ИНФОРМАТИКА И УПРАВЛЕНИЕ В ТЕХНИЧЕСКИХ И СОЦИАЛЬНЫХ СИСТЕМАХ

УДК 519.713 ***EDN:*** **SKLBKO**

УСТАНОВОЧНЫЕ ЭКСПЕРИМЕНТЫ ДЛЯ ТЕЛЕКОММУНИКАЦИОННЫХ КОМПОНЕНТОВ

А.В. Лапутенко
ORCID: **0009-0007-7810-1732** e-mail: **laputenko.av@gmail.com**
Национальный исследовательский Томский государственный университет
Томск, Россия

А.С. Твардовский
ORCID: **0000-0001-7705-7214** e-mail: **tvardal@mail.ru**
Национальный исследовательский Томский государственный университет
Томск, Россия

Н.В. Евтушенко
ORCID: **0000-0002-4006-1161** e-mail: **evtushenko@ispras.ru**
Институт системного программирования им. Иванникова РАН
Москва, Россия

Исследованы свойства экспериментов / последовательностей для идентификации текущего состояния компонентов телекоммуникационных систем. Знание текущего состояния тестируемой системы может снизить стоимость пассивного тестирования, поскольку в ряде случаев достаточно проверить только критические свойства в данном состоянии. Для подобной идентификации используются установочные / синхронизирующие последовательности / трассы, построенные по различным формальным моделям. В качестве таких моделей для описания компонентов современных телекоммуникационных систем широко используются расширенные и временные автоматы. Рассматриваются известные подходы к построению установочных последовательностей для расширенных и временных автоматов на основе соответствующих конечно-автоматных абстракций, анализируется их эффективность, исследуются свойства установочных / синхронизирующих последовательностей.

Ключевые слова: конечные автоматы, временные автоматы, расширенные автоматы, установочная / синхронизирующая последовательность, конечно-автоматная абстракция.

ДЛЯ ЦИТИРОВАНИЯ: Лапутенко, А.В. Установочные эксперименты для телекоммуникационных компонентов / А.В. Лапутенко, А.С. Твардовский, Н.В. Евтушенко // Труды НГТУ им. Р.Е. Алексеева. 2024. № 2. С. 7-19. EDN: SKLBKO

HOMING EXPERIMENTS FOR TELECOMMUNICATION COMPONENTS

A.V. Laputenko
ORCID: **0009-0007-7810-1732** e-mail: **laputenko.av@gmail.com**
National Research Tomsk State University
Tomsk, Russia

A.S. Tvardovskii
ORCID: **0000-0001-7705-7214** e-mail: **tvardal@mail.ru**
National Research Tomsk State University
Tomsk, Russia

N.V. Yevtushenko
ORCID: **0000-0002-4006-1161** e-mail: **evtushenko@ispras.ru**
Ivannikov Institute for System Programming of the Russian Academy of Sciences
Moscow, Russia

Abstract. The paper is devoted to the study of the properties of experiments or sequences for identifying the current state of components of telecommunication systems. It is necessary to know the current state of a system under test, as this can reduce the cost of passive testing, since in some cases it is sufficient to test only critical properties in this state. Homing or synchronizing sequences or traces constructed according to various formal models are used for identification of current state. Extended and timed finite state machines are widely used to describe the components of modern telecommunication systems. Paper presents the well-known approaches to derive homing sequences for extended and timed finite state machines based on the corresponding finite state machine abstractions, and analyzes the effectiveness of this approach. In addition, it includes the study of the properties of the existing homing or synchronizing sequences.

***Key words**:* finite state machines, timed finite state machines, extended finite state machines, homing / synchronizing sequence, finite state machine abstraction.

FOR CITATION: A.V. Laputenko, A.S. Tvardovskii, N.V. Yevtushenko. Homing experiments for telecommunication components. Transactions of NNSTU n.a. R.E. Alekseev. 2024. № 2. Pp. 7-19. EDN: SKLBKO

1. Введение

Установочные последовательности используются для идентификации текущего состояния тестируемой системы [1-5], в которое она переходит после подачи такой последовательности в любом состоянии, идентифицируемое по наблюдаемой реакции. Если достигнутое таким образом состояние не зависит от выходной реакции, входная последовательность называется синхронизирующей. В режиме активного тестирования исследуемая система может быть переведена в известное начальное состояние перед подачей последующих тестовых последовательностей. В процессе мониторинга наблюдение установочных трасс позволяет сократить проверку критических свойств после идентификации текущего состояния, таким образом, знание текущего состояния позволяет минимизировать затраты в режимах как активного, так и пассивного тестирования за счет проверки только критических свойств системы в известных состояниях [6-8]. В ряде случаев для описания поведения особенностей (компонентов) современных систем вместо модели классического конечного автомата используются другие модели с конечным числом состояний. Это расширенные и временные автоматы, которые позволяют, с одной стороны, детализировать поведение системы, с другой – использовать более компактные (с меньшим числом состояний и переходов) описания. Расширенный автомат [9-10] дополняется контекстными переменными, входными и выходными параметрами, а также предикатами, влияющими на выполнение переходов; значения переменных и параметров могут соответствовать критическим показателям, например, таким как объем памяти или количество подключений. Временной автомат [11-13] расширяется таймаутами и/или временными ограничениями, например, для сброса состояния в телекоммуникационных протоколах после истечения допустимого времени ожидания или для переключения в режим энергосбережения.

При решении задач идентификации состояний неклассических автоматов предлагаются различные виды конечно-автоматных абстракций. В их основе лежит идея построения классического конечного автомата, который с заданной точностью описывает поведение временного или расширенного автомата. Для расширенного автомата т.н. l-эквивалент описывает поведение автомата при подаче входных последовательностей, длина которых не превосходит l [14]. Конечно-автоматная абстракция временного автомата описывает поведение автомата на множестве временных входных последовательностей [13]. Для конечных автоматов известны достаточные и необходимые условия существования установочных последовательностей, и на базе конечно-автоматных абстракций такие условия были определены для

временных автоматов [15]. В настоящее время не установлено, каким образом для расширенных автоматов можно проверить наличие или отсутствие установочных последовательностей, за исключением построения полного конечно-автоматного эквивалента посредством полного моделирования расширенного автомата, если эквивалент существует [9]. В данной работе проанализирована эффективность построения установочных последовательностей для некоторых компонентов телекоммуникационных систем, поведение которых описано посредством расширенных и временных автоматов, с использованием их различных конечно-автоматных абстракций.

