И. Бурдонов, А. Косачев
Симуляция систем с отказами и разрушением
The Workshop on Program Semantics, Specification and Verification: Theory and Applications (PSSV 2010, June 14-15, 2010)
affiliated with 5th International Computer Science Symposium in Russia (CSR-2010, June 16-20, 2010)
http://csr2010.antat.ru,
Kazan, Russia.

Слайд 1

Тема доклада - симуляция систем с отказами и разрушением.

Слайд 2

- Правильность исследуемой системы понимается как ее соответствие заданным требованиям.
- Верификация правильности на основе формальных моделей предполагает, что исследуемая система и требования моделируются, соответственно, реализационной и спецификационной моделями.
- Отношение «соответствует» моделируется формальным отношением моделей, называемое конформностью.
- Для того, чтобы конформность можно было проверять при тестировании, требования должны быть функциональными, то есть выражаться в терминах взаимодействия исследуемой системы с окружением.

Слайд 3

Взаимодействие описывается в терминах воздействий на реализацию и ответных наблюдений за ее поведением.

- Под наблюдением понимается некоторое действие, выполняемое реализацией и наблюдаемое извне ее.

Такое действие называется внешним.

- Воздействия, которые могут приводить к одним и тем же наблюдениям, эквивалентны.

Можно считать, что реализация находится внутри некоторой машины тестирования, снабженной клавиатурой и дисплеем.

Воздействие – это нажатие той или иной кнопки клавиатуры, на которой написано множество разрешаемых действий.

Ответное действие наблюдается на экране дисплея.

- Семантика взаимодействия определяется алфавитом внешних действий и набором воздействий, т.е. кнопок.

- Кроме внешних, то есть наблюдаемых, действий реализация может выполнять внутренние, то есть ненаблюдаемые, действия.

Поскольку они не наблюдаемы, они неразличимы между собой и обозначаются одним символом **τ.**

Такие действия всегда разрешены независимо от того, какая кнопка клавиатуры нажата, и нажата ли вообще.

Слайд 4

В качестве модели выбирается система помеченных переходов LTS.

Она определяется как совокупность множества состояний, одно из которых выделено и называется начальным состоянием, алфавита внешних действий
и множества переходов.

- Переход определяется тремя компонентами:
  состоянием в начале перехода – пресостоянием,
  внешним действием или символом τ, которым помечен переход,
  и состоянием в конце перехода – постсостоянием.
- Мы будем обозначать переход с помощью одинарных стрелок.
- Трассой называется последовательность внешних действий.

Двойная стрелка вводится для описания того факта, что из пресостояния по некоторой трассе можно попасть в некоторое постсостояние.

Если LTS недетерминирована, то трасса может заканчиваться в нескольких постсостояниях.

Оператор after описывает множество таких постсостояний после трассы.

Слайд 5

На этом слайде даны три эквивалентных определения конформности, которая называется слабой симуляцией.
Первые два из этих определений принадлежат Милнеру.

- Во всех случаях требуется существование такого соответствия R состояний реализации и спецификации, которое удовлетворяет следующему требованию:

если пресостояния реализации i и спецификации s соответствуют друг другу,
а из пресостояния реализации i по некоторой трассе σ или по некоторому действию достижимо некоторое постсостояние реализации i`, то найдется такое постсостояние спецификации s`, которое также достижимо из пресостояния спецификации s по той же трассе или действию, а, кроме того, соответствует постсостоянию реализации i`.

Такое соответствие R называется конформным соответствием.

Слайд 6

Теперь введем ещё один вид наблюдаемого поведения реализации, которое называется отказом (refusal).

- Оно означает остановку реализации из-за отсутствия в ней разрешенных внешних действий.

Отказ задаётся множеством тех действий, которые могли бы наблюдаться в ответ на воздействие.
Отказ, вообще говоря, не всегда наблюдаем.
Мы будем считать, что наблюдаемость или ненаблюдаемость отказа определяется только воздействием.

- Тогда семантика взаимодействия определяется двумя наборами кнопок:

R – с наблюдаемыми отказами
и Q – с ненаблюдаемыми отказами.

Очевидно, пересечение этих наборов пусто.

Слайд 7

Отказы в LTS.

