Модел. и анализ информ. систем. Т. **18**, № **4** (2011) 144–156

УДК 519.681.3

Верификация телекоммуникационных систем, специфицированных взаимодействующими конечными автоматами, с помощью раскрашенных сетей Петри[1]

Белоглазов Д.М., Машуков М.Ю., Непомнящий В.А.

Институт систем информатики им. А.П. Ершова СО РАН

e-mail: dmitry.beloglazov@gmail.com,mashukovm@mail.ru, vnep@iis.nsk.su

получена 18 ноября 2011 года

Ключевые слова: телекоммуникационные системы, расширенные конечные автоматы, раскрашенные сети Петри, кольцевые протоколы, взаимодействие функциональностей, телефонные сети, верификация, метод проверки моделей

Рассматриваются униформные системы взаимодействующих расширенных конечных автоматов, которые удобны для задания исходной спецификации телекоммуникационных систем, таких как кольцевые протоколы и телефонные сети. Цель работы – представить программный комплекс Automata Systems Verifier (ASV), предназначенный для анализа и верификации автоматных спецификаций. Он базируется на алгоритме трансляции систем автоматов в раскрашенные сети Петри (РСП), представленном и обоснованном в [4]. Для анализа РСП комплекс ASV использует систему CPN Tools [10], а для их верификации методом проверки моделей относительно свойств, заданных формулами мю-исчисления, он использует систему Petri Net Verifier [12]. Описано применение этого комплекса к верификации кольцевого RE-протокола и к исследованию взаимодействия функциональностей в телефонных сетях.

1. Введение

Развитие методов и средств для анализа и верификации телекоммуникационных систем является известной проблемой современного программирования. Широко используемый язык выполнимых спецификаций телекоммуникационных систем SDL, принятый в качестве стандарта, базируется на взаимодействующих конечных автоматах. Однако значительная выразительность языка SDL затрудняет анализ и верификацию SDL-спецификаций телекоммуникационных систем.

Для преодоления этих трудностей обычно применяются такие модели, как взаимодействующие конечные автоматы и различные виды сетей Петри. Среди этих

[1]Работа частично поддержана грантом РФФИ 11-07-90412-Ukr_f_a

моделей выделяются раскрашенные сети Петри (РСП) [9] ввиду их значительной выразительности, а также наличия мощных средств симуляции и анализа, реализованных в программном комплексе CPN Tools [10]. Метод трансляции из языка SDL в РСП и программная система, реализующая этот метод, описаны в [16]. Однако формальное обоснование корректности таких трансляторов остается открытой трудной проблемой.

Отметим, что взаимодействующие конечные автоматы удобны для задания исходной спецификации ряда программных систем. В частности, иерархические системы взаимодействующих конечных автоматов используются для спецификации реактивных систем [1,2,3], а униформные системы взаимодействующих конечных автоматов используются для спецификации кольцевых протоколов и телефонных сетей [4,6,8,17]. Трансляция иерархических систем взаимодействующих расширенных конечных автоматов в РСП и применение системы CPN Tools для их анализа и верификации описаны в [1]. Алгоритм трансляции униформных систем взаимодействующих расширенных конечных автоматов в РСП предложен и формально обоснован в [4].

Цель настоящей работы – представить программный комплекс Automata Systems Verifier (ASV), предназначенный для анализа и верификации униформных систем взаимодействующих расширенных конечных автоматов. Комплекс ASV включает транслятор из таких систем в РСП и использует системы CPN Tools и Petri Net Verifier [12] для анализа и верификации РСП методом проверки моделей относительно свойств, выраженных в мю-исчислении. В настоящей работе описывается также применение комплекса ASV для верификации кольцевого RE-протокола [7] и взаимодействия функциональностей в телефонных сетях [11].

Данная работа состоит из 6 разделов. В разделе 2 определяются униформные системы взаимодействующих расширенных конечных автоматов и описывается алгоритм трансляции этих систем в РСП. В разделе 3 дано описание программного комплекса ASV. Применение этого комплекса к верификации кольцевого RE-протокола представлено в разделе 4, а к верификации взаимодействия функциональностей в телефонной сети – в разделе 5. Обсуждению результатов и перспектив нашего подхода посвящен раздел 6.

