Тестирование системы автоматов с буферизацией сообщений

И.Б. Бурдонов <igor@ispras.ru>
А.С. Косачев <kos@ispras.ru>
Институт системного программирования РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25

Аннотация. Статья посвящена проблеме тестирования составных систем, компоненты которых моделируются конечными автоматами, а взаимодействие между ними – обменом сообщениями по симплексным каналам связи. Система описывается ориентированным графом связей, вершины которого соответствуют автоматам компонентов, а дуги – каналам связи. Предполагается выполненной следующая гипотеза о связях: граф связей статический, а отображаемая им структура связей не содержит ошибок. Автомат, находящийся в вершине графа, в каждом состоянии может принимать несколько сообщений по входным дугам (не более одного по каждой дуге) и посылать несколько сообщений по выходным дугам (не более одного по каждой дуге). Целью тестирования является покрытие переходов автоматов компонентов, которые достижимы при работе этих автоматов в системе. Предполагается, что при тестировании возможно наблюдение изменения состояний автоматов в вершинах графа и сообщений на дугах графа. Сначала рассматривается упрощённая модель системы, в которой циркулирует только одно сообщение. На её примере показывается, что гипотеза о связях позволяет существенно сократить время тестирования. Полное тестирование системы автоматов без учёта гипотезы о связях может потребовать число тестовых воздействий порядка произведения чисел состояний автоматов компонентов, а с учётом гипотезы о связях – порядка суммы этих чисел. При равном числе состояний всех автоматов это даёт экспоненциальное уменьшение числа тестовых воздействий. Затем рассматривается более общая модель, когда в системе может быть одновременно много сообщений, но не более одного на каждой дуге. Определяется композиция автоматов системы и показывается, при каких ограничениях на автоматы их композиция детерминирована. Для детерминированной композиции предлагается алгоритм генерации тестов, основанный на фильтрации тестов, генерируемых для покрытия всех переходов композиции. Тест отбрасывается, если он покрывает только такие переходы в компонентах системы, которые покрываются остающимися тестами. В заключение определяются направления дальнейших исследований.

Ключевые слова: ориентированные графы, покрытие графа, взаимодействующие автоматы, распределенные системы, тестирование, сети.

DOI: 10.15514/ISPRAS-2016-28(1)-7

Для цитирования: Бурдонов И.Б., Косачев А.С. Тестирование системы автоматов с буферизацией сообщений. Труды ИСП РАН, том 28, вып. 1, 2016 г., с. 103-130. DOI: 10.15514/ISPRAS-2016-28(1)-7

1. Введение

Большинство сложных, особенно распределённых, систем представляет собой набор взаимодействующих компонентов. В данной статье компоненты моделируются конечными автоматами, а взаимодействие – обменом сообщениями между автоматами. Мы будем предполагать, что структура связей между компонентами моделируется ориентированным графом (будем называть его *графом связей*), в вершинах которого находятся автоматы, а дуги соответствуют симплексным каналам передачи сообщений. Дугу можно понимать как очередь длины 1, буферизующую сообщение, передаваемое из автомата в начале дуги в автомат в конце дуги. Кроме *внутренних* дуг, то есть дуг, соединяющих автоматы между собой, существуют также *внешние* дуги, связывающие систему с её *окружением*. Если такая дуга ведёт из окружения в некоторый автомат, то будем называть её *внешней входной* дугой, а если дуга ведёт из автомата в окружение, то – *внешней выходной* дугой. При тестировании тест подменяет собой окружение: он передаёт сообщения в систему по внешним входным дугам и принимает от системы сообщения по внешним выходным дугам.

Мы будем считать, что граф связей статический, то есть не меняющийся в процессе работы системы. В этом случае система (также как её компоненты) может моделироваться конечным автоматом, получающимся из автоматов-компонентов с помощью подходящего оператора композиции, учитывающего граф связей. Частью состояния системы является набор состояний входящих в неё автоматов, поэтому число состояний системы не меньше произведения $n_1 n_2 ... n_k$, где n_i – число состояний i-го автомата, а k – число автоматов. Даже если учитывать только *достижимые состояния* системы, то есть те, которые достижимы из её начального состояния, то их число может иметь тот же порядок, что и произведение $n_1 n_2 ... n_k$.

Система работает правильно, если структура её связей правильна, и каждый автомат-компонент работает правильно. Обратное, вообще говоря, не верно, если требования к системе не однозначно определяют её структуру, например, функциональные требования к системе, связывающие получаемую от системы реакцию с последовательностью подаваемых в систему воздействий. В данной статье целью тестирования является покрытие переходов автоматов компонентов, которые достижимы при работе этих автоматов в системе. Поэтому, если в структуре связей нет ошибок (будем называть это *гипотезой о связях*), то есть граф связей автоматов совпадает с заданным, то такое тестирование системы сводится к проверке правильности переходов каждого автомата. Проблема в том, что каждый автомат может тестироваться только как часть системы. Это означает, что тест не имеет непосредственного доступа

к автомату-компоненту, и вынужден осуществлять тестовые воздействия с помощью сообщений, посылаемых по внешним входным дугам, которые ведут, быть может, в другие автоматы. Тестирование компонента такой системы похоже на тестирование в контексте ([1], [2], [3], [4]), когда этот компонент рассматривается как тестируемая система, а остальные – как контекст. Существенное отличие, однако, в том, что в таком контексте тоже могут быть ошибки, хотя, если верна гипотеза о связях, то только в компонентах, а не в структуре связей между ними. С другой стороны, такое тестирование может проверять работу сразу нескольких компонентов, через которые проходят сообщения.

Поскольку компонент тестируется как часть системы, не все переходы автомата компонента, которые можно проверить при автономном тестировании с прямым доступом к компоненту, можно проверить при тестировании системы. Поэтому речь должна идти только о *достижимых переходах* автоматов, то есть таких переходах, которые могут быть выполнены при работе автомата как части системы.

Тестирование автомата системы (получающегося композицией автоматов компонентов для заданного графа связей) обеспечивает проход по всем достижимым переходам автомата системы. При этом, конечно, проверяются все достижимые переходы автоматов компонентов, но, вообще говоря, делается много «лишней работы». Гипотеза о связях позволяет получить существенный выигрыш во времени тестирования. В следующем разделе статьи приведён пример, для которого соотношение времени тестирования системы без учёта и с учётом правильности графа связей такое же, как соотношение произведения $n_1 n_2 ... n_k$ (число состояний автомата системы) и суммы чисел состояний автоматов-компонентов $n_1+n_2+...+n_k$. В частности, если $n_1=n_2=...=n_k=n$, имеем соотношение n^k и nk, то есть для фиксированного числа k компонентов получаем экспоненциальное уменьшение времени тестирования.

В данной статье предлагается алгоритм построения набора тестов, который является полным (проверяет все переходы автоматов компонентов, достижимые при работе этих компонентов в системе) при выполнении двух условий: 1) верна гипотеза о связях, 2) система детерминирована. Дополнительно этот алгоритм определяет недостижимые переходы автоматов компонентов. Предполагается, во-первых, что нам известно, каким должен быть автомат каждого компонента (задан граф переходов автомата с точностью до изоморфизма) и именно это должно проверяться при тестировании. Во-вторых, тест может наблюдать как состояния автоматов системы, так и сообщения, передаваемые по дугам графа связей. Поскольку мы не налагаем никаких ограничений на связность графов переходов автоматов в вершинах, вообще говоря, полный набор тестов содержит более одного теста. Это означает, что требуется рестарт системы при переходе от одного теста к другому. Такие предположения могут быть оправданы,

например, при имитационном тестировании аппаратуры (simulation-based verification) (см. например, [5]).

Сначала, в разделе 2, изучается простейшая модель системы, в которой может циркулировать только одно сообщение. Даётся формальное определение автомата компонента и формулируются требования к автомату, обеспечивающие детерминизм системы. В разделе 3 модель усложняется: в системе может одновременно передаваться несколько сообщений, но не более одного сообщения по каждой дуге графа связей. Фактически, дуга представляет собой очередь сообщений длины 1. Обобщается формальное определение автомата компонента и формулируются требования к автомату, обеспечивающие детерминизм системы. В заключении намечаются направления дальнейших исследований.

2. Первая модель: только одно сообщение в системе

2.1. Определение модели

В этом разделе мы рассмотрим простейшую модель системы, в которой одновременно может быть не более одного сообщения; если в данный момент времени сообщение есть в системе, оно находится на одной из дуг. Если сообщение передаётся по внутренней дуге $a \to b$, то оно было послано автоматом, находящимся в начале дуги, в вершине a, и будет принято автоматом, находящимся в конце дуги, в вершине b. Если $a \to b$ – это внешняя входная дуга, то a – это окружение системы. Если $a \to b$ – это внешняя выходная дуга, то b – это окружение системы.

Дугу будем рассматривать как очередь сообщений длины 1. По дуге можно послать сообщение, если дуга *пуста* (на ней нет сообщений), и с дуги можно принять сообщение, если дуга не пуста (на ней есть сообщение). Принимая сообщение с дуги, автомат может сам послать любое сообщение, но только одно и только по одной из дуг, выходящих из его вершины, либо не посылать никаких сообщений. Автомат предполагается детерминированным.

Введём формальные определения и обозначения.

Автомат – это набор (S,X,Y,T,s_0), где

S – конечное множество *состояний* автомата,

X – множество *стимулов* (входных символов),

Y – множество *реакций* (выходных символов),

$T \subseteq S \times X \times Y \times S$ – конечное множество *переходов* автомата,

$s_0 \in S$ – *начальное состояние*.

Обозначим: $s \text{—}?x!y \to t \triangleq (s,x,y,t) \in T$, $s \text{—}?x!y \to \triangleq \exists t\, (s,x,y,t) \in T$.

Там, где это не приведёт к недоразумению, мы будем в неформальном тексте сам переход (s,x,y,t) обозначать стрелкой $s \text{—}?x!y \to t$. Состояние t будем

называть *постсостоянием* перехода. Если постсостояние несущественно, то переход *(s,x,y,t)* будем обозначать стрелкой $s—?x!y\rightarrow$.