LTS-модель может остановиться только в стабильном состоянии, в котором нет τ-переходов, поскольку они всегда разрешены.

- Вот здесь на картинке есть 3 стабильных состояния
- Отказ P возникает в таком стабильном состоянии, в котором нет переходов по действиям из P.

Отказ наблюдаем, если P – это R-кнопка.

- Вот здесь есть три отказа в двух состояниях. Один из них ненаблюдаемый, помечен красным цветом.
- В этой семантике рассматриваются трассы как последовательности действий и отказов, то есть трассы действий LTS, в которой добавлены виртуальные переходы-петли по наблюдаемым отказам исходной LTS.

Слайд 8

Здесь дано определение слабой симуляции в семантике с отказами. Красным цветом выделены изменения по сравнению с 3-им определением слабой симуляции без отказов:
наблюдение u теперь может быть не только внешним действием из L, но и наблюдаемым отказом из R.

- На классе реализаций, в которых все отказы наблюдаемы, это определение совпадает с 3-им определением слабой симуляции.

Слайд 9

Будем говорить, что воздействие опасно, если оно не гарантирует ответного наблюдения или, если оно может вызвать такое поведение реализации, которого следует избегать при тестировании по каким-то иным причинам.

Можно выделить три вида опасных воздействий.

- Первая опасность: воздействие, которое может привести к ненаблюдаемому отказу.
- Вторая опасность тоже означает отсутствие наблюдения, но не из-за остановки реализации, а, наоборот, из-за бесконечного выполнения ненаблюдаемых действий.

Здесь предполагается, что бесконечная последовательность любых действий выполняется бесконечное время, а конечная последовательность – конечное время.

Бесконечная последовательность ненаблюдаемых, то есть τ-действий, называется дивергенцией и обозначается символом Δ.

После возникновения дивергенции любое воздействие опасно, так как вместо внешнего действия или остановки реализация может бесконечно долго выполнять τ-действия.

- Третья опасность связана с поведением реализации, которое нежелательно по каким-то особым причинам.

Такое поведение мы называем разрушением и обозначаем символом γ.

Воздействие опасно, если оно разрешает внешнее действие, после которого возможно разрушение.

- Взаимодействие будем называть безопасным, если не возникает ненаблюдаемых отказов, не выполняются воздействия при дивергенции и воздействия, которые могут вызвать разрушение.

Слайд 10

Здесь приведены примеры всех трех видов опасности в LTS:
Ненаблюдаемый отказ {ab} в состоянии 3.

- Дивергенция в состоянии 4: моделируем ее виртуальным переходом-петлей по символу Δ.
- Разрушение в состоянии 1: моделируем его переходом-петлей по символу γ. Этот переход не виртуальный. Символ γ, фактически, добавляется в алфавит LTS.

Слайд 11

Здесь дано определение слабой симуляции в семантике с отказами с учетом безопасности.

Кнопка безопасна в состоянии, если в этом состоянии нет разрушения и дивергенции, нет отказа по этой кнопке, если такой отказ ненаблюдаем, и после любого действия, разрешаемого этой кнопкой, нет разрушения.

- Красным цветом выделены изменения по сравнению с предыдущим, 4-ым определением слабой симуляции с отказами.

Если в начальном состоянии спецификации возможно разрушение, то к реализации не предъявляется никаких требований.
Если кнопка P безопасна в состоянии реализации, то требования к реализации предъявляются теми соответствующими ему состояниями спецификации, в которых кнопка P тоже безопасна.

- На классе реализаций, в которых все отказы наблюдаемы, нет дивергенции и разрушения это определение совпадает с предыдущим.

Слайд 12

О безопасности воздействий в состояниях спецификации мы можем судить по заданной LTS-модели спецификации.

Если LTS-модель реализации тоже задана, мы можем определять и безопасность воздействий в состояниях реализации.

- Однако при тестировании модель реализации обычно неизвестна.

Предполагается только, что такая модель существует: это называется тестовой гипотезой.

В этом случае судить о безопасности воздействий в реализации мы может только на основании спецификации, для чего предлагается следующая гипотеза о безопасности, основанная на соответствии состояний реализации и спецификации – H-соответствии.

- Здесь дано формальное определение соответствия H как минимального соответствия, порождаемого двумя правилами вывода.

