ВЕСТНИК ТОМСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
2017 Управление, вычислительная техника и информатика № 38

ДИСКРЕТНЫЕ ФУНКЦИИ И АВТОМАТЫ

УДК 004
DOI: 10.17223/19988605/38/10

И.Б. Бурдонов, А.С. Косачев

ТЕСТИРОВАНИЕ СИСТЕМЫ АВТОМАТОВ

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. При тестировании возможно наблюдение состояний автоматов и передаваемых сообщений. Гипотеза о связях предполагает, что граф связей статический и не содержит ошибок. Это позволяет существенно сократить время тестирования детерминированной системы вплоть до экспоненциального уменьшения числа тестовых воздействий.
Ключевые слова: ориентированный граф; покрытие графа; взаимодействующие автоматы; тестирование; сети.

Большинство сложных, особенно распределённых систем представляет собой набор взаимодействующих компонентов. В данной статье компоненты моделируются конечными автоматами, а взаимодействие – обменом сообщениями между автоматами. Структура связей между компонентами моделируется ориентированным графом (*графом связей*), в вершинах которого находятся автоматы, а дуги соответствуют симплексным каналам передачи сообщений. *Внутренние* дуги соединяют автоматы между собой, а *внешние* дуги связывают систему с её *окружением*. *Внешняя входная дуга* ведёт из внешней среды в один из автоматов, а *внешняя выходная дуга* ведёт из автомата во внешнюю среду. При тестировании тест играет роль внешней среды: передаёт сообщения в систему по внешним входным дугам и принимает от системы сообщения по внешним выходным дугам.

Если граф связей статический, т.е. не меняющийся в процессе работы системы, система (так же как её компоненты) может моделироваться конечным автоматом, получающимся из автоматов-компонентов с помощью подходящего оператора композиции, учитывающего граф связей.

Система работает правильно, если структура её связей правильна и каждый автомат в системе работает правильно. Обратное, вообще говоря, не верно, если требования к системе неоднозначно определяют её структуру, например, функциональные требования к системе, связывающие сообщения, получаемые от системы, с сообщениями, посылаемыми в систему. В данной статье целью тестирования является покрытие переходов автоматов системы, достижимых при работе этих автоматов в системе. Поэтому если в структуре связей нет ошибок (*гипотеза о связях*), т.е. граф связей автоматов совпадает с заданным, то такое тестирование сводится к проверке правильности переходов каждого автомата. Проблема в том, что автомат может тестироваться только как часть системы, т.е. тест не имеет непосредственного доступа к автомату, 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 [1–5], когда этот компонент рассматривается как тестируемая система, а остальные – как контекст. Существенное отличие, однако, в том, что в таком контексте тоже возможны ошибки, но если верна гипотеза о связях, то только в компонентах, а не в графе связей. С другой стороны, при тестировании проверяется работа сразу нескольких компонентов, через которые проходят сообщения. Поскольку автомат тестируется как часть системы, могут быть проверены не все его переходы, которые проверяются при автономном тестировании с прямым доступом к автомату. Речь идет только о переходах автоматов, *достижимых* при его работе в составе системы.

Тестирование композиционного автомата системы, получающегося композицией автоматов по заданному графу связей, обеспечивает проход по всем его достижимым переходам. При этом, конечно, проверяются все достижимые переходы автоматов-компонентов, но может делаться много «лишней работы». Гипотеза о связях позволяет существенно сократить время тестирования.

В статье предлагается алгоритм построения набора тестов, который является полным (проверяет все достижимые переходы автоматов-компонентов) при выполнении двух условий: 1) верна гипотеза о связях; 2) система является детерминированной. Дополнительно алгоритм определяет недостижимые переходы автоматов. Предполагается, что нам известно, каким должен быть каждый автомат (задан граф переходов автомата с точностью до изоморфизма), и именно это проверяется при тестировании. Кроме того, тест может наблюдать как состояния автоматов в вершинах графа связей, так и сообщения на его дугах. Поскольку не налагается ограничений на связность графов переходов автоматов, полный набор тестов может содержать более одного теста. При переходе от одного теста к другому требуется рестарт системы. Такие предположения могут быть оправданы, например, при имитационном тестировании аппаратуры (simulation-based verification) (см. например, [6]).