2. Взаимодействующие расширенные конечные автоматы

В данном разделе даётся определение системы взаимодействующих расширенных конечных автоматов (РКА). За основу были взяты определения из [13] и внесены следующие существенные изменения: вместо чтения и записи символов автомат получает входной сигнал определённого формата из окружения и посылает в окружение выходные сигналы. Также, кроме локальных переменных, автомат может использовать глобальные переменные системы. Допустимые типы для локальных и глобальных переменных: целочисленный, булевый, перечислимый, запись и список.

Неформально система взаимодействующих РКА – это набор автоматов, которые изменяют значения своих локальных и глобальных переменных и взаимодействуют

друг с другом, отправляя сигналы. Сигналы могут иметь разные типы, включающие идентификаторы отправителя и получателя, а также некоторую дополнительную информацию. Такая система работает пошагово. На каждом шагу может сработать некоторый переход автомата, если этот переход разрешён, т.е. все условия его срабатывания выполнены (локальные и глобальные переменные имеют определённые значения, и есть входной сигнал определённого типа). Когда переход срабатывает, автомат меняет своё состояние, удаляет соответствующий входной сигнал из окружения, выполняет некоторые вычисления на локальных и глобальных переменных и посылает выходные сигналы в окружение.

Определение 1. Расширенный конечный автомат есть кортеж $\alpha = <S,\ V,\ G,\ I,\ O,\ T>$, где

- S – набор состояний;
- V – набор локальных переменных автомата, который включает специальную переменную id –идентификатор автомата, используемый для взаимодействия с другими автоматами;
- G – набор глобальных переменных;
- I – набор входных сигналов;
- O – набор выходных сигналов.

Входные и выходные сигналы имеют следующий формат: *[Src, Dest, Type, Param]*, где *Src* – это идентификатор отправителя, *Dest* – идентификатор получателя, *Type* – перечислимый тип сигнала и *Param* – параметр сигнала (если параметра нет, то $Param = null$).

- T – набор переходов. Каждый переход из – это кортеж

$t = \{Ss,\ Se, Os(G,V,I_0),\ P(G,V,I_0),\ E(G,V,\ I_0)\}$, где

- Ss – исходное состояние;
- Se – результирующее состояние;
- I_0 – входной сигнал из окружения Env (он может отсутствовать и тогда срабатывание перехода не зависит от входных сигналов);
- $Os(G,V,I_0)$ – множество выходных сигналов, которое является подмножеством O;
- $P(G, V, I_0)$ – предикат, который задает условие срабатывания перехода;
- $E(G, V, I_0)$ – множество операторов присваивания, которые задают вычисления автомата, используя локальные и глобальные переменные.

Определение 2. Система взаимодействующих РКА – это кортеж $\Sigma = <A,\ G,\ Env,\ Init>$, где

- A – множество расширенных конечных автоматов;
- G – множество глобальных переменных всех автоматов в системе;
- *Env* – окружение, которое является множеством сигналов в системе, ожидающих приёма;
- *Init* – инициализирующая функция, определённая на G и A, которая задаёт начальные состояния для всех РКА из множества A, начальные значения локальных и глобальных переменных, а также начальное множество сигналов *Env*. Инициализирующая функция *Init* может быть не определена и тогда переменным не будут присвоены исходные значения, а множество *Env* будет пустым в начальном состоянии.

Все множества в этих определениях должны быть конечными, а множество глобальных переменных не пересекается с множествами локальных переменных автоматов. Выполнение системы определяется относительно глобального времени, заданного положительными целыми числами.

Определение 3. Конфигурацией автомата α системы Σ на шаге k называется пара *<S(k), V(k)>*, где $S(k)$ – состояние автомата, а $V(k)$ – значения его локальных переменных на шаге k. Конфигурация системы на шаге k – это множество конфигураций всех автоматов системы, множество значений глобальных переменных и значение переменной *Env* на данном шаге.