Во-первых, если разрушения нет в начальных состояниях реализации и спецификации, то любые два состояния, достижимые из начальных по пустой трассе, соответствуют друг другу по H. В частности, сами начальные состояния.

Во-вторых, если пресостояния соответствуют друг другу и кнопка P, разрешающая некоторое наблюдение u, безопасна в этих пресостояниях, то постсостояния после такого безопасного наблюдения u также соответствуют друг другу по H.

- Гипотеза о безопасности требует:

Во-первых, чтобы в начальном состоянии реализации не было разрушения, если это не разрешено спецификацией, то есть разрушения нет в начальном состоянии спецификации.

Во-вторых. кнопка должна быть безопасна в состоянии реализации, если она H-безопасна, что означает: безопасна хотя бы в одном соответствующем по H состоянии спецификации.

- Внизу дано формальное определение гипотезы о безопасности.

Слайд 13

Здесь дано определение слабой симуляции в семантике с отказами с учетом безопасности и гипотезы о безопасности.

Мы назвали ее безопасной симуляцией – safe simulation - сокращенно ss.

Красным цветом выделены изменения по сравнению с предыдущим, 5-ым определением слабой симуляции.

Во-первых, должна быть выполнена гипотеза о безопасности. Это не проверяется при тестировании, а является его предусловием. Остальная часть определения – это как раз условие, которое должно проверяться при тестировании.

Во-вторых, рассматриваются воздействия, H-безопасные в состояниях реализации.

- Это определение совпадает с 5-ым определением для спецификаций без ненаблюдаемых отказов, дивергенции и разрушения и H-безопасных реализаций.
- Отношение ***ss*** транзитивно. Кроме того, на классе спецификаций, удовлетворяющих собственной H-гипотезе, оно рефлексивно, то есть является предпорядком на этом классе спецификаций.
- Если соответствие R конформно, то его пересечение с H-соответствием также конформно. Здесь дано независимое определение безопасной симуляции для такого соответствия.
- Объединение конформных соответствий конформно. Поэтому мы можем рассматривать два естественных конформных соответствия: R1 – объединение всех конформных соответствий и R2 – его пересечение с H-соответствием.

Слайд 14

Тестирование Безопасной симуляции

Слайд 15

Тестирование – это проверка конформности в процессе эксперимента.

- **Для конформности типа симуляции** требуется операция опроса текущего состояния реализации.
- **Тест** – это инструкция, каждый пункт которой описывает либо рестарт системы, либо тестовое воздействие (кнопку) и в зависимости от полученных наблюдения и постсостояния – следующий пункт или вердикт (***pass*** или ***fail***).
- Реализация ***проходит*** тест, если тестирование ***никогда*** не дает вердикт ***fail***. Никогда – это значит ни при каком возможном проявлении недетерминизма реализации.

Тест ***значимый***, если каждая конформная реализация его проходит.
Тест ***исчерпывающий***, если каждая неконформная реализация его не проходит.
Тест ***полный***, если он значимый и исчерпывающий.

- Для проверки симуляции нужно проверить все переходы, где наблюдение разрешается кнопкой H-безопасной в пресостоянии, которое достижимо при безопасном тестировании.
- Для этого предполагается выполненной следующая ***гипотеза глобального тестирования:*** любую пару из наблюдения, разрешаемого данной кнопкой, и постсостояния можно получить за конечное число нажатий этой кнопки в пресостоянии.

Кроме того, любое состояние, достижимое из начального состояния по пустой трассе, то есть по τ-переходам, можно получить за конечное число рестартов.

Слайд 16

Сначала рассмотрим тестирование теоретически, то есть без учета практических ограничений.

Если спецификация разрешает разрушение с самого начала до нажатия какой-либо кнопки, то все реализации формально конформны и тестирование не требуется.

- В противном случае определим множество N неконформных пар состояний реализации i и спецификации s из соответствия H.

Во-первых, пара (i,s) неконформна, если в реализационном состоянии i при нажатии некоторой H-безопасной кнопки P можно получить такое наблюдение u, которое нельзя получить в состоянии спецификации s, хотя кнопка P безопасна в состоянии s.