1. Модель

Дуга рассматривается как очередь сообщений длины 1. По дуге можно послать сообщение, если она *пуста* (на ней нет сообщений), и с дуги можно принять сообщение, если она не пуста (на ней есть сообщение). Если сообщение передаётся по внутренней дуге $a \to b$, то оно было послано автоматом в вершине a и будет принято автоматом в вершине b. Если $a \to b$ – внешняя входная дуга, то a – внешняя среда. Если $a \to b$ – внешняя выходная дуга, то b – внешняя среда. Автомат может принимать одновременно несколько сообщений по нескольким (не обязательно всем) входным дугам, но не более одного сообщения по каждой дуге, и посылать несколько сообщений по нескольким (не обязательно всем) выходным дугам, но не более одного сообщения по каждой дуге. Автомат предполагается детерминированным. 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 внешней среде. Введём формальные определения и обозначения.

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 внешней среды 0 нет дуг-петель). Дуга (v, z, w) задана начальной вершиной v, пометкой z и конечной вершиной w. Дуга $(0, z, w)$ – внешняя входная дуга, $(v, z, 0)$ – внешняя выходная дуга. Кроме того, $I_v = \{(a, z, v) \mid (a, z, v) \in E\}$ – множество дуг, заканчивающихся в вершине v, $O_v = \{(v, z, b) \mid (v, z, b) \in E\}$ – множество дуг, начинающихся в вершине v. I_0 содержит все внешние входные дуги, O_0 – внешние выходные дуги, $E \setminus (I_0 \cup O_0)$ – внутренние дуги.

Пусть M – множество всех возможных *сообщений*, общее для всех автоматов в системе.

Для множеств A и B через A^B обозначим множество всех отображений из B в A.

Система автоматов определяется как граф связей G, в котором каждой вершине v поставлен в соответствие автомат $A_v = (M, S_v, X_v, Y_v, T_v, s_{0v})$, где S_v – конечное множество *состояний* автомата, $X_v = \{x \mid \exists f (x \subseteq f \,\&\, f \in M^{I_v})\}$ – множество *стимулов* (входных символов), $Y_v = \{y \mid \exists f (y \subseteq f \,\&\, f \in M^{O_v})\}$ – множество *реакций* (выходных символов), $T_v \subseteq S_v \times X_v \times Y_v \times S_v$ – конечное множество *переходов*, $s_{0v} \in S_v$ – *начальное состояние автомата*. Обозначим: $s \text{ — } ?x!y \to t \triangleq (s, x, y, t) \in T_v$, $s \text{ — } ?x!y \to \triangleq \exists t\, (s, x, y, t) \in T_v$. 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 $s \text{ — } ?x!y \to$. Дугу из I_v будем называть *входной дугой* автомата в вершине v, а дугу из O_v – *выходной дугой* этого автомата. 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:

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

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

Определим формальное условие выполнения перехода $s - ?x!y \to t$: $x \in X_v^{\#} \;\&\; y \in Y_v^{\#}$.

2. Детерминизм

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

1) $\forall s,x,x',y,y',t,t' \quad s - ?x!y \to t \;\&\; s - ?x!y' \to t' \Rightarrow y = y' \;\&\; t = t'$.

Распределение $x_v^{\#}$ сообщений по входным дугам автомата неоднозначно определяет стимул, принимаемый автоматом, поскольку автомат может как принимать, так и не принимать сообщения с непустых входных дуг. Будем говорить, что стимулы x и x' *совместимы*, и обозначать $x \approx x'$, если при некотором отображении $x_v^{\#}$ автомат может принять любой из этих стимулов, т.е. $x \in X_v^{\#}$ и $x' \in X_v^{\#}$. Формально $x \approx x' \triangleq \forall i \in \boldsymbol{Dom}(x) \cap \boldsymbol{Dom}(x')\ (x(i) = x'(i))$. 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:

$\forall s,x,x',y,y',t,t'\ x \neq x' \;\&\; x \approx x' \Rightarrow \neg(s - ?x!y \to \;\&\; s - ?x'!y' \to)$.