В графе связей мы допускаем кратные дуги, для различения которых будем помечать их символами из алфавита *Z*. Для простоты будем считать, что все вершины перенумерованы *0,1,2,...,k*, где *k* – число вершин, в которых находятся автоматы, а номер *0* соответствует окружению.

Граф связей определяется как набор $G = (V,Z,E)$, где $V = \{0,1,...,k\}$ – конечное множество вершин, $E \subseteq (V\times Z\times V) \setminus (\{0\}\times Z\times\{0\})$ – конечное множество дуг (у окружения *0* нет дуг-петель). Дуга *(v,z,w)* задаётся начальной вершиной v, пометкой *z* и конечной вершиной *w*. Дуга *(0,z,w)* – внешняя входная дуга, *(v,z,0)* – внешняя выходная дуга. Обозначим:

$I_v = \{(a,z,v)|(a,z,v)\in E\}$ – множество дуг, заканчивающих в вершине *v*,

$O_v = \{(v,z,b)|(v,z,b)\in E\}$ – множество дуг, начинающихся в вершине *v*.

Если $v=0$, то I_0 – это множество внешних входных дуг, O_0 – это множество внешних выходных дуг. Множество внутренних дуг равно $E \setminus (I_0 \cup O_0)$. См. рис. 1.

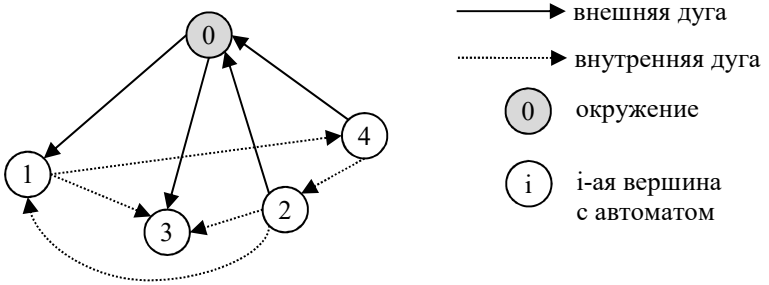

Рис. 1. Граф связей системы автоматов
Fig.1. Communication graph for a system of automatons

Обозначим через *M* множество всех возможных *сообщений*.

Система автоматов – это граф связей *G*, для которого каждой вершине *v* поставлен в соответствие автомат (S_v,X_v,Y_v,T_v,s_{v0}) такой, что

$X_v = \{(i,m) \mid i\in I_v \,\&\, m\in M\}$ и $Y_v = \{(j,m) \mid j\in O_v \,\&\, m\in M\} \cup \{\varnothing\}$.

Такой автомат будем называть автоматом в вершине *v*. Дугу, заканчивающуюся в вершине *v*, будем называть *входной дугой автомата*, а дугу, начинающуюся в вершине *v*, будем называть *выходной дугой автомата*. Стимул автомата – это пара *(i,m)*, где *i* – входная дуга автомата, а *m* – сообщение, принимаемое автоматом по этой дуге. Реакция автомата – это либо пара *(j,m)*, где *j* – выходная дуга автомата, а *m* – сообщение, посылаемое автоматом по этой дуге, либо пустое множество $\varnothing$, означающее отсутствие реакции (пустая реакция).

Определим условия выполнения перехода автомата $s—?x!y\rightarrow t$: 1) автомат находится в состоянии *s*, 2) если $x = (i,m)$, то на входной дуге *i* находится

сообщение *m*, 3) если $y=(j,m`)$, то выходная дуга *j* должна быть пуста. В результате выполнения перехода автомат оказывается в постсостоянии *t*, сообщение *m* принимается и входная дуга *i* становится пустой, если $y=(j,m`)$, то на выходную дугу *j* помещается сообщение *m`*.

Замечание 1: о дуге-петле. В рассматриваемой модели в системе имеется не более одного сообщения. Поэтому переход $s—?x!y\rightarrow t$, где $x = (i,m)$ и $y=(j,m`)$, не выполняется из-за того, что выходная дуга *j* занята, только в том случае, когда $i=j$, то есть сообщение должно быть послано по той же дуге-петле, с которой принимается сообщение. Такой переход (когда $i=j$) никогда не может быть выполнен, и его можно удалить из автомата.

Будем говорить, что автомат *детерминирован*, если в любой момент времени может выполниться не более одного перехода. В этой модели это означает, что состояние и стимул однозначно определяют реакцию и постсостояние перехода: $\forall s,x,y,y`,t,t` \; (s—?x!y\rightarrow t \,\&\, s—?x!y`\rightarrow t` \Rightarrow y=y` \,\&\, t=t`)$. Далее в этом разделе мы будем рассматривать только детерминированные автоматы.

Состоянием системы является набор состояний её автоматов $s_1,s_2,...,s_k$, а также указание, имеется ли какое-либо сообщение на дуге и, если имеется, то какое именно и на какой дуге. Формально состояние системы – это набор $(s_1,s_2,...,s_k,(e,m))$, если на дуге *e* есть сообщение *m*, или набор $(s_1,s_2,...,s_k,\varnothing)$, если нет сообщений на дугах. *Финальным* состоянием системы назовём такое её состояние, когда на дугах нет сообщений, то есть состояние вида $(s_1,s_2,...,s_k,\varnothing)$. Начальным состоянием системы будем считать финальное состояние, в котором каждый автомат находится в своём начальном состоянии, то есть состояние $(s_{10},s_{20},...,s_{k0},\varnothing)$.

Тестирование такой системы выполняется как подача в систему внешних стимулов, то есть посылка сообщений из окружения по внешним входным дугам. Для того чтобы оставаться в рамках первой модели (не более одного сообщения в системе), на работу окружения (теста – при тестировании) наложим ограничение: окружение может подавать внешний стимул только тогда, когда система находится в финальном состоянии (в частности, в начальном состоянии), и только один внешний стимул, то есть только одно сообщение по одной внешней входной дуге. Если сообщение в системе находится на внешней выходной дуге, то такое состояние нефинально, но окружение может принять это сообщение, освободив дугу и переведя систему в финальное состояние, а потом посылать следующий стимул. Однако окружение может подать внешний стимул и в том случае, когда система переходит в финальное состояние из-за того, что некоторый автомат принимает стимул, но не выдаёт никакой реакции. Очевидно, что для детерминированных автоматов в вершинах такое ограничение на поведение окружения гарантирует, что система не выйдет за пределы первой модели, то есть в системе будет циркулировать не более одного сообщения.

2.2. Композиция автоматов системы

Определим композицию автоматов при заданном графе связей, то есть автомат, отражающий работу системы в целом. Входными дугами такого автомата системы будут внешние входные дуги из I_0, а выходными дугами – внешние выходные дуги из O_0. Состояния системы описаны выше. Переходы делятся на три категории: 1) *переходы по стимулам* (без выдачи реакции) вида $s—?x!\varnothing\rightarrow t$, 2) *переходы по реакциям* (без приёма стимулов), которые будем понимать как переходы по пустому стимулу и обозначать $s—?\varnothing!y\rightarrow t$, 3) *внутренние переходы* (без приёма стимулов и без выдачи реакций), которые мы будем понимать как переходы по пустому стимулу и пустой реакции вида $s—?\varnothing!\varnothing\rightarrow t$, которые для краткости будем обозначать $s\rightarrow t$.

Определим переходы формально:

1. В финальном состоянии $s=(s_1,s_2,...,s_k,\varnothing)$ окружение может послать по любой внешней входной дуге любое сообщение. Если по дуге $(0,z,v)$ посылается сообщение m, то посылается непустой внешний стимул $x=((0,z,v),m)$. В автомате системы имеется переход $s—?x!\varnothing\rightarrow t$, где $t=(s_1,s_2,...,s_k,x)$ нефинальное состояние.
2. Пусть есть нефинальное состояние $s=(s_1,s_2,...,s_k,((v,z,w),m))$, когда на внешней входной или внутренней дуге (v,z,w) находится сообщение m, то есть $w\neq 0$. Пусть автомат в вершине w находится в состоянии s_w. Обозначим $x=((v,z,w),m)$.
   2.1. Пусть в состоянии s_w имеется переход по приёму стимула x, то есть переход вида $s_w—?x!y\rightarrow t_w$. Если реакция отсутствует, то есть $y=\varnothing$, то в автомате системы имеется внутренний переход $s\rightarrow t$, где $t=(s_1,s_2,...,s_{w-1},t_w,s_{w+1},...,s_k,\varnothing)$ финальное состояние. Если реакция $y=((w,z,w`),m`)$, то в автомате системы имеется внутренний переход $s\rightarrow t`$, где $t`=(s_1,s_2,...,s_{w-1},t_w,s_{w+1},...,s_k,y)$ нефинальное состояние. В состоянии s будет только один переход.
   2.2. Пусть в состоянии s_w нет перехода по стимулу x, то есть, нет перехода вида $s_w—?x!y\rightarrow t_w$. Тогда в состоянии s нет переходов.
3. Пусть есть нефинальное состояние $s=(s_1,s_2,...,s_k,((v,z,0),m))$, когда на внешней выходной дуге $(v,z,0)$ находится сообщение m. Обозначим $y=((v,z,0),m)$. В состоянии s ни один автомат не выполняет перехода, окружение не может послать стимул, так как состояние не финально. Единственное, что может произойти, это приём окружением непустой внешней реакции y, то есть сообщения m, находящегося на дуге $(v,z,0)$. В автомате системы это изображается переходом $s—?\varnothing!y\rightarrow t$, где $t=(s_1,s_2,...,s_k,\varnothing)$ финальное состояние.

Будем говорить, что композиция автоматов системы *детерминирована*, если в каждом её состоянии определено не более одного перехода по каждому стимулу, включая стимул $\varnothing$.

Утверждение 1: В первой модели композиция детерминированных автоматов детерминирована.