Во-вторых, если после такого наблюдения u мы можем оказаться в реализации в таком постсостоянии i`, что любое постсостояние спецификации s` после того же наблюдения u образует с состоянием i` неконформную пару.

- Конформность эквивалентна тому, что пара начальных состояний реализации и спецификации конформна, т.е. не принадлежит множеству N.
- Правила вывода для множества N определяют граф вывода.

Вершины – это неконформные пары состояний.

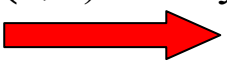
Дуга ведет из пары (i,s) в пару (i`,s`) и помечена парой (u,i`).

Она соответствует 2-ому правилу вывода, когда постсостояния i` и s` достижимы из пресостояний i и s, соответственно, по наблюдению u.

- Деревом вывода будем называть дерево маршрутов графа вывода, начинающихся в паре начальных состояний.

Корень дерева – пустой маршрут, а листья – маршруты, заканчивающиеся в вершинах, полученных по 1-ому правилу вывода.

Если маршрут дерева не листовой, он должен продолжаться в дереве теми и только теми дугами, которые соответствуют одному применению правила вывода 2, то есть помеченными одной парой (u,i`) и ведущими во все пары (i`,s`).



Понятно, что если пара начальных состояний не принадлежит множеству N, то есть реализация конформна, то нет ни одного дерева вывода. В противном случае по данному графу вывода, вообще говоря, может быть построено много различных деревьев вывода.

Слайд 17

В общем случае, если полный тест существует,
он обнаруживает ошибку за конечное время,
но при отсутствии ошибок может выполняться бесконечено долго.

Если пара начальных состояний неконформна, то есть реализация неконформна,
но все деревья выводы бесконечны,
то любой тест за конечное время не определит неконформность.

Есть пример такой реализации,
т.е. на классе всех H-безопасных реализаций нет полных тестов,
и существуют только значимые тесты.

Слайд 18

На этом слайде приведен пример спецификации и неконформной реализации, в которой никакой тест не может обнаружить ошибку.

Алфавит внешних действий состоит из двух действий: x и y.

- Рассматривается любая семантика, в которой либо обе кнопки {x} и {y} являются **R**-кнопками, либо имеется кнопка {xy} – она может быть как **R**-, так и **Q**-кнопкой.

Покажем, что реализация I неконформна спецификации S для любой семантики с указанными ограничениями.

Действительно, действия `x` и `y` безопасны по всех состояниях реализации и спецификации.

- Рассмотрим состояние 1 в реализации.
- Оно не соответствует s_0, т.к. $1\langle =y\Rightarrow\rangle$, но $s_0\langle =y\rangle\not\Rightarrow$.
- Но состояние `1` реализации не соответствует и никакому другому состоянию спецификации, т.к. $1\langle =x,x,\ldots\Rightarrow\rangle$ для любого числа `x`, а в спецификации это есть только для s_0.
- Через **`I(n)`** обозначим под-LTS реализации **`I`**, содержащую состояния `0,1,…,n` и все переходы между ними.

`I(n)` ***ss*** **`S`** для каждого `n`, т.к. **`I(n)`** совпадает с частью **`S`**.

Любой тест за конечное время `n` может исследовать часть реализации, содержащую не более `n` состояний, и, тем самым, не может различить **`I(n)`** и **`I`**.

Следовательно, никакой полный тест не обнаружит ошибку в реализации I.

Нужно сказать, что это явление – отсутствие полного теста – характерно именно для симуляций. Для трассовых конформностей, основанных только на трассах реализации и спецификации и не опирающихся на соответствие состояний, полный тест всегда существует.

Слайд 19

Существует тест, который полный на ***подклассе*** таких реализаций, в которых либо нет ни одного дерева вывода (что означает конформность), либо есть деревья вывода (что означает неконформность), и одно из таких деревьев ***конечно***.

- Для работы такого теста, кроме глобального тестирования, требуется перечислимость следующих множеств:
  - Множество состояний спецификации, достижимых по пустой трассе.
  - Множество P(s) кнопок, безопасных в состоянии s, для каждого состояния s, достижимого при безопасном тестировании.
  - Множество s ***after*** ⟨u⟩ постсостояний, достижимых из данного состояния s по безопасному наблюдению u, для каждого состояния s, достижимого при безопасном тестировании.
- Теперь сформулируем условия, при которых существует полный тест, то есть когда для неконформной реализации существует конечное дерево вывода.

