И. Бурдонов, А. Косачев

Тестирование соответствия (conformance testing)

[ИСПРАН](http://www.ispras.ru/)

239 слайдов



**СТАЦИОНАРНОЕ ТЕСТИРОВАНИЕ**

[Ограничение на среду передачи. 3](#_Toc104991482)

[Осцилляция. 3](#_Toc104991483)

[Стационарный автомат. 4](#_Toc104991484)

[Работа тестовой системы. 5](#_Toc104991485)

[Недетерминизм. 6](#_Toc104991486)

[Перечисление в состоянии спецификации. 7](#_Toc104991487)

[Полнота тестирования. 8](#_Toc104991488)

**НЕСТАЦИОНАРНОЕ ТЕСТИРОВАНИЕ**

[Ограничение на среду передачи. 9](#_Toc104991489)

[Цикл стационарного и нестационарного тестирования. 9](#_Toc104991490)

[Наблюдаемые и гипотетические трассы. 10](#_Toc104991491)

[Завершённые трассы. 11](#_Toc104991492)

[Мультистимульное стационарное тестирование. 13](#_Toc104991493)

[Незавершённые трассы. 14](#_Toc104991494)

[Безопасность тестирования. 15](#_Toc104991495)

[Верификация наблюдаемых трасс. 16](#_Toc104991496)

[Сериализация (шаг тестирования). 17](#_Toc104991497)

[Перечисление. 18](#_Toc104991498)

[ПРОБЛЕМЫ: Выбор стратегии тестирования (аналог обхода автомата). 18](#_Toc104991499)

**ГИПЕРТЕСТИРОВАНИЕ**

[Среда передачи. 20](#_Toc104991500)

[Среда: несколько очередей. 21](#_Toc104991501)

[Передача стимулов и реакций, минуя среду. 23](#_Toc104991502)

[Передача стимулов и реакций, минуя среду (пример). 23](#_Toc104991503)

[Среда: «куча» (множество). 24](#_Toc104991504)

[Среда: стек. 24](#_Toc104991505)

[Среда: приоритеты очередей, куч и стеков. 24](#_Toc104991506)

[Зависимость между стимулами и реакциями. 25](#_Toc104991507)

[Автомат среды: Передача стимулов и реакций. 26](#_Toc104991508)

[Автомат среды: Конвергентность композиции среды и реализации. 27](#_Toc104991509)

[Автомат среды: Таблица условий. 28](#_Toc104991510)

[Автомат среды: Правила работы. 29](#_Toc104991511)

[Автомат среды: Дополнительные примеры сред. 30](#_Toc104991512)

[ПРОБЛЕМЫ: Автомат среды: как по автомату среды построить отображение наблюдаемых трасс в гипотетические трассы? 30](#_Toc104991513)

[Внешняя блокировка: Адаптивное тестирование. 30](#_Toc104991514)

[Внешняя блокировка: Таблица условий. 31](#_Toc104991515)

[Внешняя блокировка: Правила работы среды. 32](#_Toc104991516)

[Внутренняя блокировка. Порты. 33](#_Toc104991517)

[Внутренняя блокировка: требования к реализации. 34](#_Toc104991518)

[Внутренняя блокировка: отображение трасс. 34](#_Toc104991519)

[Внутренняя блокировка: Таблица условий. 35](#_Toc104991520)

[Внутренняя блокировка: Правила работы среды. 36](#_Toc104991521)

[Внешняя и внутренняя блокировка. 37](#_Toc104991522)

[Внешняя и внутренняя блокировка: Таблица условий. 37](#_Toc104991523)

[Внешняя и внутренняя блокировка: Правила работы среды. 38](#_Toc104991524)

[ПРОБЛЕМЫ: Осциллирующая реализация. Проблема дивергенции. 39](#_Toc104991525)

[ПРОБЛЕМЫ: Осциллирующая реализация. Проблема незавершённости шага тестирования. 40](#_Toc104991526)



### Ограничение на среду передачи.

Мы будем рассматривать стационарное тестирование только для среды передачи, которая представляет собой одну неограниченную очередь стимулов и одну неограниченную очередь реакций.

### Осцилляция.

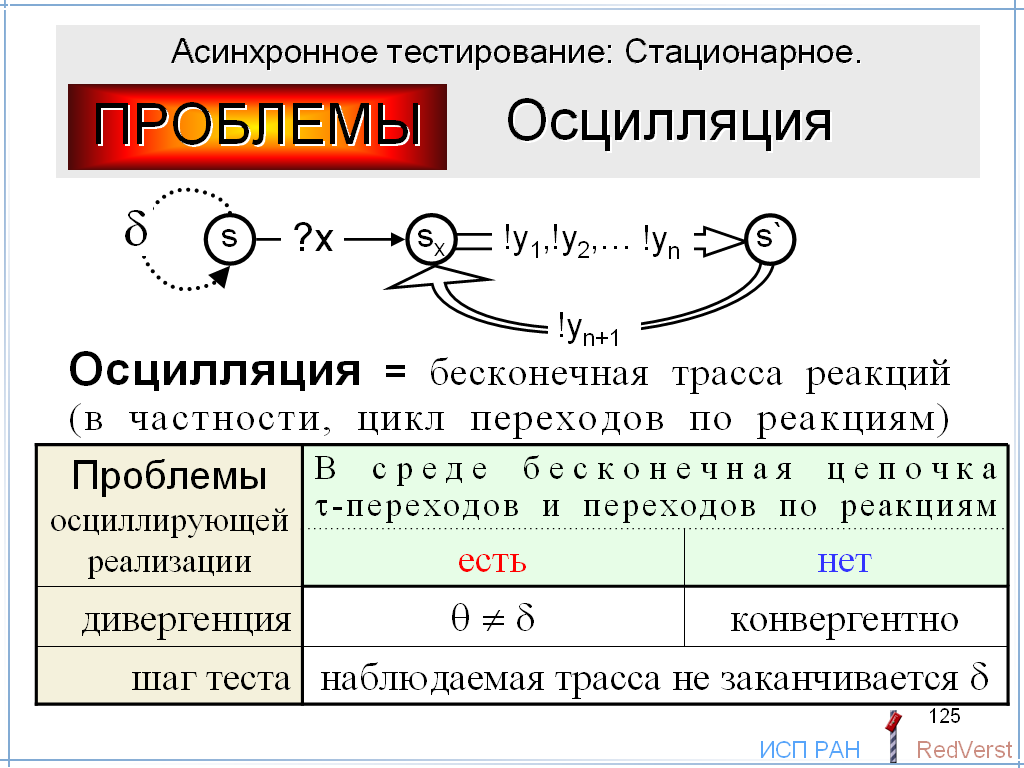
Идея стационарного тестирования заключается в том, чтобы давать стимулы только в стационарных состояниях. В таких состояниях ничего не меняется: автомат стоит и ждёт стимулов.

Получив стимул, автомат может выдать последовательность реакций. Он двигается по некоторой цепочке переходов, начинающейся в стационарном состоянии с перехода по стимулу и далее состоящей из внутренних переходов и переходов по реакциям. Заканчивается такая цепочка в стационарном состоянии, где можно давать следующий стимул.

Проблема возникает, если автомат может выдать бесконечную последовательность реакций. В этом случае говорят, что автомат *осциллирует*.

Бесконечная последовательность реакций может выдаваться в цикле из τ-переходов и переходов по реакциям.

Осцилляция создаёт две проблемы при асинхронном тестировании: проблему дивергенции и проблему шага тестирования.

Проблема дивергенции. Если среда передачи всегда готова принимать реакции от реализации, то в цикле осцилляции реализации мистический синхронизатор может всегда выбирать синхронный переход по передаче реакции из реализации в среду. В результате в композиции среды и реализации возникает бесконечная цепочка τ-переходов, то есть дивергенция. Иными словами, среда всё время принимает реакции, но не выдаёт их в тест, который этих реакций ждёт. Истечение тайм-аута в тесте в этом случае не обязательно означает стационарность. Примером такой среды может служить среда с неограниченной выходной очередью.

Если в среде нет бесконечной цепочки τ-переходов и переходов по приему реакций из реализации, то композиция реализации и среды будет конвергентной. Примером может служить среда с ограниченной выходной очередью.

Проблема шага тестирования. Шаг тестирования, который мы будем рассматривать, представляет собой последовательность посылаемых стимулов и принимаемых реакций, которая имеет конечную длину и завершается стационарностью, когда всё «успокаивается».

Однако осцилляция реализации может привести к тому, что тест будет всё время принимать реакции, но так и не дойдёт до стационарного состояния реализации. Проблема в том, что мы не имеем возможности столкнуть реализацию с цикла осцилляции даже в том случае, когда в этом цикле определены также переходы по приёму стимулов. Мы не можем заставить реализацию принять стимул, поскольку приём стимулов не имеет большего приоритета, чем выдача реакций. К проблеме осцилляции и приоритетов переходов мы ещё вернёмся.

А пока первое требование, которое предъявляется к реализации при стационарном тестировании, – реализация должна быть неосциллирующей.

### Стационарный автомат.

Стационарное тестирование неосциллирующего асинхронного автомата очень похоже на тестирование автомата Мили. В обоих случаях мы даём один стимул, а потом ждём реакций. Отличие в том, что в асинхронном автомате мы можем получить не одну реакцию, а конечную, в частности, пустую последовательность реакций, заканчивающуюся символом стационарности δ.

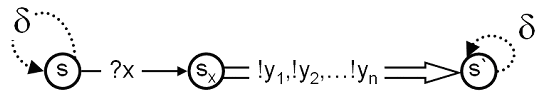
Эта аналогия может быть выражена вполне формально: преобразованием асинхронного автомата в автомат Мили, который можно назвать стационарным автоматом.

Состояния стационарного автомата – это стационарные состояния исходного автомата.

Стимулы стационарного автомата – это стимулы исходного асинхронного автомата.

Реакции стационарного автомата – это конечные последовательности реакций исходного автомата, заканчивающиеся символом δ.

В стационарном автомате определяется переход sx,(y1,y2,…yn,δ)s`, если в исходном автомате состояния s и s` стационарны и имеется цепочка переходов из s в s` с трассой x,y1,y2,…yn.



126a

Небольшая проблема возникает, если начальное состояние автомата нестационарно. Тогда первый шаг тестирования отличается от остальных шагов тем, что начинается в нестационарном состоянии. Поэтому на этом шаге мы не посылаем стимул в реализацию, а только принимаем реакции от неё. Для построения стационарного автомата нужно ввести фиктивный пустой стимул и на переходе из начального состояния написать пару (пустой стимул, последовательность реакций). Пустой стимул – это абстракция, смысл которой в том, что мы не посылаем никакого стимула. В дальнейшем, для простоты, мы не будем специально оговаривать этот особый случай.

126b

Все требования конечности и детерминированности, которые мы предъявляли к автомату Мили для того, чтобы тот или иной метод тестирования был возможен, теперь при стационарном тестировании предъявляются к стационарному автомату. Тестирование с открытым состоянием предполагает теперь открытость стационарных состояний асинхронного автомата. Иными словами, задача сводится к предыдущей.

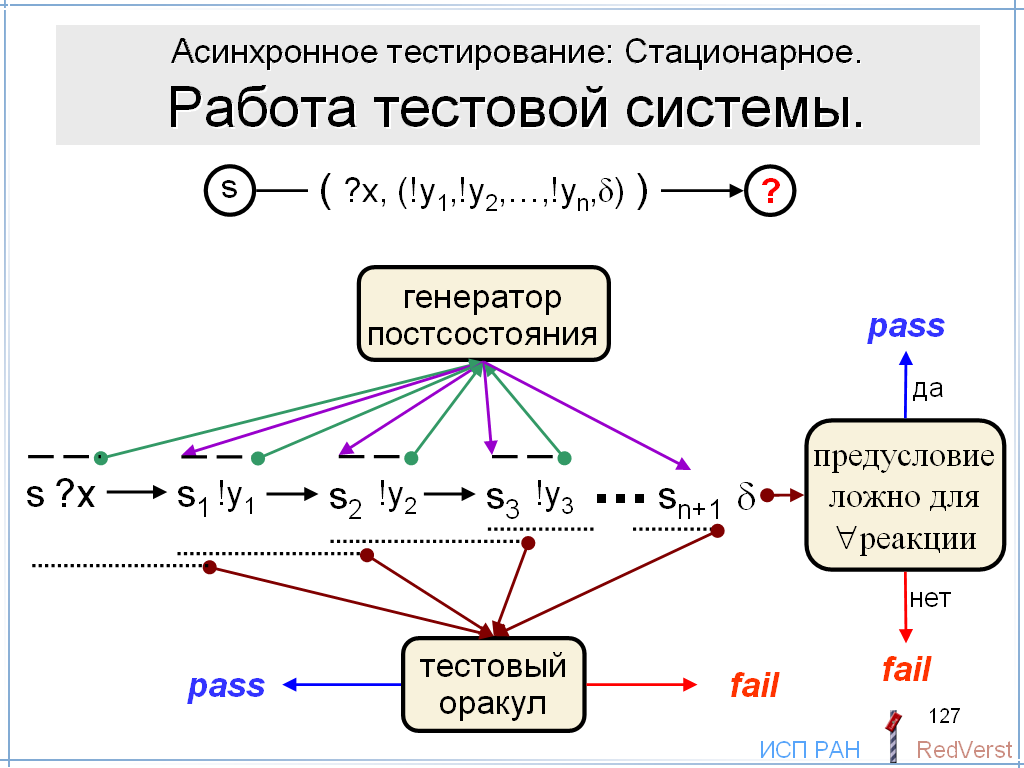
### Работа тестовой системы.

Несколько слов нужно сказать о том, каким образом при стационарном тестировании работает тестовый оракул и генератор постсостояний.

Пусть мы в стационарном состоянии s даём стимул x.

Сначала генератор постсостояния определяет постсостояние s1 для перехода из стационарного состояния s по стимулу x.

Тестовый оракул проверяет, что тройка (s,x,s1) удовлетворяет постусловию стимула.

Затем генератор постсостояния определяет постсостояние s2 для перехода из состояния s1 по первой реакции y1, а тестовый оракул проверяет, что тройка (s1,y1,s2) удовлетворяет постусловию реакции.

Далее такая процедура повторяется для реакций y2,y3,…yn.

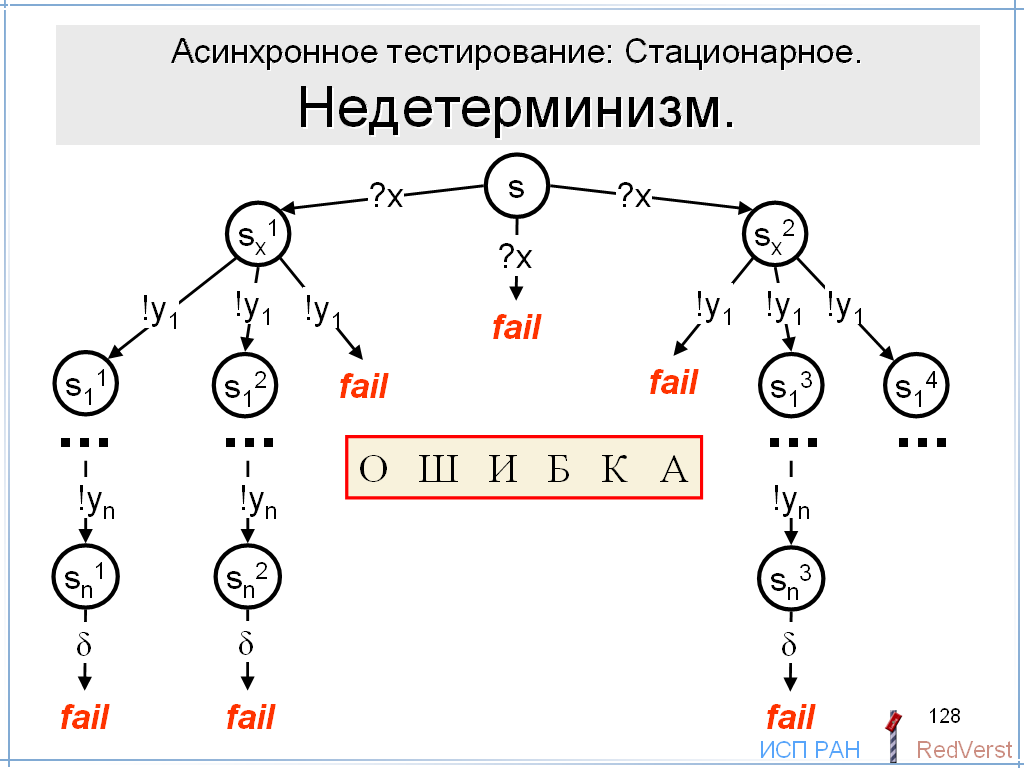
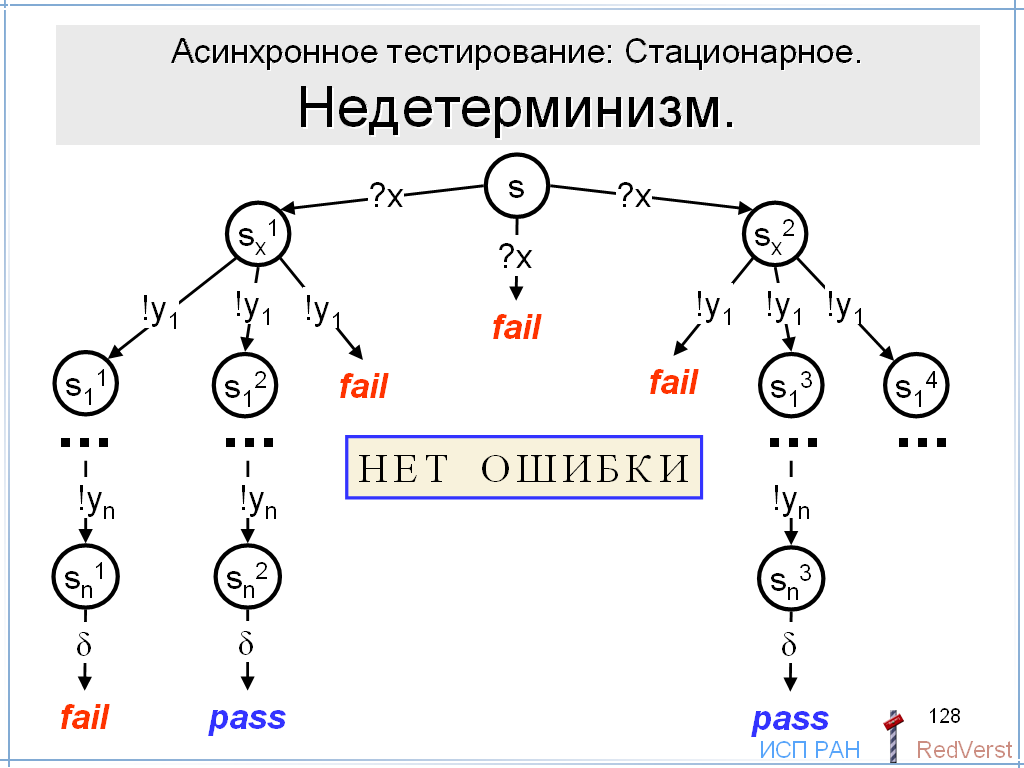
В конце проверяем, что полученное постсостояние sn+1 стационарно, то есть для любой реакции y предусловие реакции PRE(sn+1,y) ложно.

