Синтез проверяющих последовательностей для недетерминированных автоматов относительно редукции

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

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

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

1. Введение

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

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

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

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

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

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

2. Основные определения и обозначения

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

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

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

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

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

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

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

Автомат P является *подавтоматом* автомата S, если $P \subseteq S$, $p_0 = s_0$ и $h_P \subseteq h_S$.

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

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

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

последовательность является разделяющей для любых двух различных состояний автомата.

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

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

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

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

В качестве примера рассмотрим автомат на рис. 1.

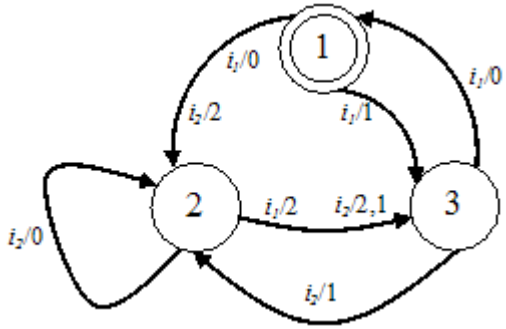

Рис. 1 Автомат-спецификация S.

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

Табл. 1 Выходные реакции на разделяющую последовательность.

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

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

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

3. Алгоритм построения адаптивной проверяющей последовательности

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

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

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

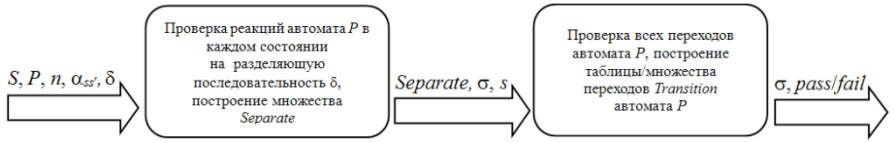

Рис. 2 Алгоритм построения адаптивной проверяющей последовательности

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

3.1. Построение множества Separate

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

Алгоритм 1 идентификации состояний проверяемого автомата

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

σ: = ε;
Separate: = $\varnothing$;
s: = s_0;
flag: = 0;

Шаг 1.

Пока *flag* = 0 выполнить:

{

Подать на автомат *P* последовательность δ;

σ: = σδ;

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

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

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

/*произошло зацикливание по разделяющей последовательности;*/

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

s: = *s*′;

}

Шаг 2.

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

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

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

{

Подать $\alpha_{ss'}$ на автомат *P*

σ: = σ $\alpha_{ss'}$;

}

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

Иначе Шаг 1.

3.2. Построение множества Transition

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

Алгоритм 2 проверки всех переходов предъявленного автомата *P*

Вход. Автомат-спецификация *S*, разделяющая последовательность δ и текущее состояние автомата после подачи первой части проверяющей последовательности на проверяемый автомат *P*, множество *Separate*;

Выход. Вердикт *pass*, если *P* есть редукция *S*, или вердикт *fail*, если *P* не есть редукция *S*; множество переходов автомата *P*;

Transition: = ∅;

Пока мощность множества *Transition* не равна произведению *n* на число входных символов выполнить:

{

Если в текущем состоянии *s* есть переход по входному символу *i*, не входящий в множество *Transition*, **то**

Подать на автомат *P* последовательность *i* δ;

σ: = σ *i* δ;

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

Иначе по множеству *Separate* определить тройку (*s*′, ρ, *s*′′) ∈ *Separate*;

Занести четверку (*s*, *i*, *o*, *s*′) в множество *Transition*;

s: = *s*′′;

Иначе по множеству *Transition* найти входную последовательность γ для перехода из текущего состояния *s* в состояние *s*′, в котором есть переход по входному символу *i*, не входящий в множество *Transition*;

/* Последовательность существует, т.к. по свойствам автомата-спецификации *S* редукциями *S* могут быть только автоматы с тем же числом состояний*/

Подать на проверяемый автомат последовательность γ;

σ: = σ γ;

s: = *s*′;

}

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

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

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

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

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

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

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

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

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

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

4. Иллюстрация алгоритма на примере

Рассмотрим работу алгоритма построения проверяющей последовательности на примере, приведенном на рисунке 1. Для данного автомата разделяющая последовательность δ: i_2i_1. В качестве реализации рассмотрим автомат на рис. 3

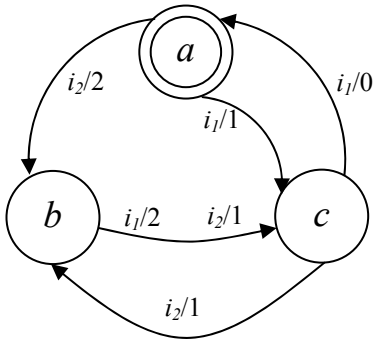

Рис. 3 Проверяемый автомат P, рассматриваемый как реализация автомата на рис. 1

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

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

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

Таблица 2.

s	ρ	*s′*
1	22	3
2	10	1
3	12	3

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

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

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

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

5. Заключение

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

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

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

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

Список литературы

[1]. Petrenko A. Adaptive Testing of Deterministic Implementations Specified by Nondeterministic FSMs / A. Petrenko, N. Yevtushenko // ICTSS, 2011. P. 162-178.

[2]. М.В. Ветрова. Разработка алгоритмов синтеза и тестирования конечно автоматных компенсаторов. Дисс. На соискание степени канд. техн. наук, Томский госуниверситет, 2004..

[3]. Н.Г. Кушик. Методы синтеза установочных и различающих экспериментов с недетерминированными автоматами. Дисс. На соискание степени канд. физ.-мат. наук, Томский госуниверситет, 2013.

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

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

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

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

References

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