Дерево вывода конечно тогда и только тогда, когда оно имеет конечное ветвление.

А это бывает тогда и только тогда, когда конечно каждое из множеств s ***after*** ⟨u⟩ .

- Для этого ***достаточно***, чтобы в каждом безопасно достижимом состоянии s были конечны,

во-первых, число переходов по каждому безопасному действию, и во-вторых, конечно множество состояний, достижимых из s по пустой трассе.

Слайд 20

Понятно, что на практике не приемлемо бесконечное выполнение теста. Естественным практическим ограничением является требование, чтобы тест всегда заканчивался за конечное время.

- Можно предложить следующие достаточные ограничения, обеспечивающие как существование полных тестов, так и выполнение требования конечности тестирования:

1. Число кнопок конечно и каждая кнопка (как множество действий) разрешима относительно алфавита всех внешних действий L.
2. Спецификация конечна, то есть конечно число ее состояний и переходов.
3. Та часть I~ реализации I, которая может проходиться при безопасном тестировании, тоже конечна.
4. Последнее ограничение касается недетерминизма реализации.

Если реализация недетерминирована и на ее недетерминизм не налагается никаких ограничений, то конформная реализация должна тестироваться бесконечно долго, конечно, при условии, что реализационная модель нам неизвестна.

Гипотеза о глобальном тестировании здесь не помогает, поскольку она обеспечивает лишь возможность полного тестирования, но не его конечность.

Предлагается следующее ограничение, которое мы назвали t-недетерминизмом.

Оно означает, что если мы нажимаем некоторую кнопку в некотором пресостоянии реализации t раз, получаем наблюдение и опрашиваем постсостояние, то гарантированно будут получены все возможные в реализации пары (наблюдение, постсостяоние).

При t=1 реализация детерминирована.

Слайд 21

Предлагаемое тестирование симуляции разбивается на два этапа:
Сначала мы с помощью тестирования исследуем реализацию I, строя ее часть I~, которая может быть получена при безопасном тестировании.
Потом проводим верификацию симуляции по этой части.
Заметим, что такое разбиение на два этапа можно делать только при наших ограничениях на конечность реализации. Иначе I~ может оказаться просто бесконечной LTS. Поэтому в алгоритме теоретического тестирования эти два этапа слиты вместе как один процесс: мы одновременно исследуем реализацию и верифицируем симуляцию по построенной части реализации. Но для практических нужд нам вполне достаточно алгоритма тестирования из двух этапов. Это попроще.

Итак, исследование реализации: построение I~.

Переход i—u→i` добавляется тогда, когда после опроса состояния i нажимаем кнопку P, получаем наблюдение u и опрашиваем постсостояние i`.
Эту кнопку P назовем *управляющей* кнопкой этого перехода и обозначим P(i—u→i`).

- Для каждого полученного при опросе состояния реализации i мы будем хранить следующие постепенно растущие множества:
  1. Множество H(i) – это те состояния спецификации, которые H-соответствуют состоянию i.
  2. Множество P(i) - это кнопки, которые H-безопасны в состоянии i, то есть безопасны в спецификационных состояниях из H(i).

Кроме того, для каждой такой кнопки P будем хранить счетчик C(i,P) числа нажатий кнопки P в состоянии i.
Будем говорить, что кнопка P полна в состоянии i, если c(P,i)=t.
Состояние i полное, если в нем полны все кнопки из P(i).

- В начале тестирования мы опрашиваем текущее состояние i_0` - это должно быть некоторое состояние, достижимое из начального состояния реализации по пустой трассе. Для этого состояния определяем:

$H(i_0`)$ состоит из всех состояний спецификации, достижимых из ее начального состояния по пустой трассе.

$P(i_0`)$ состоит из всех кнопок, безопасных в этих состояниях спецификации.

Для каждой кнопки ее счетчик равен нулю.

Слайд 22

На этом слайде изображена схема алгоритма исследования реализации.

Сначала проверяем полноту текущего состояния i.

- Если состояние i неполное, в нем есть неполная кнопка P, которую мы нажимали меньше t раз.