2. Конечные автоматы

Конечный автомат (далее – автомат) есть четверка $\mathsf{P} = (\mathrm{P}, \mathrm{I}, \mathrm{O}, \mathrm{h_P})$, где P, I и O – конечные непустые множества состояний, входных и выходных символов. Соответственно, $\mathrm{h_P} \subseteq (\mathrm{P} \times \mathrm{I} \times \mathrm{O} \times \mathrm{P})$ – отношение переходов. Кортеж $(\mathrm{p}, \mathrm{i}, \mathrm{o}, \mathrm{p}')$ представляет переход под воздействием входного символа i из текущего состояния p в состояние p' с выходным символом o. Автомат P называется полностью определенным и детерминированным, если для каждой пары $(\mathrm{p}, \mathrm{i}) \in \mathrm{P} \times \mathrm{I}$ существует ровно одна пара $(\mathrm{o}, \mathrm{p}') \in \mathrm{O} \times \mathrm{P}$ такая, что $(\mathrm{p}, \mathrm{i}, \mathrm{o}, \mathrm{p}') \in \mathrm{h_P}$. В данном исследовании рассматриваются только полностью определенные автоматы. Единственное исключение составляет абстракция расширенных автоматов, частичность которой будет устраняться при помощи так называемого «безразличного» состояния (состояние DNC).

Вход-выходная (*IO-*) *последовательность* $\alpha/\gamma = i_1/o_1, i_2/o_2, \ldots, i_l/o_l$ существует в состоянии p_1 автомата P, если существуют переходы $(p_1, i_1, o_1, p_2), (p_2, i_2, o_2, p_3), \ldots, (p_l, i_l, o_l, p_{l+1}) \in h_P$. Соответственно, l – длина последовательности α/γ, $\alpha = i_1, i_2, \ldots, i_l$ – ее входная проекция, а $\gamma = o_1, o_2, \ldots, o_l$ – выходная проекция последовательности.

Далее представлены некоторые определения и обозначения для расширенных и временных автоматов.

Расширенный конечный автомат (далее – расширенный автомат) дополняет классический автомат контекстными переменными, входными / выходными параметрами и предикатами (ограничениями) на переходах, определенными на множестве значений контекстных переменных и входных параметров. Формально расширенный конечный автомат S представляет собой семерку $(S, I, I_p, O, O_p, V, \lambda_S)$, где S – конечное множество состояний, I – конечное множество входных символов, I_p – конечное множество входных параметров, O – конечное множество выходных символов, O_p – конечное множество выходных параметров, V – конечное множество контекстных переменных, λ_S – конечное множество переходов между состояниями множества S.

Переход $t \in \lambda_S$ есть семерка (s, i, P, op, up, o, s'), где:

- s и s' – начальное и конечное состояния перехода t;
- $i \in I$ – входной символ, который может быть дополнен вектором значений входных параметров $\boldsymbol{x}$ из множества $D_{inp\text{-}i}$, где $D_{inp\text{-}i}$ – множество векторов возможных значений входных параметров для входного символа i;
- $o \in O$ – выходной символ, который может быть дополнен вектором выходных параметров $\boldsymbol{y}$ из множества $D_{out\text{-}o}$, где $D_{out\text{-}o}$ – множество векторов возможных значений выходных параметров для выходного символа o;
- P: $D_{inp\text{-}i} \times D_V \to \{\mathsf{true}, \mathsf{false}\}$ – предикат, где D_V – множество векторов возможных значений контекстных переменных и $i \in I$;
- op: $D_{inp\text{-}i} \times D_V \to D_{out\text{-}o}$ – функция вычисления выходных параметров, где $i \in I$ и $o \in O$;
- up: $D_{inp\text{-}i} \times D_V \to D_V$ – функция обновления контекстных переменных, где $i \in I$.

Таким образом, когда входной символ i с вектором значений входных параметров $\boldsymbol{x} = (x_1, x_2, \ldots) \in D_{inp\text{-}i}$ подается в состоянии s на расширенный автомат S и вектор контекстных

переменных равен $\boldsymbol{v} = (v_1, v_2, \ldots) \in D_V$, автомат выполняет переход (s, i, P, op, up, o, s') тогда и только тогда, когда $P(\boldsymbol{x}, \boldsymbol{v}) = \text{true}$. В этом случае S переходит в состояние s', выдавая выходной символ o, возможно, с вектором значений выходных параметров $op(\boldsymbol{x}, \boldsymbol{v}) = \boldsymbol{y} = (y_1, y_2, \ldots) \in D_{out\text{-}o}$, и следующий вектор значений контекстных переменных есть $\boldsymbol{v}' = up(\boldsymbol{x}, \boldsymbol{v})$. Пара $(s, \boldsymbol{v})$ представляет собой *конфигурацию* в расширенном автомате, $(i, \boldsymbol{x})$ – параметризованный входной символ, $(o, \boldsymbol{y})$ – параметризованный выходной символ.

Как уже было отмечено, в данной статье рассмотрены полностью определенные расширенные автоматы, для которых в каждой конфигурации определен переход по каждому параметризованному входному символу. Если по некоторому параметризованному входному символу переход не определен, автомат доопределяется переходом-петлей в этом состоянии со специальным выходным символом «*ignore*» или «*null*», не меняющим значения контекстных переменных. Предполагается также, что в каждом состоянии для каждого параметризированного входного символа один и только один предикат принимает значение true. Таким образом, далее рассматриваются *полностью определенные детерминированные* автоматы, которые достаточно часто используются для описания компонентов телекоммуникационных систем. Параметризованная входная (выходная) последовательность $\alpha = (i_0, \boldsymbol{x}_0) \ldots (i_n, \boldsymbol{x}_n)$ $(\gamma = (o_0, \boldsymbol{y}_0) \ldots (o_n, \boldsymbol{y}_n))$ представляет собой последовательность параметризованных и, возможно, непараметризованных, входных (выходных) символов. Последовательность конфигураций и переходов, которые проходит расширенный автомат при обработке параметризованной входной последовательности α в конфигурации $(s, \boldsymbol{v})$ с соответствующей параметризованной выходной последовательностью γ, называется *трассой tr с IO-проекцией* α/γ. Последняя конфигурация трассы *tr* является *tr-преемником* конфигурации $(s, \boldsymbol{v})$.

