Модел. и анализ информ. систем. Т. **17**, № **4** (2010) 27–40

УДК 517.51+514.17

Безопасное тестирование симуляции систем с отказами и разрушением

Бурдонов И.Б., Косачев А.С.[1]

Институт системного программирования РАН

e-mail: igor@ispras.ru, kos@ispras.ru

получена 13 октября 2010

Ключевые слова: формальные модели, тестирование по формальным моделям, полное тестирование, симуляция, системы с отказами и разрушением

Статья посвящена тестированию соответствия (конформности) реализации требованиям спецификации. Идея безопасного тестирования предложена авторами для конформности, основанной на трассах наблюдений. Эта идея распространяется на случай (слабой) симуляции, основанной на соответствии состояний реализации и спецификации. Предлагается теория безопасной симуляции для систем с отказами и разрушением. Обсуждаются вопросы полноты тестирования и достаточные условия существования полного набора тестов. Предлагается алгоритм полного тестирования для практического применения, опирающийся на некоторые ограничения на реализацию и спецификацию.

1. Введение

Тестирование конформности – это проверка в процессе эксперимента соответствия (конформности) реализации требованиям, заданным в виде спецификации. Это соответствие определяется семантикой тестового взаимодействия, которая описывает возможные тестовые воздействия и возможные наблюдения ответного поведения реализации.

Если в ответ на тестовые воздействия можно наблюдать только (внешние) действия, выполняемые реализацией, или отсутствие таких действий (отказ), то конформность определяется через трассы наблюдений – последовательностей действий и отказов. Спецификация описывает трассы, которые допускаются в реализации.

Тестирование, при котором гарантировано конечное время ожидания наблюдения после тестового воздействия, называется безопасным. Возможны две причины бесконечного ожидания: дивергенция и ненаблюдаемые отказы. Дивергенция — это

[1]Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 10-07-00147-а).

бесконечное выполнение реализацией внутренних (ненаблюдаемых) действий. Ненаблюдаемый отказ — это отсутствие выполняемых реализацией внешних действий, которое тест не может определить за конечное время[2]. В обоих случаях тест не может ни продолжить тестирование, ни закончить его, так как неизвестно, нужно ли ждать наблюдения или никакого наблюдения не будет.

Мы также вводим специальное, не регулируемое тестовыми воздействиями, действие реализации, называемое *разрушением*. Оно моделирует любое нежелательное поведение системы, в том числе и ее реальное разрушение. Тестирование, при котором не возникает дивергенции, ненаблюдаемых отказов и разрушения, называется безопасным.

Спецификация описывает ситуации, когда тестовое воздействие должно быть безопасно в реализации. Соответственно, конформность основана только на безопасном поведении реализации. Общая теория такой конформности развита в работах авторов [1, 2, 3].

В литературе также рассматриваются конформности, основанные на соответствии состояний реализации и спецификации (обзор см. в [5]) и называемые симуляциями. Симуляция требует, чтобы правильным было не только наблюдаемое внешнее поведение реализации, но и изменение ее состояний. Все рассматриваемые в литературе симуляции либо не учитывают безопасности, предполагая отсутствие дивергенции и ненаблюдаемых отказов, либо рассчитаны на прямое наблюдение дивергенции и всех отказов. Также они не учитывают возможность разрушения.

Целью данной работы является распространение общего подхода, учитывающего отказы и разрушение, на симуляции. Спецификация определяет не только класс конформных ей реализаций, но и более широкий класс реализаций, которые можно безопасно тестировать для проверки конформности (гипотеза о безопасности).

Выбор симуляции в качестве конформности наиболее естественен, когда состояния реализации доступны для их наблюдения. Тестирование, при котором в любой момент времени можно опросить текущее состояние реализации, называется тестированием с открытым состоянием. Задача тестирования – обнаружение ошибок в реализации, понимаемое как несоответствие ее поведения спецификационным требованиям. Тестирование полное, если обнаруживается любая ошибка и не фиксируются «ложные» ошибки. В данной статье рассматривается полное тестирование с открытым состоянием безопасной симуляции.

Это рассмотрение проводится как в общетеоретическом, так и в практическом планах. Теоретическое полное тестирование должно обнаруживать любую ошибку за конечное время, но при отсутствии ошибок может продолжаться бесконечно. Причины бесконечного тестирования – это бесконечность реализации и/или спецификации, а также неограниченный недетерминизм поведения реализации. При некоторых ограничениях возможно построение полных тестов, завершающих свою работу за конечное время. Такие тесты уже можно использовать на практике.

2–4 разделы статьи содержат основные положения теории конформности: семантика взаимодействия и безопасное тестирование, математическая модель реа-

[2]Например, если интервал времени между тестовым воздействием и ответным внешним действием ограничен некоторым тайм-аутом, то превышение тайм-аута при ожидании внешних действий означает наблюдение отказа. При отсутствии такого рода ограничений отказ ненаблюдаем.

лизации и спецификации, определение симуляции, гипотеза о безопасности и определение безопасной симуляции. В 5-м разделе рассматривается связь безопасной симуляции с трассовой конформностью. В 6-м разделе рассматривается полнота тестирования. 7-й раздел посвящен теоретическому, а 8-й раздел — практическому тестированию. Определяются ограничения на реализацию и спецификацию, позволяющие полное тестирование за конечное время, и приводится алгоритм такого тестирования. В разделе 9 приводится пример верификации симуляции.

2. Семантика безопасного взаимодействия

Семантика взаимодействия формализуется в терминах *внешних действий и кнопок*. Действие – это поведение реализации, наблюдаемое в ответ на внешнее воздействие. Множество действий называется алфавитом действий и обозначается $\mathbf{L}$. Кнопка – это подмножество $P \subseteq L$; нажатие кнопки P моделирует воздействие на реализацию, сводящееся к разрешению выполнять любое действие из P. Наблюдаться может либо действие $a \in P$, либо (для некоторых кнопок) отсутствие таких действий, называемое отказом P. Семантика взаимодействия задается алфавитом $\mathbf{L}$ и двумя наборами кнопок: с наблюдением соответствующих отказов – семейство $\mathbf{R} \subseteq 2^{\mathbf{L}}$ и без наблюдения отказов – семейство $\mathbf{Q} \subseteq 2^{\mathbf{L}}$. Предполагается, что $\mathbf{R} \cap \mathbf{Q} = \varnothing$ и $(\cup\mathbf{R}) \cup (\cup\mathbf{Q}) = \mathbf{L}$.

При нажатии кнопки $Q \in \mathbf{Q}$ в общем случае неизвестно, нужно ли ждать наблюдения $a \in Q$, или никакого наблюдения не будет, поскольку возник ненаблюдаемый отказ Q. При правильном взаимодействии такая кнопка нажимается, только если в реализации нет отказа.

Кроме внешних действий реализация может совершать внутренние (ненаблюдаемые) действия, обозначаемые τ. Эти действия всегда разрешены. Предполагается, что любая конечная последовательность любых действий совершается за конечное время, а бесконечная – за бесконечное время. Бесконечная последовательность τ-действий («зацикливание») называется *дивергенцией* и обозначается Δ. Дивергенция сама по себе не опасна, но при попытке выхода из нее (нажатии любой кнопки) неизвестно, нужно ли ждать наблюдения или бесконечно долго будут выполняться τ-действия. Поэтому при правильном взаимодействии кнопки нажимаются, только если в реализации нет дивергенции.