Доказательство: Из формального определения переходов композиции следует (рис. 2): 1) Из финального состояния ведут только переходы $s—?x!\varnothing\rightarrow t$ по непустым стимулам без выдачи реакций – не более одного перехода по каждому стимулу. Постсостояние такого перехода нефинально. 2) Из нефинального состояния ведёт не более одного перехода: либо внутренний переход $s\rightarrow t$ с финальным или нефинальным постсостоянием, либо переход $s—?\varnothing!y\rightarrow t$ без приёма стимула с выдачей реакции и финальным постсостоянием.

Утверждение доказано.

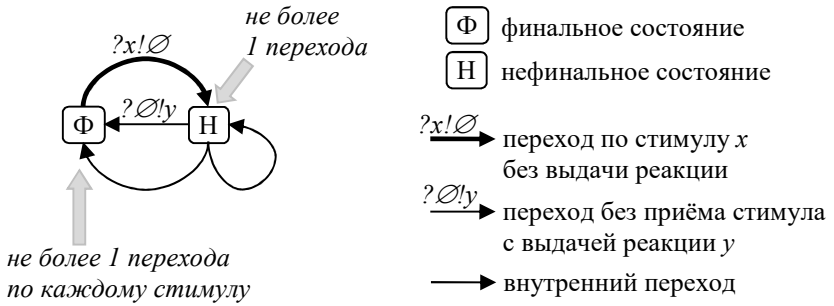

Рис. 2. Типы состояний и переходов автомата системы
Fig. 2. Types of states and transitions of automaton system

Тупиком назовём нефинальное состояние системы, из которого нет перехода. *Состоянием зацикливания* назовём нефинальное состояние системы, расположенное на цикле переходов, не проходящем через финальные состояния. Очевидно, все переходы такого цикла внутренние. На рис. 3 изображены возможные цепочки переходов после приёма системой (непустого) внешнего стимула.

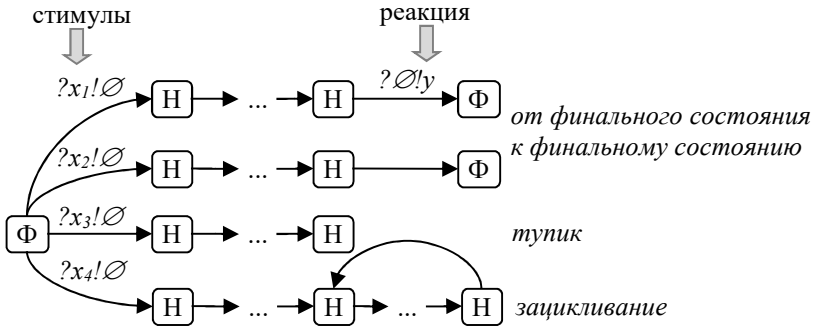

Рис. 3. Типы цепочек переходов в системе после приёма стимула
Fig.3. Types of transitions chains in the system after receiving a stimulus

Наличие пустого стимула, то есть дополнительных внутренних переходов и переходов по выдаче реакции без приёма стимула, отличает автомат системы от того типа автомата, который мы определили выше как автомат в вершине. Однако автомат системы A может быть с помощью следующей процедуры *финализации* f преобразован к автомату $f(A)$ того же типа, что автомат в вершине:

1) Цепочка переходов $s—?x_1!\varnothing\rightarrow\rightarrow...\rightarrow—?\varnothing!y\rightarrow t_1$, ведущая из финального состояния s в финальное состояние t_1 заменяется на один переход $s—?x_1!y\rightarrow t_1$.
2) Цепочка переходов $s—?x_2!\varnothing\rightarrow\rightarrow...\rightarrow t_2$, ведущая из финального состояния s в финальное состояние t_2 заменяется на один переход $s—?x_2!\varnothing\rightarrow t_2$.
3) Цепочка переходов $s—?x_3!\varnothing\rightarrow\rightarrow...\rightarrow t_3$, ведущая из финального состояния s в тупиковое нефинальное состояние t_3 заменяется на один переход $s—?x_3!\varnothing\rightarrow t_3$.
4) Цепочка переходов $s—?x_4!\varnothing\rightarrow\rightarrow...\rightarrow t_4\rightarrow...\rightarrow t_4$, ведущая из финального состояния s в нефинальное состояние зацикливания t_4 с однократным проходом по циклу заменяется на один переход $s—?x_4!\varnothing\rightarrow t_4$. Более строго: в последовательности состояний этой цепочки переход каждое состояние, кроме t_4, встречается по одному разу, а состояние t_4 два раза.

Состояния t_1 и t_2 финальные, а состояния t_3 и t_4 после финализации будут тупиковыми, т.е. в них нет никаких переходов. Автомат A будем называть *нефинальной композицией* автоматов компонентов, а автомат $f(A)$ – *финальной композицией*. Будем говорить, что два автомата *эквивалентны*, если любой набор тестов, покрывающий все достижимые переходы одного автомата, покрывает все достижимые переходы другого автомата.

Утверждение 2: Финальная композиция $f(A)$ детерминированных автоматов детерминирована и эквивалентна нефинальной композиции A.

Доказательство: Поскольку в нефинальной композиции A из финального состояния выходит не более одного перехода по каждому стимулу, это же будет и в автомате $f(A)$. В автомате $f(A)$ нефинальные состояния автомата A, оставшиеся достижимыми после финализации, являются терминальными. Поэтому финальная композиция $f(A)$ детерминирована. Из нефинального состояния нефинальной композиции выходит не более одного перехода. Поэтому при тестировании проход какой-либо цепочки переходов в автомате A из тех, что указаны в процедуре финализации, вызовет в автомате $f(A)$ проход соответствующего перехода. И наоборот, проход перехода в автомате $f(A)$ соответствует проходу соответствующей цепочки переходов в автомате A. Следовательно, A и $f(A)$ эквивалентны.

Утверждение доказано.

2.3. Генерация тестов

Целью тестирования рассматриваемой системы автоматов является покрытие всех достижимых переходов автоматов компонентов системы. Иногда приходится различать тест как конечную последовательность тестовых воздействий на тестируемую систему и тест как программу, реализующую эти тестовые воздействия, или модель такой программы. Там, где по контексту не ясно, что имеется в виду, мы будем в первом случае писать *тестовая последовательность*. В первой модели тестовая последовательность – это последовательность стимулов, подаваемых из теста (подменяющего собой окружение) в тестируемую систему.

Стимул можно подать в систему только в финальном состоянии. Если стимул приводит систему в нефинальное состояние, при котором сообщение находится на внешней выходной дуге, тест должен сначала принять это сообщение, а потом подать следующий стимул. Поскольку такие действия как приём тестом сообщения с внешней выходной дуги очевидны и обязательны, мы опускаем указания на них в тестовой последовательности.

Какие проверки выполняются при прогоне теста? В финальной композиции переходу $s—?x!y\rightarrow t$, где $x\neq\varnothing$, соответствует цепочка переходов в нефинальной композиции. Первый в цепочке переход имеет вид $s—?x!\varnothing\rightarrow t`$ и означает для $x=(i,m)$ размещение тестом сообщения $m$ на внешней входной дуге $i$. Последний в цепочке переход имеет вид $s`—?\varnothing!y\rightarrow t$ и для $y=(j,m`)$ означает передачу сообщения m с внешней выходной дуги j в тест. Остальные переходы внутренние и имеют вид $s``\rightarrow t``$, только такие переходы нужно проверять, поскольку только их выполнение означает выполнение перехода в каком-либо автомате-компоненте (но не более, чем в одном). Проверяется, выполнил ли этот автомат какой-либо переход или нет, и правильно ли это. Если автомат выполнил переход (и это правильно), то проверяется, правильную ли реакцию автомат выдал, и правильно ли изменилось его состояние.

Прогон теста покрывает некоторое множество переходов автоматов компонентов системы. Поскольку система детерминирована, это множество одно и то же при разных прогонах данного теста (с рестартом между прогонами), поэтому тест достаточно прогонять один раз. Мы будем рассматривать только конечные наборы тестов, после завершения прогона одного теста выполняется рестарт системы и прогоняется следующий тест из набора. Набор тестов покрывает множество переходов автоматов компонентов, которое является объединением множеств переходов автоматов компонентов, покрываемых тестами из этого набора. Набор тестов будем называть *полным*, если он покрывает все переходы автоматов компонентов, достижимые при работе этих компонентов в системе. Ставится задача генерации полного набора тестов.

Для решения этой задачи мы предлагаем использовать любой алгоритм генерации полного набора тестов для одного автомата. Таких алгоритмов предложено довольно много, по сути, они сводятся к построению набора маршрутов, покрывающих граф переходов автомата, достижимых из его начального состояния (см., например, [6]). В качестве такого автомата для наших целей выбирается автомат системы, получаемый с помощью описанной в предыдущем подразделе композиции автоматов. Понятно, что покрывая все достижимые переходы композиционного автомата системы, мы покрываем все достижимые переходы автоматов компонентов. Однако такой набор тестов может оказаться сильно избыточным для решения нашей задачи: покрытие всех достижимых переходов автоматов компонентов не обязательно требует покрытия всех достижимых переходов композиционного автомата системы.

Поэтому предлагается в процессе генерации полного набора тестов для композиционного автомата системы применять *процедуру фильтрации*, которая будет отбрасывать «лишние» тесты. Эта процедура работает следующим образом. С самого начала создаётся пустое множество T генерируемого набора тестов и множество P непокрытых переходов автоматов компонентов, которое сначала равно множеству всех переходов всех автоматов компонентов. Когда генерируется очередной i-ый тест T_i для композиционного автомата системы, вычисляется множество P_i переходов автоматов компонентов, покрываемое этим тестом. Тесту соответствует маршрут в композиционном автомате. Если рассматривается нефинальная композиция, то каждому внутреннему переходу этого маршрута соответствует один переход одного из автоматов компонентов; множество таких переходов и есть множество P_i. Напомним, что при переходе по внешнему стимулу или внешней реакции в автоматах компонентов никаких переходов не происходит. Если рассматривается финальная композиция, то каждому переходу этого маршрута соответствует множество переходов некоторых автоматов компонентов; объединение этих множеств и есть множество P_i. Далее алгоритм фильтрации проверяет, покрывает ли i-ый тест какой-либо новый, ещё не покрытый переход какого-либо автомата компонента. Если $P_i \cap P = \varnothing$, то никаких новых переходов i-ый тест не покрывает, и он отбрасывается. В противном случае тест добавляется к набору тестов $T := T \cup \{T_i\}$, а из множества непокрытых переходов удаляются новые переходы $P := P \setminus P_i$. После того как все тесты сгенерированы и отфильтрованы, получившееся множество T является полным набором тестов, а множество P – множеством недостижимых переходов автоматов компонентов.