Теорема 1. 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, порождающем 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 $t_v^{\wedge}$.

Если имеется выбираемый стимул $x_v \in X_v^{\#}$, то он единственный по второму требованию детерминизма. При этом, если для некоторой реакции y есть переход $s_v - ?x!y \to t$ и $y \in Y_v^{\#}$, т.е. реакция y может быть послана (нужные выходные дуги пусты), то автомат выполнит этот переход (он единственный по первому требованию детерминизма), $x_v^{\wedge} = x$, $y_v^{\wedge} = y$, $t_v^{\wedge} = t$. В противном случае автомат не выполнит никакого перехода, $x_v^{\wedge} = \varnothing$, $y_v^{\wedge} = \varnothing$, $t_v^{\wedge} = s_v$. Последнее имеет место и в случае отсутствия выбираемого стимула. Теорема 1 доказана.

3. Композиция

Определим композицию детерминированных автоматов по заданному графу связей. Результатом композиции будет автомат (S, X, Y, T, s_0), отражающий работу системы в целом, включая все автоматы-компоненты и все дуги.

Состояние системы есть набор $s = (s_1, s_2, ..., s_k, D)$, где $s_1, s_2, ..., s_k$ есть набор состояний её автоматов, а D: $E \to M$ – частично-определённое отображение, задающее распределение сообщений по дугам графа связей: оно для каждой непустой дуги указывает находящееся на ней сообщение. Начальное состояние системы $s_0 = (s_{10}, s_{20}, ..., s_{k0}, \varnothing)$, в котором каждый автомат находится в своём начальном состоянии, а сообщений на дугах нет.

Определение переходов композиции зависит от предполагаемого режима работы. В синхронном режиме за один такт срабатывают все автоматы, которые могут выполнить переход, а в асинхронном – только один такой автомат (вообще говоря, некоторое подмножество автоматов), выбираемый недетерминированным образом. Поскольку в рамках данной статьи нас интересуют только детерминированные системы, асинхронный режим далее не рассматривается.

Определим переходы композиции формально. В состоянии системы s внешняя среда может послать в систему сообщения по любым пустым внешним входным дугам и принять сообщения с любых занятых внешних выходных дуг. Это определяет допустимые внешние стимулы и реакции. Стимул x: $I_0 \to M$ *допустим*, если $\boldsymbol{Dom}(x) \cap \boldsymbol{Dom}(D) = \varnothing$, в частности всегда допустим пустой стимул $x = \varnothing$. Реакция y: $O_0 \to M$ *допустима*, 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 — $?x!y \to \hat{t}$, где $\hat{t} = (\hat{t_1}, \hat{t_2}, ..., \hat{t_k}, \hat{D})$. Определим для каждого v постсостояние $\hat{t_v}$ и распределение сообщений $\hat{D}$.

Рассмотрим вершину v. Для состояния системы s однозначно определяются отображение $x_v^{\#}$ как сужение отображения D на множество I_v входных дуг v-го 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: $y_v^{\#} = \{(j, M) \mid j \in O_v \backslash \boldsymbol{Dom}(D)\}$. По теореме 1 при заданных s_v, $x_v^{\#}$ и $y_v^{\#}$ автомат в вершине v выполняет не более одного перехода, и однозначно определяются принимаемый стимул $\hat{x_v}$ и посылаемая реакция $\hat{y_v}$, а также постсостояние $\hat{t_v}$, которое и становится частью состояния $\hat{t}$.

Определим $\hat{D}$. Сначала положим $\hat{D} := D$. Рассмотрим, как должно меняться расположение сообщений на дуге $e = (i, z, j)$.

1. В состоянии s дуга e была пустой, т.е. $e \notin \boldsymbol{Dom}(D)$. Если $j \neq 0$, то j-й автомат не принимает с неё сообщения, т.е. $e \notin \boldsymbol{Dom}(\hat{x_j})$. Если $i \neq 0$, то i-й автомат посылает по этой дуге сообщение m, если $e \in \boldsymbol{Dom}(\hat{y_i}) \,\&\, \hat{y_i}(e) = m$, тогда пара (e, m) добавляется в $\hat{D}$. Если $j = 0$, то $e \notin \boldsymbol{Dom}(y)$. Если $i = 0$, то $e \in \boldsymbol{Dom}(x) \Rightarrow e \in \boldsymbol{Dom}(\hat{D}) \,\&\, \hat{D}(e) = x(e)$, т.е. если внешняя среда посылает по внешней входной дуге e сообщение $x(e)$, то пара $(e, x(e))$ добавляется в $\hat{D}$.