Мы также вводим специальное, не регулируемое кнопками, действие, называемое *разрушением* и обозначаемое γ. Оно моделирует любое нежелательное поведение системы, в том числе и ее реальное разрушение. Семантика разрушения предполагает, что правильное взаимодействие должно его избегать.

Такое правильное взаимодействие, при котором не возникает ненаблюдаемых отказов, попыток выхода из дивергенции и разрушения, называется безопасным.

3. LTS-модель

В качестве модели реализации и спецификации используется LTS (Labelled Transition System), определяемая как $\mathbf{S} = LTS(V_{\mathbf{S}}, \mathbf{L}, E_{\mathbf{S}}, s_0)$, где $V_{\mathbf{S}}$ – непустое множество со-

стояний, $\mathbf{L}$ – алфавит внешних действий, $E_{\mathbf{S}} \subseteq V_{\mathbf{S}} \times (\mathbf{L} \cup \{\tau, \gamma\}) \times V_{\mathbf{S}}$ – множество переходов, $s_0 \in V_{\mathbf{S}}$ – начальное состояние. Переход из состояния s в состояние s' по действию z обозначается $s \xrightarrow{z} s'$. Маршрут – это цепочка смежных переходов: первый переход начинается в начальном состоянии, а каждый другой переход – в конце предыдущего перехода.

Состояние *дивергентно*, если в нем начинается бесконечный τ-маршрут. Состояние *стабильно*, если из него не выходят τ- и γ-переходы. Отказ $P \in \mathbf{R} \cup \mathbf{Q}$ порождается стабильным состоянием, из которого нет переходов по действиям из P.

Для определения трасс (с отказами из $\mathbf{R} \cup \mathbf{Q}$) LTS $\mathbf{S}$ в каждом ее стабильном состоянии добавляются виртуальные петли $s \xrightarrow{P} s$, помеченные порождаемыми отказами, и Δ-петли в дивергентных состояниях $s \xrightarrow{\Delta} s$. Затем рассматриваются маршруты, не продолжающиеся после γ- и Δ-переходов, и трассой называется последовательность пометок на переходах такого маршрута с пропуском символов τ. Обозначим для $s, s' \in V_{\mathbf{S}}, u \in \mathbf{L} \cup \mathbf{R} \cup \mathbf{Q} \cup \{\gamma, \Delta\}$, $\sigma = <u_1, \ldots, u_n> \in (\mathbf{L} \cup \mathbf{R} \cup \mathbf{Q} \cup \{\gamma, \Delta\})^*$:

$s \Rightarrow s' \stackrel{\Delta}{=} s = s' \vee \exists s_1, ..., s_n \; s = s_1 \xrightarrow{\tau} s_2 \xrightarrow{\tau} ... \xrightarrow{\tau} s_n = s'$,

$s \stackrel{<u>}{\Longrightarrow} s' \stackrel{\Delta}{=} \exists s_1, s_2 \; s \Rightarrow s_1 \xrightarrow{u} s_2 \Rightarrow s'$,

$s \stackrel{\sigma}{\Longrightarrow} s' \stackrel{\Delta}{=} \exists s_1, ..., s_{n+1} \; s = s_1 \stackrel{<u_1>}{\Longrightarrow} s_2 ... s_n \stackrel{<u_n>}{\Longrightarrow} s_{n+1} = s'$,

$s \stackrel{\sigma}{\Longrightarrow} \stackrel{\Delta}{=} \exists s' \; s \stackrel{\sigma}{\Longrightarrow} s'$,

$s =\sigma\not\Rightarrow \stackrel{\Delta}{=} \nexists s' \; s \stackrel{\sigma}{\Longrightarrow} s'$,

$s \; \textbf{\textit{after}} \; \sigma \stackrel{\Delta}{=} \{s' \mid s \stackrel{\sigma}{\Longrightarrow} s'\}$.

4. Слабая симуляция

Симуляция требует, чтобы каждое наблюдение u, возможное в реализационном состоянии i с постсостоянием i', было возможно в каждом соответствующем ему спецификационном состоянии s, и в спецификации для s и u нашлось бы постсостояние s', соответствующее i'. Разные симуляции отличаются друг от друга, главным образом, отношением к наблюдаемости внутренних действий (τ). В данной статье мы исходим из основного допущения о принципиальной ненаблюдаемости τ-действий: при взаимодействии невозможно различить наличие и отсутствие τ-действий как до, так и после внешнего действия. Этому соответствует слабая симуляция (weak simulation), называемая также наблюдаемой симуляцией (observation simulation). Дадим три эквивалентных определения слабой симуляции (два первых принадлежат Милнеру [6, 7]) .

$\mathbf{I} \leq^1_{ws} \mathbf{S} \stackrel{\Delta}{=} \exists R \subseteq V_{\mathbf{I}} \times V_{\mathbf{S}} \; (i_0, s_0) \in R \; \& \; \forall (i, s) \in R \; \forall \sigma \in \mathbf{L}^* \; \forall i'$
$(i \stackrel{\sigma}{\Longrightarrow} \Rightarrow \; \exists \, s \stackrel{\sigma}{\Longrightarrow} s' \; \& \; (i', s') \in R)$
(рис. 1 слева).

$\mathbf{I} \leq^2_{ws} \mathbf{S} \stackrel{\Delta}{=} \exists R \subseteq V_{\mathbf{I}} \times V_{\mathbf{S}} \; (i_0, s_0) \in R \; \& \; \forall (i, s) \in R \; \forall u \in \mathbf{L} \; \forall i'$
$(i \xrightarrow{\tau} i' \; \Rightarrow \; \exists s' s \Rightarrow s' \; \& \; (i', s') \in R)$
$\&\; (i \xrightarrow{u} i' \; \Rightarrow \; \exists s' s \stackrel{u}{\Longrightarrow} s' \; \& \; (i', s') \in R)$
(рис. 1 в центре).

$\mathbf{I} \leq^3_{ws} \mathbf{S} \stackrel{\Delta}{=} \exists R \subseteq V_{\mathbf{I}} \times V_{\mathbf{S}} \; (i_0, s_0) \in R \; \& \; \forall (i, s) \in R \; \forall u \in \mathbf{L} \; \forall i'$

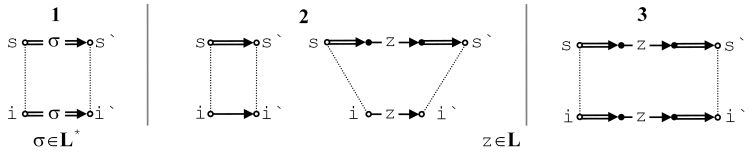

Рис. 1. Три определения слабой симуляции

$(i \stackrel{<u>}{\Longrightarrow} i' \;\Rightarrow\; \exists s' s \stackrel{<u>}{\Longrightarrow} s' \;\&\; (i', s') \in R)$

(рис. 1 справа).