Замечание 2: о рестарте. Необходимость в рестарте возникает из-за отсутствия сильной связности графа переходов композиции. Можно выделить два особых случая, когда прогон теста заканчивается в терминальном состоянии композиции. 1) Состояние тупика, когда сообщение «зависает» на внешней входной или внутренней дуге. «Убрать» это сообщение можно только рестартом. 2) Зацикливание, когда сообщение бесконечно циркулирует по

внутренним дугам, что, начиная с какого-то момента времени, не приводит к увеличению множества покрытых переходов автоматов компонентов. «Убрать» это зацикливание можно только рестартом.

2.4. Пример

Утверждение 3: Для любых чисел состояний автоматов компонентов $n_1,n_2,...,n_k$ существует такая система, что число внешних стимулов, которые надо дать для покрытия всех достижимых переходов композиции, равно $\boldsymbol{\Omega}(n_1 n_2 ... n_k)$, а минимальное число внешних стимулов, которых достаточно для покрытия всех достижимых переходов автоматов компонентов, равно $\boldsymbol{O}(n_1+n_2+...+n_k)$.

Доказательство: Рассмотрим систему, изображённую на рис. 4.

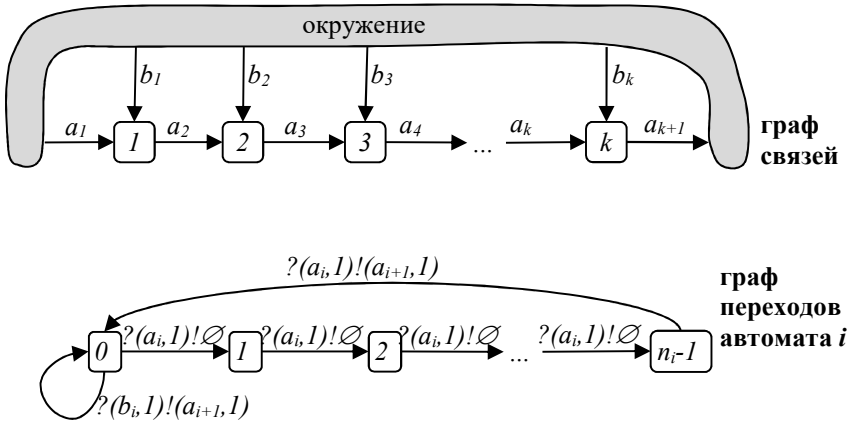

Рис. 4. Пример 1
Fig. 4. Example 1

Здесь имеется единственное сообщение *1*, $M = \{1\}$. Все автоматы в вершинах однотипные и различаются числом состояний n_i. Автомат в вершине i имеет две входные дуги, обозначенные a_i и b_i, и одну выходную дугу a_{i+1}. Соответственно, имеются два стимула $(a_i,1)$ и $(b_i,1)$ и единственная непустая реакция – $(a_{i+1},1)$. Автомат i в любом состоянии $j \neq n_i-1$, получая по входной дуге a_i сообщение *1*, переходит в следующее состояние $j+1$ без выдачи реакции. В последнем состоянии n_i-1 автомат, получая по входной дуге a_i сообщение *1*, переходит в начальное состояние *0* с посылкой сообщения *1* по единственной выходной дуге a_{i+1}. Тем самым, автомат i на каждую порцию из принятых им n_i сообщений по входной дуге a_i посылает одно сообщение следующему автомату $i+1$, или окружению, если $i=k$. Кроме того, имеется переход-петля в состоянии *0* по приёму сообщения по дуге b_i с выдачей сообщения следующему автомату $i+1$, или окружению, если $i=k$.

В финальной композиции нет тупиков и зацикливания, все состояния финальные и имеют вид $s=(s_1,s_2,...,s_k,\varnothing)$, где состояние i-ого автомата $s_i=0..n_{i-1}$. Начальное состояние $s_0=(0,0,...,0,\varnothing)$. Обозначим $s_{max}=(n_1\text{-}1,n_2\text{-}1,...,n_k\text{-}1,\varnothing)$.

Финальное состояние s можно понимать как пару $s=(s^{\sim},\varnothing)$, где $s^{\sim}$ – число, записанное в позиционной системе счисления слева направо от младшей позиции *1* к старшей позиции k: *1,2,...k*, и n_i – основание системы счисления в позиции i.

Рассмотрим в финальной композиции переходы по приёму стимула $(a_1,1)$, то есть по приёму сообщения по внешней входной дуге a_1, ведущей в автомат *1*. Для каждого состояния $s\neq s_{max}$ есть переход $(s^{\sim},\varnothing)—?(a_1,1)!\varnothing\rightarrow(s^{\sim}+1,\varnothing)$, а также есть переход $(s_{max}{}^{\sim},\varnothing)—?(a_1,1)!(a_{k+1},1)\rightarrow(s_0{}^{\sim},\varnothing)$. Это не все переходы композиции, но число таких переходов равно произведению $n_1n_2...n_k$. Следовательно, число переходов в финальной композиции не менее произведения $n_1n_2...n_k$. Поскольку в финальной композиции один внешний стимул вызывает ровно один переход, число внешних стимулов, которые надо дать для покрытия композиции, равно $\boldsymbol{\Omega}(n_1n_2...n_k)$.

Как наиболее быстро покрыть все переходы автомата i? Для этого достаточно 1) n_i раз послать сообщение по дуге a_i и 2) один раз по дуге b_i. Для автомата *0* пункт 1 тест может выполнить непосредственно, поскольку дуга a_1 внешняя. Для автомата $i>1$ пункт 1 можно выполнить, посылая n_i раз сообщение по внешней дуге b_{i-1} в предыдущий автомат $i\text{-}1$. Пункт 2 тест также может выполнить непосредственно, поскольку дуга b_i внешняя. В целом получается следующий тест:

1. n_1 раз посылается сообщение в автомат *1* по внешней дуге a_1,
   автомат *1* проходит цикл переходов по состояниям $0,...,n_1\text{-}1,0$, во время последнего перехода будет послано сообщение по дуге a_2, при получении этого сообщения автомат *2* переходит из состояния *0* в состояние *1*, остальные автоматы находятся в состоянии *0*;
2. $n_2\text{-}1$ раз посылается сообщение в автомат *1* по внешней дуге b_1,
   автомат *1* проходит $n_2\text{-}1$ раз по петле в состоянии *0*, каждый раз посылая сообщение по дуге a_2, при получении всех этих сообщений автомат *2* проходит цепочку переходов из состояния *1* в состояние *0*, при последнем переходе посылается сообщение по дуге a_3, при получении этого сообщения автомат *3* переходит в состояние *1*, остальные автоматы остаются в состоянии *0*;
3. $n_3\text{-}1$ раз посылается сообщение в автомат *2* по внешней дуге b_2,
   автомат *2* проходит $n_3\text{-}1$ раз по петле в состоянии *0*, каждый раз посылая сообщение по дуге a_3, при получении всех этих сообщений автомат *3* проходит цепочку переходов из состояния *1* в состояние *0*, при последнем переходе посылается сообщение по дуге a_4, при получении этого сообщения автомат *4* переходит в состояние *1*, остальные автоматы остаются в состоянии *0*;

...

k-1. $n_{k-1}\text{-}1$ раз посылается сообщение в автомат $k\text{-}2$ по внешней дуге b_{k-2},
   автомат $k\text{-}2$ проходит $n_{k-1}\text{-}1$ раз по петле в состоянии *0*, каждый раз посылая сообщение по дуге a_{k-1}, при получении всех этих сообщений автомат $k\text{-}1$ проходит цепочку переходов из состояния *1* в состояние *0*, при последнем переходе посылается сообщение по дуге a_k, при получении этого сообщения автомат k переходит в состояние *1*, остальные автоматы остаются в состоянии *0*;

k. $n_k\text{-}1$ раз посылается сообщение в автомат $k\text{-}1$ по внешней дуге b_{k-1},
   автомат $k\text{-}1$ проходит $n_k\text{-}1$ раз по петле в состоянии *0*, каждый раз посылая сообщение по дуге a_k, при получении всех этих сообщений автомат k проходит цепочку переходов из состояния *1* в состояние *0*, при последнем переходе посылается внешнее сообщение по дуге a_{k+1}, которое принимается тестом, остальные автоматы остаются в состоянии *0*;

k+1. *1* раз посылается сообщение в автомат k по внешней дуге b_k,
   автомат k проходит петлю в состоянии *0* и по внешней выходной дуге a_{k+1} посылает сообщение.

Длина тестовой последовательности $n_1+(n_2\text{-}1)+...+(n_k\text{-}1)+1 = (n_1+...+n_k)\text{-}k+2 = \boldsymbol{O}(n_1+...+n_k)$.

Утверждение доказано.

3. Вторая модель: обобщение автомата в вершине

3.1. Определение модели

В этом разделе мы сохраняем предположения о дуге (дуга реализует очередь сообщений длины 1), но обобщаем понятие автомата в вершине. Такой обобщённый автомат может принимать одновременно несколько сообщений по нескольким (необязательно всем) входным дугам, но не более одного сообщения по каждой дуге, и посылать несколько сообщений по нескольким (необязательно всем) выходным дугам, но не более одного сообщения по каждой дуге.