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

Тем самым, $\hat{D} = (D \cup x \cup \hat{y_1} \cup ... \cup \hat{y_k}) \setminus (y \cup \hat{x_1} \cup ... \cup \hat{x_k})$.

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.

Теорема 2. Композиция детерминированных автоматов детерминирована.

Доказательство. Нужно показать, что 1) каждый автомат вершины графа связей может выполнить не более одного перехода и 2) распределение $\hat{D}$ определяется однозначно. И то и другое следует из теоремы 1 и определения композиции. Теорема 2 доказана.

4. Генерация тестов

В данной статье целью тестирования системы автоматов является покрытие всех достижимых переходов автоматов. 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 есть пара состояний системы и теста. Переход композиции соответствует паре (допустимый стимул, допустимая реакция), что определяет возможные постсостояния системы и теста, т.е. возможные постсостояния композиции. Заметим, что приём или неприём тестом сообщений с внешних выходных дуг системы, по сути, является дополнительным тестовым воздействием на систему, поскольку меняет выполнимость тех или иных переходов. Сам переход композиции является внутренним, т.е. ничем не помечен, поскольку композиция системы и теста замкнута и ни с чем не взаимодействует. Формально переходы композиции системы и теста определяются следующим правилом вывода: $s - ?x!y \to t \;\&\; s' — ?y!x \to t' \vdash ss' \to tt'$.

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

Тестовая последовательность есть конечная последовательность пар (x_1, y_1), ..., (x_n, y_n), которой в тестируемой системе соответствует маршрут (цепочка смежных переходов) $s_0 - ?x_1!y_1 \to s_1 - \ldots \to s_{n-1} - ?x_n!y_n \to s_n$, а в тесте – маршрут $s'_0 - ?y_1!x_1 \to s'_1 - \ldots \to s'_{n-1} - ?y_n!x_n \to s'_n$. Каждое состояние s_i и s'_i соответствует префиксу тестовой последовательности длиной i. Для такой тестовой последовательности детерминированный тест состоит только из указанной выше цепочки переходов.

Какие проверки выполняются при прогоне теста на каждом i-м такте? 1) Выполнился ли в тесте переход $s'_i - ?y_{i+1}!x_{i+1} \to s'_{i+1}$; если не выполнился, то фиксируется ошибка. 2) Для каждого автомата в системе: правильно ли изменилось его состояние, правильно ли он выполнил приём стимула (с каких входных дуг принял сообщения), правильно ли он выполнил выдачу реакции (на какие выходные дуги послал сообщения, и правильные ли эти сообщения).

Прогон тестовой последовательности покрывает некоторое множество переходов автоматов системы. Поскольку система детерминирована, это множество одно и то же при разных прогонах данной тестовой последовательности (с рестартом между прогонами), поэтому тестовую последовательность 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, которое является объединением множеств переходов автоматов, покрываемых тестами из этого набора. Набор тестов будем называть *полным*, если он покрывает все переходы всех автоматов, достижимые при работе этих автоматов 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, [7]). В качестве такого автомата для наших целей берётся автомат системы, получаемый с помощью композиции, описанной в предыдущем разделе. Покрывая все достижимые переходы композиционного автомата системы, мы, конечно, покрываем все достижимые переходы автоматов-компонентов. Однако такой набор тестов может быть 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, которое сначала равно множеству всех переходов всех автоматов. Когда генерируется очередной i-й тест T_i для композиционного автомата системы, вычисляется множество P_i переходов автоматов, покрываемое этим тестом. Тесту соответствует маршрут в композиционном автомате. Каждому переходу этого маршрута соответствует множество переходов в автоматах-компонентах (не более одного в каждом автомате); объединение этих множеств по всем переходам маршрута и есть множество P_i. Далее алгоритм фильтрации проверяет, покрывает ли i-й тест какой-либо новый, ещё не покрытый переход какого-либо автомата компонента. Если $P_i \cap P = \varnothing$, то никаких новых переходов

i-й 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.