Определение 4. Пусть автомат α системы Σ находится в состоянии Ss. Переход t автомата α разрешён, если $P(G, V, I_0)$ истинно для некоторого сигнала I_0 из *Env*. Срабатывание разрешённого перехода означает:

- изменение состояния *Ss* на *Se*,
- удаление входного сигнала I_0 из окружения *Env*, если I_0 задан,
- изменение значений переменных согласно выражению $E(G, V, I_0)$,
- помещение выходных сигналов *Os(G,V,*I_0*)* в окружение *Env*.

Мы говорим, что *автомат срабатывает*, если срабатывает по крайней мере один из его разрешённых переходов (недетерминированный выбор). Мы говорим, что *система срабатывает на шаге* k, если какой-либо автомат в системе срабатывает на этом шаге (недетерминированный выбор).

Ниже приведена схема алгоритма трансляции системы Σ =*<A, G, Env, Init>* в РСП N [1].

1. Определить вспомогательный тип данных (цвет) *Signal* для работы с сигналами. Построить место *Env* цвета *Signal*.
2. Для всех переменных системы (глобальных и локальных) определить соответствующие типы данных.
3. Для каждой глобальной переменной системы создать соответствующее место.

4. Для каждого автомата α из A совершить следующие действия:

   (a) Для каждой локальной переменной v из V построить соответствующее место α_v.

   (b) Для каждого состояния s из S построить соответствующее целочисленное место α_s.

   (c) Для каждого перехода t из T построить такой переход α_t, что условия на этом переходе определяются предикатами из $P(G, V, I_0)$, а вычисления производятся согласно $E(G, V, I_0)$. Соединить каждый переход с местами, соответствующими переменным, используемым в этом переходе.

Полный алгоритм приведён в [4] вместе с доказательством его корректности.

Рассмотрим простой пример – систему, состоящую из двух взаимодействующих автоматов – Отправителя и Получателя. Начальное состояние Отправителя – *IDLE*. Алгоритм его следующий: в состоянии *IDLE* отправить Получателю сигнал типа *msg* и перейти в состояние ожидания *AWAITING_ACK,* а затем в этом состоянии при получении входного сигнала о подтверждении типа *ack* вернуться в состояние *IDLE*. Алгоритм Получателя следующий: если получено сообщение, то вернуть Отправителю этого сообщения сигнал типа *ack*. Автомат Отправителя имеет вид $A = <S,\ V,\ I,\ T>$, где $S = \{IDLE,\ AWAITING\_ACK\}$, $V = \{Receiver\_id\}$, $T = \{t_1, t_2\}$, $t_1 = \{IDLE,\ AWAITING\_ACK,\ [id,\ Receiver\_id,\ msg,\ null],\ true,\ \emptyset\}$, $t_2 = \{AWAITING\_ACK,\ IDLE,\ \emptyset, I_0.type = ack,\ \emptyset\}$, а для автомата Получателя $S = \{IDLE\}, V = \{\}$, $T = \{t_1\}$, где $t_1 = \{IDLE,\ IDLE,\ [id,\ I_0.src,\ ack,\ null],\ I_0.type = msg,\ \emptyset\}$. На Рис. 1 приведено графическое представление РСП, полученной в результате трансляции системы “Отправитель – Получатель”.