### Недетерминизм.

Конечно, может оказаться, что постсостояние si определяется неоднозначно.

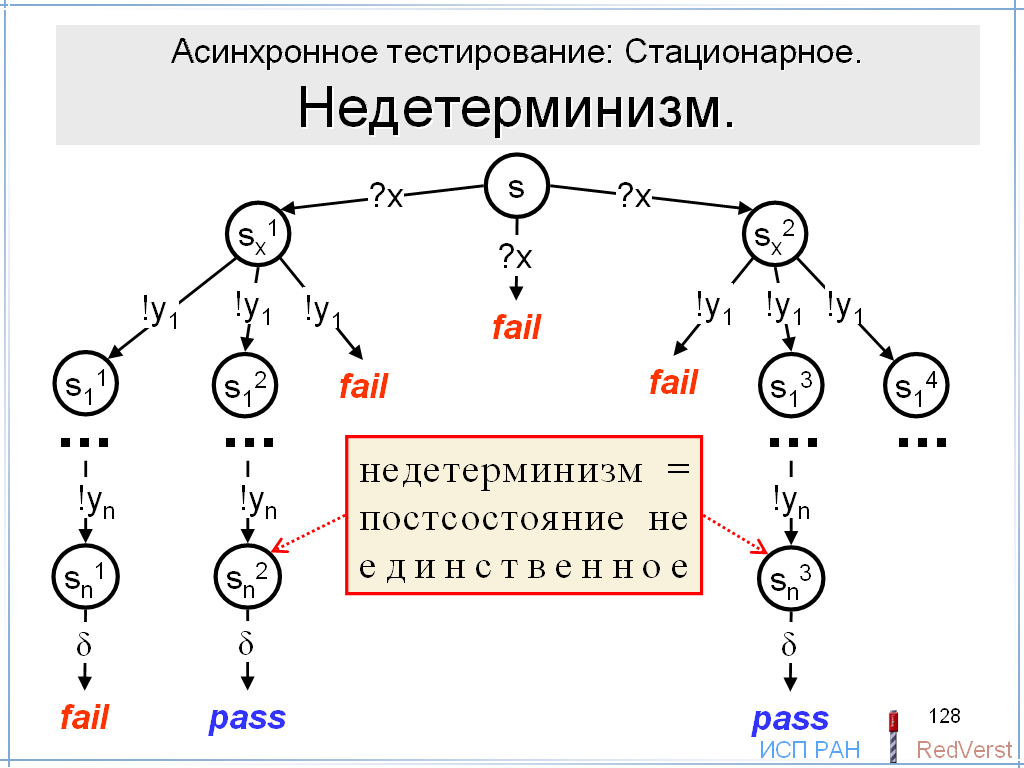
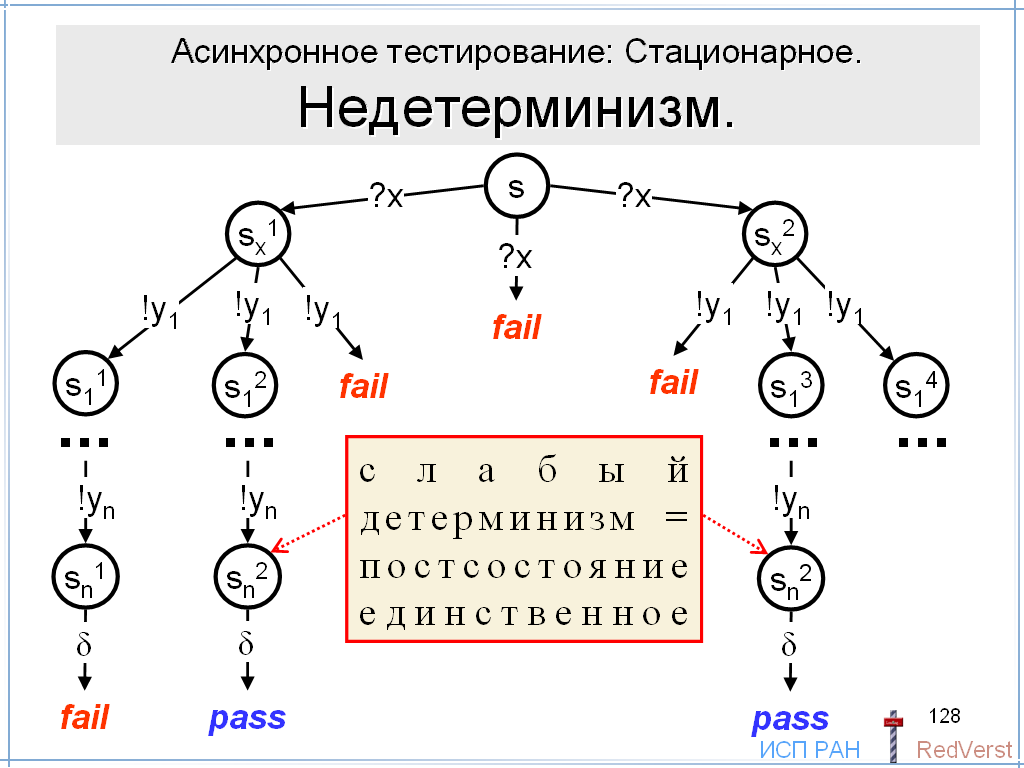
Если генератор постсостояния способен перечислять возможные постсостояния, то мы сможем строить несколько цепочек переходов. Некоторые из них мы отфильтруем, если на каком-то шаге их отбраковывает тестовый оракул.

Ошибка фиксируется только в том случае, когда ни одна из цепочек не годится.

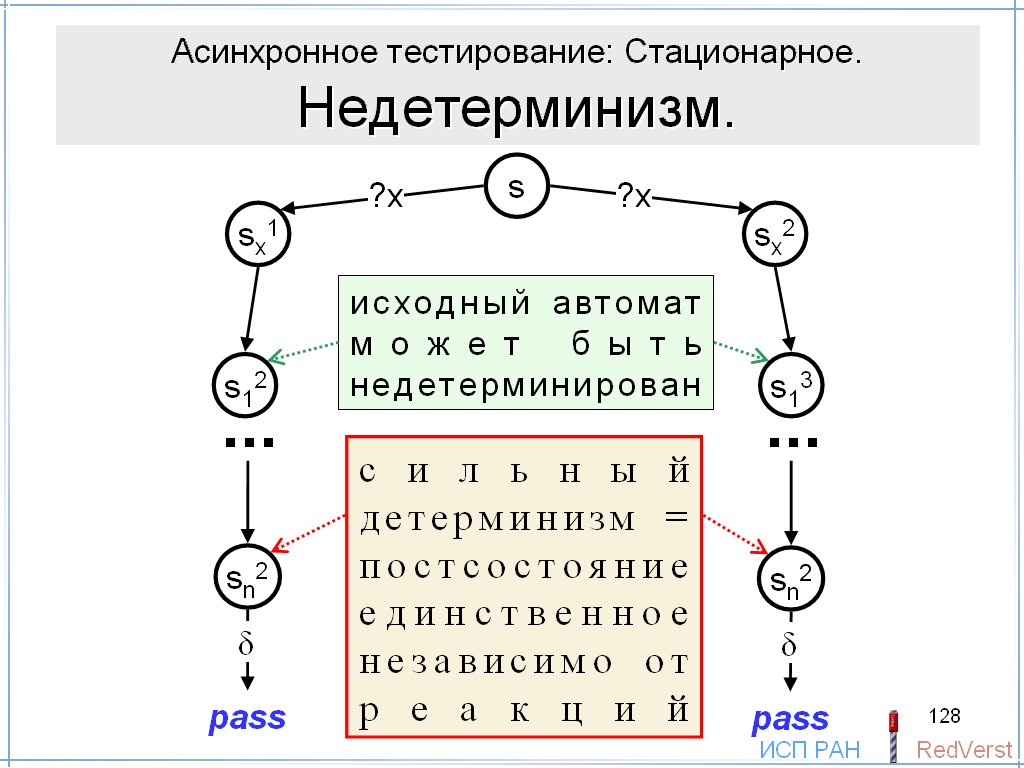
 

Если годится хотя бы одна цепочка, ошибки нет.

Если годится несколько цепочек, то возможен недетерминизм, если эти цепочки заканчиваются в разных постсостояниях.

Тем не менее, само по себе наличие нескольких цепочек не обязательно означает недетерминизм. В асинхронном автомате может быть несколько цепочек переходов для одного и того же стимула и одной и той же последовательности реакций, заканчивающихся, тем не менее, в одном стационарном состоянии. Если для разных реакций постсостояние окажется разным, стационарный автомат будет слабо детерминирован.



Если все цепочки для данного стимула заканчиваются в одном состоянии, независимо от последовательности реакций, то стационарный автомат будет сильно детерминирован.

Заметим, что исходный автомат может быть недетерминированным.

### 

### Перечисление в состоянии спецификации.

Для верификации наблюдаемой трассы нам нужно уметь перечислять стимулы и постсостояния для переходов, определённых в состоянии спецификации, достижимом по наблюдаемой трассе.

Поскольку тестирование должно быть безопасным, нам нужно проверять безопасность стимулов, которые мы посылаем в реализацию в стационарных состояниях.

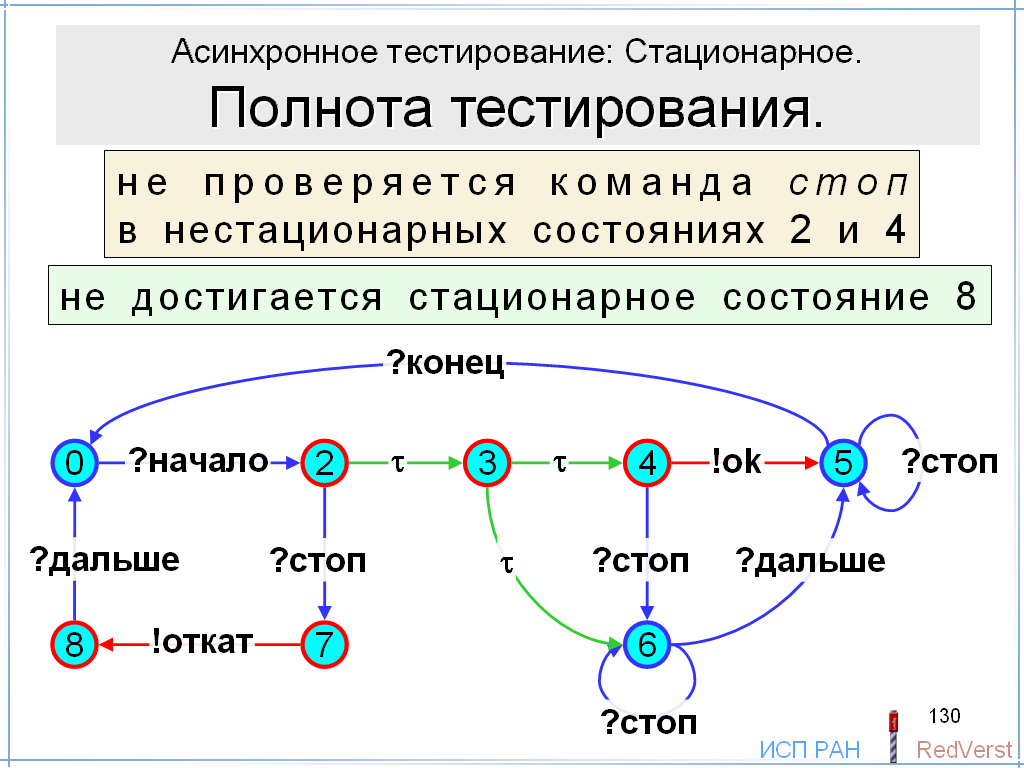
Если предусловие спецификации описывает безопасные стимулы (как принимаемые, так и блокируемые), то стимул безопасен в стационарном состоянии, если он в нём специфицирован.

Однако, если предусловие описывает принимаемые стимулы (как безопасные, так и разрушающие), то для проверки безопасности стимула нам нужно дополнительно уметь перечислять реакции во всех тех нестационарных состояниях, в которые мы можем попасть в спецификации после приёма этого стимула в стационарном состоянии и далее через τ-переходы и переходы по реакциям.

### Полнота тестирования.

Понятно, что стационарное тестирование не является полным, поскольку мы не пытаемся проверить приём реализацией стимулов в нестационарных состояниях.

Для нашего примера мы не проверим команду *стоп* в состояниях 2 и 4.



Кроме того, нетрудно изменить наш пример так, чтобы при стационарном тестировании достигались не все стационарные состояния. Например, по реакции *откат* автомат не сразу возвращается в начальное состояние 1, а сначала переходит в стационарное состояние 8, где ждёт команду *дальше*, чтобы только после этого вернуться в состояние 1.

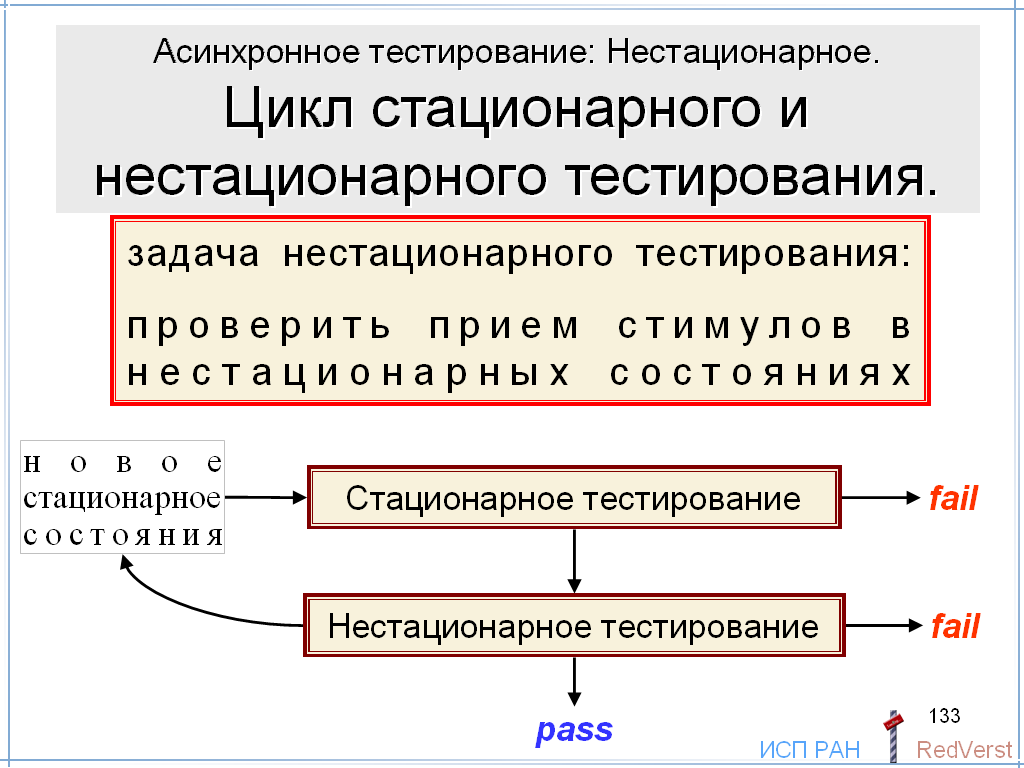
Как естественное обобщение автомата Мили можно рассматривать асинхронный автомат, который принимает стимулы только в стационарных состояниях. Такой автомат, фактически, эквивалентен своему стационарному автомату. Стационарное тестирование может быть полным, разумеется, со всеми теми оговорками, которые мы делали для автоматов Мили.