Дадим формальное определение автомата в вершине. Автомат в вершине такой же, как в разделе 2, но с изменёнными множествами стимулов и реакций, поскольку стимул и реакция теперь могут содержать несколько сообщений принимаемых или посылаемых по нескольким дугам. Формально для графа связей $G=(V,Z,E)$ и множества сообщений M автомат в вершине v – это такой автомат (S_v,X_v,Y_v,T_v,s_{v0}), что

$X_v = \{x \mid \exists f\, x \subseteq f \,\&\, f\in M^{I_v}\}$,

$Y_v = \{y \mid \exists f\, y \subseteq f \,\&\, f\in M^{O_v}\}$, где через A^B обозначено множество всех отображений из B в A.

Стимул автомата – это частично определённое отображение $x:I_v \to M$, которое каждой входной дуге $i \in \mathbf{Dom}(x)$ ставит в соответствие сообщение $x(i)$, принимаемое по этой дуге. Реакция автомата – это частично определённое отображение $y:O_v \to M$, которое каждой выходной дуге $j \in \mathbf{Dom}(y)$ ставит в соответствие сообщение $y(j)$, посылаемое по этой дуге.

Для описания состояния входных и выходных дуг автомата в вершине v введём два частично-определённых отображения $x_v^{\#}:I_v \to M$ и $y_v^{\#}:O_v \to \{M\}$. Первое отображение каждой непустой входной дуге $i \in I_v$ ставит в соответствие сообщение $m \in M$, находящееся на этой дуге. Второе отображение каждой пустой выходной дуге $j \in O_v$ ставит в соответствие множество сообщений M. Эти отображения $x_v^{\#}$ и $y_v^{\#}$ порождают множества *потенциальных* стимулов и *потенциальных* реакций, которые автомат может, соответственно, принять и послать при данном состоянии входных и выходных дуг, если, конечно, в текущем состоянии автомата определены соответствующие переходы:

$X_v^{\#} = \{x:I_v \to M \mid \mathbf{Dom}(x) \subseteq \mathbf{Dom}(x_v^{\#}) \ \& \ \forall i \in \mathbf{Dom}(x)\ x(i)=x_v^{\#}(i)\}$,

$Y_v^{\#} = \{y:O_v \to M \mid \mathbf{Dom}(y) \subseteq \mathbf{Dom}(y_v^{\#})\}$.

Теперь можно формально определить условие выполнения перехода $s \text{—}?x!y \to t$: $x \in X_v^{\#} \ \& \ y \in Y_v^{\#}$.

3.2. Детерминированный автомат

Для того чтобы система автоматов была детерминирована, потребуем детерминированности каждого из этих автоматов. Прежде всего, как и для первой модели, состояние и стимул должны однозначно определять реакцию и постсостояние перехода:

1) $\forall s,x,x`,y,y`,t,t` \quad s \text{—}?x!y \to t \ \& \ s \text{—}?x!y` \to t` \Rightarrow y=y` \ \& \ t=t`$.

В отличие от первой модели во второй модели распределение сообщений по входным дугам автомата не однозначно определяет стимул, который может принять автомат, поскольку автомат может не принимать сообщения с некоторых входных дуг, даже если дуги не пусты. Например, если на входных дугах i_1 и i_2 находятся сообщения m_1 и m_2, соответственно, то может быть принят любой из стимулов $\{(i_1,m_1),(i_2,m_2)\}$, $\{(i_1,m_1)\}$, $\{(i_2,m_2)\}$ или $\varnothing$. Если в автомате в текущем состоянии есть переходы хотя бы по двум из этих стимулов, то поведение автомата недетерминировано: может быть выбран приём любого из этих стимулов.

Будем говорить, что два стимула x и $x`$ *совместимы*, и обозначать $x \approx x`$, если при некоторых сообщениях на входных дугах автомата, т.е. при некотором отображении $x_v^{\#}$, автомат может принять любой из этих стимулов, т.е. $x \in X_v^{\#}$ и $x` \in X_v^{\#}$. Формально: $x \approx x` \triangleq \forall i \in \mathbf{Dom}(x) \cap \mathbf{Dom}(x`)\ x(i) = x`(i)$. Заметим, что все стимулы во множестве $X_v^{\#}$ совместимы друг с другом.

Отсюда вытекает второе требование детерминизма автомата:

2) Нет переходов из одного состояния по разным совместимым стимулам:

$\forall s,x,x`,y,y` \quad x \not\approx x` \ \& \ x \approx x` \Rightarrow \neg(s \text{—}?x!y \to \ \& \ s \text{—}?x`!y` \to)$.

Утверждение 4: Если выполнены оба требования детерминизма, то состояние s_v автомата в вершине v и отображения $x_v^{\#}$ и $y_v^{\#}$ однозначно определяют, выполняет ли автомат какой-либо переход, и, если выполняет, то сам переход, т.е. однозначно определяют принимаемый стимул $x_v^{\wedge}$, посылаемую реакцию $y_v^{\wedge}$ и постсостояние $t_v^{\wedge}$.

Доказательство: По определению отображения $x_v^{\#}$ и $y_v^{\#}$ однозначно определяют множества $X_v^{\#}$ и $Y_v^{\#}$. Из второго требования детерминизма следует, что при любом распределении $x_v^{\#}$ сообщений на входных дугах автомата, порождающим множество $X_v^{\#}$ потенциальных стимулов, не более одного из этих стимулов $x_v \in X_v^{\#}$ может быть принято автоматом в данном состоянии s_v. Такой стимул x_v будем называть *выбираемым*, он может отсутствовать. Правда, это не означает, что выбираемый стимул x_v (если он есть) обязательно будет принят автоматом, поскольку выполнение перехода с приёмом этого стимула обусловлено возможностью послать реакцию. Рассмотрим все случаи поведения автомата в состоянии s_v при заданных $x_v^{\#}$ и $y_v^{\#}$ (однозначно определяющих $X_v^{\#}$ и $Y_v^{\#}$), определяя, будет ли выполнен переход и, если будет, то сам переход, т.е. принимаемый стимул $x_v^{\wedge}$, посылаемую реакцию $y_v^{\wedge}$ и постсостояние $t_v^{\wedge}$.

1. Имеется выбираемый стимул $x_v \in X_v^{\#}$, он единственный по 2-ому требованию детерминизма.

1.1. Для некоторой реакции y есть переход $s_v \text{—}?x!y \to t$ и $y \in Y_v^{\#}$, т.е. реакция y может быть послана (нужные выходные дуги пусты). Автомат выполнит этот переход (он единственный по 1-ому требованию детерминизма), $x_v^{\wedge}=x$, $y_v^{\wedge}=y$, $t_v^{\wedge}=t$.

1.2. Для любой реакции y либо нет перехода $s \text{—}?x!y \to$, либо $y \notin Y_v^{\#}$. Автомат не выполнит никакого перехода, $x_v^{\wedge}=\varnothing$, $y_v^{\wedge}=\varnothing$, $t_v^{\wedge}=s_v$.

2. Нет выбираемого стимула. Автомат не выполнит никакого перехода, $x_v^{\wedge}=\varnothing$, $y_v^{\wedge}=\varnothing$, $t_v^{\wedge}=s_v$.

Утверждение доказано.

3.3. Интерпретация 1-ой модели в терминах 2-ой модели

Утверждение 5: Автомат 1-ой модели (раздел 2) – частный случай автомата 2-ой модели (раздел 3).

Доказательство: Утверждение очевидно, если применить интерпретацию 1-ой модели в терминах 2-ой модели так, как отображено на табл.1., и сравнить определения выполнимости перехода автомата в обеих моделях.

Табл. 1. Интерпретация 1-ой модели в терминах 2-ой модели
Table. 1. Interpretation of the first model in terms of the second model

	1-ая модель	интерпретация	особенности 1-ой модели
стимул	$x=(i,m)$	$x=\{(i,m)\}$	принимается ровно одно сообщение
реакция	$y=(j,m)$ или $y=\varnothing$	$y=\{(j,m)\}$ или $y=\varnothing$	посылается не более одного сообщения
переход	$s—?x!y\rightarrow t$	$s—?x!y\rightarrow t$	

Утверждение доказано.

В то же время детерминированный автомат в 1-ой модели, вообще говоря, не является детерминированным автоматом во 2-ой модели. Например, в 1-ом случае могут быть два перехода $s—?(i,m)!y\rightarrow t$ и $s—?(i`,m`)!y\rightarrow t$, где $i\neq i`$, наличие которых во 2-ой модели означает нарушение 2-го требования детерминизма. Это объясняется тем, что 1-ая модель ограничивает циркуляцию сообщений в системе, допуская только одно сообщение, поэтому требования детерминизма в 1-ой модели оказываются слабее, чем во 2-ой.

3.4. Композиция автоматов системы

Определим композицию детерминированных автоматов с данным графом связей. Результатом композиции будет автомат (S,X,Y,T,s_0), отражающий работу системы в целом, включая все автоматы-компоненты и все дуги. Поскольку теперь, в отличие от первой модели, сообщения могут быть на нескольких дугах одновременно, но не более одного сообщения на каждой дуге, состояние системы – это набор состояний её автоматов $s_1,s_2,...,s_k$, а также распределение сообщений по дугам графа связей. Формально состояние системы – это набор $s=(s_1,s_2,...,s_k,D)$, где $D:E\rightarrow M$ – частично-определённое отображение, которое для каждой непустой дуги указывает находящееся на ней сообщение. Начальным состоянием системы, как и для первой модели, будем считать состояние, в котором каждый автомат находится в своём начальном состоянии, а сообщений на дугах нет, то есть состояние $s_0=(s_{10},s_{20},...,s_{k0},\varnothing)$.

Переходы композиции из T определяются в зависимости от того, предполагается ли синхронный или асинхронный режим работы автоматов. В каждом состоянии системы имеется множество A автоматов, которые могут выполнять переходы. В синхронном режиме за один такт срабатывают все автоматы из A, а в асинхронном – только один автомат (вообще говоря, некоторое подмножество A), выбираемый недетерминированным образом. Поскольку в рамках данной статьи нас интересуют только детерминированные системы, асинхронный режим мы далее не рассматриваем. Заметим, что для первой модели эти два режима работы не различаются, поскольку в каждом состоянии системы может сработать не

более одного автомата. В отличие от первой модели теперь окружение может подавать в систему следующий стимул не только в финальном состоянии, когда все дуги пусты, но и в любом состоянии, при котором пусты те внешние входные дуги, по которым окружение хочет послать сообщения.

