ВЕСТНИК ТОМСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2008 Управление, вычислительная техника и информатика № 3(4)

ДИСКРЕТНЫЕ ФУНКЦИИ И АВТОМАТЫ

УДК 618.5:519.68

Ю.В. Михайлов, А.В. Коломеец

ПРОВЕРКА ПЕРЕХОДОВ В РАСШИРЕННОМ АВТОМАТЕ НА ОСНОВЕ СРЕЗОВ

В данной работе предлагается метод построения проверяющего теста для расширенного автомата на основе различающих последовательностей в специальном срезе автомата исходного расширенного автомата, т.е. в его упрощенной версии. Устанавливаются достаточные условия, при которых последовательности, различающие два состояния в срезе, различают эти состояния в исходном расширенном автомате.

Ключевые слова: *расширенный автомат*, *контекстно-свободный расширенный автомат*, *различающая последовательность.*

При построении тестов для телекоммуникационных протоколов очень важным является понятие различимости между состояниями спецификации. Для модели конечного автомата отношение различимости хорошо изучено. Однако для реальных протоколов соответствующий конечный автомат, если существует, имеет слишком много переходов, что затрудняет использование этой модели при построении тестов. Одной из модификаций конечного автомата является расширенный автомат; расширенный автомат содержит дополнительные переменные, и переходы снабжены условиями, при которых может выполниться данный переход. Известные методы построения тестов по расширенному автомату используют последовательности, различающие конфигурации в расширенном автомате. В данной работе мы предлагаем для построения таких различающих последовательностей использовать срезы расширенного автомата. Каждый срез, с одной стороны, имеет меньший размер, чем исходный расширенный автомат, а с другой – сохраняет свойства расширенного автомата, необходимые для построения последовательностей, различающих состояния и конфигурации расширенного автомата, т.е. эти срезы строятся таким образом, чтобы сохранить возможности построения полного проверяющего теста.

1. Основные определения

Под *расширенным автоматом* [1] понимается пятёрка $M = (S, X, Y, V, T)$, где S – непустое конечное множество состояний автомата, X – непустое множество входных символов, называемое входным алфавитом, Y – непустое множество выходных символов, называемое выходным алфавитом, V – конечное, возможно пустое множество контекстных переменных, T – множество переходов между состояниями из S. Каждый переход t из T – это семёрка (s, x, P, op, y, up, s'), где $s, s' \in S$ являются начальным и конечным состояниями перехода; $x \in X$ есть вход-

ной символ и $D_{inp\text{-}x}$ обозначает множество векторов, компонентами которых являются значения параметров, соответствующих входному символу x (далее – входные параметры); $y \in Y$ – выходной символ и $D_{out\text{-}y}$ обозначает множество векторов, компонентами которых являются значения параметров, соответствующих выходному символу y (далее – выходные параметры); P, op и up – функции, определенные над входными параметрами и контекстными переменными из V:

P: $D_{inp\text{-}x} \times D_V \rightarrow$ {истина, ложь} – предикат, где D_V – множество контекстных векторов;

op: $D_{inp\text{-}x} \times D_V \rightarrow D_{out\text{-}y}$ – функции для определения значений выходных параметров;

up: $D_{inp\text{-}x} \times D_V \rightarrow D_V$ – функции для определения значений контекстных переменных.

Параметризованным входным символом называется пара «входной символ x, входной вектор a из $D_{inp\text{-}x}$», т.е. пара (x, **a**).

Конфигурацией расширенного автомата M называется пара «состояние s, контекстный вектор v», то есть (s, **v**).

Переход называется *возбужденным* для конфигурации (s, **v**) и параметризованного входа (x, **ρ**), если предикат на переходе из состояния s под действием входного символа x принимает значение «Истина» для пары (**v**, **ρ**).

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

Рассмотрим расширенный автомат ES_1 на рис. 1. В автомате 3 состояния, два параметризированных входных символа $a(x)$ и $b(y)$, один непараметризированный входной символ c, два параметризированных выходных символа $O(z)$ и $F(w)$, два непараметризированных выходных символа R и NULL (нет выхода) и две контекстные переменные v и w.