Рассмотрено также построение установочных последовательностей для систем с временными аспектами (временные системы с таймаутами), поскольку в телекоммуникационных протоколах, таких как TCP, HTTPS, TFTP и др., могут быть разные временные ограничения, например, таймаут сервера обычно определяет максимальное время ожидания ответа от клиента. По истечении времени ожидания соединение разрывается. Временной автомат представляет собой пятерку $\mathsf{S} = (S, I, O, \lambda_S, \Delta_S)$, где множества S, I и O обозначают те же объекты, что в классическом автомате, $\lambda_S \subseteq S \times I \times O \times S$ – *отношение переходов*, а Δ_S – *функция таймаута*. Функция *таймаута* Δ_S: $S \to S \times (N \cup \{\infty\})$ определяет для каждого состояния допустимое время ожидания входного символа, выраженное положительным целым числом абстрактных тактов. Если $\Delta_S(s) = (s', T)$ и никакой входной символ не подается *до истечения таймаута* T, то автомат S переходит в состояние s' и часы «сбрасываются» в ноль. Неограниченное значение таймаута ∞ означает, что автомат может оставаться в текущем состоянии сколь угодно долго до тех пор, пока не будет подан входной символ. Переход (s, i, o, s') определяется аналогично переходу классического конечного автомата. Полностью определенные и частичные, детерминированные и недетерминированные временные автоматы определяются аналогично классическим автоматам.

Временное состояние автомата S – это пара $(s, x) \in S_T = \{(s, x) : (s, x) \in S \times \{\mathbb{R}^+ \cup 0\}$ и $x < T$ и $\Delta_S(s) = (s', T)\}$, т.е. отражает текущее состояние s и значение временной переменной x, которое может достигать значений не больше таймаута в состоянии s. Переходы между временными состояниями под действием входных символов (вход-выходное отношение переходов $_{io}\!\!\to$) и с течением времени (временное отношение переходов $_{t}\!\!\to$) определяются аналогично [13].

Для каждого временного состояния (s, x) и задержки $t \geq 0$, тройка $\langle (s, x), t, (s', x') \rangle \in {}_{t}\!\!\to$, где (s', x') – временное состояние, достигнутое временным автоматом за время t, если входные символы не подавались. Далее такой переход обозначается как $(s, x)\ {}_{t}\!\!\to (s', x')$. Переход $\langle (s, x), i, o, (s', 0) \rangle \in {}_{io}\!\!\to$ если $(s, i, o, s') \in \lambda_S$ (или $(s, x)\ {}_{io}\!\!\to (s', 0)$ для краткости).

Временным входным символом называется пара (i, t), где $i \in I$ и t – неотрицательное действительное число; временной входной символ (i, t) означает, что входной символ i пода-

ется на автомат в момент времени $t \geq 0$, отсчитываемый от некоторого начального момента времени. Последовательность временных входных символов $\alpha = (i_0, t_0) \ldots (i_n, t_n)$, $t_0 < \ldots < t_n$, представляет собой *временную входную последовательность*; соответствующая выходная последовательность во временном состоянии (s, x) определяется итеративно по указанным выше правилам. Последовательность состояний и переходов, которые проходит автомат при подаче временной входной последовательности α во временном состоянии $(s, 0)$ с соответствующей выходной последовательностью γ, представляет собой трассу *tr* с временной *IO*-проекцией α/γ [15]. Финальное временное состояние трассы *tr* является *tr*-преемником.

3. Установочные последовательности

Для классических полностью определенных автоматов установочной называют входную последовательность, определяющую состояние исследуемого автомата после ее подачи и наблюдения выходной последовательности, независимо от состояния, в котором она подается. Для полностью определенного автомата установочная последовательность может быть получена с использованием соответствующего усеченного дерева преемников [1]. Установочная последовательность называется *синхронизирующей*, если достигнутое состояние не зависит от выходной последовательности.

Ниже рассматриваются понятия установочной последовательности и соответствующие подходы к ее построению для расширенных и временных автоматов.

3.1. Построение установочной последовательности для расширенного автомата

Для расширенного автомата S (параметризованная) вход-выходная последовательность α/γ называется *c-установочной*, если для каждой пары трасс tr_1 и tr_2 в конфигурациях $(s_1, \boldsymbol{v}_1)$, $(s_2, \boldsymbol{v}_2)$ с одной и той же *IO*-проекцией α/γ, tr_1 -преемник $(s_1, \boldsymbol{v}_1)$ и tr_2 -преемник $(s_2, \boldsymbol{v}_2)$ совпадают. Параметризованная входная последовательность α является *c-установочной* для расширенного автомата S, если в любой конфигурации α порождает только трассы с *c*-установочными *IO*-проекциями. Параметризованная вход-выходная последовательность α/γ называется *s-установочной* для S, если для каждой пары трасс tr_1 и tr_2 в конфигурациях $(s_1, \boldsymbol{v}_1)$, $(s_2, \boldsymbol{v}_2)$ с одной и той же *IO*-проекцией α/γ, состояния в tr_1-преемнике $(s_1, \boldsymbol{v}_1)$ и tr_2-преемнике $(s_2, \boldsymbol{v}_2)$ совпадают. Параметризованная входная последовательность α является *s-установочной* для расширенного автомата S, если α порождает только трассы с *s*-установочными *IO*-проекциями. Таким образом, *c*-установочные последовательности позволяют идентифицировать конфигурацию расширенного автомата в то время как *s*-установочные определяют только состояние.