### Ограничение на среду передачи.

Мы будем рассматривать нестационарное тестирование только для среды передачи, которая представляет собой одну неограниченную очередь стимулов и одну неограниченную очередь реакций.

### Цикл стационарного и нестационарного тестирования.

Нестационарное тестирование является следующим после стационарного этапом тестирования.

Его задача – проверить приём реализацией стимулов в нестационарных состояниях.

Если при этом достигаются новые стационарные состояния, мы можем переключиться обратно в режим стационарного тестирования.

Таким образом, возникает цикл из стационарного и нестационарного тестирования.

### Наблюдаемые и гипотетические трассы.

Каким образом может быть реализовано нестационарное тестирование?

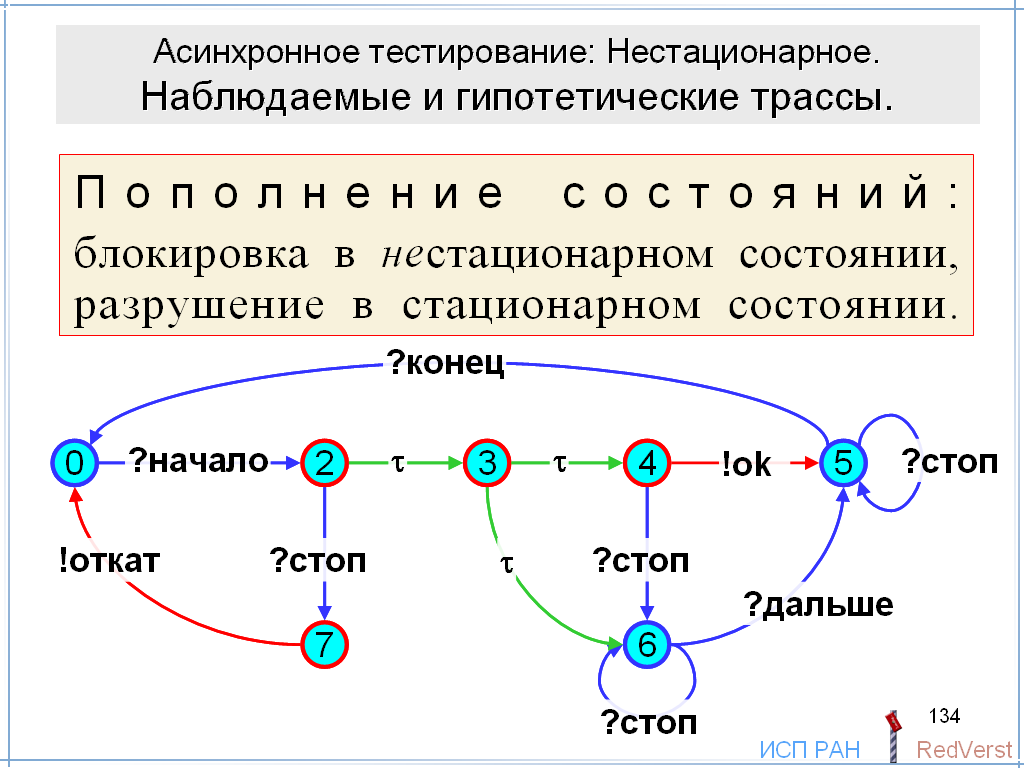
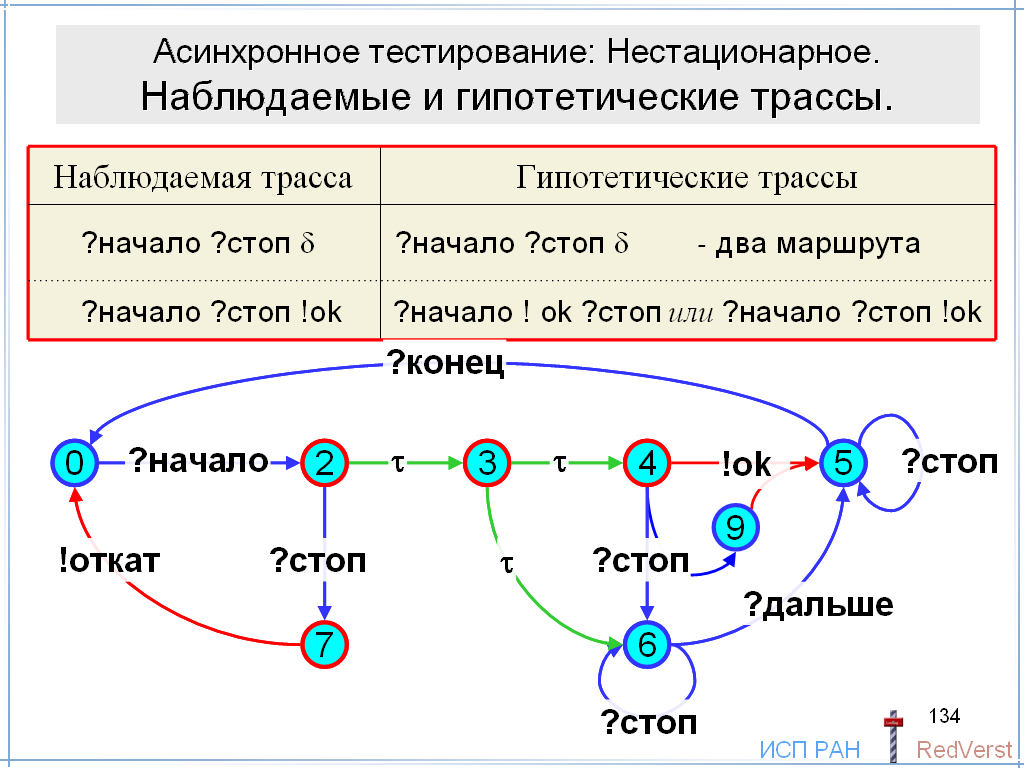
Прежде всего, заметим, что при нестационарном асинхронном тестировании наблюдаемые трассы, вообще говоря, не совпадают с реальными трассами, которые проходит реализация. *Гипотетической* трассой будем называть трассу, которую могла бы пройти реализация при условии, что мы имеем данную наблюдаемую трассу.

Посмотрим на наш пример при нашем допущении полноты.

Пусть наблюдается трасса *?начало?стоп*δ.

В реализации эту трассу имеют два разных маршрута. Один маршрут из состояния 3 переходит по τ-переходу в состояние 6, где принимается команда *стоп*, а потом обнаруживается, что состояние стационарное. Другой маршрут из состояния 3 переходит по τ-переходу в состояние 4, где принимается команда *стоп* с переходом в то же самое стационарное состояние 6. На самом деле, здесь нам повезло, что состояние то же самое; оно могло бы быть и другим.

Теперь пусть наблюдается трасса *?начало?стоп!ok*. Хотя тест выдаёт команду *стоп* сразу после команды *начало*, однако, реализация выдаёт реакцию *ok* перед тем, как она примет команду *стоп*. Здесь происходит рассинхронизация: можно считать, что одновременно тест выдаёт команду *стоп*, а реализация – реакцию *ok*. В тот момент, когда тест выдаёт команду *стоп*, реакция *ok* уже могла оказаться в выходной очереди. Реализация пройдёт трассу *?начало!ok?стоп*.

Однако реализация могла бы быть устроена по-другому. Она могла бы в состоянии 4 принимать команду *стоп* недетерминированно: с переходом не только в состояние 6, но и в какое-нибудь состояние 9, промежуточное между 4 и 5, а при переходе из состояния 9 в состояние 5 выдавать реакцию *ok*.

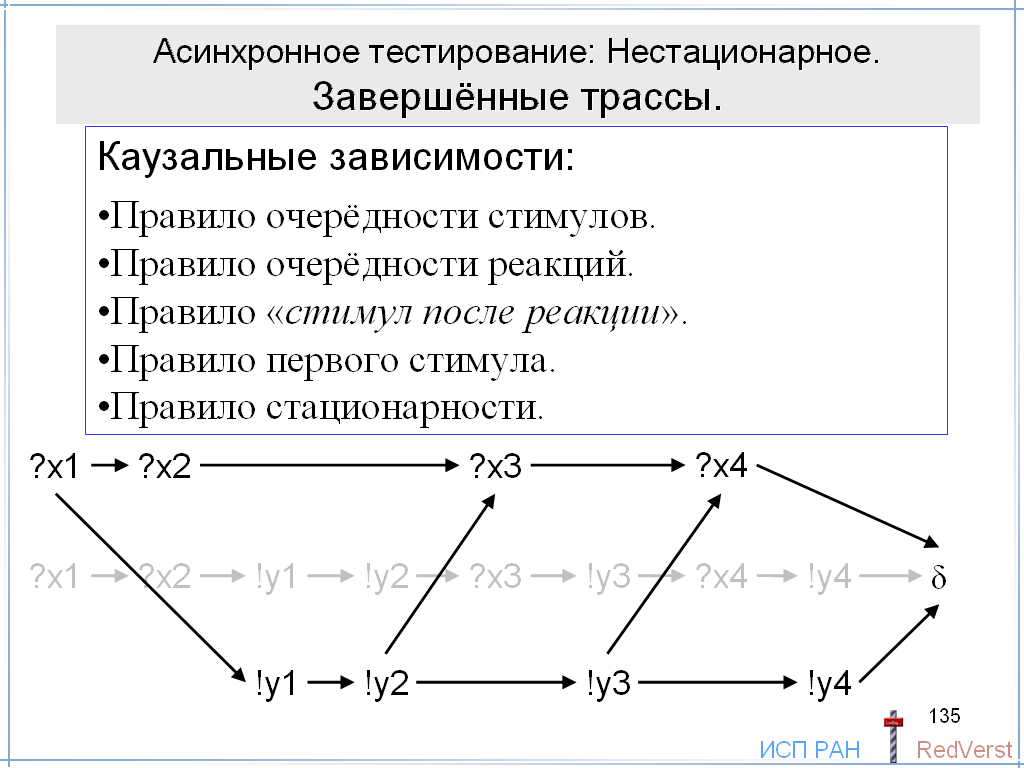
Тогда в ней были бы две разные трассы, соответствующие одной наблюдаемой трассе.

### Завершённые трассы.

Шаг тестового эксперимента при нестационарном тестировании так же, как при стационарном тестировании, начинается и заканчивается в стационарном состоянии. Отличие в том, что, кроме первого стимула, с выдачи которого начинается шаг тестирования, мы можем выдавать дополнительные стимулы, не дожидаясь перехода в стационарное постсостояние. В целом шаг тестирования – это последовательность выдачи стимулов в реализацию и приёма реакций от неё, начинающаяся с выдачи первого стимула и заканчивающаяся тогда, когда при ожидании реакций мы обнаруживаем их отсутствие, то есть стационарность. Заметим, что отсюда вовсе не следует, что все промежуточные состояния нестационарны. Например, выдача подряд двух стимулов может привести к тому, что второй стимул реализация примет в стационарном состоянии.

Итак, наблюдаемая трасса, соответствующая одному шагу нестационарного тестирования, представляет собой последовательность стимулов, реакций и символа δ, начинающуюся со стимула и заканчивающаяся символом δ, причём это единственное его вхождение в трассу. При асинхронном тестировании наблюдаемая трасса, вообще говоря, не совпадает с трассой, которую прошла реализация, но она определяет множество гипотетических трасс. Это определение зависит от среды передачи.

Мы рассматриваем среду, состоящую из неограниченных входной и выходной очередей. Для этой среды сформулируем каузальные правила, то есть правила, определяющие порядок стимулов и реакций в гипотетической трассе.



Правило очерёдности стимулов: Стимулы выбираются реализацией из входной очереди, очевидно, в том же порядке, в каком они поступали в очередь из теста. Поэтому порядок следования стимулов в гипотетической трассе такой же, как в наблюдаемой трассе.

Правило очерёдности реакций: Аналогично, тест выбирает реакции из выходной очереди в том же порядке, в каком реализация выдавала их в очередь. Поэтому порядок следования реакций в гипотетической трассе такой же, как в наблюдаемой трассе.

Правило «стимул после реакции»: Далее, если мы выдавали некоторый стимул после того, как получили некоторую реакцию, то, очевидно, в гипотетической трассе эта реакция предшествует этому стимулу. Это правило имеет общий характер – для любой среды.

Правило первого стимула: Поскольку шаг тестирования начинается в стационарном состоянии, гипотетическая трасса начинается со стимула, но не с реакции. Для одной входной очереди этим стимулом, очевидно, будет первый стимул наблюдаемой трассы.

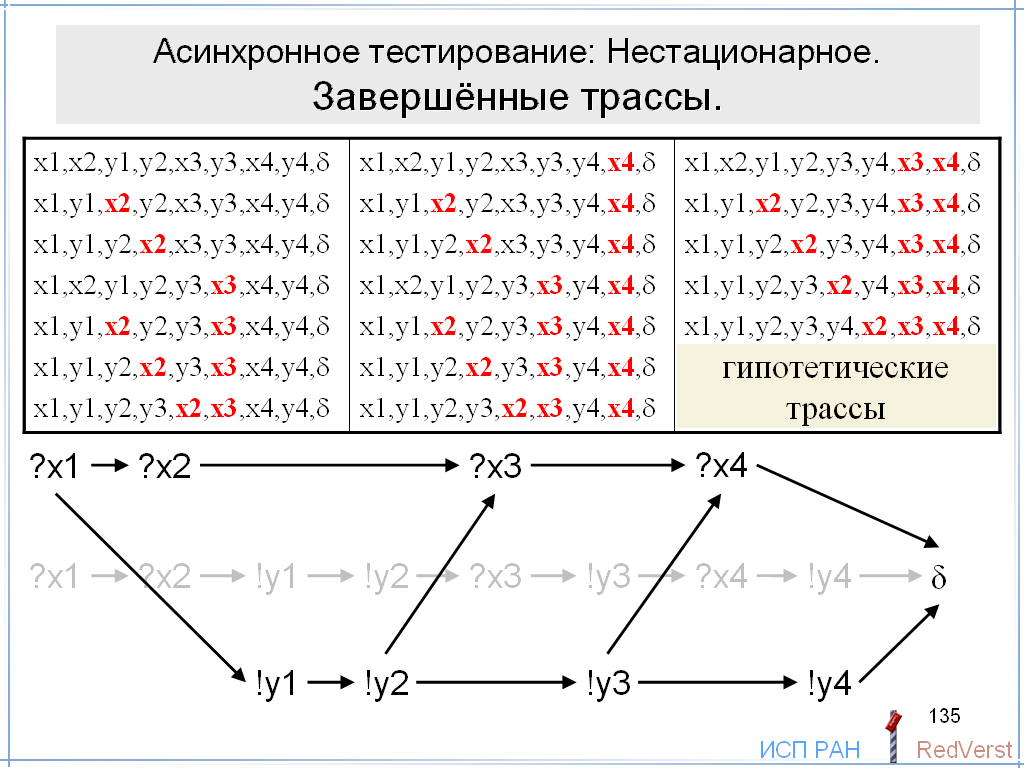
Правило стационарности: Наконец, трасса заканчивается стационарностью.

Мы перечислили все каузальные зависимости, которым должны подчиняться символы гипотетической трассы для данной среды передачи. В целом эти правила определяют частичный порядок своих символов.

Гипотетическая трасса – это любая трасса, то есть любой линейный порядок, не противоречащий такому частичному порядку.

Как мы уже говорили, гипотетическая трасса может проходить через стационарные состояния. Это означает, что гипотетическая трасса может не только заканчиваться стационарностью, но и содержать её символ внутри. Разумеется, за внутренней стационарностью может следовать только стимул или стационарность (это петля и её можно повторять сколько угодно раз), но не реакция. Для нашей среды это не имеет значения, поскольку тест такую внутреннюю стационарность всё равно не видит. Поэтому для простоты мы будем рассматривать гипотетические трассы без внутренней стационарности.

Для трассы на рисунке мы имеем 19 гипотетических трасс. Они отличаются тем, что стимулы могут сдвигаться к концу трассы до тех пор, пока это не нарушает каузальные зависимости. Красным жирным шрифтом выделены такие сдвинутые стимулы.