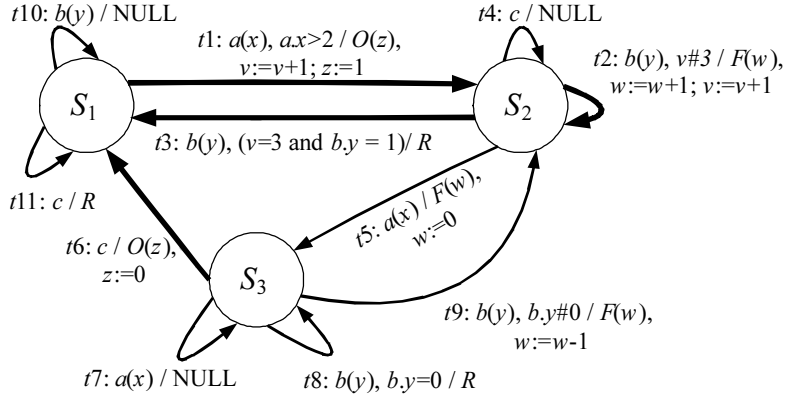

Рис. 1. Расширенный автомат ES_1

Состояния s_1 и s_2 расширенного автомата *ES различимы*, если существует параметризованная входная последовательность α, на которую автомат *ES* в состояниях s_1 и s_2 имеет различные выходные реакции, в противном случае состояния s_1 и s_2 называются *неразличимыми*.

Полностью определенные расширенные автоматы *ES* и *ET* называются *эквивалентными*, если их начальные состояния не различимы, т.е. для любой параметризованной входной последовательности α множества параметризованных выходных последовательностей автоматов совпадают. Если начальные состояния автоматов различимы, т.е. существует параметризованная входная последовательность α, такая, что выходные реакции *ES* и *ET* на α различны, то расширенные автоматы *ES* и *ET различимы* (последовательностью α), обозначение $EF \not\cong_{\alpha} ES$ или просто $EF \not\cong ES$.

Мы далее полагаем, что эталонное поведение проверяемой системы описано расширенным автоматом *ES*. Проверяемая система является реализацией эталонной системы, и в процессе реализации или функционирования проверяемой системы могли произойти ошибки, которые предполагаются постоянными во времени.

Рассмотрим переход $t = (s, x, P, op, y, up, s')$ эталонного автомата. Говорят, что на переходе t в проверяемой системе имеет место *ошибка перехода* [2], если конечное состояние перехода в проверяемом автомате (реализации) отличается от конечного состояния этого перехода в эталонном автомате. Иначе говоря, в проверяемом автомате вместо перехода t существует переход $(s, x, P, op, y, up, s'')$, в котором состояние $s' \neq s''$.

Переход $t = (s, x, P, op, y, up, s')$ имеет *выходную ошибку* [2], если выходной символ проверяемого автомата на данном переходе отличается от выходного символа y, т.е. в проверяемом автомате вместо перехода t существует переход $(s, x, P, op, y', up, s')$.

В данной работе на каждом шаге мы предлагаем строить фрагмент теста, который гарантированно обнаруживает ошибки только на одном переходе; этот переход называется *подозрительным*, остальные переходы называются *проверенными* переходами. Напомним, что мы рассматриваем только ошибки в финальном состоянии перехода и/или выходном символе.

Для построения полного проверяющего теста непосредственно по расширенному автомату мы предлагаем использовать срезы (т.е. уменьшенные образы) расширенного эталонного автомата *ES*; каждый из срезов должен сохранять, хотя бы частично, свойства, необходимые для построения полного теста.