Для построения установочных последовательностей использована конечно-автоматная абстракция, частично моделирующая расширенный автомат. Состояния конечно-автоматной абстракции представляют собой конфигурации, в то время как входной (выходной) алфавит расширяется за счет векторов значений выходных (выходных) параметров. Если такую автоматную абстракцию невозможно построить из-за бесконечного или слишком большого набора конфигураций (или параметризованных входных символов), можно использовать *l*-эквивалент расширенного автомата, описывающий поведение автомата в начальной конфигурации для всех входных последовательностей длины, не превышающей *l* [14]. В этом случае рассматриваемая модель представляет собой инициальный автомат, где состояниями *l*-эквивалента являются конфигурации расширенного автомата, достигаемые при подаче последовательности длины не больше *l* из заданной начальной конфигурации. Начальная конфигурация для построения *l*-эквивалента выбирается исходя из семантики системы; для телекоммуникационных систем это обычно предусмотрено спецификацией. Для расширенного автомата $\mathsf{S} = (S, I, I_p, O, O_p, V, \lambda_\mathsf{S})$ с начальной конфигурацией $(s_0, \boldsymbol{v_0})$, *l*-эквивалент $\mathsf{A_S}(l)$ есть автомат $(S_A, I_A, O_A, h_{\mathsf{SA}})$, где $S_A \subseteq S \times D_V$, $I_A \subseteq \{i \times D_{inp\text{-}i} : i \in I\}$, $O_A \subseteq \{o \times D_{inp\text{-}o} : o \in O\}$. Для состояния $(s, \boldsymbol{v}) \in S_A$ автомата $\mathsf{A_S}(l)$ в *l*-эквиваленте существует

переход $((s, \boldsymbol{v}), (i, \boldsymbol{x}), (o, \boldsymbol{y}), (s', \boldsymbol{v}'))$, если в расширенном автомате существует переход $(s, i, P, op, up, o, s') \in \lambda_S$, где $P(\boldsymbol{v}, \boldsymbol{x}) = true$, $op(\boldsymbol{v}, \boldsymbol{x}) = \boldsymbol{y}$, $up(\boldsymbol{v}, \boldsymbol{x}) = \boldsymbol{v}'$. l-эквивалент *полностью определенного* детерминированного расширенного автомата S может быть частичным, если существует переход в некоторую конфигурацию S, недостижимую в l-эквиваленте. В таком случае неопределенные переходы рассматриваются как переходы в специальное DNC-состояние со специальным выходным символом *null*. В состоянии DNC определены петлевые переходы по всем входным символам с выходным символом *null*. Конечно-автоматной абстракцией (*полным* l-эквивалентом) называется l-эквивалент расширенного автомата, который совпадает с $(l+1)$-эквивалентом, если такой l-эквивалент существует.

При построении установочной последовательности все l-эквиваленты и конечно-автоматная абстракция, если она существует, рассматриваются как неинициальные автоматы, и установочная последовательность для расширенного автомата может быть построена с использованием таких абстракций, если состояние DNC не достигается в дереве преемников и, соответственно, ни одна индуцированная установочной последовательностью трасса не достигает состояния DNC. Построены установочные последовательности для расширенного автомата, используя метод, предложенный в работе [10]: параметризованная установочная последовательность (если она существует) строится на основе усеченного дерева преемников для l-эквивалента расширенного автомата. Входная последовательность α l-эквивалента называется *l-ограниченной установочной последовательностью* для расширенного автомата, если она является установочной последовательностью для l-эквивалента, в множество начальных состояний (т. е. состояний, формирующих корень дерева преемников) которого не входит DNC. Согласно [10], в любом состоянии, не совпадающем с DNC, любая индуцируемая α трасса не переводит l-эквивалент в состояние DNC и обладает установочной *IO*-проекцией. Может оказаться, что для l-эквивалента установочная последовательность не существует, но существует для $(l+1)$-эквивалента, и возникает вопрос, можно ли определить значение l, которого достаточно для построения установочной последовательности, если такая последовательность существует.

Утверждение. Пусть S – расширенный автомат, в котором контекстные переменные и входные параметры принимают значения из конечных множеств, и $A_S(l)$ – его конечно-автоматная абстракция (полный l-эквивалент). Входная (возможно, параметризированная) последовательность α является c-установочной последовательностью для S, если и только если она является установочной для $A_S(l)$.

Действительно, если переменные и входные параметры принимают значения из конечных множеств, расширенный автомат вырождается в классический конечный и может быть полностью описан полным l-эквивалентом. По определению, в этом случае l-ограниченная установочная последовательность есть c-установочная последовательность. По определению, c-установочная последовательность для S является s-установочной последовательностью для S. Обратное, в общем случае, неверно. Отметим также, что если l-эквивалент $A_S(l)$ не является полным, то его состояния не отражают всех конфигураций расширенного автомата S и, соответственно, справедливы следующие замечания:

1) l-ограниченная установочная последовательность, построенная по l-эквиваленту $A_S(l)$, не является, в общем случае, c-(s-)установочной для расширенного автомата S;

2) C-(s-)установочная последовательность расширенного автомата S, не является, в общем случае, l-ограниченной установочной для l-эквивалента $A_S(l)$, поскольку по этой последовательности в $A_S(l)$ может достигаться состояние DNC.

В разделе, содержащем экспериментальные результаты, проверяется, насколько часто можно построить установочную последовательность для расширенного автомата по его l-эквиваленту.

3.2. Построение установочных последовательностей для временных автоматов

Вход-выходная временная последовательность α/γ называется *установочной* для полностью определенного детерминированного временного автомата S, если для каждой пары трасс tr_1 и tr_2 во временных состояниях $(s_1, 0)$, $(s_2, 0) \in S_T$ с одной и той же *IO*-проекцией α/γ , tr_1 -преемник $(s_1, 0)$ и tr_2 -преемник $(s_2, 0)$ совпадают.

Временная входная последовательность α является *установочной последовательностью* для неинициального полностью определенного детерминированного временного автомата S, если α порождает только трассы с установочными *IO*-проекциями. Временное состояние (s, x) временного автомата S называется *стабильным*, если таймаут в состоянии s равен бесконечности, т.е. $\Delta_S(s) = (s, \infty)$, и соответственно, временной автомат может оставаться в этом состоянии сколь угодно долго.

Установочная вход-выходная последовательность α/γ называется *стабильной* для автомата S, если для любой трассы tr во временном состоянии $(s, 0) \in S_T$ с *IO*-проекцией α/γ , tr-преемник $(s, 0)$ является *стабильным состоянием*. Установочная последовательность α называется *стабильной* для временного автомата S, если α порождает только трассы со стабильными установочными *IO*-проекциями.