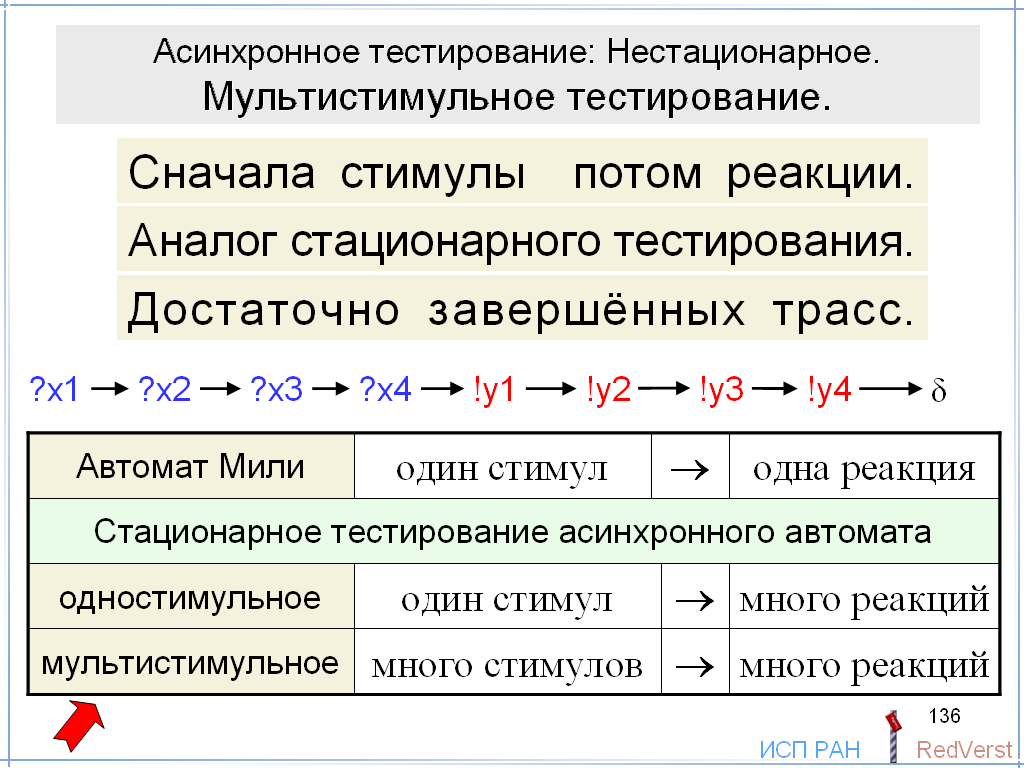
Наблюдаемая трасса, соответствующая одному шагу нестационарного тестирования, заканчивается стационарностью, но не содержит стационарность внутри. Тем самым, это завершённая трасса. Нужно только помнить, что мы рассматриваем трассы, начинающиеся не в начальном, а в текущем состоянии автомата.

Заметим, что при стационарном тестировании гипотетическая завершённая трасса всегда совпадает с наблюдаемой завершённой трассой. Действительно, стимул, с которого начинается наблюдаемая трасса, всегда является первым символом гипотетической трассы. При стационарном тестировании далее идут одни реакции, которые сохраняют свой порядок, а заканчивается всё стационарностью.

### Мультистимульное стационарное тестирование.

Частный, но важный и полезный случай тестирования, когда мы сначала выдаём последовательность стимулов, а потом получаем реакции до стационарности. В наблюдаемой трассе сначала идут все стимулы, потом все реакции и в конце – стационарность. Такое тестирование отличается от стационарного тестирования, которое мы рассматривали раньше, только тем, что мы выдаём не один стимул, а последовательность стимулов. В этом случае достаточно ограничиться рассмотрением только завершённых трасс – так же, как при обычном стационарном тестировании.

В целом у нас есть три случая, когда тестирование асинхронного автомата сводится к тестированию автомата Мили, получаемого из исходного автомата специальным преобразованием.

Тестирование автомата Мили: в ответ на один стимул мы получаем одну реакцию. В этом случае преобразование тождественное.

Одностимульное стационарное тестирование асинхронного автомата: в ответ на один стимул, подаваемый в стационарном состоянии, мы получаем последовательность реакций. Соответствующий автомат Мили – это стационарный автомат. Его состояния – это стационарные состояния исходного автомата. Переход из одного стационарного состояния в другое стационарное состояние помечен одним стимулом и последовательностью реакций.

Мультистимульное стационарное тестирование асинхронного автомата: в ответ на последовательность стимулов, которая начинает подаваться в стационарном состоянии, мы получаем последовательность реакций. Соответствующий автомат Мили – это мультистимульный стационарный автомат. Его состояния – это стационарные состояния исходного автомата. Переход из одного стационарного состояния в другое стационарное состояние помечен последовательностью стимулов и последовательностью реакций.

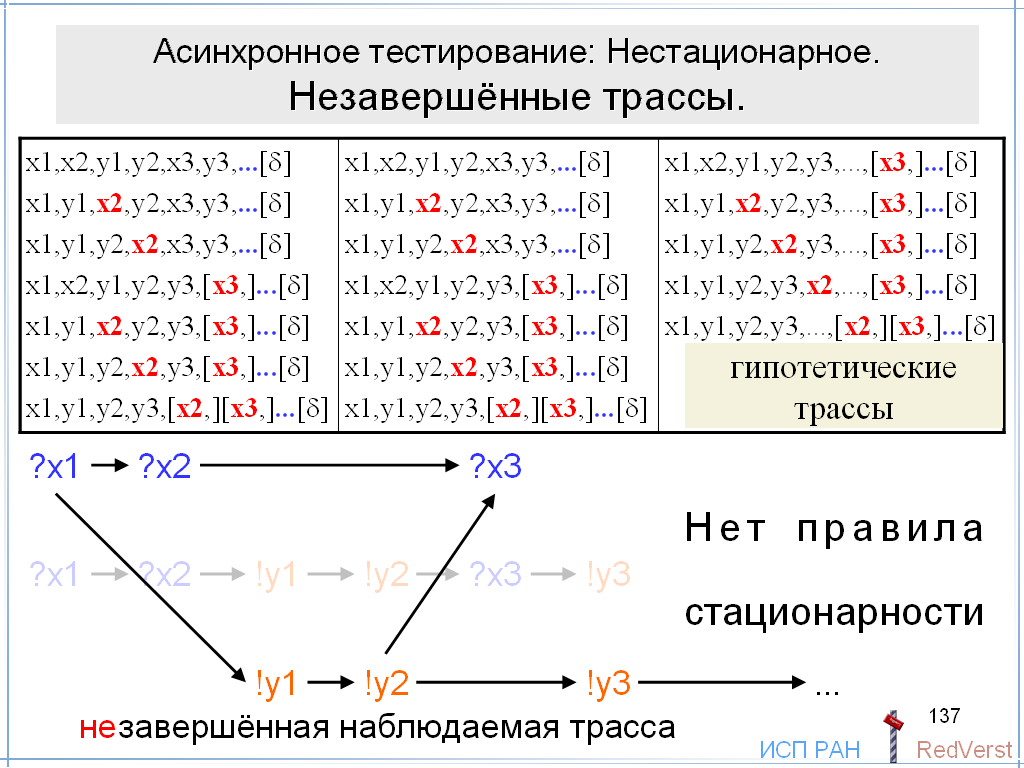
Можно сказать, что мультистимульное тестирование занимает промежуточное положение между стационарным и нестационарным тестированиями. По стратегии тестирования, основанной на обходе автомата Мили, это тестирование стационарное. Однако оно нуждается в умении перечислять реакции, допускаемые спецификацией после некоторой трассы, для того, чтобы определить безопасность последовательности подаваемых стимулов. Кроме того, одной наблюдаемой трассе здесь может соответствовать несколько гипотетических трасс, получающихся сдвигом стимулов к концу трассы. В этом оно схоже с нестационарным тестированием.

### Незавершённые трассы.

В общем случае мы можем перемежать выдачу стимулов и приём реакций. Безопасность выдаваемого стимула зависит от реакций, принятых к этому моменту времени. Поэтому мы должны по ходу дела, то есть ещё до завершения трассы, во-первых, проверять правильность получаемых реакций и, во-вторых, если реакции правильные, определять безопасные стимулы, которые можно было бы выдать в этот момент времени.

Незавершённая наблюдаемая трасса – это трасса, которая не заканчивается стационарностью.

Если трасса не завершена, то мы не знаем, получили мы все реакции, которые реализация уже выдала, или ещё не все. Поэтому мы считаем, что гипотетическая трасса может содержать дополнительные реакции. Эти дополнительные реакции, по правилу очерёдности реакций, идут после уже принятых реакций. Однако порядок следования ещё не принятых реакций и уже посланных стимулов может быть любым с учётом правила первого стимула.



Кроме того, мы не знаем, приняла ли реализация все посланные стимулы или ещё нет. Гарантированно можно говорить лишь о том, что, если тест получил хотя бы одну реакцию, то реализация приняла первый стимул. (Поскольку в стационарном состоянии, с которого мы начинаем, реакции не выдаются.)

Гипотетические трассы нам будут нужны для двух целей:

* Верификация: проверка, что реализация выдала правильные реакции.
* Безопасность: вычисление стимулов, которые можно безопасно посылать в реализацию после наблюдаемой трассы.

### Безопасность тестирования.

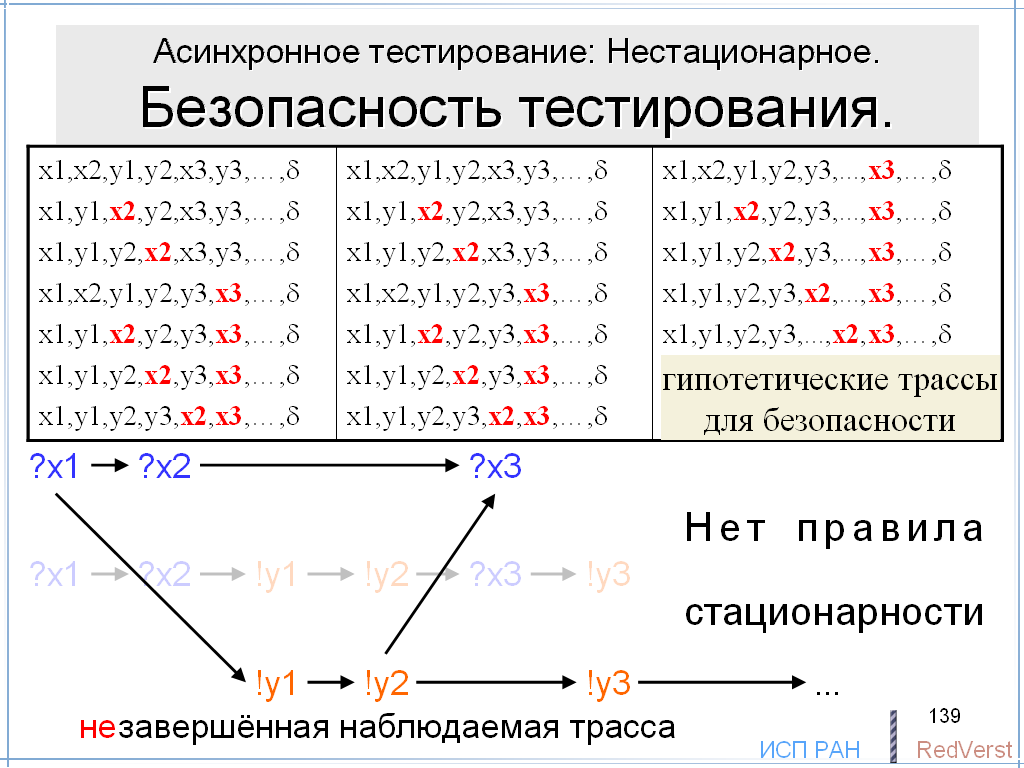
Как мы уже говорили, тестовый эксперимент должен быть безопасным, то есть не приводить к разрушению реализации. О безопасности данного тестового эксперимента при синхронном тестировании мы судили на основании гипотезы о безопасности, которая утверждает, что после трассы, которая есть как в реализации, так и в спецификации, можно безопасно давать только такой стимул, которым эта трасса безопасно продолжается в спецификации. Поскольку при стационарном асинхронном тестировании наблюдаемая трасса совпадает с гипотетической, нам достаточно этой же гипотезы о безопасности. Но для нестационарного асинхронного тестирования мы должны гипотезу о безопасности усилить.

Если в тестовом эксперименте мы получили некоторую наблюдаемую трассу, то мы должны посмотреть, какие трассы могла бы пройти спецификация, если бы в этом тестовом эксперименте она занимала место реализации. Иными словами, мы должны взять пересечение гипотетических трасс, порождаемых данной наблюдаемой трассой, и трасс спецификации, начинающихся с текущего спецификационного состояния.

Стимул, который мы можем послать после наблюдаемой трассы, будет принят реализацией после того, как она примет все стимулы и выдаст все реакции, которые есть в наблюдаемой трассе. При этом реализация пройдёт такую гипотетическую трассу, которая содержат все посланные стимулы, все полученные реакции и, быть может, ещё какие-то дополнительные реакции. Поскольку по нашему допущению полноты в нестационарном состоянии неспецифицированный стимул не приводит к разрушению, нас интересуют только стационарные состояния, в которых могут заканчиваться такие гипотетические трассы.

Иными словами, нам достаточно ограничиться завершёнными гипотетическими трассами, которые содержат все посланные стимулы и все полученные реакции.

Вот эти трассы для нашего примера.



Гипотеза о безопасности будет звучать так:

если стимул безопасен после каждой спецификационной трассы, порождаемой данной наблюдаемой трассой,

то этот стимул безопасен после любой реализационной трассы, порождаемой данной наблюдаемой трассой.

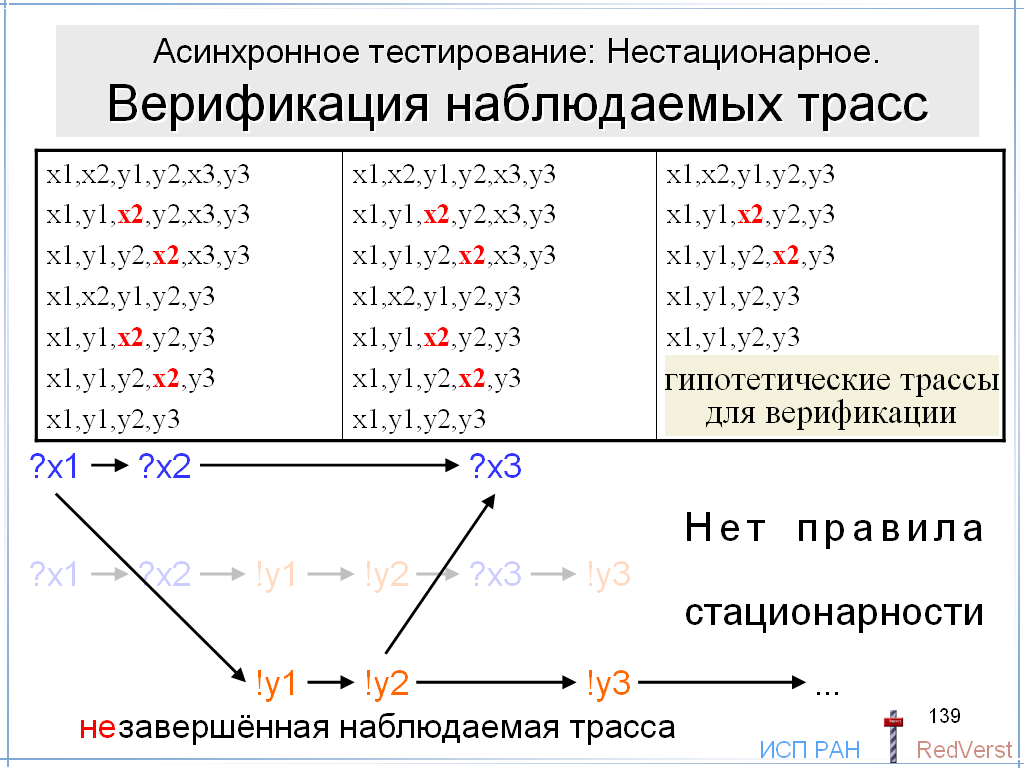
Эта гипотеза сильнее, поскольку реализационная трасса может не совпадать ни с одной из спецификационных трасс, порождаемых данной наблюдаемой трассой.

### Верификация наблюдаемых трасс.

Верификация наблюдаемой незавершённой трассы – это проверка того, что реализация выдала правильные реакции.

Для этого годятся завершённые гипотетические трассы, которые мы использовали для проверки безопасности стимулов. Однако они несут лишнюю информацию: для верификации нам не важны реакции и стационарность, которые мы ещё не получили.

Достаточно ограничиться трассами, которые содержат все полученные реакции, даже если они содержат не все посланные стимулы.