2. Построение срезов расширенного автомата

При синтезе полного теста мы должны пройти в эталонном автомате все подозрительные (а иногда и проверенные переходы) и, кроме того, различить финальное состояние перехода с другими состояниями эталонного автомата [3]. Решение проблемы достижимости и различимости явно или неявно базируется на моделировании поведения расширенного автомата, сложность которого в значительной степени зависит от числа контекстных переменных. Поэтому на первом шаге мы уменьшаем число контекстных переменных, сохраняя свойство достижимости для состояний расширенного автомата.

2.1. Построение среза расширенного автомата с сохранением достижимости состояний

Пусть $V = \{v_1, v_2, ... v_n\}$ – множество контекстных переменных расширенного автомата *ES*. Переменная v_i *непосредственно зависит* от переменной v_j, если на некотором переходе функция определения значения контекстной переменной v_i зависит от переменной v_j. Переменная v_i *зависит* от переменной v_j, если существуют переменные $v_1, \ldots, v_k$, такие, что $v_i = v_1$, $v_j = v_k$ и переменная v_p непосредственно зависит от переменной v_{p+1}, $p = 1, \ldots, k-1$.

Для упрощения решения задачи достижимости в расширенном автомате мы предлагаем построить срез $Slice_R(ES)$, который сохранит свойства достижимости исходного расширенного автомата *ES*. Мы называем такой срез *R-срезом*. Для построения среза $Slice_R(ES)$ предлагается следующий алгоритм.

Алгоритм 2.1. [4] Построения *R*-среза расширенного автомата.

Вход: Расширенный автомат *ES*.

Выход: Срез $Slice_R(ES)$ автомата *ES*.

Шаг 1: Сокращаем множество контекстных переменных: переменная v_i, удаляется из множества контекстных переменных *V*, если ни одна из свободных переменных любого предиката не зависит от переменной v_i. Другими словами, мы удаляем все контекстные переменные, не влияющие на значение предикатов. Обозначим $V' \subseteq V$ сокращенное множество контекстных переменных.

Шаг 2: Все соотношения для определения значения контекстной переменной из множества $V \setminus V'$ удаляются из расширенного автомата.

Шаг 3: Если существует переход в расширенном автомате *ES*, на котором функция определения выходного параметра зависит от контекстной переменной из множества $V \setminus V'$, то данное соотношение удаляется из расширенного автомата. Соответствующему выходному параметру присваивается любое допустимое значение. Конец алгоритма.

Утверждение 1. Пусть *ES* – расширенный автомат и его срез $Slice_R(ES)$ получен по алгоритму 2.1. Если параметризованная входная последовательность α переводит расширенный автомат $Slice_R(ES)$ из начального состояния *s* в состояние *s'*, то последовательность α также переводит автомат *ES* из начального состояния *s* в состояние *s'*. Более того, если под действием параметризированного входного символа **α** в автомате $Slice_R(ES)$ выполним переход $t = (s,x,P,op,y',up,s')$, то соответствующий переход выполним в автомате *ES* под действием того же параметризированного входного символа.

2.2. Построение *FSM*-среза расширенного автомата

Для того чтобы упростить проблему различимости состояний в расширенном автомате, мы предлагаем построить срез $Slice_{FSM}(ES)$ с конечно-автоматным поведением, т.е. срез, который не содержит контекстных переменных и выходных параметров. Как известно, задачи достижимости и различимости для конечных автоматов решаются достаточно просто (по сравнению с расширенными автоматами). Мы называем такой срез *FSM*-срезом. Следуя [4], мы обозначаем такой срез как $Slice_{FSM}(ES)$; однако наш алгоритм позволяет построить такой срез с сохранением большего числа переходов исходного расширенного автомата.

Идея построения среза состоит в удалении переходов, на которых предикат зависит от контекстных переменных, т.е. переходов, выполнение которых зависит

от значений контекстных переменных. Тем не менее, некоторые переходы, имеющие предикаты, зависящие от контекстных переменных можно сохранить, воспользовавшись следующим свойством. Пусть, например, $P = P_1 \vee P_2$ и предикат P_1 не зависит от контекстных переменных. Тогда переход с предикатом P выполним, если выполним переход с предикатом P_1. В общем случае такая замена предиката возможна, если предикат P можно представить в виде суперпозиции предикатов P_1 и P_2, $P = f(P_1, P_2)$, из которых следует, что предикат P_1 зависит только от входных параметров и существует такое значение предиката P_1, при котором f существенно не зависит от значения предиката P_2. Соответственно предикат P можно заменить предикатом P_1.