Установочная последовательность (в том числе стабильная) для временного автомата с таймаутами может быть построена с использованием его конечно-автоматной абстракции [15]. Конечно-автоматная абстракция временного автомата является классическим конечным автоматом, описывающим поведение временного автомата на подмножестве входных временных последовательностей. Каждое состояние такой абстракции представляет собой пару (s, t), где s – состояние исходного временного автомата, а t – целочисленное значение временной переменной, строго меньшее таймаута в состоянии s; ее увеличение на один такт и переходы по таймауту моделируются специальной вход-выходной парой 1/1. Если таймаут в состоянии s равен ∞, то конечно-автоматная абстракция содержит только состояние $(s, 0)$ с переходом-петлей по паре 1/1. Для детерминированного полностью определенного временного автомата конечно-автоматная абстракция является детерминированным и полностью определенным автоматом.

При построении установочной последовательности для временного автомата на первом шаге проверяется, существует ли установочная последовательность для его конечно-автоматной абстракции, построив усеченное дерево преемников. Если конечно-автоматная абстракция не обладает установочной последовательностью, то такой последовательности нет и во временном автомате. Если установочная последовательность построена для конечно-автоматной абстракции, то абстрактная установочная последовательность преобразуется во временную входную последовательность, которая является установочной для временного автомата [15]. Для синтеза стабильных установочных последовательностей на дерево преемников накладываются дополнительные ограничения, и по абстракции строится S'-установочная последовательность, переводящая временной автомат только в стабильные состояния множества S'.

4. Экспериментальные результаты для расширенных автоматов

Телекоммуникационные протоколы часто используют клиент-серверную архитектуру, где клиент отправляет запросы серверу, а сервер отвечает на запросы клиента. Сообщения запроса обычно содержат дополнительные параметры, которые влияют на ответ и изменение состояния клиента и сервера. Соответственно, для их адекватного описания используются расширенные автоматы, и в этом разделе рассмотрено построение установочных последовательностей для ряда телекоммуникационных протоколов на основе расширенных конечных автоматов.

4.1. Автоматные описания телекоммуникационных протоколов

Simple Connection Protocol (SCP) представляет собой систему, которая отражает основные свойства телекоммуникационных протоколов и описывает обмен данными между клиентом и сервером при установлении соединения с определенным уровнем качества (qos). Расширенный автомат для этого протокола представлен на рис. 1 [10].

В данной работе предполагается, что уровень качества передачи данных ограничен значениями 0, 1, 2 и, соответственно, значения входных параметров CONreq.qos и Accept.qos принадлежат множеству {0, 1, 2}. Значения выходных параметров «NONsupport.ReqQos», «connect.ReqQos» и «data_out.ReqQos» ограничены аналогичным образом. Расширенный автомат имеет 3 состояния, 5 входных, 5 выходных символов и 3 контекстные переменные ReqQos, TryCount, FinQos.

В состоянии s_1 сервер ожидает запрос на входящие соединения с заданным уровнем качества соединения (CONreq.qos) не выше 1. При появлении подходящего запроса, система переходит в состояние s_2, в котором клиент ожидает подтверждения готовности сервера. Если соединение установлено успешно, система переходит в состояние s_3, в котором происходит передача данных. Расширенный автомат (рис. 1) был доопределен до полностью определенного посредством введения соответствующих петель.

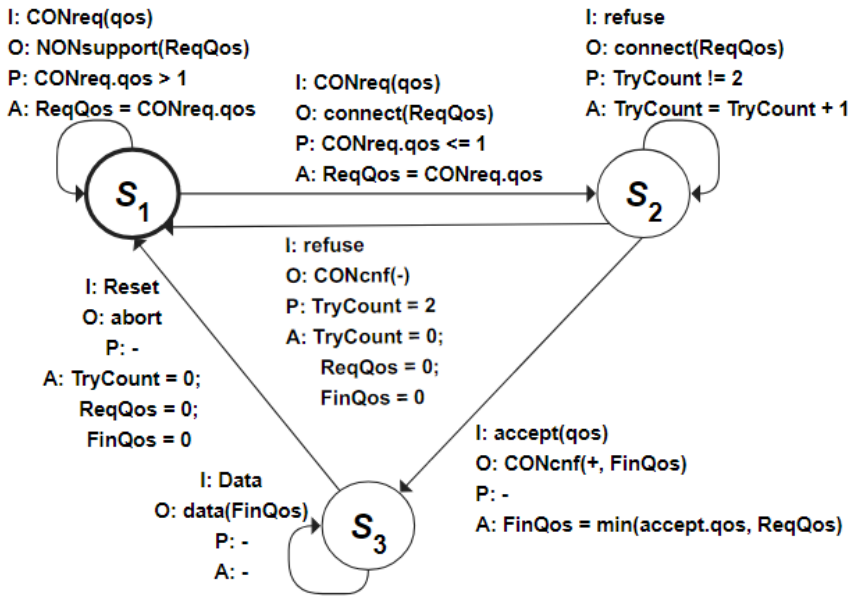

Рис. 1. Расширенный автомат для протокола SCP

Fig. 1. Extended finite state machines for SCP protocol

В ходе экспериментов были построены l-эквиваленты для l от 2 до 5, причем 5-эквивалент является конечно-автоматной абстракцией расширенного автомата на рис. 1 и имеет 17 состояний, 8 абстрактных входных и 6 выходных символов. Согласно описанию протокола, l-эквивалент строился для начальной конфигурации (s_1, (0, 0, 0)). Практически все построенные l-эквиваленты были частичными автоматами, и для использования программы [16] при построении установочных последовательностей неопределенные переходы были доопределены как переходы в специальное состояние DNC. Установочные последовательности, переводящие автомат в состояние DNC, в полученных результатах не учитывались. С помощью 5-эквивалента было найдено 726 059 входных последовательностей длины не больше 7, являющихся s-установочными для расширенного автомата (рис. 1). Более длинные последовательности не рассматривались ввиду их большого числа. Ни один из построенных l-эквивалентов (l < 5) не обладал s-установочными последовательностями. Из всех

установочных последовательностей 409 268 являются синхронизирующими. Также для 5-эквивалента всего было найдено 585 *c*-установочных последовательностей (максимальная длина последовательностей – 7), из которых 304 являются синхронизирующими.