Соответствие R, для которого выполнены условия слабой симуляции, называется *конформным соответствием.*

Отказы. Если под наблюдениями понимать не только внешние действия из $\mathbf{L}$, но и наблюдаемые отказы из $\mathbf{R}$, то определение слабой симуляции модифицируется (изменения по сравнению с $\leq^3_{ws}$ подчеркнуты):

$\mathbf{I} \leq^4_{ws} \mathbf{S} \stackrel{\Delta}{=} \exists R \subseteq V_\mathbf{I} \times V_\mathbf{S}\ (i_0, s_0) \in R \;\&\; \forall (i, s) \in R\ \forall u \in \mathbf{L} \underline{\cup \mathbf{R}}\ \forall i'$

$(i \stackrel{<u>}{\Longrightarrow} i' \;\Rightarrow\; \exists s' s \stackrel{<u>}{\Longrightarrow} s' \;\&\; (i', s') \in R)$

На классе реализаций без наблюдаемых отказов эти соответствия совпадают: $\leq^3_{ws} = \leq^4_{ws}$.

Безопасность. Состояние s назовем *безопасным*, если в этом состоянии не начинается γ-трасса: $s = \gamma \not\Rightarrow$. При безопасном взаимодействии проходятся только безопасные состояния реализации. Кнопку $P \in \mathbf{R} \cup \mathbf{Q}$ назовем *безопасной в (безопасном) состоянии* s, если ее можно нажимать при безопасном взаимодействии:

$P\ \textbf{\textit{safe}}\ s \stackrel{\Delta}{=} s =< \gamma > \not\Rightarrow \;\&\; s =< \Delta > \not\Rightarrow$

$\&\ (P \in \mathbf{Q} \;\Rightarrow\; s =< P > \not\Rightarrow) \;\&\; \forall z \in P\ s =< z, \gamma > \not\Rightarrow$

Наблюдение *безопасно*, если оно разрешается безопасной кнопкой. Состояние *безопасно достижимо*, если оно достижимо из начального состояния последовательностью нажатий безопасных кнопок. Модификация слабой симуляции с отказами и безопасностью выглядит так (изменения по сравнению с $\leq^4_{ws}$ подчеркнуты):

$\mathbf{I} \leq^5_{ws} \mathbf{S} \stackrel{\Delta}{=} \exists R \subseteq V_\mathbf{I} \times V_\mathbf{S}\ (\underline{s =< \gamma > \not\Rightarrow} \Rightarrow (i_0, s_0) \in R)$

$\&\ \forall (i, s) \in R\ \underline{\forall P \textbf{\textit{safe}}\ i}\ \forall u \in \underline{P \cup \{P\}}\ \forall i'$

$(\underline{P \textbf{\textit{safe}}\ s \;\&}\ i \stackrel{<u>}{\Longrightarrow} i' \;\Rightarrow\; \exists s' s \stackrel{<u>}{\Longrightarrow} s' \;\&\; (i', s') \in R)$

На классе реализаций и спецификаций, в которых все отказы наблюдаемы, нет дивергенции и разрушения, эти соответствия совпадают: $\leq^4_{ws} = \leq^5_{ws}$.

Гипотеза о безопасности. Поскольку спецификация задана, по ней можно проверять условие $P\textbf{\textit{safe}}\ s$. Условие $P\textbf{\textit{safe}}\ i$ можно проверять, если реализация также известна. В противном случае (при тестировании) судить о безопасности кнопок в состояниях реализации мы можем только на основании некоторой *гипотезы о безопасности.* Эта гипотеза основана на соответствии $H \subseteq V_\mathbf{I} \times V_\mathbf{S}$ состояний реализации и спецификации, и называется *H-гипотезой.* Она предполагает 1) безопасность начального состояния i_0 реализации, если безопасно начальное состояние s_0 спецификации, 2) безопасность кнопки в состоянии реализации, если она безопасна хотя бы в одном соответствующем по H состоянии спецификации.

Определим соответствие H рекурсивно. Если начальные состояния безопасны, то они, а также любые два состояния, достижимые из начальных состояний по пустой

трассе, соответствуют друг другу. Состояния i' и s' соответствуют друг другу, если они достижимы из соответствующих друг другу состояний i и s по наблюдению u, разрешаемому кнопкой P, безопасной в обоих состояниях i и s. Соответствие H – это минимальное соответствие, порождаемое следующими правилами вывода:

$\forall i, i' \in V_{\mathbf{I}}\ \forall s, s' \in V_{\mathbf{S}}\ \forall P \in \mathbf{R} \cup \mathbf{Q}\ \forall u \in P \cup \{P\}$

$s_0 =\!\!<\gamma>\!\!\not\Rightarrow\ \&\ i_0 =\!\!<\gamma>\!\!\not\Rightarrow\ \&\ i_0 \Rightarrow i\ \&\ s_0 \Rightarrow s\ \ \vdash (i, s) \in H,$

$(i, s) \in H\ \&\ P\,\textbf{\textit{safe}}\ i\ \&\ P\,\textbf{\textit{safe}}\ s$

$\&\ i \stackrel{<u>}{\Longrightarrow} i'\ \&\ s \stackrel{<u>}{\Longrightarrow} s'\ \ \vdash (i', s') \in H.$

Кнопку P будем называть *H-безопасной* в реализационном состоянии i, если она безопасна хотя бы в одном соответствующем i спецификационном состоянии s:

$P\,H\textbf{-}\textbf{\textit{safe}}\ i\ \stackrel{\Delta}{=}\ \exists s\ (i, s) \in H\ \&\ P\ \textbf{\textit{safe}}\ s$

Теперь дадим формальное определение H-гипотезы:

$\mathbf{I}\,H\textbf{-}\textbf{\textit{safe}}\ \mathbf{S}\ \stackrel{\Delta}{=}\ (s_0 =\!\!<\gamma>\!\!\not\Rightarrow\ \&\ i_0 =\!\!<\gamma>\!\!\not\Rightarrow$

$\&\ \forall i \in V_{\mathbf{I}}\ \forall P \in \mathbf{R} \cup \mathbf{Q}\ (P\,H\textbf{-}\textbf{\textit{safe}}\ i\ \Rightarrow\ P\ \textbf{\textit{safe}}\ i).$

Безопасная симуляция. Соединив H-гипотезу о безопасности и слабую симуляцию, получаем *безопасную симуляцию*, которую обозначим ***ss*** (изменения по сравнению с $\leq^5_{ws}$ подчеркнуты):

$\mathbf{I}\ \textbf{\textit{ss}}\ \mathbf{S}\ \stackrel{\Delta}{=}\ \underline{\mathbf{I}\,H\textbf{-}\textbf{\textit{safe}}\ \mathbf{S}}\ \&\ \exists R \subseteq V_{\mathbf{I}} \times V_{\mathbf{S}}\ (\underline{s =\!\!<\gamma>\!\!\not\Rightarrow \Rightarrow}\ (i_0, s_0) \in R)$

$\&\ \forall (i, s) \in R\ \forall P\textbf{\textit{safe}}\ \ \mathrm{i}\ \ \forall u \in \mathbf{L} \cup \mathbf{R}\ \forall i'$

$(P\textbf{\textit{safe}}\ s\ \&\ i \stackrel{<u>}{\Longrightarrow} i'\ \Rightarrow\ \exists s' s \stackrel{<u>}{\Longrightarrow} s'\ \&\ (i', s') \in R)$

Отношение ***ss*** транзитивно и на классе спецификаций, удовлетворяющих собственной H-гипотезе, рефлексивно, то есть является предпорядком.