Алгоритм 2.2. Построения *FSM*-среза расширенного автомата.

Вход: Расширенный автомат *ES.*

Выход: Конечно-автоматный срез $Slice_{FSM}(ES)$ автомата *ES.*

Шаг 1: Удалить из расширенного автомата *ES* все переходы, на которых предикат зависит только от контекстных переменных. Если предикат P можно представить в виде суперпозиции предикатов P_1 и P_2, $P = f(P_1, P_2)$, из которых следует, что предикат P_1 зависит только от входных параметров и существует такое значение предиката P_1, при котором f существенно не зависит от значения предиката P_2, то заменяем предикат P предикатом P_1.

Шаг 2: Удалить из расширенного автомата все контекстные переменные и функции вычисления контекстных переменных (*up*).

Шаг 3: Удалить из расширенного автомата все выходные параметры и соответствующие функции вычисления выходных параметров (*op*); полученный автомат обозначить $Slice_{FSM}(ES)$. Конец алгоритма.

По построению расширенный автомат $Slice_{FSM}(ES)$ не содержит контекстных переменных и выходных параметров, и соответственно можно считать, что каждый переход вида $t = (s,x,P,op,y,up,s')$, на котором предикат зависит от входных параметров, заменяется множеством переходов $(s,\mathbf{x},y,s')$ для всех параметризированных входных символов **x,** которые обращают предикат P в ИСТИНУ. Соответственно построенный срез имеет конечно-автоматное поведение; для детерминированного расширенного автомата соответствующий автомат является детерминированным, но в общем случае является частичным и не связным и более того не обязательно является конечным, так как число входных параметризированных символов может быть бесконечным. Однако такой срез имеет хорошие свойства, которые можно использовать при синтезе тестов для расширенного автомата *ES*. В частности, в таком срезе достаточно просто найти последовательность, которая переводит расширенный автомат *ES* из начального в заданное состояние (если такая последовательность существует в срезе $Slice_{FSM}(ES)$), а также различить две конфигурации $(s,\mathbf{v})$ и $(s',\mathbf{v}')$ исходного расширенного автомата, если состояния s и s' различимы в срезе $Slice_{FSM}(ES)$. Срез $Slice_{FSM}(ES_1)$ для расширенного автомата ES_1 на рис. 1 приведен на рис. 2. В автомате, представленном на рисунке 2, несмотря на то, что множество контекстных переменных пусто, так же как и множество выходных параметров, выполнение некоторых переходов зависит от предиката, определенного на множестве входных параметров.

Заметим, что *FSM*-срез расширенного автомата имеет конечно-автоматное поведение, однако в общем случае этот срез не является конечным автоматом. Более того, отношение переходов автомата явно не представлено четверками; в описании среза используются предикаты, зависящие от входных параметров. Таким об-

разом, *FSM*-срез, скорее, является расширенным автоматов с пустым множеством контекстных переменных и выходных параметров. Расширенный автомат с пустым множеством контекстных переменных и выходных параметров будем далее называть *контекстно-свободным* расширенным автоматом, и в следующих разделах мы исследуем задачи достижимости и различимости состояний для таких расширенных автоматов.