Рассмотрим далее *Trivial File Transfer Protocol* [17], представляющий собой довольно простой инструмент передачи файлов между двумя узлами сети. Расширенный конечный автомат для протокола TFTP представлен на рис. 2. Чтобы упростить анализ, рассмотрим модель в которой только один файл может быть передан между узлами сети. При этом в файле может быть не более 3 блоков по 512 байт каждый.

В процессе экспериментов для расширенного автомата TFTP (рис. 2) были построены *l*-эквиваленты для *l* от 2 до 6, причем 6-эквивалент является полной конечно-автоматной абстракцией расширенного автомата на рис. 2 и имеет 9 состояний, 6 абстрактных входных и 5 выходных символов. Для 6-эквивалента было построено 468 919 последовательностей длины не больше 9, являющихся *s*-установочными для расширенного автомата на рис. 2, из которых 157240 являются синхронизирующими. Более длинные последовательности не рассматривались ввиду их большого числа. Число всех *c*-установочных последовательностей составило 713 (максимальная длина последовательностей – 9), из которых 168 являются синхронизирующими. На данный момент авторы не готовы сформулировать свойства *l*-эквивалентов, по которым можно построить установочные последовательности для полной абстракции, и таким образом, в отличие от тестовых последовательностей [14], использование *l*-ограниченных установочных последовательностей представляется неэффективным при построении *s*-установочных последовательностей для телекоммуникационных протоколов.

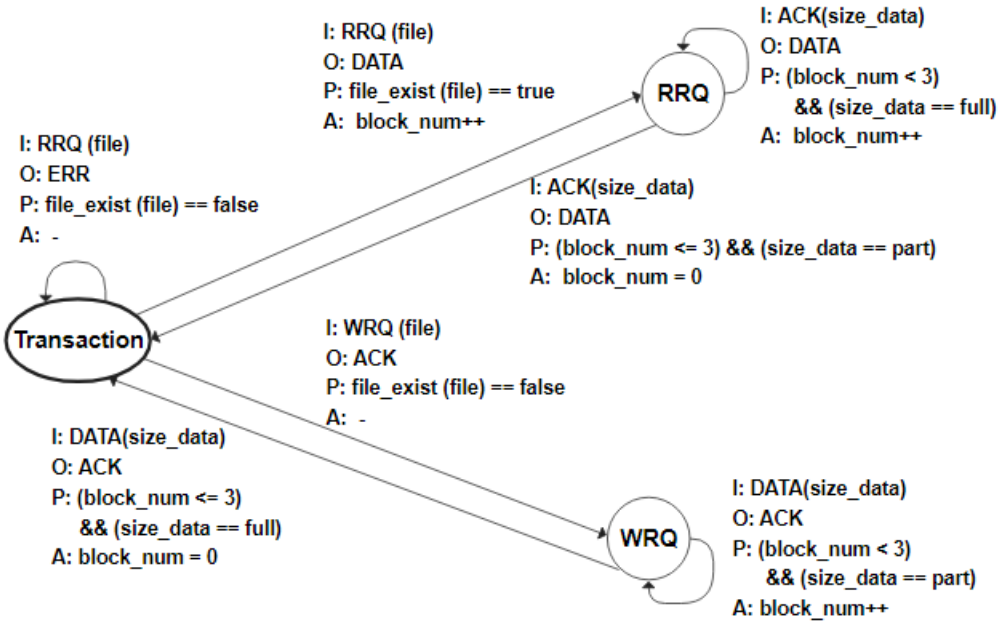

Рис. 2. Расширенный автомат для протокола TFTP

Fig. 2. Extended finite state machines for TFTP protocol

На начальном этапе (до построения *l*-эквивалента) неопределенные переходы в расширенном автомате доопределялись как петля в текущем состоянии с выходным символом *ignore* или *null*. Поэтому для использования этих последовательностей на практике важно, чтобы при реализации системы разработчик добавил обработку системой неопределенных в спецификации переходов как петлю по неопределенным переходам с соответствующим выходным символом.

5. Экспериментальные результаты для временных автоматов

Рассмотрим построение установочных последовательностей для временных систем с таймаутами. В телекоммуникационных протоколах (TCP, HTTPS, TFTP и т. д.) таймаут сервера часто определяет максимальное время ожидания запроса от клиента. По истечении времени ожидания соединение разрывается. Это свойство добавляет пустую установочную последовательность в соответствующий временной автомат, по которой серверная реализация сбрасывается в начальное состояние по истечении таймаута. Для автоматов с таймаутами описанного выше класса установочная последовательность может быть построена на основе конечно-автоматной проекции временного автомата, полученной путем удаления всех переходов по таймауту. Можно показать, что, если каждое состояние, кроме начального s_{in}, полностью определенного детерминированного временного автомата, имеет конечный таймаут, $\Delta(s) = (s_{in}, T)$, и таймаут в состоянии s_{in} равен ∞, то временная входная последовательность α является *установочной* для автомата S, если и только если соответствующая абстрактная последовательность является установочной для конечно автоматной проекции автомата S. Исключением являются последовательности, во временных символах которых присутствует временная задержка $t \geq T$. Отметим, что такую проекцию можно использовать для поиска установочных последовательностей в произвольных автоматах с таймаутами, но в таком случае не гарантируется обнаружение таких последовательностей.

Временной конечный автомат для протокола TFTP представлен на рис. 3 [18]. Эта модель описывает серверную часть протокола, исходя из предположения, что каждый файл состоит из трех частей, и повторные передачи не допускаются. Временной автомат, рассмотренный в данном разделе, отличается от расширенного автомата на рис. 2, в частности, в данной модели не учитываются параметры входных символов.