Если реализация задана явно, можно аналитически проверять как H-гипотезу, так и безопасную симуляцию. Когда реализация неизвестна, требуется тестирование, а H-гипотеза становится предусловием безопасности тестирования. Если $s_0 \stackrel{<\gamma>}{\Longrightarrow}$, то $H = \varnothing$, безопасное тестирование невозможно, но и не нужно, так как любая реализация конформна (при любом R). Если $s_0 =\!\!<\gamma>\!\!\not\Rightarrow$, то тестирование заключается в проверке *тестируемого условия* (нижние две строки определения ***ss***). Нажимается каждая кнопка $P\,H\textbf{-}\textbf{\textit{safe}}\ i$, и полученные наблюдение u и постсостояние i' проверяются по спецификации: наблюдение u должно быть в каждом состоянии спецификации s, которое соответствует по R состоянию i, и в котором кнопка P безопасна, а среди постсостояний s' хотя бы одно должно соответствовать i' по R.

Для класса спецификаций без ненаблюдаемых отказов, дивергенции и разрушения, имеем: $H\textbf{-}\textbf{\textit{safe}} \cap\ \leq^5_{ws} =\ \textbf{\textit{ss}}$, а на поддомене безопасных реализаций $\leq^5_{ws} =\ \textbf{\textit{ss}}$.

Для конформного по ***ss*** соответствия R соответствие $R \cap H$ тоже конформно. Мы можем переформулировать определение безопасной симуляции следующим образом:

$\mathbf{I}\ \textbf{\textit{ss}}\ \mathbf{S}\ \stackrel{\Delta}{=}\ \mathbf{I}\,H\textbf{-}\textbf{\textit{safe}}\ \mathbf{S}\ \&\ \exists R \in H\ (\underline{s =\!\!<\gamma>\!\!\not\Rightarrow\ \Rightarrow}\ (i_0, s_0) \in R)$

$\&\ \forall (i, s) \in R\ \forall P\textbf{\textit{safe}}\ \ \mathrm{s}\ \forall u \in P \cup \{P\}\ \forall i'$

$(i \stackrel{<u>}{\Longrightarrow} i'\ \Rightarrow\ \exists s' s \stackrel{<u>}{\Longrightarrow} s'\ \&\ (i', s') \in R)$

Мы можем ограничиться такими соответствиями R, которые вложены в H. Объединение конформных по ***ss*** соответствий конформно, что дает два естественных конформных соответствия: R_1 – объединение всех конформных соответствий, и $R_2 = R_1 \cap H$.

5. Связь симуляции с трассовой конформностью

В трассовой теории конформности гипотеза о безопасности основывается на трассах реализации и спецификации [1, 2, 3] и не требует соответствия состояний. Воспроизведем здесь основные определения этой теории. Для LTS $\mathbf{S}$ множество ее трасс с отказами из $\mathbf{R}\cup\mathbf{Q}$ обозначим через $\mathbf{T}(\mathbf{S})$. Для реализации $\mathbf{I}$ определяется отношение ***safe in*** безопасности кнопки $P \in \mathbf{R}\cup\mathbf{Q}$ после трассы $\sigma \in \mathbf{T}(\mathbf{I})$:

P ***safe in*** $\mathbf{I}$ ***after*** $\sigma \triangleq (P \in \mathbf{R} \vee \sigma\bullet\!<P>\!\notin \mathbf{T}(\mathbf{I})$
$\&\ \forall z \in P\ \sigma\bullet\!<z,\gamma>\!\notin \mathbf{T}(\mathbf{I})\ \&\ \sigma\bullet\!<\Delta>\!\notin \mathbf{T}(\mathbf{I})$

Очевидно, что если кнопка безопасна по ***safe in*** после трассы (P ***safe in*** $\mathbf{I}$ ***after*** σ), то она безопасна в каждом состоянии после этой трассы: $\forall i \in (\mathbf{I}$ ***after*** $\sigma)\ P$ ***safe*** i.

Для спецификации отношение ***safe by*** безопасности кнопок после трасс определяется неоднозначно: это любое отношение, удовлетворяющее трем требованиям: $\forall\sigma \in \mathbf{T}(\mathbf{S})\ \forall R \in \mathbf{R}; \forall z \in \mathbf{L}; \forall Q \in \mathbf{Q}$

1) P ***safe by*** $\mathbf{S}$ ***after*** $\sigma \Leftrightarrow \forall u \in R\ \sigma\bullet\!<u,\gamma>\!\notin \mathbf{T}(\mathbf{S})\ \&\ \sigma\bullet\!<\Delta>\!\notin \mathbf{T}(\mathbf{S})$

2) $\sigma\bullet\!<z>\!\in \mathbf{T}(\mathbf{S})\ \&\ T \in \mathbf{R}\cup\mathbf{Q}\ z \in T\ \&\ \forall u \in T\ \sigma\bullet\!<u,\gamma>\!\notin \mathbf{T}(\mathbf{S})$
$\&\ \sigma\bullet\!<\Delta>\!\notin \mathbf{T}(\mathbf{S}) \Rightarrow \exists P \in \mathbf{R}\cup\mathbf{Q}\ z \in P\ \&\ P$ ***safe by*** $\mathbf{S}$ ***after*** σ ,

3) Q ***safe by*** $\mathbf{S}$ ***after*** $\sigma \Leftrightarrow \exists v \in Q\ \sigma\bullet\!<v>\!\in \mathbf{T}(\mathbf{S})$
$\&\ \forall u \in Q\ \sigma\bullet\!<u,\gamma>\!\notin \mathbf{T}(\mathbf{S})\ \&\ \sigma\bullet\!<\Delta>\!\notin \mathbf{T}(\mathbf{S})$

Будем считать, что вместе со спецификацией задано отношение ***safe by***, отвечающее этим трем требованиям.

Трасса σ спецификации $\mathbf{S}$ называется безопасной, если спецификация не содержит трассу $<\gamma>$, а трасса σ не заканчивается на дивергенцию и разрушение, и каждый встречающийся в ней символ u является внешним действием или $\mathbf{R}$-отказом, разрешаемым кнопкой, которая безопасна после непосредственно предшествующего этому символу префикса трассы. Множество безопасных трасс обозначим:

SafeBy$(\mathbf{S}) \triangleq \{\sigma \in \mathbf{T}(\mathbf{S})\ |\!<\gamma>\!\notin \mathbf{T}(\mathbf{S})\ \&\ \forall u \in \mathbf{L}\cup\mathbf{R}$
$(\sigma = \mu\bullet\!<u>\!\bullet\lambda \Rightarrow \exists P \in \mathbf{R}\cup\mathbf{Q}\ P$ ***safe by*** $\mathbf{S}$ ***after*** $\mu\ \&\ u \in P\cup\{P\})\}$

Для кнопок $R \in \mathbf{R}$ отношения ***safe by*** и ***safe in*** совпадают. Поэтому, если кнопка $R \in \mathbf{R}$ безопасна по ***safe by*** после некоторой трассы σ (R ***safe by*** $\mathbf{S}$ ***after*** σ), то она безопасна в каждом состоянии после этой трассы: $\forall s \in (\mathbf{S}$ ***after*** $\sigma)\ R$ ***safe*** s. Однако кнопка $Q \in \mathbf{Q}$, которая безопасна по ***safe by*** после трассы, может быть опасна в некоторых (но не всех) состояниях $s \in (\mathbf{S}$ ***after*** $\sigma)$.