Нажимаем кнопку P, получаем наблюдение u и опрашиваем состояние i`.

В результате строим переход i—u→i`.

Состояние i` становится новым текущим состоянием.

Корректируем счетчик кнопки, увеличивая его не единицу.

- Анализируем полученный переход.
- Если он уже был получен ранее, возвращаемся к началу.

Слайд 23

Если полученный переход новый, корректируем данные.

- Сначала локальная коррекция:

Добавляем полученный переход в **I~**.

Кнопку P, которую мы нажимали, запоминаем как управляющую кнопку этого перехода.

Обновляем множество H(i`), добавляя в него постсостояния спецификации, достижимые из тех состояний множества H(i), в которых кнопка P безопасна.

- Затем выполняем глобальную коррекцию:

Когда некоторое новое состояние s добавляется во множество H(i) для некоторого состояния i,

Корректируем множество P(i), добавляя в него кнопки, безопасные в состоянии s.

Также для каждого полученного ранее перехода i—u→i` при условии, что управляющая кнопка этого перехода безопасна в состоянии s, обновляем множество H(i`), добавляя в него постсостояния спецификации, достижимые из состояния s по наблюдению u.

Вновь добавленные состояния, то есть те, которых ранее не было, отмечаем как новые.

Эта рекурсивная процедура повторяется пока возможно.

- Условия конечности гарантируют, что процедура глобальной коррекции закончится за конечное число шагов.

Слайд 24

Если текущее состояние полное, смотрим, есть ли в **I~** неполные состояния?

- Если есть неполные состояния, переходим в любое из них.

Для этого выбираем в **I~** лес непересекающихся деревьев, покрывающих все состояния и ориентированных к своим корням, которыми являются все неполные состояния.

С каждым переходом $x \text{—} u \rightarrow y$ этого леса свяжем управляющую кнопку $A(x)=P(x \text{—} u \rightarrow y)$.

Заметим, что из состояния x выходит не более одного перехода, принадлежащего какому-либо дереву леса.

Будем двигаться, нажимая в каждом текущем состоянии x кнопку A(x).

Из-за недетерминизма можем оказаться не в состоянии y, а в другом состоянии z, где будем нажимать кнопку A(z).

Переход в неполное состояние гарантируется t-недетерминизмом реализации.

- Если неполных состояний нет, заканчиваем исследование реализации.

Слайд 25

После исследования реализации, когда уже построена LTS **I~**, проводим верификацию симуляции.

Мы будем пытаться строить конформное соответствие R_2. Если такое соответствие существует, то мы его построим и вынесем вердикт ***pass***. В противном случае – вердикт ***fail***.

- Сначала строим двудольный граф.

Вершины 1-го типа – это пары состояний (i,s), где s принадлежит H(i), то есть H-соответствует i.

Вершины 2-го типа – пары из перехода и состояния спецификации, которое H-соответствует пресостоянию этого перехода.

В каждую вершину 2-го типа входит ровно одна дуга 1-го типа, вычисляемая однозначно.

Дуга 2-го типа, ведет из вершины 2-го типа в вершину 1-го типа (i`,s`), если s` достижимо из s по наблюдению u.

Одновременно составляется список терминальных вершин 2-го типа.

- Далее для каждой терминальной вершины v_2 2-го типа удаляется входящая в нее дуга 1-го типа $v_1 \rightarrow v_2$.

Вершина v_1 и каждая входящая в нее дуга 2-го типа $v_1` \rightarrow v_1$ также удаляются.

Для удаляемой вершины 1-го типа $v_1=(i,s)$ из множества H(i) удаляем состояние s.

Повторяем эту процедуру удаления, пока возможно.

Если удаляется вершина $(i_0`,s_0)$, где состояние $i_0`$ достижимо из начального состояния реализации по пустой трассе, выносим вердикт ***fail***. В противном случае выносим вердикт ***pass***.

Соответствие R_2 – это множество пар (i,s), где s принадлежит множеству H(i), из которого удалены «лишние» состояния.

- Если при построении двудольного графа каждое множество H(i) заменить на множество всех состояний спецификации, то есть вначале выбрать все возможные пары состояний реализации и спецификации, а не только те, которые H-соответствуют друг другу, то будет построено конформное соответствие R_1.