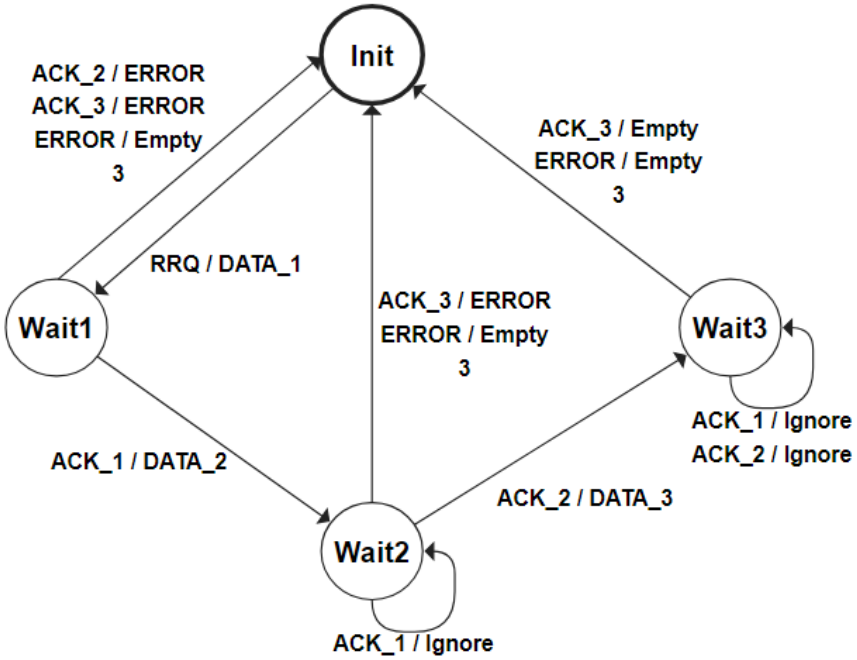

Рис. 3. Автомат с таймаутами для протокола TFTP

Fig. 3. Finite state machines with timeouts for TFTP protocol

Конечно-автоматная абстракция [19] для TFTP имеет 11 состояний, 6 входных и 7 выходных символов. Для конечно-автоматной абстракции существует стабильная установочная последовательность (1, 1, 1), в которой входной символ 1 обозначает ожидание одного такта времени. Отметим, что такая пустая установочная последовательность также является синхронизирующей. Соответственно, временной автомат обладает пустой стабильной

синхронизирующей последовательностью, поскольку из любого состояния после ожидания 3 тактов, автомат перейдет в начальное состояние, которое является стабильным. Ввиду данного свойства, остальные установочные последовательности могут быть построены по конечно-автоматной проекции временного автомата. В результате экспериментов для TFTP была построена 1 пустая синхронизирующая / установочная последовательность, 30 временных установочных последовательностей длины 2 и 90 установочных последовательностей длины 3. Всего существует 85 стабильных установочных последовательностей и 112 синхронизирующих последовательностей.

Были проведены эксперименты с моделью системы управления поездом [20]. Конечный автомат с таймаутами (TAB = 3, TBC = 2) представлен на рис. 4. Система состоит из поезда и базовой станции (БС), которая управляет скоростью поезда в зависимости от его текущего положения, скорости и ускорения. На рисунке представлена только составляющая системы, относящаяся к управлению поездом. В исходном состоянии управление поездом отсутствует. После получения управляющего сигнала от базовой станции поезд начинает движение (состояние «Moving»). При получении сигнала *neg* поезд замедляет ход и входит в состояние согласования дальнейших параметров движения («Negotiation»). Переход из состояния «Moving» в состояние «Negotiation» возможен по тайм-ауту TAB. При отсутствии сигналов в состоянии «Negotiation» поезд переходит в состояние «Stop», снижая скорость до 0 (тайм-аут TBC). После получения сигналов *control*, *neg* и *move* от базовой станции поезд набирает скорость и переходит в состояние «Negotiation» для получения дальнейших команд управления. Конечно-автоматная абстракция [19] для этой модели имеет 8 состояний, 6 входных и 4 выходных символов.

Заметим, что состояния «Start» и «Stop» являются стабильными, но, поскольку состояние «Start» не имеет входящего перехода, достаточно рассмотреть только состояние «Stop». Также, существует пустая последовательность, которая переводит систему из состояний «Moving», «Negotiation» и «Stop» в состояние «Stop» по таймаутам TAB и/или TBC и, таким образом, если известно, что система покинула начальное состояние, то пустая последовательность, соответствующая ожиданию суммы двух этих таймаутов, является стабильной синхронизирующей / установочной последовательностью.

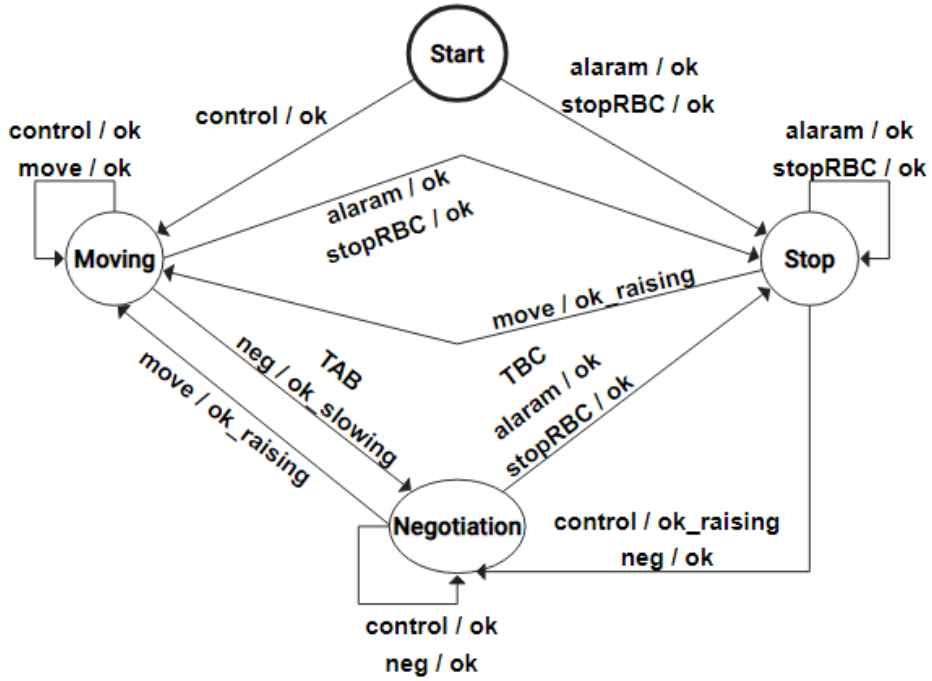

Рис. 4. Временной автомат с таймаутами для системы управления поездом

Fig. 4. Timed finite state machines with timeouts for a train control system

Таблица 1.
Анализ установочных последовательностей для временных систем

Table 1.
Analysis of homing sequences for timed systems

Система	**HS существует**	**Стабильная HS существует**	**Пустая HS существует**	**Синхронизирующая последовательность существует**
TFTP	ДА	ДА	ДА	ДА
Система управления поездом	ДА	НЕТ	НЕТ	ДА

Для системы управления поездом существует 14 временных установочных последовательностей длины 1 и 96 последовательностей длины 2, из которых 104 последовательности являются синхронизирующими. Пустых и стабильных установочных последовательностей нет. В табл. 1 представлены результаты по наличию / отсутствию установочных / синхронизирующих последовательностей для систем с временными аспектами.