5. О размере тестового набора

Теорема 3. Для любого числа состояний автоматов компонентов n_1, n_2 ,..., n_k существует такая система, что время тестирования (в тактах), необходимое для покрытия всех достижимых переходов композиции, равно $\boldsymbol{\Omega}(n_1 n_2 ... n_k)$, а минимальное время, достаточное для покрытия всех достижимых переходов автоматов-компонентов, равно $\boldsymbol{O}(n_1+n_2+...+n_k)$.

Доказательство. Рассмотрим систему, изображённую на рис. 1.

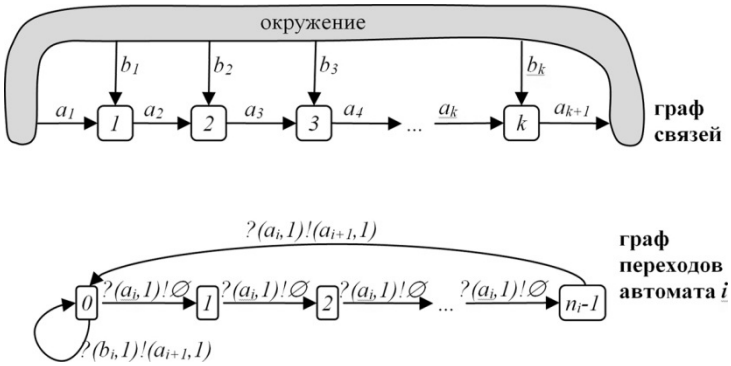

Рис. 1. Пример системы автоматов

Здесь имеется единственное сообщение: $M = \{1\}$. 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 состоянии $j \neq n_i - 1$, получая по входной дуге a_i сообщение 1, переходит в следующее состояние j+1 без выдачи реакции, а в состоянии n_i –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 внешней среде, если $i = k$. Кроме того, в состоянии 0 есть переход-петля по приёму сообщения по дуге b_i с выдачей сообщения следующему автомату $i + 1$ или окружению, если $i = k$.

Рассмотрим состояние композиции вида $s = (s_1, s_2, ..., s_k, \varnothing)$, где состояние i-го автомата $s_i = 0, ..., n_{i-1}$. Начальное состояние $s_0 = (0, 0, ..., 0, \varnothing)$. Обозначим $s_{\max} = (n_1 - 1, n_2 - 1, ..., n_k - 1, \varnothing)$. 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. Число таких состояний равно $n_1 n_2 ... n_k$. Поскольку все они достижимы из начального состояния, время тестирования композиционной системы равно $\boldsymbol{\Omega}(n_1 n_2 ... n_k)$.

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 с номером $i - 1$. Пункт 2 тест также может выполнить непосредственно, поскольку дуга b_i внешняя. Тем самым, время тестирования всех переходов автоматов-компонентов равно $\boldsymbol{O}(n_1 + ... + n_k)$. Теорема 3 доказана.

Заметим, что для $n_1 = n_2 = ... = n_k = n$ имеем соотношение n^k и nk, т.е. для фиксированного числа k компонентов получаем экспоненциальное уменьшение времени тестирования.

Заключение

Сформулируем направления дальнейших исследований.

1. Оптимизация. Предложенный алгоритм фильтрации строит набор тестов не обязательно оптимальный по времени тестирования и / или числу тестов. Возникает задача поиска оптимального набора, которая, вообще говоря, сводима к задаче о поиске минимального покрытия [8, 9].

2. Недетерминизм. Нужно определить такие ограничения на недетерминизм системы и / или составляющих её автоматов, которые позволяли бы выполнять полное тестирование за конечное время и разработать соответствующие алгоритмы тестирования. Неплохие решения этой задачи предложены для автономного тестирования, когда автомат находится под непосредственным управлением теста [10–13] (не в контексте окружающей его части системы).