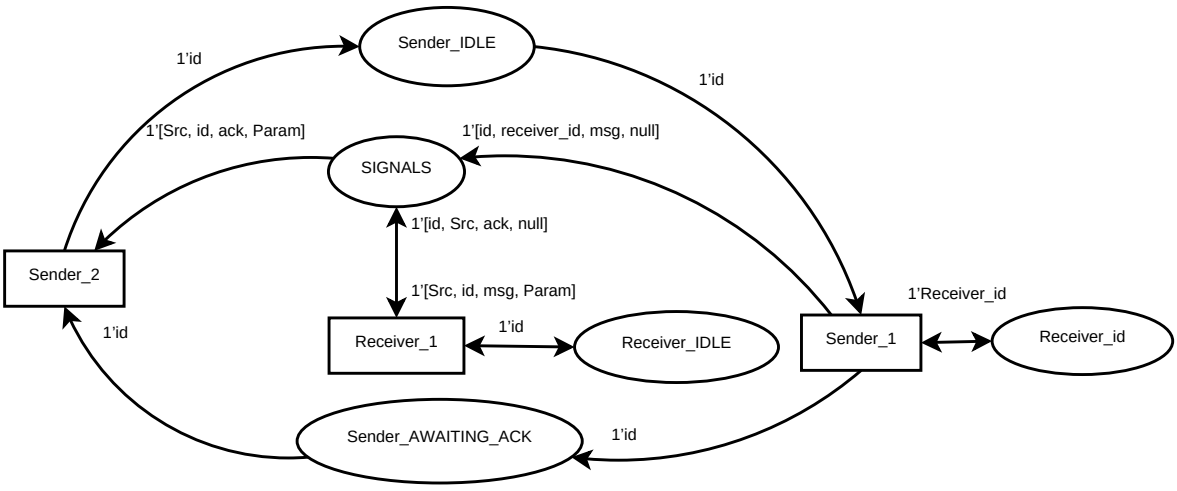

Рис. 1. Сетевая модель системы “Отправитель – Получатель”

3. Программный комплекс ASV

Программный комплекс ASV, предназначенный для моделирования и верификации телекоммуникационных систем, заданных в виде систем взаимодействующих РКА, состоит из следующих модулей:

- **Транслятор** из систем РКА в РСП. Для системы РКА этот транслятор строит эквивалентную сетевую модель.
- **Симулятор РСП**. Для симуляции РСП используется система CPN Tools [10].
- **Модуль построения графа достижимости РСП**. CPN Tools строит граф достижимости во внутреннем формате. Наш модуль преобразует этот формат во входной формат верификатора PNV [12] и строит модель – полное описание вершин графа, соответствующих состояниям РКА.
- **Построитель предикатов**. По графу достижимости с полным описанием вершин и файлу с описанием свойств РСП этот модуль строит описание предикатов – множеств вершин графа достижимости, на которых истинны данные свойства.
- **Система проверки моделей PNV** [12]. Этот модуль получает на вход 3 файла: модель, мю-формулу и описание предикатов. На выходе он выдает результат верификации.

Структура комплекса ASV показана на Рис. 2.

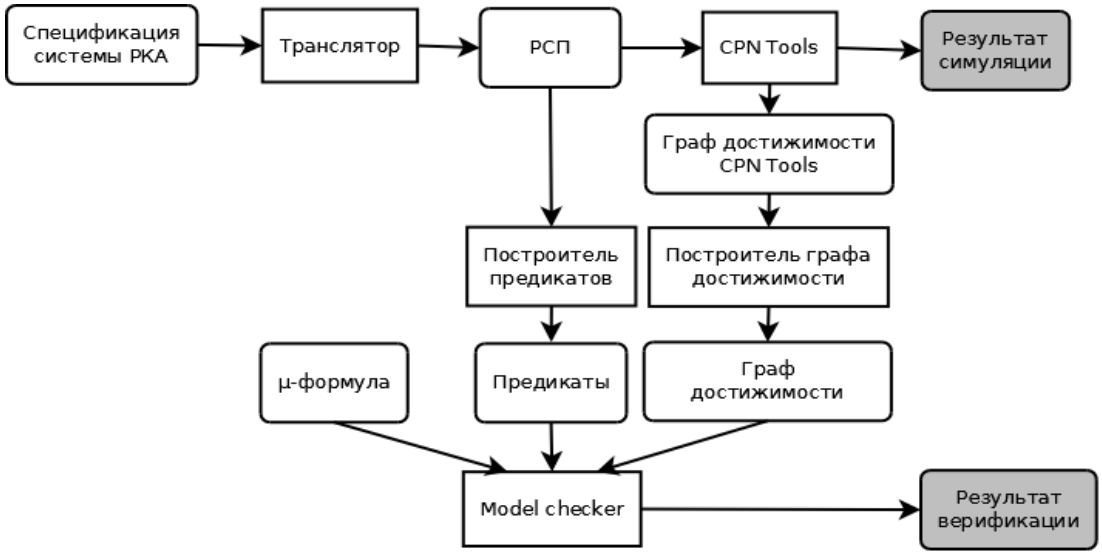

Рис. 2. Программный комплекс ASV

Опишем транслятор из систем РКА в РСП. Спецификации систем РКА подаются на вход транслятора в текстовом виде. Язык для описания таких систем был разработан на основе языка SDL.