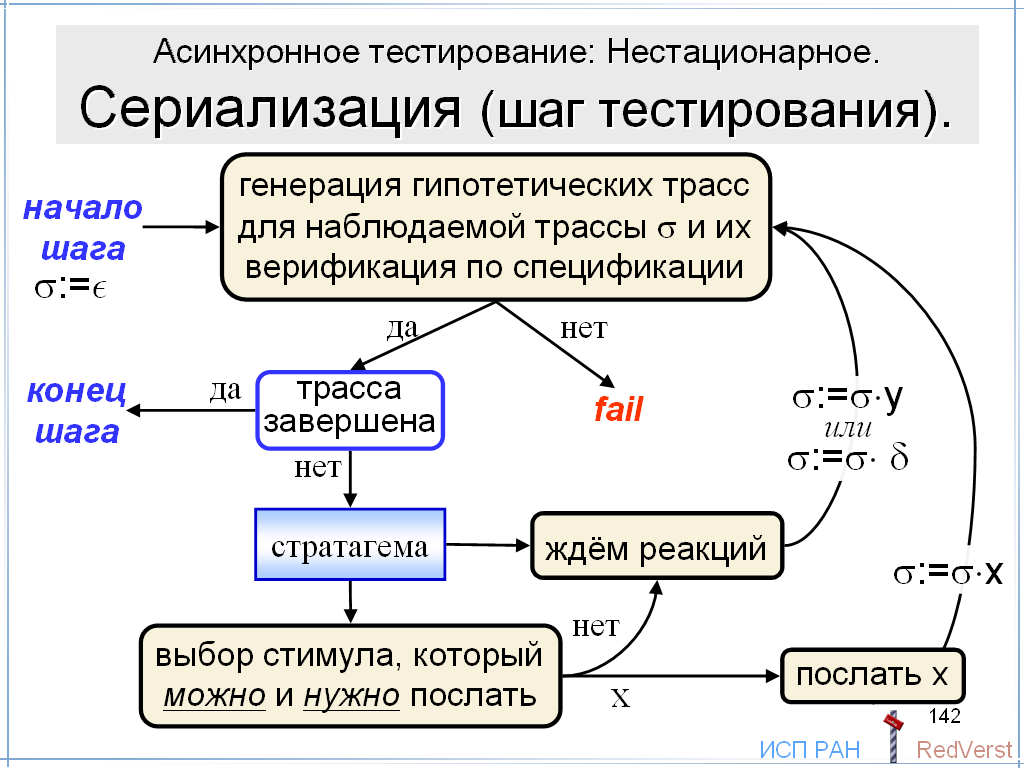
### Сериализация (шаг тестирования).

Теперь посмотрим, как работает тестовая система на одном шаге нестационарного тестирования.

Эта работа похожа на случай стационарного тестирования, Отличие в том, что мы можем давать стимулы, не дожидаясь стационарного состояния, и у нас могут получаться разные спецификационные трассы, порождаемые одной и той же наблюдаемой трассой.

Пусть в какой-то момент времени мы получили некоторую наблюдаемую трассу. Под трассой мы будем понимать трассу, которая наблюдалась не с начала тестирования, а с начала текущего шага тестирования.

В начале шага тестирования это пустая трасса.

Мы должны определить все спецификационные трассы, порождаемые этой наблюдаемой трассой. Это называется сериализацией наблюдаемой трассы.

Для этого сначала вычисляем все гипотетические трассы, порождаемые наблюдаемой трассой, которые нужны для верификации: трассы, которые содержат все полученные реакции, даже если они содержат не все посланные стимулы. Каждую такую гипотетическую трассу верифицируем по спецификации аналогично тому, как мы это делали в стационарном случае.

Если гипотетическая трасса отвергается, то, в отличие от стационарного случая, это не означает ошибку. Просто мы эту трассу отбрасываем и верифицируем следующую гипотетическую трассу.

Ошибка фиксируется, когда сериализация закончена, и все гипотетические трассы отброшены.

Если ошибок нет, то мы смотрим, завершена ли наблюдаемая трасса, то есть, заканчивается она стационарностью или нет.

Если трасса завершена, шаг тестирования завершён.

В противном случае, мы должны принять решение: дать следующий стимул или ждать реакций. Выбор определяется общей стратегией нестационарного тестирования.

Если мы решаем ждать реакций, то ждём реакций, а, получив реакцию или стационарность, повторяем описанный процесс для новой наблюдаемой трассы.

Если мы решаем выдать стимул, то стратегия тестирования определяет, какие стимулы нам нужно давать в текущей ситуации, то есть после пройденной наблюдаемой трассы. С другой стороны, мы должны определить, какие стимулы можно безопасно давать в реализацию. Здесь важно, что у нас может получиться не одна, а несколько спецификационных трасс. Стимул, который можно безопасно давать после данной наблюдаемой трассы, – это стимул, который в спецификации безопасен после каждой из полученных спецификационных трасс.

В целом мы ищем те стимулы, которые *можно* и *нужно* давать.

Если таких стимулов нет, нам ничего другого не остаётся, кроме как ждать реакций.

Если такой стимул есть, мы можем его дать и повторить процесс для новой наблюдаемой трассы.

### Перечисление.

При нестационарном тестировании нам требуется перечислять в состоянии спецификации стимулы, постсостояния и реакция. Это нужно как для верификации, так и для определения безопасных стимулов.

### ПРОБЛЕМЫ: Выбор стратегии тестирования (аналог обхода автомата).

В целом у теста должна быть некоторая стратегия тестирования, которая определяет, какие стимулы нужно давать в данной ситуации, а какие не нужно, например, потому, что тест уже давал этот стимул в этой ситуации. В то же время тест адаптивно подстраивается под текущую ситуацию, определяемую получаемыми от реализации реакциями. Стимул даётся тогда, когда его не только можно, но и нужно давать в данной ситуации. В противном случае тест ожидает реакций, изменяющих ситуацию.

Прежде всего, отметим, что без каких-либо дополнительных условий нестационарный тест не может гарантированно проверить ничего, кроме того, что проверяет стационарный тест. Например, если тест работает достаточно медленно, то, пока он «думает», реализация успевает принять стимул, выдать все реакции и перейти в стационарное состояние. Таким образом, в любом тестировании стимулы всё равно будут приниматься только в стационарных состояниях, и, следовательно, тестирование будет стационарным.

Для того чтобы говорить о каких-то гарантиях при нестационарном тестировании, нужны дополнительные тестовые возможности и соответствующие реализационные гипотезы.

Что можно предложить?

Временные ограничения. Если считать, что, наоборот, реализация работает достаточно медленно, а среда и тест – достаточно быстро, то мы можем выдать во входную очередь серию стимулов. При этом мы будем уверены, что реализация может принять второй стимул в первом же состоянии, в которое она попадёт после первого стимула, и в котором этот стимул принимается. Аналогично для остальных стимулов. Однако само по себе это недостаточно для того, чтобы реализация действительно принимала стимулы тогда, когда они уже есть во входной очереди.

Поэтому дополнительно требуется реализационная гипотеза о приоритете приёма стимулов над выдачей реакций и τ-переходами.

Управление погодой и мистическим синхронизатором. Это тестовые возможности общего вида, которые заставляют реализацию вести себя так или иначе и обеспечивают передачу стимулов из входной очереди в реализацию или передачу реакций из реализации в выходную очередь тогда, когда нам это нужно. Фактически, все такие тестовые возможности вносят в тестирование элементы синхронизации, присущие синхронному тестированию.

Вероятности переходов. Введение вероятностей переходов позволяет строить нестационарные тесты, которые эти переходы проверяют с той или иной вероятностью.

Дополнительно нужно сделать следующее важное замечание:

Формально, если тестирование прошло при всех «погодных условиях», после этого в спецификации могут остаться непроверенные стимулы, определённые в нестационарных состояниях. Однако это такие стимулы, которые ни при каких условиях нельзя безопасно давать в реализацию. Можно сказать, что такие стимулы избыточны в спецификации. Причин, по которым эти стимулы всё же определены, несколько.

Первая причина: это могут быть стимулы, которые опасны при асинхронном тестировании, но безопасны при синхронном тестировании.

Вторую причину мы уже обсуждали, когда говорили о маркировке разрушением поведения системы, которое нежелательно тестировать по тем или иным причинам. Для одной и той же спецификации можно было бы применять различные маркировки и, тем самым, выполнять то или иное тестирование. В зависимости от маркировки одни и те же стимулы могут быть как разрушающими, так и безопасными.



### Среда передачи.

Стационарное и нестационарное тестирование мы рассматривали для фиксированной среды передачи – две неограниченные очереди: очередь стимулов и очередь реакций. Если среда передачи другая, то в тестировании многое меняется.

Тестирование, которое можно вести для разных сред из некоторого класса сред,  
мы будем называть *гипертестированием*.

Мы будем исходить из общей схемы тестирования, которая состоит из последовательности шагов тестирования. На каждом шаге мы имеем наблюдаемую трассу, а среда задаёт *отображение наблюдаемой трассы во множество гипотетических трасс*.

Для спецификации мы пока по-прежнему будем предполагать наше *комбинированное допущение полноты*:

неспецифицированный стимул блокируется в стабильном нестационарном состоянии и ведёт к разрушению в стационарном состоянии.

Итак, мы рассматриваем общий случай среды передачи, но такой, который удовлетворяет следующим ограничениям:

* Среда передаёт стимулы и реакции без потерь.
* Среда не генерирует лишних стимулов и реакций.

Теперь нам нужны ограничения, которые,

* *во-первых*, позволяли бы определять стационарность реализации по тайм-ауту при ожидании реакций;
* *во-вторых*, шаг тестирования предполагает, что при срабатывании тайм-аута среда пуста: реализация уже приняла все стимулы, посланные тестом, а тест принял все реакции, выданные реализацией;
* *в-третьих*, нам по-прежнему нужно, чтобы не возникало d`adlock`а при передаче стимула из теста в среду.

Асинхронное тестирование реализации мы определили как синхронное тестирование композиции реализации со средой передачи. Поэтому тайм-аут в тесте при ожидании реакций для любой среды означает дивергенцию или стационарное состояние этой композиции. В общем случае такое состояние вовсе не обязательно означает стационарность реализации или пустоту среды.

Мы налагаем на среду и реализацию следующие дополнительные ограничения:

* Композиция среды и реализации конвергентна.
* В стационарном состоянии композиции среда пуста, а реализация находится в стационарном состоянии.
* В стабильном состоянии композиция принимает от теста все стимулы.

Из наших ограничений следует, что гипотетическая трасса, порождаемая данной наблюдаемой трассой, состоит из тех же самых стимулов и реакций, но, быть может, расположенных в другом порядке.

Здесь мы по-прежнему рассматриваем наблюдаемые трассы, начинающиеся в текущем стационарном состоянии.

У нас по-прежнему будут работать каузальные правила: правило «стимул после реакции», правило первого стимула и правило стационарности.

Отличие от одной входной очереди в том, что для правила первого стимула таким первым стимулом гипотетической трассы будет не обязательно первый стимул наблюдаемой трассы.

Эти три каузальных правила являются общими для всех таких сред.

Однако правило очерёдности стимулов и правило очерёдности реакций верны только для среды с одной неограниченной очередью стимулов и одной неограниченной очередью реакций. Для других сред могут быть другие каузальные правила, играющие аналогичную роль.

Сначала мы рассмотрим несколько характерных примеров различных сред передачи.

### Среда: несколько очередей.

Первым примером является среда, моделируемая не одной входной и одной выходной очередями, а большим числом очередей – входных и выходных. Будем считать, что алфавит стимулов разбит на столько классов, сколько есть входных очередей, так что стимул из i-го класса помещается в i-ую очередь. Аналогично, алфавит реакций разбит на столько классов, сколько есть выходных очередей, так что реакция из j-го класса помещается в j-ую очередь.

Пусть, например, стимулы и реакции – это натуральные числа. Среда имеет две входные очереди: для чётных и нечётных стимулов, и две выходные очереди: для чётных и нечётных реакций.

Множество гипотетических трасс определяется,

* во-первых, линейным порядком стимулов в каждой входной очереди,
* во-вторых, линейным порядком реакций в каждой выходной очереди.

Это модифицированные правила очерёдности стимулов и реакций, которые у нас были для одной входной и одной выходной очереди.



Кроме того, работают общие каузальные правила:

* правило «стимул после реакции»,
* правило первого стимула (но не обязательно первый посылаемый стимул),
* и правило стационарности.

Эти трассы нельзя задать частичным порядком на множестве стимулов и реакций наблюдаемой трассы. Причина – в модифицированном правиле первого стимула. Первый символ трассы должен быть стимулом, но здесь у нас на эту роль претендуют два стимула – первые стимулы двух входных очередей. Один из них будет первым символом трассы, а другой может быть принят реализацией позже некоторых реакций.

### Передача стимулов и реакций, минуя среду.



При асинхронном тестировании у нас может быть возможность передавать часть стимулов и реакций непосредственно между реализацией и тестом, минуя среду. Такие стимулы и реакции будем называть *прямыми*. Это допускается оператором параллельной композиции, если алфавиты реализации, среды и теста удовлетворяют соответствующему условию.

### Передача стимулов и реакций, минуя среду (пример).

Рассмотрим пример, аналогичный предыдущему примеру: 2 входные очереди и 2 выходные очереди. Но только у нас будет стимул и реакция, которые передаются, минуя среду. Они имеют номер 3.



У нас появляются новые каузальные зависимости:

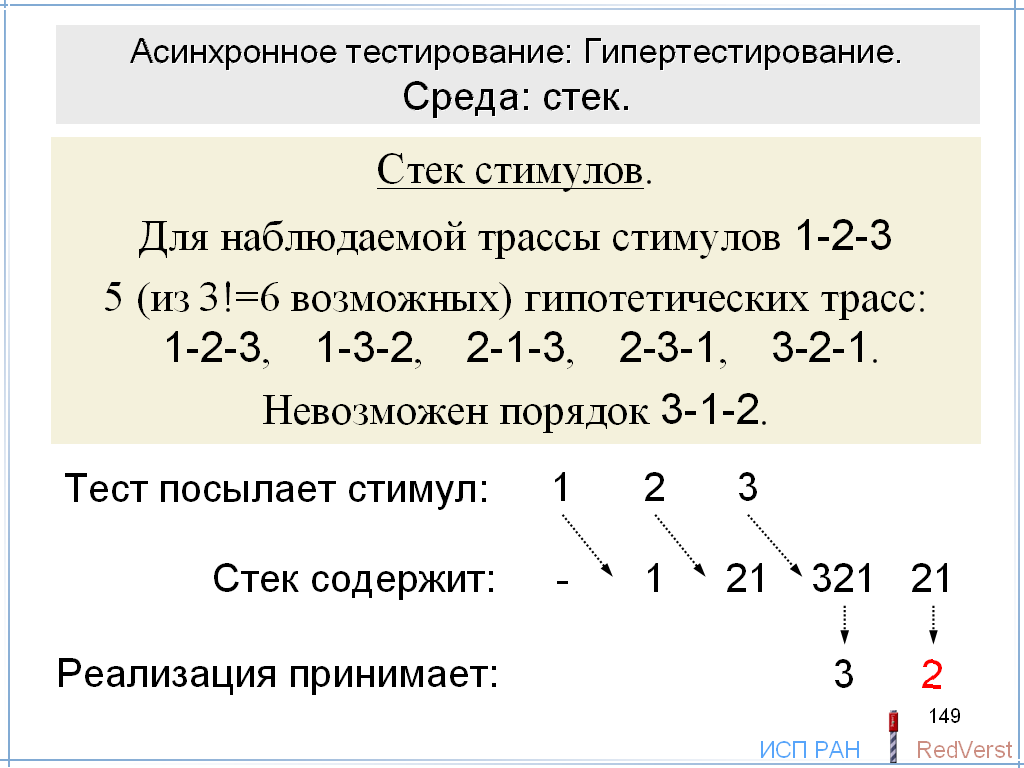
* Порядок прямых стимулов и реакций сохраняется при переходе от наблюдаемой трассы к гипотетической трассе.
* Прямой стимул принимается реализацией раньше, чем любой стимул, посланный тестом после этого прямого стимула.
* Прямая реакция выдаётся реализацией позже, чем любая реакция, полученная тестом до этой прямой реакции.

### Среда: «куча» (множество).

Другая среда – это «куча», то есть неупорядоченное множество.

Например, если вместо входной очереди используется куча стимулов, то для наблюдаемой трассы из трёх стимулов 1,2,3 любая их линейная последовательность является гипотетической трассой.