Трассовая гипотеза о безопасности определяется следующим образом:

$\mathbf{I}$ ***safe for*** $\mathbf{S} \triangleq (<\gamma>\!\notin \mathbf{T}(\mathbf{S}) \Rightarrow <\gamma>\!\notin \mathbf{T}(\mathbf{I})$
$\&\ \forall\sigma \in$ ***SafeBy***$(\mathbf{S}) \cap \mathbf{T}(\mathbf{I})\ \forall P \in \mathbf{R}\cup\mathbf{Q}$
$(P$ ***safe by*** $\mathbf{S}$ ***after*** $\sigma \Rightarrow P$ ***safe in*** $\mathbf{I}$ ***after*** $\sigma)$

Трассовая конформность определяется так:

$\mathbf{I}$ ***saco*** $\mathbf{S} \triangleq \mathbf{I}$ ***safe for*** $\mathbf{S}\ \&\ \forall\sigma \in$ ***SafeBy***$(\mathbf{S}) \cap \mathbf{T}(\mathbf{I})$
$\forall P$ ***safe by*** $\mathbf{S}$ ***after*** $\sigma\ \forall i \in (\mathbf{I}$ ***after*** $\sigma)\ \forall u \in P\cup\{P\}$
$(i \stackrel{<u>}{\Longrightarrow} \Longrightarrow \exists s \in (\mathbf{S}$ ***after*** $\sigma)\ s \stackrel{<u>}{\Longrightarrow})$

Можно рассматривать симуляцию с трассовой гипотезой о безопасности, которая определяется так (изменения по сравнению с ***ss*** подчеркнуты):

$\mathbf{I}$ ***sst*** $\mathbf{S} \triangleq \mathbf{I}$ (<u>***safe for***</u> $\mathbf{S}\ \&\ \exists R \subseteq V_{\mathbf{I}} \times V_{\mathbf{S}}\ (s_0 =\!<\gamma>\!\not\Rightarrow \Rightarrow\ (i_0, s_0) \in R)$

$\& \forall (i,s) \in R \, \forall \sigma \in \underline{\boldsymbol{SafeBy}(\mathbf{S}) \cap \mathbf{T}(\mathbf{I}) \, \forall P \; \boldsymbol{safe \; by} \; \mathbf{S} \; \boldsymbol{after} \; \sigma}$
$\forall u \in P \cup \{P\} \, \forall i'$
$(P \boldsymbol{safe} \; s \; \& \; \underline{i \in (\mathbf{I} \; \boldsymbol{after} \; \sigma)} \; \& \; i \stackrel{\langle u \rangle}{\Longrightarrow} i' \; \Rightarrow \; \exists s' s \stackrel{\langle u \rangle}{\Longrightarrow} s' \; \& \; (i', s') \in R)$

Симуляция с H-гипотезой о безопасности предъявляет более сильные требования к реализации, чем симуляция с трассовой гипотезой о безопасности. Симуляция с трассовой гипотезой о безопасности, в свою очередь, предъявляет более сильные требования к реализации, чем трассовая конформность с той же гипотезой о безопасности.

Как итог мы имеем следующие соотношения трассовой конформности и безопасных симуляций с различными гипотезами о безопасности: $\boldsymbol{ss} \subset \boldsymbol{sst} \subset \boldsymbol{saco}$. Усиление требований к реализации идет сначала как введение дополнительных требований к соответствию состояний реализации и спецификации при сохранении той же трассовой гипотезы о безопасности $\boldsymbol{sst} \subset \boldsymbol{saco}$. Поскольку гипотеза о безопасности одна и та же, количество безопасных нажатий кнопок в состояниях и, тем самым, число тестовых испытаний одно и то же. Различие лишь в объеме аналитических проверок после тестовых испытаний. После этого идет усиление гипотезы о безопасности $\boldsymbol{ss} \subset \boldsymbol{sst}$, что ведет к большему количеству безопасных нажатий кнопок в состояниях и, тем самым, к большему числу проверок.

6. Полнота тестирования

При тестировании считается, что исследуемая система неизвестна, но *предполагается*, что она имеет модель (данного класса, в статье – LTS), которая при взаимодействии с ней ведет себя так же, как реальная система. В литературе это называется *тестовой гипотезой* [4]. Поэтому реальная система удовлетворяет требованиям, формализуемым в спецификации, тогда и только тогда, когда ее модель (реализация) конформна спецификации.

Для симуляции используется операция опроса состояния реализации: в начале тестирования и после каждого наблюдения (тестирование с открытым состоянием).

Тест – это инструкция, каждый пункт которой описывает либо требуемый рестарт системы[3], либо тестовое воздействие (кнопку) и в зависимости от полученных наблюдения и постсостояния – следующий пункт или вердикт (***pass*** или ***fail***). Реализация *проходит* тест, если ее тестирование всегда (при любом проявлении недетерминизма) не заканчивается с вердиктом ***fail***. Тест *значимый*, если каждая конформная реализация его проходит; *исчерпывающий*, если каждая неконформная реализация его не проходит; *полный*, если он значимый и исчерпывающий.

Ставится задача генерации по спецификации *полных тестов*, которые однозначно определяли бы конформность или неконформность любой реализации (быть может, из некоторого класса).

Для полноты тестирования симуляции ***ss*** нужно для каждого достижимого при безопасном тестировании состояния i реализации и каждой H-безопасной в нем кнопки P верифицировать каждое имеющееся в реализации наблюдение $u \in P \cup \{P\}$ и постсостояние i'. Предполагается, что любую пару (u, i') можно получить за

[3]Тест с рестартом эквивалентен набору тестов, который обычно рассматривается.

конечное число нажатий кнопки P в состоянии i. Также предполагается, что любое состояние $i'_0 \in I_0 = (i_0\ \textbf{\textit{after}}\ <>)$ можно получить за конечное число рестартов. Это называется гипотезой о *глобальном тестировании.*

7. Теоретическое тестирование

Если $s_0 \stackrel{<\gamma>}{\Longrightarrow}$, то все реализации конформны, и тестирование не требуется. Если $s =<\gamma>\not\Rightarrow$, определим минимальное множество N *неконформных пар* состояний (i, s), порождаемое следующими правилами вывода: $\forall (i, s) \in H\ \forall P\ H\textbf{-}\textbf{\textit{safe}}\ i\ \forall u \in P \cup \{P\}$

1. $i \stackrel{<u>}{\Longrightarrow}$ & $P\ \textbf{\textit{safe}}\ s$ & $s =<u>\not\Rightarrow\ \vdash\ (i, s) \in N$
2. $i \stackrel{<u>}{\Longrightarrow} i'$ & $P\ \textbf{\textit{safe}}\ s$ & $\{i'\} \times (s\ \textbf{\textit{after}}\ <u>) \subseteq N\ \ \vdash\ (i, s) \in N$

Конформность $\mathbf{I}\ \textbf{\textit{ss}}\ \mathbf{S}$ эквивалентна условию $(i_0, s_0) \notin N$. Правила вывода определяют граф вывода, вершины которого – пары $(i, s) \in N$, полученные применением 1-го (1-вершина) или 2-го (2-вершина) правила вывода. Помеченная дуга $(i, s) \xrightarrow{(u,i')} (i', s')$ соответствует правилу вывода 2 для $s' \in (s\ \textbf{\textit{after}}\ <u>)$.