Перевод систем РКА в их сетевые модели осуществляется в несколько проходов. На первом проходе по текстовому описанию автоматной системы модулем синтаксического анализатора строится синтаксическое дерево разбора. Синтаксический анализатор языка построен с помощью программного средства Bison. Во время второго прохода по этому дереву разбора начинается генерация сетевой модели во внутреннем представлении, то есть создаётся пустая страница сети для каждого автомата, по одному замещаемому переходу на главной странице сети, место для хранения сигналов *Env* и места, соответствующие глобальным переменным автоматной системы. Кроме того, определяются вспомогательные множества цветов, функции и переменные. На следующем проходе сетевая модель во внутреннем представлении расширяется так, что на страницах, соответствующих автоматам, добавляются переходы и места, моделирующие локальные переменные. Каждая фишка в месте, соответствующем локальной переменной автомата, содержит два поля – идентификатор экземпляра автомата и значение локальной переменной.

Для каждого перехода автомата в исходной спецификации обрабатываются сигналы, отправляемые автоматом при выполнении данного перехода, и соответствующие вычисления из множества E. Назовём *действиями* операции посылки сигналов и присваивания. Для каждого перехода автомата действия обрабатываются последовательно в том порядке, в котором они расположены в спецификации. В процессе их обработки в памяти хранится список переменных автомата, которые меняются в процессе выполнения перехода, и их новые значения в символьном виде. После обработки действий перехода автомата полученные значения переменных указываются на выходных дугах соответствующего перехода сетевой модели, соединяющих его с местами, которые соответствуют переменным.

Например, если автомат с идентификатором id имеет локальные переменные x и y, и первое действие – присваивание y значения $x+1$, то соответствующий переход в сетевой модели будет соединён с местами, соответствующими переменным x и y, а на дуге, идущей от этого перехода к месту, соответствующему переменной y, будет стоять выражение *(id, x + 1)*. Для всех последующих обрабатываемых действий значение переменной y будет заменяться на $x + 1$.

После создания сетевой модели производится генерация начальной разметки мест, соответствующих автоматам, причем действия раздела спецификации, описывающего инициализацию системы, транслируются последовательно в начальную разметку сети. В процессе трансляции отслеживаются текущие значения глобальных и локальных переменных.

На заключительном проходе производится вывод полученной сетевой модели во входном формате системы CPN Tools.

4. Верификация кольцевого RE-протокола

Кольцевой RE-протокол, впервые описанный в [6], используется в кольцевых сетях с тактированным доступом (slotted-ring network) – таких сетях, в которых передача данных тактируется в регулярные интервалы времени синхронно для всех станций. По кольцу от станции к станции передаётся кадр (фрейм) фиксированной длины,

разбитый на такты (slots). Для простоты изложения будем считать понятия фрейм и такт тождественными. Каждая станция передает полученный фрейм далее по кольцу, своему ближайшему соседу. Станция, желающая отправить данные, ждет фрейм, имеющий метку *пустой (empty)*, меняет ее на *занятый (full)* и загружает свои данные во фрейм. Станция, получившая фрейм, передает его неизменным, а в случае совпадения адреса назначения со своим адресом, копирует данные фрейма в локальный буфер.

RE-протокол использует два бита для метки, и структура фрейма имеет вид $<R, E, DEST, SRC, DATA, RESP>$, где R и E – биты метки, $DEST$ – поле, содержащее адрес станции-получателя, SRC – поле, содержащее адрес станции-отправителя, $DATA$ – поле для пересылаемых данных, $RESP$ – контрольная сумма. В RE-протоколе одна из станций выделяется для контроля работы протокола в случае искажения битов R и E, так как эти биты не защищаются контрольной суммой. Такая станция называется *монитор*.

Исходная спецификация RE-протокола задана в виде системы РКА. Используя комплекс ASV, мы транслировали эту спецификацию в эквивалентную РСП и затем провели симуляцию и верификацию для проверки основных свойств, таких как наличие тупиков и циклов. RE-протокол был рассмотрен в случае надёжной и ненадёжной передающей среды, количество станций варьировалось от 3 до 10 (включая монитор). Для протокола были проверены следующие свойства:

1. *Наличие тупиков.* Это свойство может быть обнаружено на этапе построения графа достижимости или при помощи проверки мю-формулы $\neg <\text{to}> true$, где *true* соответствует всем состоянием модели.