### Среда: стек.

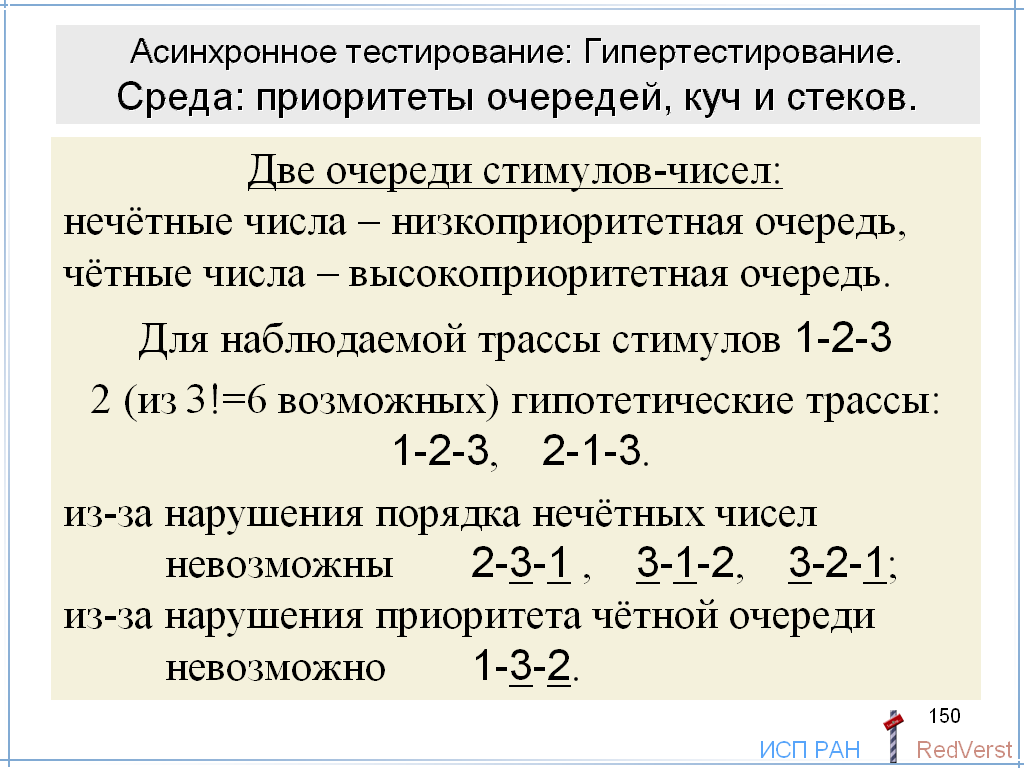
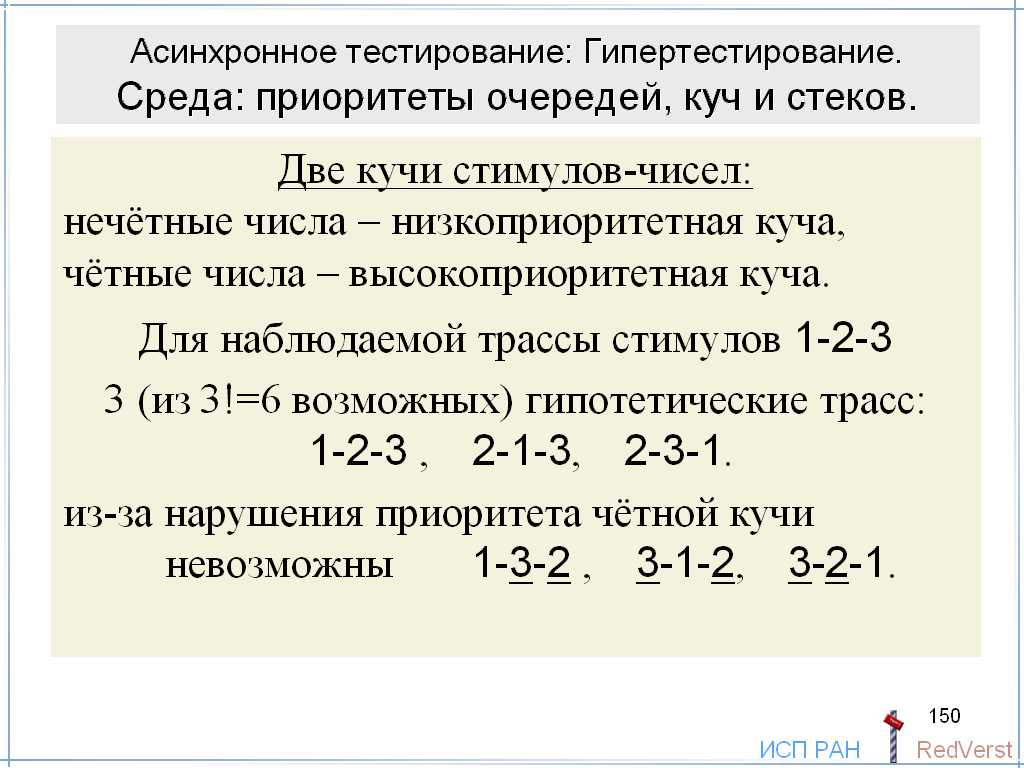
Если вместо входной очереди используется стек, то для наблюдаемой трассы из трёх стимулов 1,2,3 пять из шести линейных порядков являются гипотетическими трассами.

Один линейный порядок не может быть гипотетической трассой – это порядок 3,1,2.

Реализация может первым принять стимул 3 только в том случае, когда в стеке находятся стимулы 1 и 2. Поскольку в вершине стека расположен как раз стимул 2, реализация не может принять стимул 1 раньше стимула 2.

### Среда: приоритеты очередей, куч и стеков.

Среда передачи стимулов так же, как передачи реакций, может быть комбинированной, состоящей из набора очередей, куч и стеков. Между этими компонентами, кроме того, могут быть установлены приоритеты.

Например, стимулы – это натуральные числа и они поступают в две входные очереди с разными приоритетами. Среда предлагает реализации стимул из низкоприоритетной очереди только в том случае, когда высокоприоритетная очередь пуста. Пусть в низкоприоритетную очередь поступают нечётные числа, а в высокоприоритетную очередь – чётные числа. Для наблюдаемой трассы из трёх стимулов 1,2,3 гипотетическими трассами являются только два линейных порядка из шести. Три линейных порядка не годятся, потому что они нарушают порядок следования стимулов в одной очереди. Четвёртый порядок нарушает приоритетность очередей.

Если в этом примере заменить очереди кучами, то три линейных порядка из шести отбраковываются, потому что они нарушают приоритетность куч.

### Зависимость между стимулами и реакциями.

До сих пор мы рассматривали примеры сред, в которых стимулы и реакции передаются независимо друг от друга. Между стимулами и реакциями (или стационарностью) не было никакого порядка следования, если не считать общих каузальных правил: правило первого стимула, предшествующего всем реакциям, правило «стимул после реакции», и правило стационарности, завершающей трассу.

Однако в некоторых случаях между стимулами и реакциями (или стационарностью) есть дополнительные каузальные зависимости, которые мы можем учитывать при построении гипотетических трасс.

Первый пример – привязанные реакции. Про такую реакцию мы всегда знаем, что она выдана реализацией после приёма стимула, к которому она привязана.

Про немедленную реакцию мы знаем больше: она идёт в гипотетической трассе непосредственно после стимула, ответом на который является.

Среда может сама устанавливать дополнительные каузальные зависимости между стимулами и реакциями. Например, среда может предлагать реализации некоторые стимулы только тогда, когда реализация не выдаёт реакций, то есть, находится в стационарном состоянии. Такие стимулы можно назвать стационарными. Для обнаружения стационарности в реализации среда может использовать θ-переход при ожидании реакций от реализации аналогично тому, как это делает тест. В гипотетической трассе таким стационарным стимулам непосредственно предшествуют внутренние символы стационарности δ. Фактически, для такой среды нестационарное тестирование сводится к стационарному.

### Автомат среды: Передача стимулов и реакций.

Итак, мы рассмотрели примеры различных сред. В общем, виде среда задаётся асинхронным автоматом. Посмотрим, как наложенные нами ограничения на среду, могут быть сформулированы в терминах автомата среды.

Мы уже говорили, что формально взаимодействие теста и среды, с одной стороны, и среды и реализации, с другой стороны, происходят в разных, непересекающихся алфавитах стимулов и реакциях. Поскольку мы хотим, чтобы среда только передавала стимулы и реакции без потерь и без генерации лишних стимулов и реакций, можно считать, что между стимулами теста и стимулами реализации существует взаимно-однозначное соответствие так же, как между реакциями теста и реакциями реализации. На рисунке стимулы и реакции реализации обозначены красным цветом, а стимулы и реакции теста – синим цветом.



Среда содержит стимулы и реакции, которые выданы одной стороной (тестом или реализацией), но ещё не приняты другой стороной. Это можно описать в виде мультимножества, который определён в каждом состоянии среды. В мультимножестве, в отличие от обычного множества, стимул или реакция может входить не один, а несколько раз. Формально, мультимножество – это отображение, которое каждому элементу ставит в соответствие число вхождений. Если число вхождений стимула или реакции равно нулю, это означает, что в среде нет этого стимула или этой реакции.

Правило, согласно которому среда передаёт стимулы без потерь и не генерирует лишних стимулов, означает соответствующие правила по изменению мультимножества при передаче стимула. Если стимул принимается из теста, число вхождений этого стимула увеличивается на 1. Стимул может выдаваться в реализацию только в том случае, если число его вхождений больше нуля, в этом случае при переходе число вхождений стимула уменьшается на 1. Аналогично изменяется число вхождений реакции при её передаче в среду или из среды.

Тестирование основано на заданном отображении наблюдаемой трассы в гипотетические трассы. В общем случае это отображение зависит от состояния среды в начале шага тестирования. Мы уже потребовали, чтобы в начале шага тестирования среда была пуста, но отсюда не следует, что такое состояние среды единственное. Поэтому нам нужно дополнительно потребовать, чтобы в каждом состоянии автомата среды, соответствующем отсутствию в среде стимулов и реакций, отображение трасс было одно и то же. В начальном состоянии среда пуста.

### Автомат среды: Конвергентность композиции среды и реализации.

Теперь посмотрим, что нужно, чтобы композиция среды и реализации была конвергентна. Напомним, что конвергентность нужна для того, чтобы в тесте правильно интерпретировать истечение тайм-аута при ожидании реакций. Такая интерпретация должна означать стационарность, а не дивергенцию.

Если тест ожидает реакций, взаимодействие среды и реализации состоит из переходов четырёх типов:

* 1. асинхронные τ-переходы в реализации,
  2. асинхронные τ-переходы в среде,
  3. синхронные переходы по передаче стимула из среды в реализацию,
  4. синхронные переходы по передаче реакции из реализации в среду.

Посмотрим, сколько может быть переходов разного типа в цепочке переходов композиции. Для конвергентной реализации число переходов первого типа конечно. Поскольку тест от начала работы мог послать в среду только конечное число стимулов, а среда не генерирует лишних стимулов, число переходов третьего типа также конечно. Для конечности числа переходов второго типа требуется конвергентность среды.

Для конечности переходов четвёртого типа требуется

1. либо отсутствие осцилляции в реализации,
2. либо отсутствие в среде бесконечной цепочки τ-переходов и переходов по реакциям.

Последний случай также перекрывает требование конвергентности среды.

Заметим, что неограниченная очередь реакций конвергентна, но не удовлетворяет условию B): в ней есть бесконечная цепочка приёма реакций от реализации. Именно поэтому для такой среды мы требовали, чтобы реализация была неосциллирующей. В общем же случае возможен как вариант A), так и вариант B).

Когда мы рассматривали шаг асинхронного тестирования, мы потребовали не только конвергентности композиции, но и завершённости шага тестирования: не бывает бесконечной последовательности получаемых реакций, каждая последовательность реакций рано или поздно заканчивается стационарностью.

Поэтому пока будем требовать, чтобы

реализация была неосциллирующей.

Как можно тестировать осциллирующие реализации, мы посмотрим позже.

### Автомат среды: Таблица условий.

Теперь посмотрим, что нужно, чтобы выполнялись правила стационарности и отсутствия блокирующего deadlock`а. Напомним, что мы требуем, чтобы стационарность композиции среды и реализации означала стационарность реализации и пустоту среды. В этом случае обнаружение стационарности в конце шага тестирования означает достижение стационарного состояния реализации, все посланные тестом стимулы приняты реализацией, а все выданные реализацией реакции получены тестом. Гипотетическая трасса состоит из тех же стимулов и реакций, что и наблюдаемая трасса и выполняются общие каузальные правила.

Мы рассмотрим четыре вида стабильных состояний среды в зависимости от того, имеются или нет в ней не переданные стимулы и реакции. В терминах мультимножества это означает наличие или отсутствие стимулов или реакций с ненулевым числом вхождений.

Соответственно, мы рассмотрим два вида состояний теста: принимающее состояние, когда тест ждёт всех реакций и тайм-аута, и посылающее, когда он посылает один стимул. Эти стабильные состояния теста стационарное и нестационарное, соответственно. Точно так же мы рассмотрим два вида стабильных состояний реализации: стационарное, когда реализация принимает все стимулы, быть может, с разрушением, и не выдаёт реакции, и нестационарное, когда реализация выдаёт хотя бы одну реакцию.

Всего у нас получается 16 вариантов, которые на рисунке находятся в центральном квадрате 4x4. Левые 8 вариантов, когда тест ждёт реакций, соответствуют правилу стационарности, а правые 8 вариантов, когда тест посылает стимул, соответствуют запрету блокирующего deadlock`а.



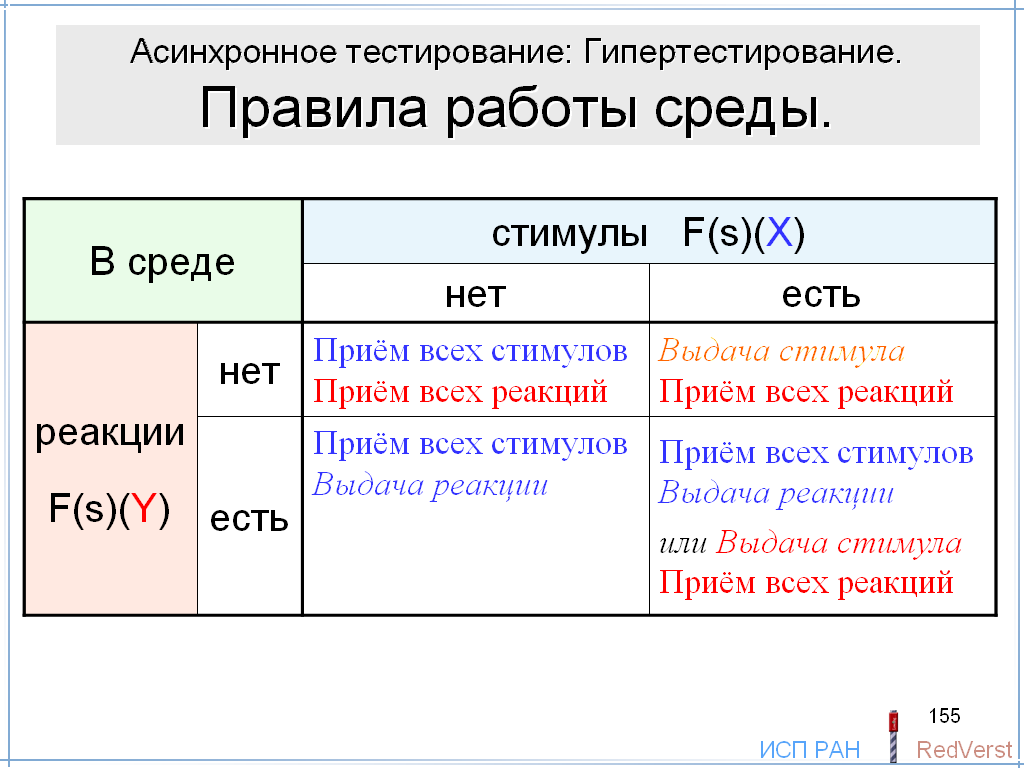
Для стационарности только один из 8 вариантов удовлетворяет правилу стационарности: реализация находится в стационарном состоянии, а среда пуста. В этом случае возникает deadlock и тест совершает θ-переход. В остальных 7 вариантах записаны необходимые и достаточные условия отсутствия deadlock`а для любой реализации. Большая буква X или Y под знаком вопроса означают приём средой всех стимулов от теста или всех реакций от реализации, соответственно. Маленькая буква x или y под восклицательным знаком означает, что среда предлагает хотя бы один стимул для передачи в реализацию или хотя бы одну реакцию для передачи в тест, соответственно.

Для блокировки имеется 8 вариантов, для каждого из которых записано необходимое и достаточное условие отсутствия deadlock`а для любой реализации.

Строка таблицы, содержащая 4 варианта, соответствует одному виду состояния среды. Общее условие для данного вида состояния среды, очевидно, является конъюнкцией условий этих 4-х вариантов. Оно записано в самом правом итоговом столбце, выделенном зелёным цветом.

### Автомат среды: Правила работы.

Этот итог можно описать отдельно как правила работы среды в зависимости от вида её состояния, то есть от наличия или отсутствия в среде не переданных стимулов и реакций.



Среда должна принимать все стимулы от теста, если в ней нет стимулов, и все реакции от реализации, если в ней нет реакций.

Среда должна выдавать в реализацию хотя бы один стимул, если у неё есть не переданные стимулы, и нет реакций.

Соответственно, среда должна выдавать хотя бы одну реакцию в тест, если у неё есть не переданные реакции, и нет стимулов.

Если же в среде есть и стимулы и реакции, то для неё допустимы два типа поведения: 1) среда может выдавать реакцию в тест и принимать все стимулы от теста; 2) среда может выдавать стимул в реализацию и принимать все реакции от реализации.

Разумеется, среда может делать что-то ещё, кроме того, что от неё требуется. Например, при наличии в ней и стимулов и реакций, она может принимать все стимулы от теста и все реакции от реализации и выдавать некоторые не переданные стимулы в реализацию и некоторые не переданные реакции в тест.