3. Конформность. В данной статье при тестировании проверяется изоморфизм автомата-компонента реализации его спецификации, заданной как автомат. В общем случае между автоматом компонента в реализации и автоматом компонента в спецификации задаётся отношение конформности, которое слабее изоморфизма: квази-редукция, симуляция и т.п. Это требует более сложного алгоритма тестирования. В то же время, если при тестировании мы можем наблюдать состояние реализации (как предполагается в данной статье), то возможно полное автономное тестирование за конечное время для конформности типа редукции или слабой симуляции [2, 3, 10–20]. Для составной системы возникает проблема декомпозиции системных требований, известная также как проблема 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 тестирования компонентов составной системы, когда верна гипотеза о связях.

4. Обобщение. В данной работе дуга графа связей реализует очередь длины 1. Но могут быть и другие дуги: очередей большей, в том числе неограниченной длины, очереди с приоритетами, стеки и т.п. Нужно обобщить понятие дуги с помощью определения автомата дуги. Более того, автомат дуги мог бы иметь несколько входов и выходов, как автомат вершины. Композиция должна быть определена для пары автоматов, выход одного из которых соединён с входом другого. На таком соединении происходит синхронное взаимодействие автоматов, где один автомат посылает сообщение тогда и только тогда, когда другой автомат это сообщение принимает. Для детерминизма системы, по-видимому, к автомату дуги нужно предъявить дополнительные (по сравнению с автоматом вершины) требования.

ЛИТЕРАТУРА

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. Бурдонов И.Б., Косачев А.С., Кулямин В.В. Теория соответствия для систем с блокировками и разрушением. М. : Наука, 2008. 412 с.
3. Бурдонов И.Б. Теория конформности (функциональное тестирование программных систем на основе формальных моделей). LAP Lambert Academic Publishing, 2011. 428 с.
4. Бурдонов И.Б., Косачев А.С. Пополнение спецификации для ioco // Программирование. 2011. № 1. С. 3–18.
5. Petrenko A., Yevtushenko N., Von Bochmann G., Dssoul Ri. Testing in context: framework and test derivation // Computer Communications. 1996. V. 19(14). P. 1236–1249.
6. Камкин А., Чупилко М. Обзор современных технологий имитационной верификации аппаратуры // Программирование. 2011. № 3. С. 42–49.
7. Бурдонов И.Б., Косачев А.С., Кулямин В.В. Неизбыточные алгоритмы обхода ориентированных графов. Детерминированный случай // Программирование. 2003. № 5. С. 59–69.
8. Левитин А.В. Алгоритмы: введение в разработку и анализ. М. : Вильямс, 2006. С. 160–163.
9. Кормен Т.К., Лейзерсон Ч.И., Ривест Р.Л., Штайн К. Алгоритмы: построение и анализ. 2-е изд. М. : Вильямс, 2006. С. 456–458.
10. Бурдонов И.Б., Косачев А.С. Полное тестирование с открытым состоянием ограниченно недетерминированных систем // Программирование. 2009. № 6. С. 3–18.
11. Бурдонов И.Б., Косачев А.С. Семантики взаимодействия с отказами, дивергенцией и разрушением. Ч. 2. Условия конечного полного тестирования // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. 2011. № 2 (15). С. 89–98.

