovy tekhnicheskoy diagnostiki* [Basics of technical diagnostics]. Moscow: Energiya.
6. Villa, T., Yevtushenko, N., Mishenko, A., Brayton, R.K., Petrenko, A. & Sangiovanni-Vincentelli, A. (2012) *The Unknown Component Problem: Theory and Applications*. Berlin: Springer.

7. Prokopenko, S. (2014) Locating a faulty component of an EFSM composition. *The Proc. of ISP RAS*. 26(6). pp. 47–55. DOI: 10.1007/s 10009-014-0357-7
8. Castagnetti, G., Piccolo, M., Villa, T., Yevtushenko, N., Mishchenko, A. & Brayton R.K. (2018) *Solving Parallel Equations with BALM-II*. Technical Report No. UCB/EECS2012-181. Electrical Engineering and Computer Sciences University of California at Berkeley. [Online] Available from: http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-181.pdf. (Accessed: 27th May 2018).
9. El-Fakih, Kh., Trenkaev, V., Spitsyna, N. & Yevtushenko, N. (2004) FSM Based Interoperability Testing Methods. *Lecture Notes in Computer Science*. 2978. pp. 60–75. DOI: 10.1007/978-3-540-24704-3_5
10. RFC. (1939) *POP3 Protocol*. [Online] Available from: http://www.rfc2.ru/1939.rfc. (Accessed: 27th May 2018).