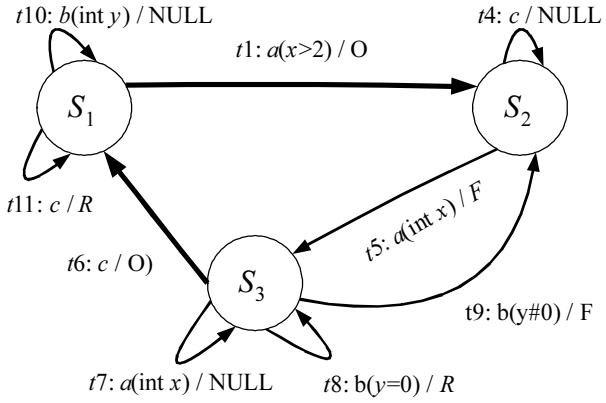

Рис. 2. $Slice_{FSM}(ES_1)$ расширенного автомата ES_1

2.3. Построение различающего автомата для двух контекстно-свободных расширенных автоматов

Пусть имеется два контекстно-свободных расширенных автомата M_1 и M_2, которые находятся в двух различных начальных состояниях s_i и s_j, обозначим их M_1/s_i и M_2/s_j соответственно. Автоматы M_1/s_i и M_2/s_j обладают одним входным и выходным алфавитами и множеством входных параметров (рис. 3).

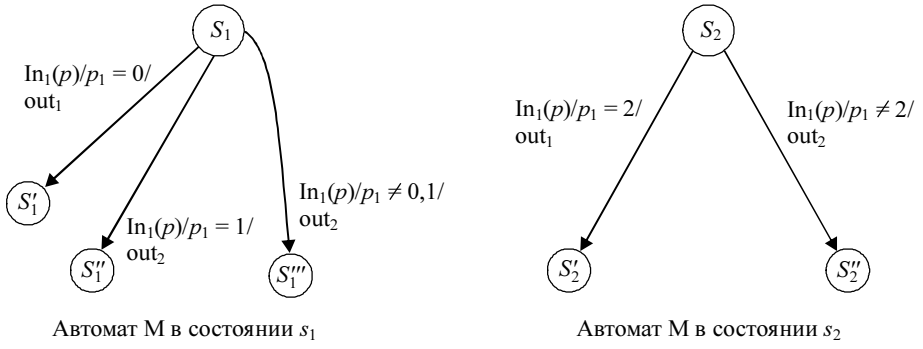

Рис. 3. Два контекстно-свободных расширенных автомата в начальных состояния s_1 и s_2

Введем понятие различающего автомата для двух контекстно-свободных расширенных автоматов M_1 и M_2 с начальными состояниями s и q (обозначение: M_1/s и M_2/q).

Различающим автоматом $(M_1/s \otimes M_2/q)/(s,q)$ контекстно-свободных расширенных автоматов M_1/s и M_2/q называется наибольший связный подавтомат расширенного автомата $<(S \times Q) \cup fail, X, Y \cup fail, T>$, где S – множество состояний автомата M_1, Q – множество состояний автомата M_2, $fail \notin S \times Q$ – специальное состояние и специальный выходной символ, X и Y – входной и выходной алфавиты, на которых определены автоматы M_1 и M_2, и множество переходов данного автомата определяется следующим образом.

Для каждой пары состояний $(s, q) \in S \times Q$ и для каждой пары переходов из состояний s и q по входному символу a, который определен в каждом из состояний s и q, с предикатами P и R, проверяем, является ли предикат $(P \,\&\, R)$ выполнимым, т.е. существует ли набор значений входных параметров, на котором предикат $(P \,\&\, R)$ принимает значение «Истина». Если такой набор ρ существует и выходной символ b на этих переходах один и тот же, то в автомате $(M_1/s \otimes M_2/q)/(s,q)$ есть переход $(s, q) \rightarrow a(\rho)/\, b \rightarrow (s', q')$.

Если выходные символы на переходах из состояний s и q по входному символу a, с предикатами P и R, не совпадают, то в различающем автомате существует переход $(s, q) \rightarrow a(\rho)/fail \rightarrow fail$.

В состоянии *fail* по всем входным символам можно добавить петлю с выходным символом *fail*.

Проверка выполнимости предиката и нахождения выполняющего набора входных параметров для ряда частных случаев обсуждается в следующем разделе.