12. Бурдонов И.Б., Косачев А.С. Тестирование конформности на основе соответствия состояний // Труды ИСП РАН. 2010. № 18. С. 183–220.
13. Бурдонов И.Б., Косачев А.С. Безопасное тестирование симуляции систем с отказами и разрушением // Моделирование и анализ информационных систем. 2010. Т. 17(4). С. 27–40.
14. Bourdonov I.B., Kossatchev A.S., Kuliamin V.V. Formal Conformance Testing of Systems with Refused Inputs and Forbidden Actions // Proceedings of the Workshop on Model Based Testing (MBT 2004). Elsevier, 2006.
15. Бурдонов И.Б., Косачев А.С., Кулямин В.В. Формализация тестового эксперимента // Программирование. 2007. № 5. С. 3–32.
16. Бурдонов И.Б., Косачев А.С., Кулямин В.В. Безопасность, верификация и теория конформности // Материалы Второй международной научной конференции по проблемам безопасности и противодействия терроризму. МГУ, 2006. М. : МЦНМО, 2007. С. 135–158.
17. Бурдонов И.Б., Косачев А.С. Системы с приоритетами: конформность, тестирование, композиция // Труды ИСП РАН. 2008. Т. 14 (1). С. 23–54.
18. Бурдонов И.Б., Косачев А.С. Тестирование с преобразованием семантик // Труды ИСП РАН. 2009. Т. 17. С.193–208.
19. Kossachev A., Burdonov I. Formal Conformance Verifcation, Short Papers of the 22nd IFIP ICTSS / eds. by Alexandre Petrenko, Adenilso Simao, Jose Carlos Maldonado. Natal, Brazil, 2010. P. 1–6.
20. Бурдонов И.Б., Косачев А.С. Семантики взаимодействия с отказами, дивергенцией и разрушением // Программирование. 2010. № 5. С. 3–23.
21. Бурдонов И.Б., Косачев А.С. Согласование конформности и композиции // Программирование. 2013. № 6. С. 3–15.

Бурдонов Игорь Борисович, д-р физ.-мат. наук. E-mail: igor@ispras.ru
Косачев Александр Сергеевич, канд. физ.-мат. наук. E-mail: kos@ispras.ru
Институт системного программирования РАН (ИСП РАН) (г. Москва)

Поступила в редакцию 25 июня 2016 г.

Igor Burdonov, Alexander Kossatchev (Institute for System Programming of the Russian Academy of Sciences, Moscow, Russian Federation).
Testing of automata system.
Keywords: directed graph; graph coverage; communicating automata; testing; networks.

DOI: 10.17223/19988605/38/10

The problem of testing of aggregate systems is considered. The system is described as an oriented graph where the nodes correspond to component automata while edges correspond to simplex communication channels. The following hypothesis is assumed: the graph of links is static and the link structure is error-free. At each state, a component automaton can accept and send multiple messages through incoming and outgoing edges (at most one message through each edge). The goal of testing is to cover transitions of component automata reachable during the system work. It is assumed that during testing it is possible to observe the state changes of automata and the messages on the edges. The general model is considered when the system can simultaneously contain multiple messages, but not more than one for each edge. An automata composition 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. A test is rejected if it covers only transitions of the components that are covered by other tests. A simplified system model with only one message in transit is considered at the end. Using this example, we show that the hypothesis on links allows considerably reduce the number of required testing actions from the product of state numbers of the component automata to the sum of these numbers. If all the automata have the same number of states then it gives the exponential reduction of the number of test actions. In conclusion, some directions of future research are considered.

REFERENCES

1. Revised Working Draft on “Framework: Formal Methods in Conformance Testing”. (1995) *JTC1/SC21/WG1/Project 54/1, ISO Interim Meeting, ITU-T on*. Paris.
2. Burdonov, I.B., Kosachev, A.S. & Kulyamin, V.V. (2008) *Teoriya sootvetstviya dlya sistem s blokirovkami i razrusheniem* [Conformance theory of the systems with refused inputs and forbidden actions]. Moscow: Nauka.
3. Bourdonov, I.B. (2011) *Teoriya konformnosti (funktsional'noe testirovanie programmnykh sistem na osnove formal'nykh modeley)* [Conformance theory (functional testing on formal model base)]. LAP Lambert Academic Publishing.
4. Bourdonov, I.B. & Kosachev, A.S. (2011) Popolnenie spetsifikatsii dlya ioco [Specification Completion for IOCO]. *Programmirovanie – Programming and Computer Software*. 37(1). pp. 3–18.
5. Petrenko, A., Yevtushenko, N., Von Bochmann, G. & Dssoul, Ri. (1996) Testing in context: framework and test derivation. *Computer Communications*. 19(14). pp. 1236–1249. DOI: 10.1016/S0140-3664(96)01157-7
6. Kamkin, A. & Chupilko, M. (2011) Obzor sovremennykh tekhnologiy imitatsionnoy verifikatsii apparatury [Survey of modern technologies of simulation-based verification of hardware]. *Programmirovanie – Programming and Computer Software*. 37(3). pp. 42–49.
7. Bourdonov, I.B., Kossachev, A.S. & Kulyamin, V.V. (2005) Neizbytochnye algoritmy obkhoda orientirovannykh grafov. Determinirovannyy sluchay [Irredundant Algorithms for Traversing Directed Graphs: The Deterministic Case]. *Programmirovanie – Programming and Computer Software*. 29(5). pp. 59–69.