В графе вывода существует дерево маршрутов (быть может, не единственное), которое назовем деревом вывода. Каждый маршрут дерева начинается в (i_0, s_0), корень дерева – пустой маршрут, а листья – маршруты, заканчивающиеся в 1-вершинах. Маршрут дерева, заканчивающийся в 2-вершине, продолжается в дереве теми и только теми дугами, которые помечены одной меткой (u, i'), что соответствует 2-му правилу вывода.

Если $(i_0, s_0) \in N$, но все деревья вывода бесконечны, то никакой тест не сможет за конечное время определить неконформность. Поэтому на классе всех H-безопасных реализаций могут существовать только значимые тесты. Один из таких тестов полный на подклассе реализаций, в которых либо не существует дерева вывода (реализация конформна), либо существует конечное дерево вывода (реализация неконформна). Кроме глобального тестирования, требуется перечислимость множества $S_0 = (s_0\ \textbf{\textit{after}}\ <>)$, множества безопасных в каждом безопасно достижимом состоянии s кнопок $P(s) = \{P \in \mathbf{R} \cup \mathbf{Q} \mid P\ \textbf{\textit{safe}}\ s\}$, и множества постсостояний $S(s, u) = (s\ \textbf{\textit{after}}\ <u>)$ для каждого безопасного в s наблюдения u.

Дерево вывода конечно тогда и только тогда, когда оно имеет конечное ветвление, что эквивалентно конечности каждого $s\ \textbf{\textit{after}}\ <u>$. Для этого достаточно, чтобы в каждом безопасно достижимом состоянии s спецификации были конечны:

1) число переходов по каждому безопасному действию, и

2) множество $s\ \textbf{\textit{after}}\ <>$.

8. Практическое тестирование

На практике нужно, чтобы тест заканчивался за конечное время. Полный тест обнаруживает ошибку за конечное время, но при отсутствии ошибок может выполняться бесконечно долго, если не предполагать специальных ограничений. Достаточно следующих ограничений. 1) Число кнопок *конечно* и каждая кнопка *разрешима* отно-

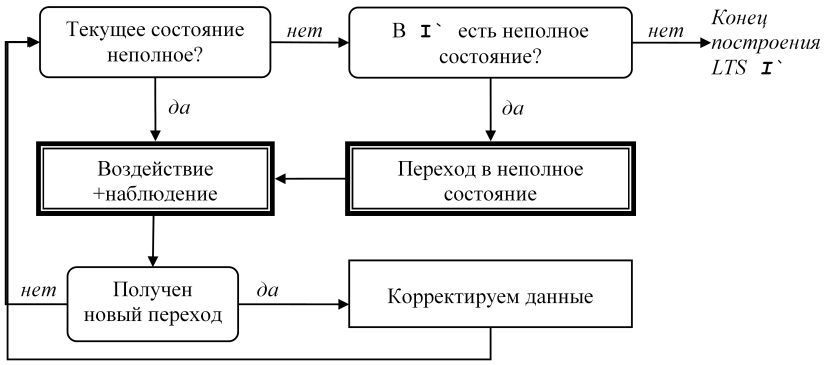

Рис. 2. Схема алгоритма исследования реализации

сительно алфавита действий. 2) Спецификация *конечна*: конечно число состояний и переходов. 3) «Часть» $\mathbf{I}'$ реализации $\mathbf{I}$, «проходимая» при безопасном тестировании, *конечна* и *t-недетерминирована*: все возможные пары (наблюдение, постсостояние) могут быть получены не просто за конечное число нажатий данной кнопки в данном состоянии, как при глобальном тестировании, а не более чем за t нажатий. При $t = 1$ реализация детерминирована.

Условия конечности позволяют сначала исследовать реализацию $\mathbf{I}$, то есть построить $\mathbf{I}'$, а потом провести верификацию. Переход $i \xrightarrow{u} i'$ добавляется в $\mathbf{I}'$ тогда, когда после опроса состояния i нажимается кнопка P, получается наблюдение u и снова опрашивается постсостояние, которым оказывается состояние i'. Эту кнопку P будем называть *управляющей* кнопкой и обозначать $P(i \xrightarrow{u} i')$. Заметим, что мы добавляем не только переходы по внешним действиям, но также виртуальные переходы по наблюдаемым отказам. Если P – это $\mathbf{R}$-кнопка и $u = P$, то добавленный переход $i \xrightarrow{P} i'$ означает, что в реализации либо есть виртуальная петля по отказу $i \xrightarrow{P} i$, если $i = i'$, либо в реализации есть τ-маршрут из i в i' и виртуальная петля по отказу $i' \xrightarrow{P} i'$.

Для каждого пройденного состояния i хранится множество $H(i)$ соответствующих ему по H спецификационных состояний, которое будет расти, и множество безопасных кнопок $P(i) = \cup\{P(s) \mid s \in H(i)\}$. Для каждой кнопки $P \in P(i)$ хранится счетчик $C(i, P)$ числа нажатий кнопки P в состоянии i. Кнопка $P \in P(i)$ *полна в состоянии* i, если $C(i, P) = t$. Состояние i *полно*, если все кнопки из $P(i)$ полны.

В начале тестирования после опроса состояния $i'_0 \in I_0$ имеем:

$H(i'_0) = S_0, P(i'_0) = \cup\{P(s) \mid s \in S_0\}, C(i'_0, P) = 0$ для каждой кнопки $P \in P(i'_0)$.

Сначала проверяем полноту текущего состояния i из $\mathbf{I}'$ (рис. 2). Если состояние неполное, то есть неполная кнопка $P \in P(i)$. Тогда нажимаем кнопку P и получаем переход $i \xrightarrow{u} i'$, постсостояние i' становится новым текущим состоянием. Увеличиваем счетчик $c(P, i) := c(P, i)+1$. Если переход $i \xrightarrow{u} i'$ новый, добавляем его в $\mathbf{I}'$, кнопку P запоминаем как управляющую $P(i \xrightarrow{u} i')$, и обновляем $H(i') := H(i') \cup S(s, u)$ для каждого $s \in H(i)$ при условии P ***safe*** s.

Когда новое состояние s добавляется в $H(i)$, корректируем $P(i) := P(i) \cup P(s)$, и для каждого полученного ранее перехода $i \xrightarrow{u} i'$ обновляем $H(i') := H(i') \cup S(s, u)$ при условии $P(i \xrightarrow{u} i')$ ***safe*** s, отмечая вновь добавленные состояния. Эта рекурсив-

ная процедура повторяется, пока возможно. Условия конечности гарантируют, что процедура закончится за конечное число шагов.