2. *Безопасность.* Это свойство означает, что все отправленные сообщения могут быть приняты.

3. *Расширенная безопасность.* Это свойство означает, что все отправленные сообщения будут приняты.

4. *Повторный приём сообщений.* Мы обнаружили, что в случае ненадёжной передающей среды сообщение, отправленное одной станцией другой, может прийти адресату более чем один раз. Проведенная верификация подтвердила, что в случае ненадёжной среды происходит повторный приём сообщений. Для решения этой проблемы было предложено исправление спецификации протокола. В моделях с надёжной средой данная проблема не возникает.

Мы выяснили, что RE-протокол не содержит тупиков и для него выполняется свойство безопасности. Свойство расширенной безопасности выполняется только в случае надёжной среды.

5. Исследование взаимодействия функциональностей в телефонных сетях

Для исследования взаимодействия функциональностей мы рассматриваем так называемую "Базовую Модель Звонков" (Basic Call State Model) [11]. Это модель базового телефонного сервиса (Basic Call Service, BCS), который позволяет абонентам общаться – набирать номер, отвечать на звонки и т.д.

Для спецификации BCS применяются взаимодействующие расширенные конечные автоматы таким образом, что отдельные автоматы специфицируют работу абонента и станции. Абоненты посылают запросы станции, которая обрабатывает запросы, и выдаёт абонентам результаты в виде сигналов. Функциональности моделируются автоматом Feature Manager, который становится посредником между абонентами и BCS. Этот автомат принимает и обрабатывает сигналы абонентов с учётом того, подключена ли определённая функциональность у вызывающего (или вызываемого) абонента. Feature Manager либо самостоятельно посылает сигналы абонентам, либо передаёт управление BCS. В результате мы построили модель с 3 различными типами автоматов: Subscriber (Абонент), BCS и Feature Manager. Автомат Subscriber имеет состояния, соответствующие состояниям абонентов: Idle, Dialing, Calling, Talking, и т.д. Автоматы BCS и Feature Manager имеют только одно состояние: они обрабатывают входные сигналы, изменяют значения переменных и отправляют выходные сигналы абонентам (экземплярам автомата Subscriber) и друг другу. Типы сигналов, которые абоненты могут посылать автомату Feature Manager, следующие: *offhook* (снять трубку), *onhook* (положить трубку) и *dial* (набрать номер). В ответ они получают сигналы следующих типов: *dialtone* (непрерывный гудок), *busytone* (короткие гудки, означающие "занято"), *incoming_ call* (входящий звонок) и т.д.

Мы моделировали следующие три функциональности:

1. *Прямое Соединение* (Direct Connect, DC). Если абонент x подключён к DC с направлением на номер абонента y, то как только x снимает трубку, он автоматически соединяется с абонентом y.

2. *Запрет Входящих* (Denied Termination, DT). Если абонент x подключён к DT, то все звонки на номер x запрещены. Абонент, набирающий номер x, автоматически получает сигнал "занято".

3. *Перевод Звонков, когда Номер Занят* (Call Forwarding when Busy, CFB). Если абонент y звонит абоненту x, а x занят и перенаправлен на номер абонента z, то y соединяется с абонентом z.

Используя комплекс ASV, мы построили РСП сначала для базовой модели звонков, а затем для всей системы автоматов, т.е. с автоматом Feature Manager. Используя эту модель, мы провели симуляцию и верификацию следующих свойств (с количеством абонентов от 2 до 7):

1. *Наличие тупиков.* То же, что и для RE-протокола.

2. *Зацикливание.* Наличие таких циклов, из которых невозможно вернуться в исходное состояние. Поскольку все проверяемые модели не имеют тупиков, зацикливание совпадает с нарушением условия "возможно вернуться в исходное состояние", выражаемого формулой $\mu X.(<to>(begin \vee X))$, где *begin* – предикат, истинный в начальном состоянии.

3. *Недетерминизм.* Наличие в сети конфликтующих переходов, соответствующих двум функциональностям.