8. Levitin, A.V. (2006) *Algoritmy: vvedenie v razrabotku i analiz* [Introduction to the design and analysis of algorithms]. Moscow: Vil'yams. pp. 160–163.
9. Cormen, T., Leiserson, C., Rivest, R. & Stein, C. (2001) *Algoritmy: postroenie i analiz* [Introduction to Algorithms]. 2nd ed. Translated from Enlgish by I. Krasikov. Moscow: Vil'yams. pp. 456–458.
10. Bourdonov, I.B. & Kossachev, A.S. (2009) Polnoe testirovanie s otkrytym sostoyaniem ogranichenno nedeterminirovannykh sistem [Complete open-state testing of limitedly nondeterministic systems]. *Programmirovanie – Programming and Computer Software*. 6. pp. 3–18.
11. Bourdonov, I.B. & Kossatchev, A.S. (2001) 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*. 2(15). pp. 89–98. (In Russian).
12. Bourdonov, I.B. & Kossatchev, A.S. (2010) Testirovanie konformnosti na osnove sootvetstviya sostoyaniy [Conformance testing based on a state relation]. *Trudy ISP RAN – Proceedings of the Institute for System Programming of the RAS*. 18. pp. 183–220.
13. Bourdonov, I.B. & Kossatchev, A.S. (2010) Safe simulation testing of systems with refusals and destructions. *Automatic Control and Computer Sciences* 17(4). pp. 27–40. DOI: 10.3103/S0146411611070042
14. Bourdonov, I.B., Kossatchev, A.S. & Kuliamin, V.V. (2006) Formal Conformance Testing of Systems with Refused Inputs and Forbidden Actions. *Electronic Notes in Theoretical Computer S*cience. 164(4). pp. 1–128. DOI: 10.1016/j.entcs.2006.09.008
15. Bourdonov, I.B., Kosachev, A.S. & Kuliamin, V.V. (2007) Formalizatsiya testovogo eksperimenta [Formalization of Test Experiments]. *Programmirovanie – Programming and Computer Software*. 5. pp. 3–32.
16. Bourdonov, I.B., Kossachev, A.S. & Kuliamin, V.V. (2006) [Safety, verification and conformance theory]. *The Proceedings of the Second International Conference on the Problems of Safety and Counteraction Against Terrorism*. Moscow: MTsNMO. pp. 135–158. (In Russian).
17. Bourdonov, I.B. & Kossatchev, A.S. (2008) Sistemy s prioritetami: konformnost', testirovanie, kompozitsiya [Systems with priority: conformance, testing, composition]. *Trudy ISP RAN – Proceedings of the Institute for System Programming of the RAS*. 14(1). pp. 23–54.
18. Burdonov, I.B. & Kosatchev, A.S. (2009) Testirovanie s preobrazovaniem semantik [Testing with semantics conversion]. *Trudy ISP RAN – Proceedings of the Institute for System Programming of the RAS*. 17. pp. 193–208.
19. Kossachev, A. & Bourdonov, I. (2010) Formal Conformance Verifcation. In: Petrenko, A., Simao, A. & Maldonado, J.C. (2010) *Short Papers of the 22nd IFIP ICTSS*. Natal, Brazil. pp. 1–6.
20. Bourdonov, I.B. & Kossachev, A.S. (2010) Semantiki vzaimodeystviya s otkazami, divergentsiey i razrusheniem [Interaction semantics with refusals, divergence, and destruction]. *Programmirovanie – Programming and Computer Software*. 5. pp. 3–23.
21. Bourdonov, I.B. & Kossachev, A.S. (2013) Soglasovanie konformnosti i kompozitsii [Agreement between conformance and composition]. *Programmirovanie – Programming and Computer Software*. 6. pp. 3–15.