Определим переходы композиции формально. В состоянии системы s окружение может послать в систему сообщение по любой пустой внешней входной дуге, а также принять сообщение с любой занятой внешней выходной дуги. Это определяет допустимые стимулы и реакции. Стимул $x:I_0\rightarrow M$ *допустимо послать* из окружения в систему, если $\mathbf{Dom}(x)\cap\mathbf{Dom}(D)=\varnothing$, в частности, всегда допустим пустой стимул $x=\varnothing$. Реакцию $y:O_0\rightarrow M$ *допустимо принять* из системы в окружение, если $y\subseteq D$, в частности, всегда допустима пустая реакция $y=\varnothing$. Если стимул x и реакция y допустимы, то в композиции определяется переход $s—?x!y\rightarrow t^\wedge$, где $t^\wedge=(t_1^\wedge,t_2^\wedge,...,t_k^\wedge,D^\wedge)$. Определим для каждой вершины v постсостояние $t_v^\wedge$, а также определим $D^\wedge$.

Для $v=1..k$ рассмотрим автомат в вершине v. Для данного состояния системы s отображение $x_v^\#$ определяется однозначно как сужение отображения D на множество I_v входных дуг v-ого автомата: $x_v^\#=\{(i,m)|(i,m)\in D\ \&\ i\in I_v\}$, и отображение $y_v^\#$, которое каждой пустой выходной дуге v-го автомата ставит в соответствие множество M всех сообщений: $y_v^\#=\{(j,M)\mid j\in O_v\backslash\mathbf{Dom}(D)\}$. По утверждению 4 при заданных s_v, $x_v^\#$ и $y_v^\#$ автомат выполняет не более одного перехода, и однозначно определяются принимаемый стимул $x_v^\wedge$ и посылаемая реакция $y_v^\wedge$, а также постсостояние $t_v^\wedge$. Именно это постсостояние и записывается в $t^\wedge$.

Определим $D^\wedge$.

Сначала положим $D^\wedge:=D$.

Рассмотрим, как должно меняться расположение сообщений на дуге $e=(i,z,j)$.

1) В состоянии s дуга e была пустой, т.е. $e\notin\mathbf{Dom}(D)$. Тогда при $j\neq 0$ j-ый автомат не принимает с неё сообщения, т.е. $e\notin\mathbf{Dom}(x_j^\wedge)$. При $i\neq 0$ i-ый автомат посылает по этой дуге сообщение m, если $e\in\mathbf{Dom}(y_i^\wedge)\ \&\ y_i^\wedge(e)=m$; тогда пара (e,m) добавляется в $D^\wedge$. Если $j=0$, то $e\notin\mathbf{Dom}(y)$. Если $i=0$, то $e\in\mathbf{Dom}(x)\Rightarrow e\in\mathbf{Dom}(D^\wedge)\ \&\ D^\wedge(e)=x(e)$, т.е. если окружение посылает по внешней входной дуге e сообщение $x(e)$, то пара $(e,x(e))$ добавляется в $D^\wedge$.

2) В состоянии s на дуге e было сообщение m, т.е. $e\in\mathbf{Dom}(D)\ \&\ D(e)=m$. Тогда при $j\neq 0$ j-ый автомат принимает это сообщение, если $e\in\mathbf{Dom}(x_j^\wedge)\ \&\ x_j^\wedge(e)=m$; тогда пара $(e,x_j^\wedge(e))$ удаляется из $D^\wedge$. При $i\neq 0$ i-ый автомат не может послать по этой дуге никакого сообщения, поскольку дуга в состоянии s занята, т.е. $e\notin\mathbf{Dom}(y_i^\wedge)$. Если $j=0$, то $e\in\mathbf{Dom}(y)\Rightarrow e\notin\mathbf{Dom}(D^\wedge)\ \&\ D(e)=y(e)$, т.е. если окружение принимает по внешней выходной дуге e сообщение $y(e)$, то пара $(e,y(e))$ удаляется из $D^\wedge$. Если $i=0$, то $e\notin\mathbf{Dom}(x)$.

Тем самым, $D^\wedge=(D\cup x\cup y_1^\wedge\cup...\cup y_k^\wedge)\setminus(y\cup x_1^\wedge\cup...\cup x_k^\wedge)$.

Будем говорить, что такая композиция детерминированных автоматов детерминирована, если в каждом достижимом (из начального состояния) состоянии каждая пара допустимых стимула и реакции однозначно определяет постсостояние системы, т.е. выполняемый переход.

Утверждение 6: Во второй модели композиция детерминированных автоматов детерминирована.

Доказательство: Достаточно показать, что 1) каждый автомат вершины графа связей может выполнить не более одного перехода, и 2) распределение $D^\wedge$ сообщений по дугам определяется однозначно. И то, и другое следует из утверждения 4 и определения композиции.

Утверждение доказано.

3.5. Генерация тестов

При тестировании на каждом такте тест посылает в тестируемую систему сообщения по пустым внешним входным дугам (не обязательно всем) и принимает от системы по занятым внешним выходным дугам (не обязательно всем) имеющиеся на них сообщения. Определим композицию системы и теста. Состояние композиции – это пара состояний системы и теста. Переход композиции соответствует паре из допустимого стимула и допустимой реакции, что определяет возможные постсостояния системы и теста, т.е. возможные постсостояния композиции. Сам переход композиции является внутренним, т.е. ничем не помечен, поскольку композиция системы и теста замкнута и ни с чем не взаимодействует. Формально в композиции системы и теста переходы определяются следующим правилом композиции: $s—?x!y→t \;\&\; s`—?y!x→t` \vdash ss`⟶tt`$.

Если тест и система оба детерминированы, то их композиция тоже будет детерминирована в следующем смысле: в каждом её состоянии определено не более одного перехода.

Тестовая последовательность – это конечная последовательность пар $(x_1,y_1),\ldots,(x_n,y_n)$, которой в тестируемой системе соответствует маршрут (цепочка смежных переходов) $s_0—?x_1!y_1→s_1—\ldots→s_{n-1}—?x_n!y_n→s_n$, а в тесте – маршрут $s`_0—?y_1!x_1→s`_1—\ldots→s`_{n-1}—?y_n!x_n→s`_n$. Каждое состояние s_i и $s`_i$ соответствует префиксу тестовой последовательности длиной i.

Для такой тестовой последовательности детерминированной тест состоит только из указанной выше цепочки переходов. Детерминированный тест выбирает только одну допустимую реакцию, принимаемую от системы, как подмножество сообщений на занятых внешних выходных дугах системы, и посылает в систему только один допустимый стимул как набор сообщений, размещаемых на подмножестве пустых внешних входных дуг системы.

Какие проверки выполняются при прогоне теста? На каждом i-ом такте проверяется следующее. 1) Выполнился ли в тесте переход $s`_i—?y_{i+1}!x_{i+1}→s`_{i+1}$; если не выполнился, то фиксируется ошибка. 2) Для

каждого автомата в системе проверяется, правильно ли изменилось его состояние, правильно ли он выполнил приём стимула (с тех или не тех входных дуг принял сообщения), правильно ли он выполнил выдачу реакции (на те или не на те выходные дуги послал сообщения, и правильные ли эти сообщения).

Как и в случае первой модели, целью тестирования является покрытие всех достижимых переходов автоматов компонентов системы, а генерация тестов основана на фильтрации набора тестов, сгенерированного для тестирования композиционной системы. В то же время тестирование для второй модели имеет ряд отличий от тестирования для первой модели.

Во-первых, теперь тест может посылать сообщения в систему не только в её финальном состоянии, а в любом состоянии.

Во-вторых, если в первой модели сообщение не посылается в систему в её финальном состоянии, то ничего не происходит: система не меняет своего состояния. Во второй модели система в финальном состоянии (т.е. при отсутствии сообщений на дугах), вообще говоря, может продолжать работать, поскольку допустимы переходы, при которых сообщения не принимаются.

В-третьих, если в первой модели сообщение (единственное в системе) находится на внешней выходной дуге, то также ничего не происходит, т.е. система не меняет своего состояния, до тех пор, пока тест не примет это сообщение и, тем самым, не переведёт систему в финальное состояние. Именно поэтому такое действие теста очевидно и обязательно, и в тестовой последовательности для 1-ой модели опущены принимаемые от системы сообщения, т.е. реакции системы. Во второй модели, с одной стороны, тестируемая система может выполнять переходы и в том случае, когда на внешних выходных дугах имеются сообщения, а, с другой стороны, тест может принимать часть сообщений с внешних выходных дуг.

В-четвёртых, в первой модели выполнимость перехода зависит только от наличия нужного сообщения на нужной входной дуге, поскольку выходные дуги всегда свободны. По замечанию 1 исключение составляет только переход $s—?(i,m)!(i,m`)→t$ по приёму сообщения с дуги-петли i с одновременной посылкой сообщения по этой же дуге, такой переход никогда не выполним. Во второй модели также всегда невыполним аналогичный переход $s—?x!y→t$, где некоторая дуга-петля $i∈\mathbf{Dom}(x)∩\mathbf{Dom}(y)$, с приёмом сообщения $x(i)$ с дуги-петли и посылкой сообщения $y(i)$ по ней же. Однако во второй модели в системе может быть несколько сообщений и выполнимость перехода всё же зависит от того, заняты или нет выходные дуги, по которым нужно послать сообщения.

Замечание 3: о приёме тестом сообщений от системы. Приём или не приём тестом сообщений с внешних выходных дуг системы, по сути, является дополнительным тестовым воздействием на систему, поскольку меняет её поведение (выполнимость тех или иных переходов). Для того чтобы описать это формально, такое тестовое воздействие нужно трактовать как часть

стимула, т.е. нужно приём тестом сообщений от системы заменить посылкой тестом сообщений в систему. Поведение дуги как очереди длины 1 можно изобразить в виде автомата с одним входом и одним выходом, изображённого на рис. 5 вверху.