На рис. 4 представлен пример автомата различающего автомата для фрагментов M/s_1 и M/s_2, показанных на рис. 3

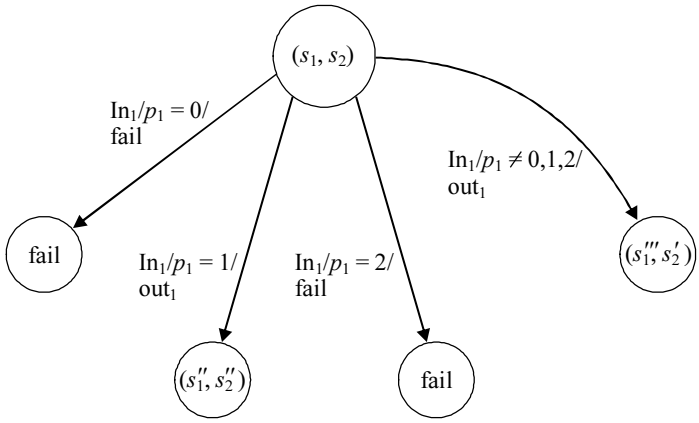

Рис. 4. Пример пересечения для среза $Slice_{FSM}(M)$

Для построения различающего автомата необходимо рассматривать каждый переход автомата M/s_1, находящегося в состоянии s_1, с каждым переходом автомата M/s_2, находящегося в состоянии s_2, по входному символу *in*. В худшем случае количество переходов равняется $m \cdot n$, где m – число таких переходов для автомата M/s_1, n –для M/s_2. Для нашего примера количество переходов сократилось до четырех, так как остальные переходы невозможно построить из-за невыполнимости конъюнкции соответствующих предикатов.

Теорема. Состояния s_i и $s_j \in S$ контекстно-свободного автомата M различимы тогда и только тогда, когда расширенный автомат $M/s_1 \otimes M/s_2$ содержит состояние *fail*. Любая параметризированная входная последовательность, ведущая из начального состояния в состояние *fail*, является различающей для состояний s_i и s_j.

Следствие (о существовании различающей последовательности для состояний на основе построения *FSM*-среза расширенного автомата). Пусть $Slice_{FSM}(M)$ есть *FSM*-срез расширенного автомата M. Состояния s_i и $s_j \in S$ автомата M различимы, если расширенный автомат $Slice_{FSM}(M)/s_1 \otimes Slice_{FSM}(M)/s_2$ содержит состояние *fail*. Любая параметризированная входная последовательность, ведущая из начального состояния в состояние *fail*, является различающей для состояний s_i и s_j.

Алгоритм 2.3. Построение различающих последовательностей для двух состояний расширенного автомата.

Вход: Расширенный автомат M, состояния s_i и s_j автомата M.

Выход: Различающая последовательность для состояний s_i и s_j, если состояния s_i и s_j различимы в автомате $Slice_{FSM}(M)$.

Шаг 1: Строим *FSM*-срез $Slice_{FSM}(M)$ расширенного автомата M.

Шаг 2: Строим различающий автомат для контекстно-свободных расширенных автоматов $Slice_{FSM}(M)\backslash s_i$ и $Slice_{FSM}(M)\backslash s_j$. Если различающий автомат содержит состояние *fail*, то находим путь, ведущий из начального состояния различающего автомата в состояние *fail*. Параметризированная входная последовательность σ, помечающая этот путь, является различающей последовательностью для состояний s_i и s_j. КОНЕЦ.

Если различающий автомат не содержит состояния *fail*, то состояния s_i и s_j не различимы в автомате $Slice_{FSM}(M)$. Конец алгоритма.

3. Алгоритм построения проверяющего теста для проверки ошибок переходов/выходов на выделенных переходах расширенного автомата

В данном разделе мы предлагаем алгоритм построения проверяющего теста в предположении, что ошибки переходов/выходов возможны только на одном заданном переходе расширенного автомата.