4. *Нарушение условий.* В данном случае условие есть только у функциональности DT: никто никогда не может звонить абоненту, подключившему услугу DT. Нарушение этого условия будет означать, что на некотором такте для некоторого x из множества идентификаторов абонентов выполнено $DT[x] = true$, и для x есть входной сигнал типа *Incloming_ call*. Это свойство обнаруживается по наличию в графе достижимости таких состояний сети, которые имеют соответствующую разметку, т.е. фишку *[x, true]* в месте DT и фишку *[Src, x, Incoming_ call, null]* в месте *Env*. Для проверки этого свойства нет необходимости проводить программную верификацию, так как оно выявляется на этапе построения предикатов.

Результаты верификации приведены в Таблице 1.

Таблица 1. Результаты исследования взаимодействия функциональностей в телефонной сети

Функциональности	**Тупики**	**Циклы**	**Недетерминизм**	**Нарушение условий**
CFB + DC	**false**	**false**	**False**	**false**
CFB + DT	**false**	**false**	**True**	**false**
DC + DT	**false**	**false**	**False**	**false**
CFB + CFB	**false**	**true**	**False**	**false**
All Features	**false**	**true**	**True**	**false**

Поясним некоторые результаты. Если у абонента подключены одновременно CFB и DT, возникает недетерминизм и не ясно, какая именно функциональность должна сработать: должен ли входящий звонок быть сброшен, либо он должен быть переведён. В случае CFB + CFB возникает зацикливание, если два абонента настроили эту услугу, указав номера друг друга в качестве номеров для перевода звонка.

6. Заключение

Наш подход к анализу и верификации телекоммуникационных систем, который базируется на их спецификации униформными системами взаимодействующих конечных автоматов и сетях Петри высокого уровня, имеет следующие преимущества:

– Исходные спецификации телекоммуникационных систем посредством униформных систем взаимодействующих конечных автоматов являются компактными и естественными для ряда широко используемых телекоммуникационных систем таких, как, например, кольцевые протоколы и телефонные сети.

– Алгоритм трансляции униформных систем взаимодействующих конечных автоматов в раскрашенные сети Петри (РСП) формально обоснован, что позволяет проводить полную верификацию телекоммуникационных систем, специфицируемых униформными системами взаимодействующих конечных автоматов.

– Программная система ASV, предназначенная для анализа и верификации телекоммуникационных систем, комбинирует различные средства анализа РСП, реализованные в системе CPN Tools, и нашу систему PNV для верификации РСП методом проверки моделей.

Эти преимущества нашего подхода иллюстрируются применением системы ASV к полной верификации кольцевого RE-протокола и взаимодействия некоторых функциональностей в телефонных сетях.

Проблема моделирования и верификации взаимодействия функциональностей в телефонных сетях рассматривалась в интересных статьях [5,6,8,14,15,17], где взаимодействующие конечные автоматы или родственные модели использовались в [6,8,17], а сети Петри высокого уровня использовались в [5,14,15]. Заметим, что различные виды систем взаимодействующих конечных автоматов рассматривались в [6], но там не было предложено подходящих программных средств для их анализа и верификации. Для верификации телефонных сетей относительно свойств, выраженных в мю-исчислении, использовался метод проверки моделей в [17].

Предполагается интегрировать систему ASV с известной системой верификации методом проверки моделей SPIN и применить расширенную систему к анализу и верификации различных коммуникационных протоколов и взаимодействия большего числа функциональностей в телефонных сетях.

Список литературы

1. Виноградов Р.А., Кузьмин Е.В., Соколов В.А. Верификация автоматных программ средствами CPN/Tools // Моделирование и анализ информационных систем. 2006. Т. 13, № 2. С. 4–15.

2. Кузьмин Е.В., Соколов В.А. Моделирование, спецификация и верификация “автоматных” программ // Программирование. 2008. №1. С. 38–60.

3. Шалыто А.А., Туккель Н.И. SWITCH-технология - автоматный подход к созданию программного обеспечения “реактивных” систем // Программирование. 2001. № 5. С. 45–62.

4. Beloglazov D., Nepomniaschy V. A Two-Level Approach for Modeling and Verification of Telecommunication Systems // Proc. PSI 2009. LNCS. 2010. V. 5947. P. 70–85.