Слайд 26

На этом слайде приведены оценки сложности предлагаемых алгоритмов.

Для исследования реализации приведены две оценки: числа тестовых воздействий и объема вычислений, не связанных со взаимодействием с реализацией.

Экспоненциальность оценки для недетерминированных реализаций следовало ожидать: она непосредственно следует из выбранного ограничения недетерминизма.

Для детерминированных реализаций число тестовых воздействий – это квадратичная функция от числа состояний реализации, а объем вычислений – кубическая.

- Объем вычислений при верфиикации симуляции линейно зависит от числа переходов реализации и квадратично – от числа состояний спецификации.

Слайд 27

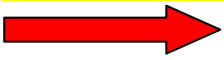
На этом слайде приведен пример верификации симуляции.
Семантика содержит как **R**-, так и **Q**-кнопки.
Спецификация **S** демонстрирует все виды опасности: ненаблюдаемый отказ {y} в состоянии 3, дивергенцию в состоянии 4 и разрушение в состоянии 5.

В спецификации имеется единственное проявление недетерминизма – это два перехода 0—y→1 и 0—y→3. Но в этих состояниях безопасна одна и та же кнопка {x}.
Поэтому спецификация удовлетворяет собственной H-гипотезе и, следовательно, конформна сама себе.

Кроме конформной реализации **S**, приведены примеры еще одной конформной реализации **I1** и одной неконформной реализации **I2.**
После исследования реализаций будут построены LTS **S~, I1~** и **I2~.**
Они отличаются наличием явных (а не виртуальных) переходов-петель по отказу {x}.

Кроме того, в **S~** отсутствуют переходы, которые опасны в спецификации **S** (являются разрушением или ведут к разрушению), а также отсутствует дивергенция.
Состояние 5 недостижимо в **S~**, поскольку оно не достижимо в **S** при безопасном тестировании.

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 **I1**.

- Поскольку начальные состояния реализации и спецификации всегда H-соответствуют друг другу, если спецификация не разрешает разрушение с самого начала, пара (0,**0**) принадлежит H, и является вершиной 1-го типа.
- Поскольку в реализации наблюдались переходы 0—y→1 и 0—x→1,
- в двудольный граф добавляются соответствующие дуги 1-го типа (0,**0**)→(0—y→1,**0**) и (0,**0**)→(0—x→1,**0**).
- Поскольку в спецификации из состояния **0** есть два перехода по y: **0**—y→**1** и **0**—y→**3**,
- в двудольный граф добавляются дуги 2-го типа (0—y→1,**0**)→(1,**1**) и (0—y→1,**0**)→(1,**3**).
- Аналогично, поскольку в спецификации из состояния **0** есть переход по x: **0**—x→**2**,
- в двудольный граф добавляется дуга 2-го типа (0—x→1,**0**)→(1,**2**).
- Поскольку в реализации наблюдался переход по отказу 1—{x}→1,
- в двудольный граф добавляется дуга 1-го типа (1,**1**)→(1—{x}→1,**1**).

В состоянии спецификации **1** нет перехода по отказу {x}, поэтому из вершины (1—{x}→1,**1**) не проводится никаких переходов – это терминальная вершина 2-го типа.

И так далее.

После построения двудольного графа анализируем терминальные вершины 2-го типа. Они отмечены серым фоном.
Для реализации **I1** такая вершина одна.

- Тогда для построения R2 из двудольного графа нужно удалить единственную входящую в эту терминальную вершину дугу (1,**1**)→(1—{x}→1,**1**), начало этой дуги – пару (1,**1**)
- и все дуги, входящие в вершину (1,**1**).
- Удаляемые дуги отмечены пунктиром.

Множество вершин 1-го типа, оставшихся после завершения этого процесса удаления, как раз и составляет конформное соответствие R2, если в нем остаётся обязательная пара начальных состояний (0,**0**).
Это имеет место для реализаций **S** и **I1**, но не для реализации **I2**.
Для конформных реализаций **S** и **I1** приведены также соответствия R1 без двудольных графов.
Эти графы строятся аналогично, но только с самого начала вершинами 1-го типа считаются все возможные пары состояний реализации и спецификаций.