Алгоритм 3.1. Построение проверяющего теста для проверки ошибок переходов/выходов на выделенном переходе расширенного автомата.

Вход: Расширенный автомат *ES* и переход $t = (s, x, P, op, y, up, s')$.

Выход: Тест, проверяющий наличие ошибок перехода/выхода на переходе t (если такой тест можно построить на основе R-среза и среза $Slice_{FSM}(ES)$).

Шаг 1: Строим *FSM*-срез $Slice_{FSM}(ES)$ расширенного автомата *ES*, и R-срез $Slice_R(ES)$. Если *FSM*-срез не содержит перехода t, то конец алгоритма.

Шаг 2: Если в *FSM*-срезе состояние s достижимо из начального состояния по входной последовательности α и существует переход из состояния s под действием входного символа x, то Шаг 3.

Если состояние s недостижимо в *FSM*-срезе, то пытаемся построить последовательность, переводящую R-срез из начального состояния в состояние s, после которой выполним предикат перехода t. Если такая последовательность α найдена, то Шаг 3.

Если такая последовательность α не найдена, то конец алгоритма.

Шаг 3: На основе алгоритма 2.3 из раздела 2.3 проверяем, с какими состояниями различимо состояние s' в *FSM*-срезе. Строим множество различающих по-

следовательностей *DS*. Если состояние *s'* в *FSM*-срезе различимо с *k* состояниями из *n* состояний, то полагаем $p = k/n \cdot 100$.

Шаг 4: Строим тест *TS* = (α ***x*** *DS)*, где ***x*** – параметризированный входной символ, при котором предикат *P* принимает значение ИСТИНА; полнота теста для проверки заданного перехода равна *p*%. Конец алгоритма.

Данный алгоритм можно использовать для проверки одиночных ошибок на множестве выделенных переходов, проверяя непосредственно каждый переход в цикле. Если в результате алгоритма 3.1 тест не построен, то полагаем полноту проверки данного перехода равной 0. Тогда полнота теста будет усреднена по всем переходам и вычисляется по следующей формуле:

$$P = \frac{\sum_{i=1}^{m} p_i}{m},$$

где m – количество рассматриваемых (подозрительных) переходов.

Заключение

В данной работе предложен метод, позволяющий строить проверяющие тесты с гарантированной полнотой для одиночных неисправностей переходов/выходов на выделенных переходах расширенного автомата. Предложены алгоритмы, позволяющие упростить исходный расширенный автомат с сохранением свойств достижимости и различимости состояний, а также алгоритм построения различающего автомата, позволяющий различить два состояния расширенного автомата, который не содержит контекстных переменных и выходных параметров. Предложенный метод построения проверяющего теста не требует построения эквивалентного конечного автомата, что существенно повышает возможности построения проверяющих тестов данным методом, так как для большинства реальных систем, поведение которых описано расширенным автоматом, эквивалентный конечный автомат практически невозможно построить.

ЛИТЕРАТУРА

1. *Petrenko A.*, *Boroday S.*, *Groz R*. Confirming configurations in EFSM // IEEE TSE. 2004. V. 30/1. P. 29 – 42.
2. *El-Fakih K.*, *S. Prokopenko*, *N. Yevtushenko*, *G. Bochmann*. Fault diagnosis in extended Finite State Machine // Lecture Notes in Computer Science. 2003. P. 197 – 210.
3. *Petrenko A.*, *Yevtushenko N.*, *Bochmann G. A*. Fault models for testing in context // Proc. of the IFIP 1st Joint International Confeence FORTE/PSTV. Chapman & Hall, 1996. P. 163 – 178.
4. *El-Fakih K.*, *Kolomeez A.*, *Prokopenko S.*, *Yevtushenko N*. Extended finite state machine based test derivation driving by user defined faults // International Conference ICST, 2008.

Статья представлена кафедрой информационных технологий в исследовании дискретных структур радиофизического факультета Томского государственного университета, поступила в научную редакцию 10 марта 2008 г.