5. Capellmann C., Dibold H., Herzog U. Using High-Level Petri Nets in the Field of Intelligent Networks // LNCS. 1999. V. 1605. P. 1–36.

6. Cavalli A., Maag S. A New Algorithm for Service Interaction Detection // Proc. ICFEM 2002. LNCS. 2002. V. 2495. P. 371–382.

7. Cohen R., Segall A. An Efficient Reliable Ring Protocol // IEEE Transactions on Communications. 1991. V. 39, № 11. P. 1616 – 1623.

8. Gibson P., Hamilton G., Mery D. Integration Problems in Telephone Feature Requirements // Proc. of the 1st Intern. Conf. on Integrated Formal Methods, York, (IFM'99). Springer, 1999. P. 129–148.

9. Jensen K., Kristensen L.M. Coloured Petri Nets: Modelling and Validation of Concurrent Systems // Springer, 2009.

10. Jensen K., Kristensen L.M., Wells L. Coloured Petri Nets and CPN Tools for modeling and validation of concurrent systems // Int. J. on Software Tools for Technology Transfer 9. 2007. P. 213–254.

11. Keck D.O., Kuehn P.J. The Feature and Service Interaction Problem in Telecommunications Systems: A Survey // IEEE Trans. on Software Eng. 1998. V. 24, № 10. P. 779–796.

12. Kozura V.E., Nepomniaschy V.A., Novikov R.M. Verification of Distributed Systems Modelled by High-level Petri Nets // Proc. Intern. Conf. on Parallel Computing in Electrical Engineering. Warsaw, Poland, 2002. P. 61–66.

13. Lee D. Principles and methods of testing finite state machines // Proc. IEEE. 1996. V. 84, № 8. P. 1090–1123.

14. Lorentsen L., Tuovinen A., Xu J. Modelling Feature Interaction Patterns in Nokia Mobile Phones using Coloured Petri Nets and Design/CPN // Proc. 3rd Workshop on Practical Use of Coloured Petri Nets (CPN'01), Aarhus Univ., DAIMI PB-554. 2001. P. 1 – 14.

15. Nakamura M. Design and Evaluation of Efficient Algorithms for Feature Interaction Detection in Telecommunication Services. // PhD dissertation, Osaka University, 1999.

16. Nepomniaschy V., Beloglazov D., Churina T., Mashukov M. Using Coloured Petri Nets to Model and Verify Telecommunications Systems // Proc. CSR 2008. LNCS. 2008. V. 5010, P. 360–371.

17. Schatz B., Salzmann Ch. Service-Based Systems Engineering: Consistent Combination of Services // Proc. ICFEM 2003. LNCS. 2003. V. 2885. P. 86–104.

Verification of Telecommunication Systems Specified by Communicating Finite Automata with the Help of Coloured Petri Nets

Beloglazov D.M., Mashukov M.Yu., Nepomniaschy V.A.

Keywords: telecommunication systems, extended finite automata, coloured Petri nets, ring protocols, feature interactions, telephone networks, verification, model checking method

Uniform systems of communicating extended finite automata are considered in the paper. These automata systems are useful for initial specification of telecommunication systems such as ring protocols and telephone networks. The goal of our paper is to represent an ASV tool (Automata Systems Verifier) intended for analysis and verification of the automata specifications. The ASV tool is based on the algorithm of translation of these automata systems into coloured Petri nets (CPN). This algorithm has been represented and justified in [4]. The ASV tool uses CPN Tools [10] for analysis of the net models and Petri Net Verifier [12] for their verification by the model checking method with respect to properties expressed in mu-calculus. Application of the ASV tool to ring protocol verification and investigation of feature interactions in telephone networks is described.

Сведения об авторах:
Белоглазов Дмитрий Михайлович,
Институт систем информатики им. А.П.Ершова СО РАН,
младший научный сотрудник;
Машуков Михаил Юрьевич,
Институт систем информатики им. А.П.Ершова СО РАН,
младший научный сотрудник;
Непомнящий Валерий Александрович,
Институт систем информатики им. А.П.Ершова СО РАН,
заведующий лабораторией.