Если текущее состояние i полное, то переходим в любое неполное состояние (если таких состояний нет, построение $\mathbf{I}'$ заканчивается). Для этого выбираем в LTS $\mathbf{I}'$ лес деревьев, покрывающих все состояния и ориентированных к своим корням, которыми являются все неполные состояния. С каждым переходом $i \xrightarrow{u} i'$ этого леса свяжем управляющую кнопку $A(i) = P(i \xrightarrow{u} i')$. Будем двигаться, нажимая в каждом текущем состоянии i кнопку $A(i)$. Из-за недетерминизма мы можем оказаться не в состоянии i', а в другом состоянии i'', где будем нажимать кнопку $A(i'')$. Переход в неполное состояние гарантируется t-недетерминизмом реализации.

Исследование реализации заканчивается за конечное время. Число тестовых воздействий равно $O(bt^n)$ для $t > 1$ и $O(bn^2)$ для $t = 1$, а объем вычислений равен $O(bnt^n) + O(bnm) + O(mk)$ для $t > 1$, для $t = 1$ первое слагаемое заменяется на $O(bn^3)$, где b – число кнопок, n – число состояний реализации, $m = O(bnt)$ – число переходов реализации, k – число состояний спецификации.

Верификация симуляции после построения LTS $\mathbf{I}'$ пытается построить конформное соответствие R_2, выдавая вердикт ***pass***, если такое соответствие существует и построено, или вердикт ***fail***, если такого соответствия быть не может. Сначала строится двудольный граф. Вершины 1-го типа – пары (i, s), где i – состояние $\mathbf{I}'$, а $s \in H(i)$ состояние $\mathbf{S}$. Вершины 2-го типа – пары $(i \xrightarrow{u} i', s)$, где $i \xrightarrow{u} i'$ – переход $\mathbf{I}'$, а $s \in H(i)$. В каждую вершину 2-го типа входит одна дуга 1-го типа $(i, s) \to (i \xrightarrow{u} i', s)$. Дуга 2-го типа $(i \xrightarrow{u} i', s) \to (i', s')$ проводится, если $s' \in S(s, u)$. Одновременно составляется список терминальных вершин 2-го типа.

После построения двудольного графа каждая терминальная вершина v_2 2-го типа удаляется вместе с входящей в нее дугой $v_1 \to v_2$, начальной вершиной v_1 этой дуги, и каждой входящей в нее дугой $v_1' \to v_1$. Одновременно для $v_1 = (i, s)$ состояние s удаляется из множества $H(i)$. Эти операции повторяются, пока не будет удалена вершина 1-го типа (i_0', s_0), где $i_0' \in I_0'$, или пока не будут удалены все терминальные вершины 2-го типа. В первом случае алгоритм заканчивается с вердиктом ***fail***, поскольку $(i_0', s_0) \in N$ влечет $(i_0, s_0) \in N$, а во втором случае – с вердиктом ***pass***. Во втором случае строится соответствие $R_2 = \{(i, s) \mid i \in V_{\mathbf{I}}\text{`\&}s \in H(i)\}$. Если при построении двудольного графа каждое $H(i)$ заменить на множество всех состояний спецификации, то алгоритм будет строить наибольшее конформное соответствие R_1.

Объем вычислений при верификации равен $O(mk^2)$.

9. Пример верификации безопасной симуляции

На рис. 3 приведен пример верификации симуляции. Семантика содержит как **R**-, так и **Q**-кнопки. Спецификация **S** демонстрирует все виды опасности: ненаблюдаемый отказ $\{y\}$ (в состоянии 3), дивергенцию (в состоянии 4) и разрушение (в состоянии 5). Единственное проявление недетерминизма в спецификации – это два перехода $0 \xrightarrow{y} 1$ и $0 \xrightarrow{y} 3$. Но в этих состояниях безопасны одни и те же кнопки (кнопка $\{x\}$), и спецификация удовлетворяет своей H-гипотезе. Поскольку отноше-

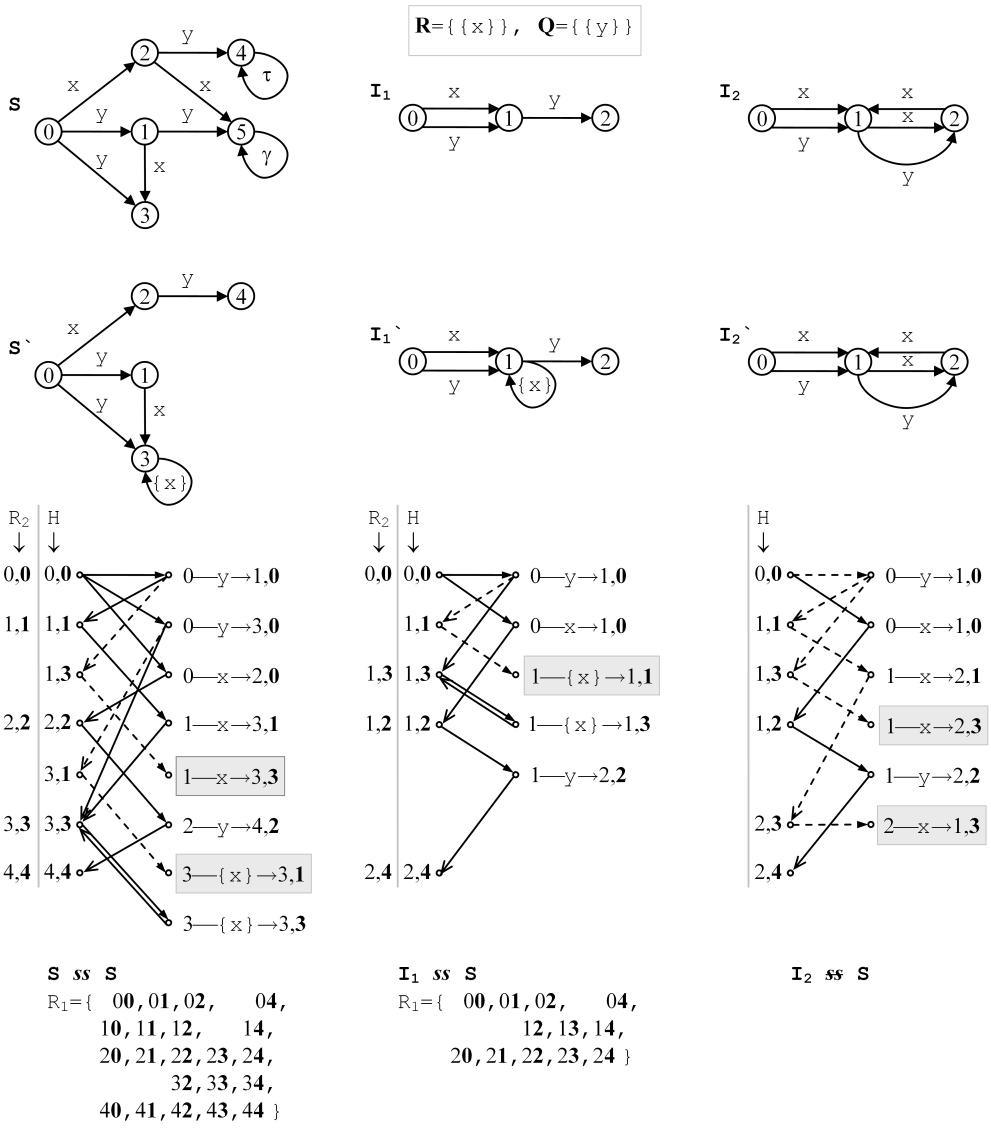

Рис. 3. Пример верификации симуляции

ние ss рефлексивно на классе спецификаций, удовлетворяющих своим H-гипотезам, спецификация $\mathbf{S}$ конформна сама себе.