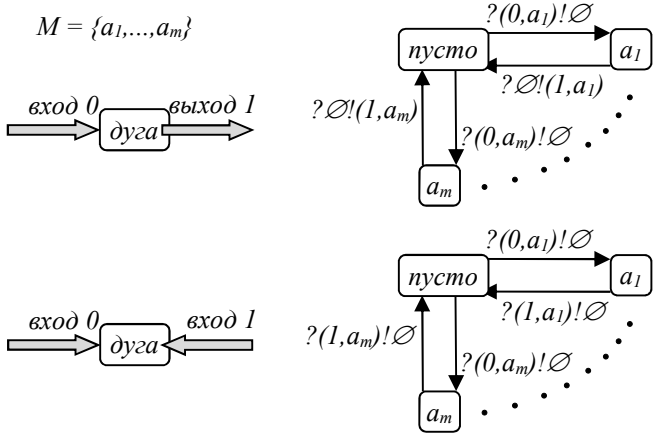

Рис. 5. Автомат очереди длины 1
Fig. 5. Automata of a queue of length 1

Такой автомат дуги взаимодействует синхронно с автоматами начальной и конечной вершин дуги: либо одновременно происходит посылка сообщения по дуге из автомата начала дуги и приём этого сообщения в автомат дуги, либо одновременно происходит посылка сообщения из автомата дуги и приём этого сообщения по дуге в автомате конца дуги. Для того чтобы в тесте заменить приём посылкой, достаточно преобразовать автоматы внешних выходных дуг, объявляя выход *1* входом *1* и заменяя на переходах автомата дуги пометку $?\varnothing!(1,a_i)$ – на пометку $?(1,a_i)!\varnothing$ (рис. 5 внизу). Соответственно, в автомате теста все входы превращаются в выходы, а на переходах вместо $?y!x$ записывается $?\varnothing!(y\cup x)$, т.е. для тестируемой системы вместо стимула x будет стимул $y\cup x$.

Замечание 4: о внешних реакциях. Для теста всё равно, какую именно реакцию y посылает система, поскольку правильность реакции y проверяется во время проверки правильности перехода системы, т.е. правильности сообщений, которые автоматы компонентов посылают на свои выходные дуги, которые являются внешними выходными дугами. Поэтому после преобразования, когда реакции становятся частью стимулов, для стимула $y\cup x$ в паре $(j,m)\in y$, где j – выходная дуга теста, бывшая до преобразования входной дугой теста, сообщение m можно заменить на специально выделенное сообщение «*принять*».

4. Заключение

В заключение сформулируем направления дальнейших исследований.

1) Предложенный алгоритм фильтрации не обязательно даёт оптимальный полный набор тестов. Оптимальность может пониматься как минимальное число или минимальная суммарная длина тестовых последовательностей. Соответственно, возникает задача оптимизации, т.е. поиска оптимального набора, которая, вообще говоря, сводима к задаче о поиске минимального покрытия ([7], [8]).

2) В этой статье определена композиция автоматов системы, наиболее удобная для тестирования. Её результатом является автомат без входных и выходных дуг, поскольку они «погружены» внутрь системы. Поэтому такой автомат не может использоваться как компонент при построении более сложной системы. В дальнейшем мы предполагаем (в рамках более общей модели) определить композицию другим способом так, чтобы её результатом был автомат с входными и выходными дугами, т.е. автомат, который можно помещать в вершину графа связей.

3) В этой статье мы предполагали, что дуга реализует очередь длины 1. Это легко обобщается на случай очередей большей длины, в том числе неограниченных очередей; более того, разные дуги могут реализовывать очереди разной длины. Длина очереди является статическим атрибутом дуги и задаётся при задании графа связей. Для второй модели отображение D, описывающее состояние дуг, теперь должно отображать дугу в последовательность сообщений, находящихся в очереди, в частности, пустая последовательность соответствует пустой дуге.

Однако для очереди длины больше 1 возникает другая проблема. В конце 3-го раздела определён автомат дуги, которая понимается как очередь длины 1. Здесь нам, что называется, «повезло», поскольку очередь большей длины невозможно изобразить в виде автомата того типа, что определён в разделе 3. Дело в том, что в «промежуточном» состоянии, когда очередь не пуста, но и не полностью заполнена, вместе с каждым переходом $s\text{—}?(0,x)!(1,y)\to t$, при котором в конец очереди вставляется сообщение x, а из головы очереди удаляется сообщение y, должны быть два других перехода. Второй переход – это переход $s\text{—}?(0,x)!\varnothing\to t$, при котором в конец очереди вставляется сообщение x, а из головы очереди не удаляется сообщение y, поскольку конец дуги его не принимает. Третий переход – это переход $s\text{—}?\varnothing!(1,y)\to t$, при котором из головы очереди удаляется сообщение y, но в конец очереди не вставляется никакого сообщения, поскольку начало дуги не посылает по дуге никакого сообщения. Однако в рамках второй модели наличие первого и второго переходов противоречит 1-ому требованию детерминизма: одному стимулу $(0,x)$ соответствуют две реакции $(1,y)$ и $\varnothing$. Кроме того, третий переход – это переход по пустому стимулу, который совместим с любым другим стимулом, в том числе со стимулом $(0,x)$, что противоречит 2-ому

требованию детерминизма. Решение этой проблемы мы предполагаем искать на пути подходящего обобщения модели автомата.

4) Кроме очереди длины больше 1, могут быть и другие дуги графа связей. Например, очередь с приоритетами, стек и т.п. Мы предполагаем обобщить понятие дуги с помощью определения автомата дуги. Для детерминизма системы, по-видимому, к автомату дуги придётся предъявить дополнительные (по сравнению с автоматом вершины) требования. Композиция должна быть определена для пары автоматов, выход одного из которых соединён с входом другого. На таком соединении происходит синхронное взаимодействие автоматов, когда один автомат посылает сообщение тогда и только тогда, когда другой автомат это сообщение принимает.

5) При тестировании, как обычно, возникает задача «огрубления», когда проверяется не «всё подряд», а проверка делается выборочно. Это называют факторизацией тестирования [9]. Например, для проверки правильности реализации числовой функции домены её аргументов разбиваются на поддомены, и из каждого поддомена выбирается хотя бы одно число для проверки. В терминах рассматриваемой здесь задачи факторизация означает, что на переходах автомата компонента определяется отношение эквивалентности, и считается, что достаточно проверить хотя бы один переход из класса эквивалентности.

6) В данной статье предлагается решение проблемы тестирования компонентов только детерминированной системы автоматов. Одним из направлений дальнейших исследований могло бы стать исследование проблемы недетерминизма. Более точно, ставится задача определить такие ограничения на недетерминизм системы и/или составляющих её автоматов, которые позволяли бы выполнять полное тестирование за конечное время. Для автономного тестирования, когда автомат находится под непосредственным управлением теста (не в контексте окружающей его части системы), предложенные неплохие решения этой задачи ([10], [11], [12], [13]).

7) В данной статье предполагается, что известно, каким должен быть автомат каждого компонента: задан граф переходов автомата с точностью до изоморфизма. В более общем случае между автоматом компонента в реализации и автоматом компонента в спецификации задаётся то или иное отношение конформности, которое слабее изоморфизма: квази-редукция, симуляция и т.п. Это требует более сложного алгоритма тестирования. Если тестирование компонента автономное, т.е. не в контексте других компонентов системы и связующих их дуг, то возможно полное тестирование за конечное время конформности типа редукции или слабой симуляции ([2], [3], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20]). Для составной системы возникает проблема декомпозиции системных требований, называемая также проблемой несохранения конформности. Она заключается в том, что композиция реализаций компонентов, конформных спецификациям этих компонентов, в общем случае неконформна спецификации системы, в

частности, композиции спецификаций компонентов. Этой проблеме посвящён ряд работ ([2], [4], [21]), но возникает задача переосмысления предложенных решений для рассматриваемой в этой статье задачи тестирования компонентов системы при условии, что верна гипотеза о связях.

Список литературы

[1]. Revised Working Draft on “Framework: Formal Methods in Conformance Testing”. JTC1/SC21/WG1/Project 54/1, ISO Interim Meeting, ITU-T on, Paris, 1995 г.

[2]. И.Б.Бурдонов, Косачев А.С., В.В.Кулямин. Теория соответствия для систем с блокировками и разрушением. «Физ-мат лит» Наука, Москва, 2008 г., 412 стр.

[3]. И.Б.Бурдонов. Теория конформности (функциональное тестирование программных систем на основе формальных моделей). LAP Lambert Academic Publishing, 2011 г., 428 стр.

[4]. И.Б.Бурдонов, А.С.Косачев. Пополнение спецификации для ioco. Программирование, 2011 г., №1, стр. 3-18.

[5]. А. Камкин, М. Чупилко. Обзор современных технологий имитационной верификации аппаратуры. Программирование, 2011 г., №3, стр. 42-49.

[6]. И.Б. Бурдонов, А.С. Косачев, В.В. Кулямин. Неизбыточные алгоритмы обхода ориентированных графов. Детерминированный случай. Программирование, 2003 г., №5, стр. 59-69.

[7]. Ананий В. Левитин. Алгоритмы: введение в разработку и анализ. М.: «Вильямс», 2006 г., стр. 160-163, ISBN 0-201-74395-7.

[8]. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ. 2-ое издание. М.: «Вильямс», 2006 г., стр. 456-458, ISBN 0-07-013151-1.

[9]. Бурдонов И.Б., Косачев А.С., Кулямин В.В. Использование конечных автоматов для тестирования программ. Программирование. 2000 г., № 2, стр.12-28.

[10]. И.Б. Бурдонов, А.С. Косачев. Полное тестирование с открытым состоянием ограниченно недетерминированных систем. Программирование, 2009 г., №6, стр. 3-18.