### Автомат среды: Дополнительные примеры сред.

Основываясь на этих правилах, мы можем рассмотреть дополнительные примеры сред.

Ограниченная очередь стимулов и неограниченная очередь реакций. Что произойдёт, когда входная очередь заполнена, а тест выдаёт ещё один стимул? В этом случае среда не принимает стимул от теста, но она предлагает реализации стимул, находящийся в голове входной очереди, и принимает от реализации любую реакцию в свою неограниченную выходную очередь. Если реализация находится в стационарном состоянии, она примет предлагаемый стимул и во входной очереди появится место для нового стимула от теста. Если реализация в нестационарном стабильном состоянии не принимает головной стимул входной очереди, она выдаёт реакцию, которую среда примет. Разумеется, поскольку выходная очередь не ограничена, среда имеет бесконечную цепочку приёма реакций от реализации и для того, чтобы композиция среды и реализации была конвергентной, реализация должна быть неосциллирующей.

Неограниченная очередь стимулов и ограниченная очередь реакций. Блокирующий deadlock возникнуть не может, поскольку среда всегда готова принять стимул в свою неограниченную входную очередь. Что произойдёт, когда будет заполнена выходная очередь? Среда принимает все стимулы от теста и предлагает тесту реакцию, находящуюся в голове выходной очереди. Если тест ждёт реакций, он эту реакцию получит. Если тест посылает стимул, он будет принят. В этой среде нет бесконечной цепочки приёма реакций от реализации, и поэтому для конвергентности композиции нам не требуется, чтобы в реализации не было осцилляции. Однако для завершённости шага тестирования требование отсутствия осцилляции в реализации пока остаётся.

Заметим, что обе очереди, входная и выходная, не могут быть ограниченными, так как это может привести к блокирующему deadlock`у. Когда обе очереди заполнены, среда не принимает стимулы и реакции. Если реализация в стабильном нестационарном состоянии не принимает головной стимул входной очереди и посылает реакции, эти реакции блокируются средой. Если тест посылает стимул в среду, этот стимул также блокируется средой.

### ПРОБЛЕМЫ: Автомат среды: как по автомату среды построить отображение наблюдаемых трасс в гипотетические трассы?

Для конкретных примеров среды, которые мы рассмотрели, легко описать отображение наблюдаемых трасс в гипотетические трассы. Однако в общем случае остаётся проблема: как по автомату среды вычислить такое отображение?

### Внешняя блокировка: Адаптивное тестирование.

До сих пор мы считали, что среда не блокирует стимул, посылаемый тестом. Если это ограничение снять, то такая блокировка, тем не менее, вовсе не означает блокировку этого стимула реализацией. Более того, такая блокировка может происходить и тогда, когда реализация по-прежнему удовлетворяет нашему допущению полноты, то есть, не блокирует стимулы в стационарном состоянии. Например, среда имеет ограниченную ёмкость для стимулов и не принимает стимул от теста, когда эта ёмкость полностью заполнена, а реализация не принимает стимулы потому, что находится в нестационарном состоянии.

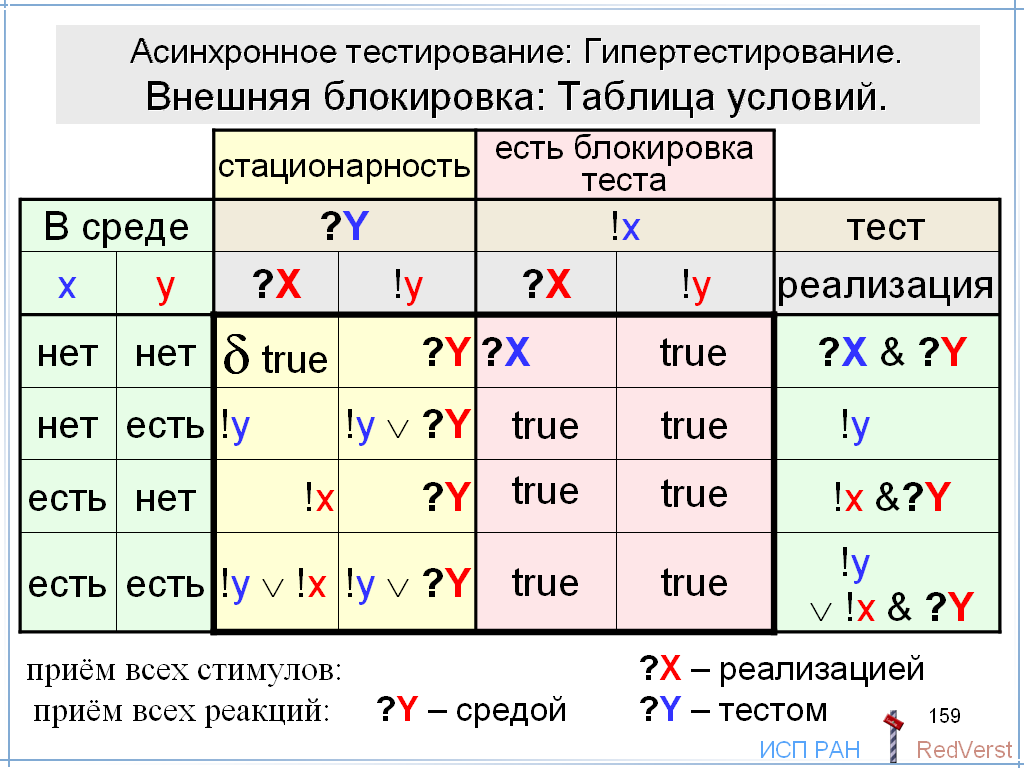
Блокировку стимулов, передаваемых из теста в среду, будем называть *внешней блокировкой*.

Внешнюю блокировку тест может обнаружить с помощью θ-перехода (тайм-аута) в посылающем состоянии.

Что может делать тест после обнаружения внешней блокировки? В общем случае он может попробовать послать другой стимул или начать ждать реакций.

### Внешняя блокировка: Таблица условий.

Посмотрим, что меняется в таблице условий для автомата среды.



Разрешая внешнюю блокировку, мы уже не можем требовать, чтобы среда в любой ситуации принимала любой стимул, посылаемый тестом.

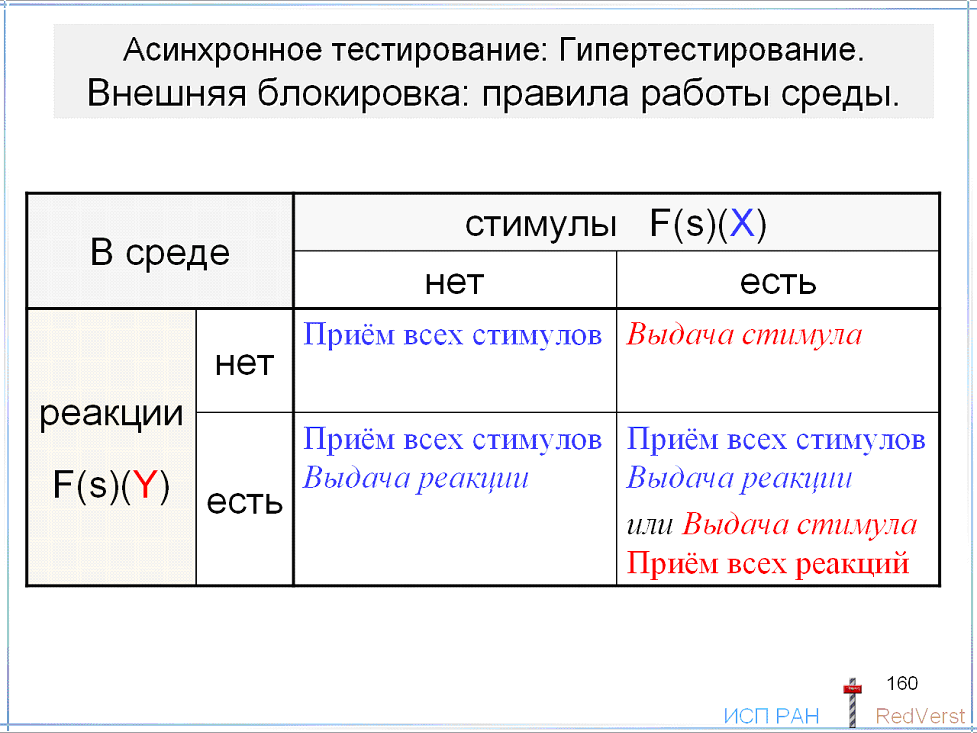
Единственный случай, когда это необходимо, – это пустая среда. Если в этой ситуации среда не примет стимул, тест начнёт ждать реакций. В среде реакций нет и, если реализация находится в стационарном состоянии, реакций и не будет. Таким образом, тест обнаружит стационарность, что правильно, но дальнейшая работа невозможна.

Итак, мы оставляем требование, чтобы среда принимала все стимулы от теста в том случае, когда среда пуста. В остальных случаях мы не накладываем на среду никаких ограничений.

В итоговом столбце условий получается, что среда не обязана принимать стимулы от теста, если в ней есть реакции. Адаптивный тест, обнаружив внешнюю блокировку, может выбрать эти реакции и тогда гарантированно среда примет от него любой стимул.

### Внешняя блокировка: Правила работы среды.

Правила работы среды меняются следующим образом.



Приём всех стимулов от теста обязателен только для пустой среды.

Если в среде нет реакций, она обязана принимать все реакции из реализации.

Если в среде есть реакции, она обязана хотя бы одну из них предлагать тесту.

Среда обязана предлагать хотя бы один стимул реализации, если в среде нет реакций. Но, если в среде есть и стимулы и реакции, то она может выбирать: выдавать реакцию в тест или выдавать стимул в реализацию и принимать все реакции от реализации.

В качестве примера годится любой входной и любой независимый выходной набор очередей, куч и стеков.

Более интересен пример среды, в которой ограничено сверху суммарное число стимулов и реакций, находящихся в среде. Без дополнительных ограничений такая среда может не удовлетворять нашим условиям. Например, среда всегда принимает стимулы из теста, пока она не заполнена. Тогда может получиться, что реализация принимает первый стимул, посланный тестом, и переходит в нестационарное состояние, намереваясь выдавать реакции. В этот момент среда пуста и тест успевает заполнить её своими стимулами. После этого тест не может послать стимул, реализация не может послать реакцию, и тест не может выбрать реакции из среды, потому что их в ней нет. В тесте возникает тайм-аут при ожидании реакций, но это ложная стационарность. Ограничение, которое нужно наложить на такую среду, следующее: она должна блокировать стимулы из теста, если в ней нет реакций и есть только одно свободное место. Это место среда резервирует для реакции от реализации.

### Внутренняя блокировка. Порты.

До сих пор мы предполагали, что в стационарных состояниях нет блокировки стимулов. Каждый стимул в таком состоянии принимается, а если он не специфицирован, то это означает приём с возможным разрушением.

Для одной входной очереди такой подход естественен, поскольку, если реализация в стационарном состоянии не принимает головной стимул очереди, то она останавливается и никакими действиями теста мы не сдвинем реализацию с мёртвой точки.

Теперь мы ослабим требование к реализации, разрешив в стационарном состоянии не только приём стимула и разрушение, но и блокировку. В нестационарном состоянии по-прежнему допускается блокировка стимула, но принимаемый стимул не разрушающий.

Ситуация, когда реализация находится в стационарном состоянии, но реализация не принимает ни один из стимулов, предлагаемых средой, хотя такие стимулы есть, называется *внутренней блокировкой*.

Для реализации, которая принимает все стимулы в каждом стационарном состоянии, не бывает внутренней блокировки. В общем же случае это не так.

Если никакими действиями теста мы не можем снять внутреннюю блокировку, она называется *постоянной*. Такая постоянная блокировка может возникнуть для одной входной очереди, если реализация в стационарном состоянии принимает не все стимулы.

Также, если стек или куча ограниченной ёмкости, то их заполнение может привести к постоянной блокировке, если реализация принимает не все стимулы из этого стека или кучи.

Однако для *не*ограниченных кучи или стека внутренняя блокировка всегда является *временной*, то есть может быть снята некоторыми действиями теста. Если реализация принимает хотя бы один стимул, то тест может послать этот стимул, и он окажется в куче или в голове стека, и будет принят реализацией.

При наличии нескольких входных очередей внутренняя блокировка будет постоянной только в том случае, когда реализация принимает стимулы только из тех очередей, которые не пусты, но принимаемые стимулы не совпадают с головными стимулами этих очередей. Если реализация принимает стимул хотя бы из одной пустой очереди, то тест может послать в эту очередь этот стимул, и реализация примет его.

Например, можно считать естественным поведение реализации, которая в некотором стационарном состоянии принимает все стимулы из первой очереди и не принимает, или принимает не все стимулы из второй очереди. Внутренняя блокировка снимается посылкой тестом любого стимула в первую очередь.

В общем случае можно рассматривать системы с *портами*. Принимаемый реализацией или передаваемый тестом стимул принимается или передаётся через некоторый порт. Иными словами, мы расширяем алфавит стимулов: теперь стимул – это пара <номер порта, стимул>. Это эквивалентно разбиению алфавита на непересекающиеся классы, каждому классу соответствует один порт. Важно, что это разбиение и этот набор портов статические и известны не только реализации, но и тесту.

С портом среда может связывать очередь, кучу или стек. Проблему может представлять случай, когда с каждым портом связана очередь, ограниченная куча или ограниченный стек, то есть то, что может вызвать постоянную блокировку. Для того чтобы постоянная блокировка никогда не возникала, реализация должна в каждом стационарном состоянии принимать все стимулы хотя бы из одного порта.

### Внутренняя блокировка: требования к реализации.

Как правило, в реальной работе реализации с окружением через среду стремятся избежать постоянной блокировки. Для этого нужно сформулировать соответствующие требования к реализации при данном наборе портов и данной среде. Поскольку спецификация как раз и является описанием требований к реализации, то возникает вопрос, в чём заключаются эти требования и как их описывать в спецификации? Фактически, речь идёт о том, что мы должны по спецификации проверить, может ли реализация, конформная этой спецификации, привести к постоянной блокировке. Эта задача, однако, уже не относится к тестированию соответствия и представляет собой предмет аналитической верификации самой спецификации. Поэтому об этом мы говорить не будем.

С точки зрения тестирования, если внутренняя блокировка, постоянная или временная, допускается спецификацией, то это нормальная ситуация. Иными словами, при тестировании таких реализаций мы уже не можем считать, что в конце шага тестирования, когда обнаружена стационарность, реализация приняла все посланные ей стимулы.

Итак, мы ослабляем требование к реализации, *разрешая в стационарном состоянии не только приём стимула и разрушение, но и блокировку*.

Мы уже говорили, что удобнее всего понимать предусловие стимулов в стационарном состоянии как описание безопасных стимулов. Тогда нужна конструкция языка, различающая безопасный принимаемый стимул и безопасный блокируемый стимул. Мы предлагали использовать ключевое слово **block** в постусловии стимула.

### Внутренняя блокировка: отображение трасс.

При завершении шага тестирования могут остаться стимулы, которые посланы тестом, но не приняты реализацией, то есть «застрявшие» в среде. При завершении шага тестирования гипотетическая трасса уже не обязана содержать все посланные стимулы, хотя она по-прежнему должна содержать все полученные тестом реакции и заканчиваться стационарностью. Здесь возникает проблема отображения наблюдаемой трассы во множество гипотетических трасс.

Как мы можем учесть застрявшие стимулы при построении отображения трасс? Сначала рассмотрим два примера.

Пример 1. Если есть одна входная очередь, то все застрявшие стимулы располагаются в ней в том порядке, в каком мы их посылали на предыдущем шаге и, кроме того, – это последние посланные стимулы. Нам нужно только знать их количество, которое зависит от гипотетических трасс предыдущих шагов. Добавляя последовательность застрявших стимулов в начало наблюдаемой трассы текущего шага, мы получаем модифицированную наблюдаемую трассу, для которой и строим отображение. Иными словами, мы строим гипотетические трассы для модифицированной наблюдаемой трассы, подаваемой в пустую среду. Застрявшие стимулы находятся в начале модифицированной трассы, но они не будут приняты, то есть снова застрянут в среде, по крайней мере, до тех пор, пока реализация не примет какой-нибудь последующий стимул.

Пример 2. Однако уже для стека такой метод не работает. Например, в стеке застряли два стимула a и b, причём b находится в вершине стека и блокируется реализацией. Если реализация не блокирует стимул a, то любая последовательность ab или ba, подаваемая в пустую среду, вызовет приём стимула a. Таким образом, застрявшие стимулы, как бы они не располагались в начале модифицированной наблюдаемой трассы, снова не застрянут.