Кроме конформной реализации $\mathbf{S}$, приведены примеры еще одной конформной реализации $\mathbf{I_1}$ и одной неконформной реализации $\mathbf{I_2}$. После исследования реализаций будут построены LTS $\mathbf{S}'$, $\mathbf{I}_1'$ и $\mathbf{I}_2'$. Они отличаются наличием явных (а не виртуальных) переходов-петель по отказу $\{x\}$. Кроме того, в $\mathbf{S}'$ отсутствуют переходы, которые опасны в спецификации $\mathbf{S}$; соответственно состояние 5 недостижимо в $\mathbf{S}'$, поскольку оно недостижимо в $\mathbf{S}$ по безопасным маршрутам.

Для каждой из этих трех реализаций ниже приведены двудольные графы для верификации соответствия R_2 с указанием соответствия H. Жирным шрифтом выделены состояния спецификации. Для пояснения рассмотрим построение двудольного графа для реализации $\mathbf{I}_1'$. Поскольку начальные состояния реализации и специфика-

ции всегда H-соответствуют друг другу, $(0, \mathbf{0}) \in H$, и пара $(0, \mathbf{0})$ является вершиной 1-го типа. Поскольку в реализации наблюдались переходы $0 \xrightarrow{y} 1$ и $0 \xrightarrow{x} 1$, в двудольный граф добавляются дуги $(0, \mathbf{0}) \to (0 \xrightarrow{y} 1, \mathbf{0})$ и $(0, \mathbf{0}) \to (0 \xrightarrow{x} 1, \mathbf{0})$. Поскольку в спецификации из состояния $\mathbf{0}$ есть два перехода $\mathbf{0} \xrightarrow{y} \mathbf{1}$ и $\mathbf{0} \xrightarrow{y} \mathbf{3}$, в двудольный граф добавляются дуги $(0 \xrightarrow{y} 1, \mathbf{0}) \to (1, \mathbf{1})$ и $(0 \xrightarrow{y} 1, \mathbf{0}) \to (1, \mathbf{3})$. Аналогично, поскольку в спецификации из состояния $\mathbf{0}$ есть переход $\mathbf{0} \xrightarrow{x} \mathbf{2}$, в двудольный граф добавляется дуга $(0 \xrightarrow{x} 1, \mathbf{0}) \to (1, \mathbf{2})$. Поскольку в реализации наблюдался переход по отказу $1 \xrightarrow{\{x\}} 1$, в двудольный граф добавляется дуга $(1, \mathbf{1}) \to (1 \xrightarrow{\{x\}} 1, \mathbf{1})$. И так далее.

После построения двудольного графа анализируем терминальные вершины 2-го типа (отмечены серым фоном). Для реализации $\mathbf{I'_1}$ такая вершина одна: $(1 \xrightarrow{\{x\}} 1, \mathbf{1})$. Она терминальна, так как в спецификации нет соответствующего (по отказу x) перехода из состояния $\mathbf{1}$. Следовательно, пара состояний $(1, \mathbf{1})$ не может принадлежать конформному соответствию. Но тогда для построения $R_2 \subseteq H$ из двудольного графа нужно удалить (единственную) входящую дугу $(1, \mathbf{1}) \to (1 \xrightarrow{\{x\}} 1, \mathbf{1})$, вершину 1-го типа $(1, \mathbf{1})$ и все входящие в нее дуги (в данном случае одну дугу $(0 \xrightarrow{y} 1, \mathbf{0}) \to (1, \mathbf{1})$. Удаляемые дуги отмечены пунктиром. Множество вершин 1-го типа, оставшихся после завершения этого процесса удаления, как раз и составляет конформное соответствие R_2, если в нем остаётся обязательная пара начальных состояний $(0, \mathbf{0})$. Это имеет место для реализаций $\mathbf{S'}$ и $\mathbf{I'_1}$, но не для реализации $\mathbf{I'_2}$.

Для конформных реализаций $\mathbf{S'}$ и $\mathbf{I'_1}$ приведены соответствия R_1 без двудольных графов, которые строятся аналогично, но только вначале вершинами 1-го типа считаются все пары состояний реализации и спецификаций.

Список литературы

1. **Бурдонов И.Б., Косачев А.С., Кулямин В.В.** Формализация тестового эксперимента // Программирование. 2007. № 5.

2. **Бурдонов И.Б.** Теория конформности для функционального тестирования программных систем на основе формальных моделей: Дис. ... д-ра физ.-мат. наук. М., 2008. http://www.ispras.ru/ RedVerst/RedVerst/Publications/TR-01-2007.pdf

3. **Бурдонов И.Б., Косачев А.С.** Полное тестирование с открытым состоянием ограниченно недетерминированных систем // Программирование. 2009. № 6.

4. **Bernot G.** Testing against formal specifications: A theoretical view // S. Abramsky and T.S.E. Maibaum, editors, TAPSOFT'91. Lecture Notes in Computer Science 494, Springer-Verlag, 1991. Volume 2. P. 99–119.

5. **van Glabbeek R.J.** The linear time - branching time spectrum II; the semantics of sequential processes with silent moves. Proceedings CONCUR '93, Hildesheim, Germany, August 1993 (E. Best, ed.), LNCS 715, Springer-Verlag, 1993. P. 66–81.

6. **Milner R.** Lectures on a calculus for communicating systems. Seminar on Concurrency, LNCS 197, Springer-Verlag. P. 197–220.

7. **Milner R.**. Communication and Concurrency, Prentice-Hall International, Englewood Cliffs, 1989.

Safe simulation testing of systems with refusals and destructions

Burdonov I.B., Kosachev A.S.

Keywords: formal model, model based testing, complete testing, simulation, system with refusals and distructions

The paper deals with conformance testing based on formal specifications. The concept of safe testing was earlier proposed by the authors for trace based conformance. This concept is propagated on the case of (weak) simulation based on a relation between specification and implementation states. The theory of safe simulation of systems with refusals and destructions is proposed. The problems of complete testing and sufficient conditions for the existense of complete test suite are discussed. The practical algorithm of complete testing for restricted classes of specifications and implementations is described.

Сведения об авторах:
Бурдонов Игорь Борисович,
Институт системного программирования РАН,
ведущий научный сотрудник;
К**осачев Александр Сергеевич**,
Институт системного программирования РАН,
ведущий научный сотрудник.
Области научных интересов авторов: операционные системы, микроядерные операционные системы, последние годы – тестирование программно-аппаратных систем, основанное на формальных моделях.