6. Заключение

Получены экспериментальные результаты по построению установочных и синхронизирующих последовательностей для телекоммуникационных протоколов на основе расширенных и временных автоматов. Анализ существования и построение установочных и синхронизирующих последовательностей проводились на основе различных конечно-автоматных абстракций. Для всех рассмотренных систем были обнаружены короткие установочные и синхронизирующие последовательности (длина последовательности меньше числа состояний). Для систем, моделируемых расширенным автоматом с большим числом конфигураций (или комбинаций входных параметров), анализ существования установочных последовательностей проводился на основе *l*-эквивалентов и на данный момент не представляется достаточно эффективным. Возможно, представляет интерес использование результатов синтеза синхронизирующих последовательностей для расширенных автоматов специального вида [21]. Для временных автоматов с пустой установочной / синхронизирующей последовательностью специального вида построение абстракции и дальнейший анализ системы упрощается за счет уменьшения числа состояний абстракции, поскольку вместо конечно-автоматной абстракции достаточно рассмотреть конечно-автоматную проекцию временного автомата, т.е. проекцию, в которой удалены все переходы по таймаутам.

Работа частично поддержана проектом РНФ № 22-29-01189.

Библиографический список

1. **Gill, A.** Introduction to the theory of finite-state machines / A. Gill. – McGraw Hill, 1962.
2. **Lee, D.** Testing finite-state machines: state identification and verification / D. Lee, M. Yannakakis // IEEE Transactions on Computers. 1994. Vol. 43. № 3. Pp. 306-320.
3. **Hibbard, T. N.** Least upper bounds on minimal terminal state experiments for two classes of sequential machines // Journal of the ACM. 1961. Vol. 8. № 4. Pp. 601-612.
4. **Wang, H.-E.** Homing Sequence Derivation with Quantified Boolean Satisfiability // Testing Software and Systems Lecture Notes in Computer Science / H.-E Wang et al. – Cham: Springer International Publishing, 2017. Pp. 230-242.
5. **Sandberg, S.** Homing and Synchronizing Sequences / S. Sandberg // Model-Based Testing of Reactive Systems Lecture Notes in Computer Science. – Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. Pp. 5-33.

6. **Wehbi, B.** Events-Based Security Monitoring Using MMT Tool / B. Wehbi et al. // 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation. Montreal, QC, Canada: IEEE, 2012. Pp. 860-863.
7. **Lopez, J.** Behavior evaluation for trust management based on formal distributed network monitoring / J. Lopez et al. // World Wide Web. 2016. Vol. 19. N 1. Pp. 21-39.
8. **Bayse, E.** A passive testing approach based on invariants: application to the WAP / E. Bayse et al. // Computer Networks. 2005. T. 48. N 2. Pp. 247-266.
9. **Petrenko, A.** Confirming configurations in efsm testing / A. Petrenko, S. Boroday, R. Groz // IIEEE Trans. Software Eng. 2004. Vol. 30. N 1. Pp. 29-42.
10. **Kushik, N.** Improving Protocol Passive Testing through «Gedanken» Experiments with Finite State Machines / N. Kushik et al. // 2016 IEEE International Conference on Software Quality, Reliability and Security (QRS). Vienna, Austria: IEEE, 2016. Pp.315-322.
11. **Krichen, M.** Conformance testing for real-time systems / M. Krichen, S. Tripakis // Form Methods Syst Des. 2009. Vol. 34. N 3. Pp. 238-304.
12. **Merayo, M. G.** Formal testing from timed finite state machines / M.G. Merayo, M. Núñez, I. Rodríguez // Computer Networks. 2008. Vol. 52. N 2. Pp.432-460.
13. **Bresolin, D.** Equivalence checking and intersection of deterministic timed finite state machines / D. Bresolin et al. // Form Methods Syst Des. 2021. Vol. 59. N 1–3. Pp. 77-102.
14. **Kushik, N.** Studying the optimal height of the EFSM equivalent for testing telecommunication protocols / N. Kushik et al. // 2nd International Conference on Advances in Computing, Communication and Information Technology CCIT 2014. Birmingham: The Institute of Research Engineers and Doctors, 2014. Pp. 159-163.
15. **Tvardovskii, A.** Deriving homing sequences for Finite State Machines with timeouts / A. Tvardovskii, N. Yevtushenko // The Computer Journal. 2023. Vol. 66. N 9. Pp. 2181–2190.
16. Home_sequence. Available at: https://mks2.cs.msu.ru/EvgeniiEM/home_sequence/-/tree/home_sequence (Accessed 01.02.2024).
17. RFC1350 – The TFTP Protocol (revision 2). Available at: https://www.ietf.org/rfc/rfc1350.txt (Accessed 01.02.2024).
18. **Zhigulin, M.** Detecting Faults in TFTP Implementations using Finite State Machines with Timeouts / M. Zhigulin, S. Prokopenko, M. Forostyanova // Proceedings of the Spring/Summer Young Researchers' Colloquium on Software Engineering. 2012. Pp. 115-118.
19. Deriving-FSM-Abstraction. Available at: https://github.com/AlexTvardFSM/Deriving-FSM-Abstraction (Accessed 01.02.2024).
20. **Janhsen, A.** Modelling and simulation of the new European train control system / A. Janhsen et al. // Proceedings of the IMACS Symposium on Mathematical Modelling. Vienna: ARGE Simulation News, Technical University Vienna, 1997. Pp. 473-478.
21. **Kushi, N.** Studying Synchronization Issues for Extended Automata / N. Kushi, N. Yevtushenko // Proceedings of the 18th International Conference on Evaluation of Novel Approaches to Software Engineering. Prague, Czech Republic: SCITEPRESS – Science and Technology Publications, 2023. Pp. 338-345.

Дата поступления
в редакцию: 04.04.2024

Дата принятия
к публикации: 10.05.2024