[11]. И.Б.Бурдонов, А.С.Косачев. Семантики взаимодействия с отказами, дивергенцией и разрушением. Часть 2. Условия конечного полного тестирования. Вестник Томского Государственного Университета, № 2(15), 2011 г., стр. 89-98.

[12]. И.Б. Бурдонов, А.С. Косачев. Тестирование конформности на основе соответствия состояний. Труды ИСП РАН, volume 18, 2010 г., стр. 183-220.

[13]. И.Б.Бурдонов, А.С.Косачев, Безопасное тестирование симуляции систем с отказами и разрушением. Моделирование и анализ информационных систем, том 17(4), 2010 г., стр. 27-40.

[14]. I.B.Bourdonov, A.S.Kossatchev, V.V.Kuliamin. Formal Conformance Testing of Systems with Refused Inputs and Forbidden Actions. Proceedings of the Workshop on Model Based Testing (MBT 2004), Elsevier, 2006.

[15]. И.Б.Бурдонов, А.С.Косачев, В.В.Кулямин. Формализация тестового эксперимента. Программирование, 2007 г., №5, стр. 3-32.

[16]. И.Б.Бурдонов, А.С.Косачев, В.В.Кулямин. Безопасность, верификация и теория конформности. Материалы второй международной научной конференции по проблемам безопасности и противодействия терроризму. МГУ 2006, М., МЦНМО, 2007 г., стр. 135-158.

[17]. И.Б.Бурдонов, Косачев А.С. Системы с приоритетами: конформность, тестирование, композиция. Труды ИСП РАН, том 14 (1), 2008 г., стр.23-54.
[18]. И.Б. Бурдонов, А.С. Косачев. Тестирование с преобразованием семантик. Труды ИСП РАН, том 17, 2009 г., стр.193-208.
[19]. A.Kossachev, I.Burdonov. Formal Conformance Verifcation, Short Papers of the 22nd IFIP ICTSS, Alexandre Petrenko, Adenilso Simao, Jose Carlos Maldonado (eds.), Nov. 08-10, 2010, Natal, Brazil, pp.1-6.
[20]. А.С. Косачев, И.Б.Бурдонов. Семантики взаимодействия с отказами, дивергенцией и разрушением. Программирование, 2010 г., №5, стр. 3-23.
[21]. И.Б.Бурдонов, А.С.Косачев. Согласование конформности и композиции. Программирование, 2013 г., №6, стр. 3-15.

Testing of automata system

I.B. Burdonov <igor@ispras.ru>
A.S. Kossatchev <kos@ispras.ru>
Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia

Abstract. The problem of testing of aggregate systems is considered. The system is described with an oriented graph of links. The nodes correspond to automata of the components and arcs correspond to simplex communication channels. The hypothesis of the links is assumed: the graph of links is static and the link structure is error-free. In each state, the automaton can accept and send multiple messages through incoming and outgoing arcs (at most one message through each arc). The goal of testing is to cover transitions of the automata reachable during the system work. It assumed that during testing it is possible to observe the state changes of automata and the messages on the arcs. A simplified system model with only one message circulating is considered at the beginning. On its example we show that the hypothesis on links allows considerably reduce the number of required testing actions from the multiplication of numbers of the component automata states to the sum of these numbers. If the numbers of states of all automata are equal, it gives exponential reduction of the number of test actions. Then the more general model is considered when the system can simultaneously contain multiple messages, but not more than one on each arc. A composition of the system automata is defined and the restrictions on automata making the system deterministic are described. An algorithm of test generation is proposed basing on test filtration generated for covering all transitions of the deterministic composition system. Test is rejected if it covers only such transitions of the components that are covered by the remaining tests. In conclusion, the directions of future research are described.

Keywords: directed graphs; graph coverage, communicating automata, distributed systems, testing, networks.

DOI: 10.15514/ISPRAS-2016-28(1)-7

For citation: Burdonov I.B., Kossatchev A.S. Testing of automata system. Trudy ISP RAN /Proc. ISP RAS, 2016, vol. 28, issue 1, pp. 103-130 (in Russian). DOI: 10.15514/ISPRAS-2016-28(1)-7

References

[1]. Revised Working Draft on “Framework: Formal Methods in Conformance Testing”. JTC1/SC21/WG1/Project 54/1, ISO Interim Meeting, ITU-T on, Paris, 1995.
[2]. Bourdonov I.B., Kossatchev A.S., Kuliamin V.V. Teoriya sootvetstviya dlya system s blokirovkami I razrusheniem [Conformance theory of the systems with Refused Inputs and Forbidden Actions]. Moscow, «Nauka», 2008, 412 p. (in Russian)
[3]. Bourdonov I. Teoriya konformnosti (funkcional'noe testirovanie prorammny'kh system na osnove formal'ny'kh modelej [Conformance theory (functional testing on formal model base)]. LAP LAMBERT Academic Publishing, Saarbrucken, Germany, 2011, ISBN 978-3-8454-1747-9, 428 p. (in Russian)
[4]. Bourdonov I.B., Kossatchev A.S. Specification Completion for IOCO. Programming and Computer Software, vol. 37(1), 2011, pp. 1-14. *DOI*: 10.1134/S0361768811010014
[5]. A. S. Kamkin, M. M. Chupilko. Survey of modern technologies of simulation-based verification of hardware. Programming and Computer Software, vol. 37 (3), 2011, pp. 147-152. *DOI*: 10.1134/S0361768811030017
[6]. I. B. Burdonov, A. S. Kossatchev, V. V. Kuliamin. Irredundant Algorithms for Traversing Directed Graphs: The Deterministic Case. Programming and Computer Software, vol. 29(5), 2003, pp. 245-258. *DOI*: 10.1023/A:1025733107700
[7]. A. Levitin. Algoritmy: vvedenie v razrabotku i analiz [Introduction to The Design and Analysis of Algorithms]. M.: «Viliams», 2006, pp. 160-163, ISBN 0-201-74395-7. (in Russian)
[8]. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. Introduction to Algorithms. 2-nd Edition. MIT Press Cambridge, MA, USA, 2001, ISBN 0-262-03293-7.
[9]. Bourdonov I.B., Kossatchev A.S., Kuliamin V.V. Application of Finite Automatons for Program Testing. Programming and Computer Software, vol. 26 (2), 2000, pp. 61-73. *DOI*: 10.1007/BF02759192
[10]. Bourdonov I.B., Kossatchev A.S. Complete Open-State Testing of Limitedly Nondeterministic Systems. Programming and Computer Software, vol. 35 (6), 2009, pp.301-313. *DOI*: 10.1109/HASE.2014.39
[11]. Bourdonov I.B., Kossatchev A.S. Semantiki vzaimodejstviya s otkazami, divergentsiej i razrusheniem. Chast' 2. Usloviya konechnogo polnogo testirovaniya. [Semantics of Interaction with Refused Inputs, Divergence and Forbidden Actions. Part 2. The condition of finite complete testing]. Vestnik Tomskogo gosudarstvennogo universiteta. Upravlenie, vychislitel'naya tekhnika i informatika. [Tomsk State University. Journal of Control and Computer Science], 2011, №2, pp. 89-98. (in Russian)
[12]. Bourdonov I.B., Kossatchev A.S. Testirovanie konformnosti na osnove sootvetstviya sostoyanij [Conformance testing based on a state relation]. Trudy ISP RAN [Proceeding of ISP RAS], vol. 18, 2010, pp. 183-320 (in Russian)
[13]. 84. Bourdonov I.B., Kossatchev A.S. Safe simulation testing of systems with refusals and destructions. Automatic Control and Computer Sciences, vol. 45(7), 2011, pp. 380-389. *DOI*: 10.3103/S0146411611070042
[14]. I.B.Bourdonov, A.S.Kossatchev, V.V.Kuliamin. Formal Conformance Testing of Systems with Refused Inputs and Forbidden Actions. Proceedings of the Workshop on Model Based Testing (MBT 2004), Elsevier, 2006. *DOI*: 10.1016/j.entcs.2006.09.008
[15]. Bourdonov I.B., Kossatchev A.S., Kuliamin V.V. Formalization of Test Experiments. Programming and Computer Software, vol. 33(5), 2007, pp. 239-260. 10.1134/S0361768807050015

[16]. Bourdonov I.B., Kossatchev A.S., Kuliamin V.V. Bezopasnost', verifikatsiya i teoriya konformnosti [Safety, Verification and Conformance Theory]. Materialy Vtoroj mezhdunarodnoj nauchnoj konferentsii po problemam bezopasnosti i protivodejstviya terrorizmu [The proceeding of the Second international conference on the problems of safety and counteraction against terrorizm], Moscow, MNCMO, 2007, pp. 135-158. (in Russian)

[17]. Bourdonov I.B., Kossatchev A.S. Sistemy s prioritetami: konformnost', testirovanie, kompozitsiya [Systems with priority: conformance, testing, composition]. Trudy ISP RAN [Proceeding of ISP RAS], vol. 14(1), 2008, pp.23-54 (in Russian)

[18]. Bourdonov I.B., Kossatchev A.S. Testirovanie s preobrazovaniem semantic [Testing with Semantics Conversion] Trudy ISP RAN [Proceeding of ISP RAS], vol. 17, 2009, pp. 193-208. (in Russian)

[19]. A.Kossachev, I.Burdonov. Formal Conformance Verifcation, Short Papers of the 22nd IFIP ICTSS, Alexandre Petrenko, Adenilso Simao, Jose Carlos Maldonado (eds.), Nov. 08-10, 2010, Natal, Brazil, pp.1-6.

[20]. Bourdonov I.B., Kossatchev A.S. Interaction Semantics with Refusals, Divergence, and Destruction. Programming and Computer Software, vol. 36(5), 2010, pp. 247-263. *DOI*: 10.1134/S0361768810050014

[21]. Bourdonov I.B., Kossatchev A.S. Agreement between Conformance and Composition. Programming and Computer Software, vol. 39(6), 2013, pp. 269–278. *DOI*: 10.1134/S0361768813060029