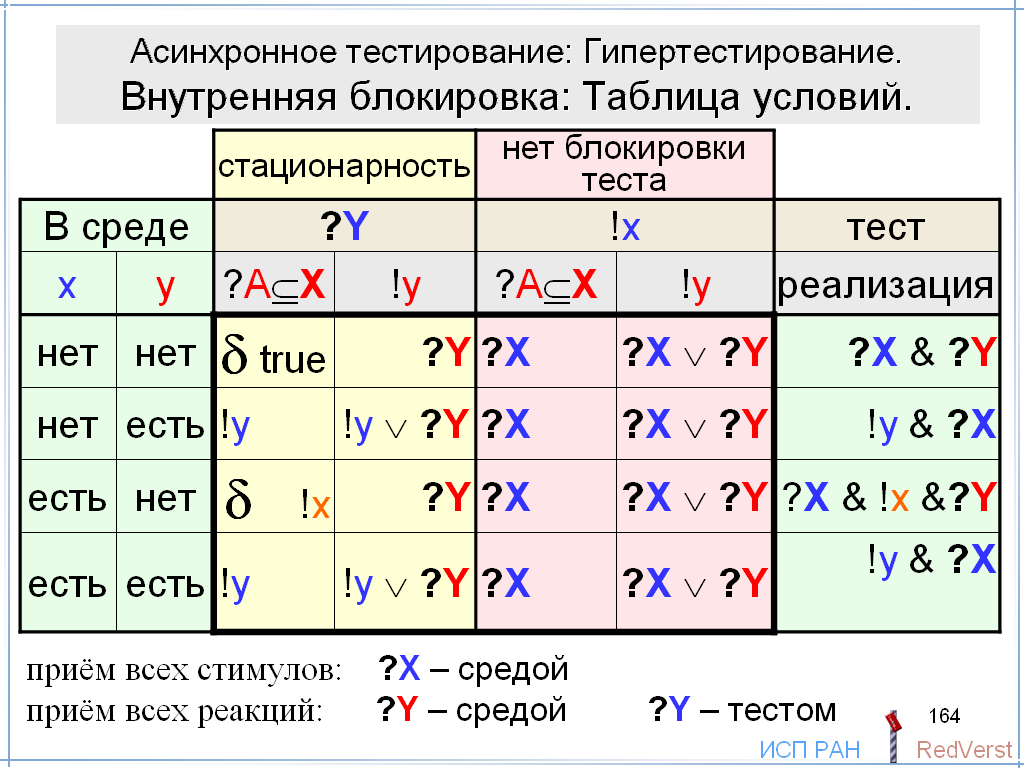
Эти примеры показывают, что проблема состоит в том, что состояние среды в начале шага тестирования может быть различным. Но тогда отображение начинает зависеть от этого состояния. Мы уже не можем задавать такое отображение для трассы, наблюдаемой между двумя стационарностями.

Общим решением является построение отображения не для наблюдаемой трассы текущего шага, а для полной трассы от начала тестирования. Верификация гипотетической трассы проводится по спецификации, начиная также от её начального состояния.

Естественно также считать, что стимулы застревают в среде по вине реализации, а не среды. Среда предлагает какие-то из застрявших стимулов реализации, но та их блокирует; остальные застрявшие стимулы просто ждут своей очереди и не предлагаются средой. Это означает, что в гипотетические трассы должны входить блокировки застрявших стимулов, предлагаемых средой, но блокируемых реализацией.

### Внутренняя блокировка: Таблица условий.

А теперь посмотрим, какая получается таблица условий для среды, когда реализация может блокировать стимулы в стационарных состояниях. Внешнюю блокировку мы пока запретим.



Теперь в стационарном состоянии реализация может принимать не все стимулы, а лишь некоторые или даже ни одного.

Поэтому в столбцах, соответствующих стационарному состоянию реализации, и строках, соответствующих наличию стимулов в среде, мы должны сделать изменения.

Теперь мы не можем требовать от среды, чтобы она предлагала реализации тот стимул, которого реализация ждёт в стационарном состоянии. Реализация может не ждать ни одного из тех стимулов, которые есть в среде.

Пусть в среде есть стимулы, но нет реакций.

Если тест ждёт реакций, то возможен deadlock, который означает стационарность реализации и отсутствие реакций в среде. При этом стимулы в среде могут оставаться. Среда должна предлагать хотя бы один стимул реализации, так как в противном случае deadlock возникнет по вине среды, а не реализации.

Если тест посылает стимул, то мы должны избежать блокировки теста. Но теперь мы уже не можем рассчитывать на то, что реализация обязательно примет предлагаемый средой стимул. Поэтому остаётся требование приёма всех стимулов от теста.

Соответственно меняется итоговый столбец условий на среду.

Пусть в среде есть и стимулы и реакции.

Если тест ждёт реакций, deadlock не должен возникать, потому что мы договорились, что в конце шага тестирования в среде не должно оставаться реакций. Поскольку мы не можем рассчитывать на то, что реализация обязательно примет предлагаемый средой стимул, остаётся требование выдачи реакции из среды в тест.

Если тест посылает стимул, то остаётся требование приёма всех стимулов от теста.

Соответственно меняется итоговый столбец условий на среду.

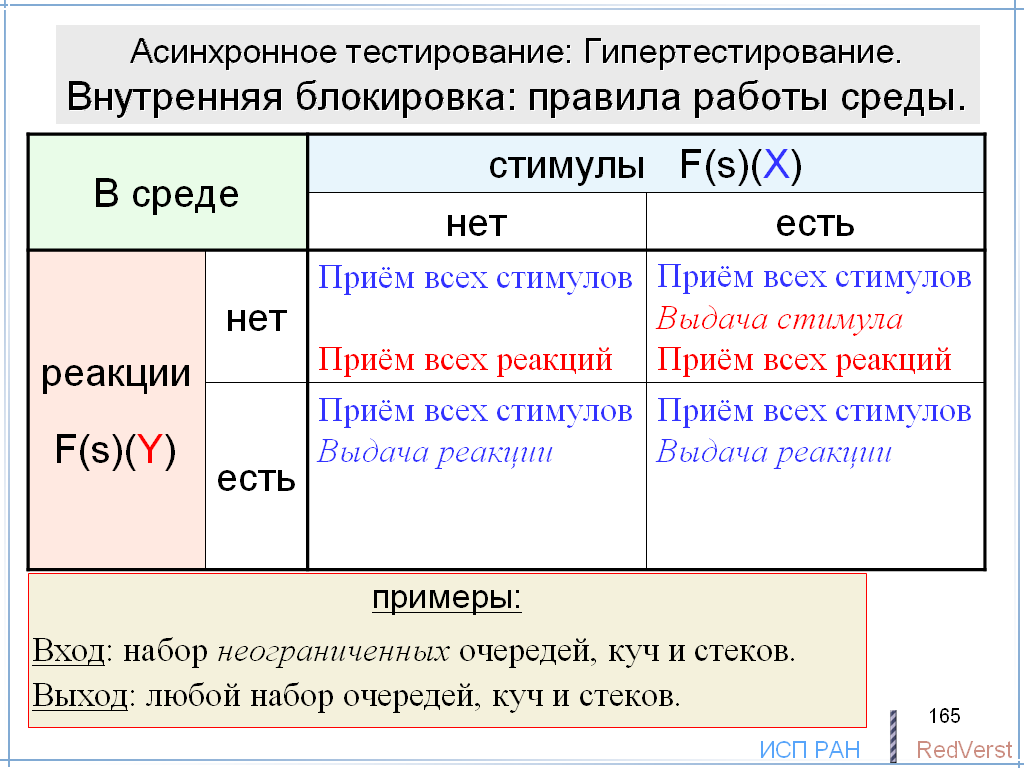
### Внутренняя блокировка: Правила работы среды.

Итоговые правила работы среды меняются.

Теперь среда должна всегда принимать все стимулы от теста.

Если в среде нет реакций, она должна ждать всех реакций от реализации. ♣Если в среде есть реакции, она должна хотя бы одну из них предлагать тесту.

Кроме того, чтобы не было deadlock`а по вине среды она должна предлагать хотя бы один стимул реализации, если в среде есть стимулы и нет реакций.



Примером может служить любой набор неограниченных входных очередей, куч и стеков. Реакции могут выдаваться реализацией через любой набор очередей, куч и стеков как неограниченной, так и ограниченной ёмкости.

### Внешняя и внутренняя блокировка.

Наконец, рассмотрим общий случай, когда возможны и внешняя и внутренняя блокировки.

Наличие внешней блокировки делает тест адаптивным: после тайм-аута в посылающем состоянии тест продолжает работать. Он может попробовать послать другой стимул или ждать реакций.

Мы ослабляем требование к реализации, разрешая в стационарном состоянии не только приём стимула и разрушение, но и блокировку.

Отображение трасс строится не для наблюдаемой трассы текущего шага, а для полной наблюдаемой трассы от начала тестирования.

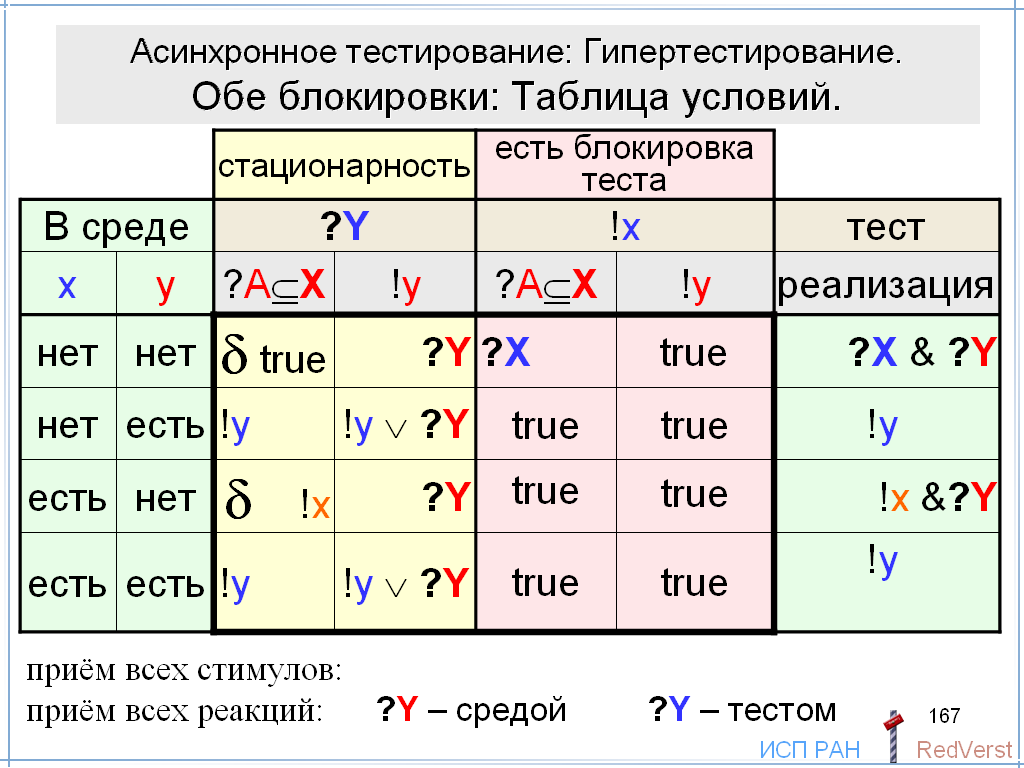
В гипотетические трассы должны входить блокировки застрявших стимулов, предлагаемых средой, но блокируемых реализацией.

### Внешняя и внутренняя блокировка: Таблица условий.

Таблица условий совмещает особенности таблиц для внешней и внутренней блокировки.

Возможность внутренней блокировки влечёт возможность обнаружения стационарности при отсутствии реакций в среде. Но стимулы могут «застрять» в среде. Если тест ожидает реакций, а в среде есть и стимулы и реакции, то среда не обязана предлагать стимул реализации, поскольку реализация всё равно может такие стимулы блокировать. Важно лишь, чтобы она предлагала реакцию тесту.

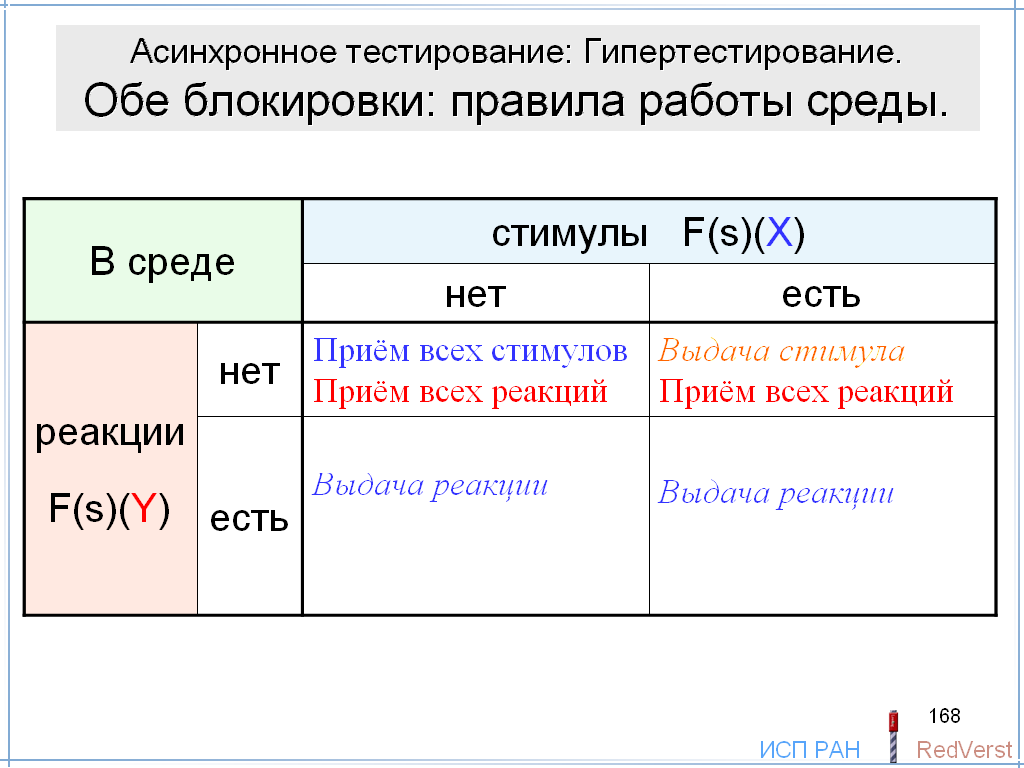
Если тест посылает стимул, то поведение среды любое, кроме случая отсутствия в ней стимулов и реакций. В этом случае среда должна принимать стимул от теста.



Соответственно меняется итоговый столбец условий.

### Внешняя и внутренняя блокировка: Правила работы среды.

Если реакции в среде нет, среда должна принимать все реакции от реализации. Если в среде есть реакции, среда обязана хотя бы одну из них предлагать тесту.



Что касается стимулов, то поведение среды ограничивается только для случая отсутствия в ней реакций. Если в среде нет стимулов, она обязана принимать все стимулы от теста. Если в среде есть стимулы, она обязана предлагать хотя бы один из них реализации.

Чем отличаются эти правила от случая, когда есть только внешняя блокировка, а внутренней блокировки нет? Только поведением среды при наличии в ней как стимулов, так и реакций.

Если внутренней блокировки нет, то мы разрешали среде на выбор предлагать реакцию тесту или принимать все реакции от реализации и предлагать стимул реализации. Если реализация находится в нестационарном состоянии, она пошлёт реакцию и среда её примет. Если же реализация находится в стационарном состоянии, она обязана принять стимул, предлагаемый средой. В любом случае взаимодействие возможно.

При наличии внутренней блокировки реализация уже не обязана принимать стимул, предлагаемый ей средой. Поэтому у среды такого выбора уже нет. Она обязана выдавать реакцию в тест. Разумеется, это не означает, что вместе с этим (а не на выбор) она не может предлагать стимулы реализации и/или принимать от неё реакции.

### ПРОБЛЕМЫ: Осциллирующая реализация. Проблема дивергенции.

Когда мы начинали говорить об асинхронном тестировании, мы наложили запрет на осцилляцию реализации по двум причинам: из-за дивергенции и из-за незавершённости шага тестирования.

Сначала рассмотрим проблему дивергенции. Для того чтобы дивергенция не возникала, среда должна периодически выдавать накопленные в ней реакции в тест, не принимая новых реакций от реализации.

Такой режим работы всегда обеспечивается средой, в которой ограничено число хранимых реакций. Например, ограниченные выходные очереди, стеки и кучи.

При неограниченной ёмкости такое поведение среды также возможно. Например, среда принимает от реализации не более n реакций подряд, то есть без выдачи реакций в тест. Если принято подряд n реакций, среда оказывается в состоянии, где только выдаёт реакцию в тест, но не принимает реакции от реализации. В остальных состояниях она может принимать реакции от реализации. Здесь число реакций, которые среда может принять из реализации без выдачи реакций в тест, всегда конечно и ограничено сверху числом n, но число реакций, накапливаемых в среде, может стать сколь угодно большим.

Другим решением было бы установление приоритета выдачи реакций из среды в тест над приёмом реакции из реализации в среду. В этом случае даже при обычной неограниченной выходной очереди тест получал бы реакцию сразу, если он задаёт приём реакций и реакция уже есть в очереди, или, если в очереди нет реакций, как только такая реакция поступит из реализации.

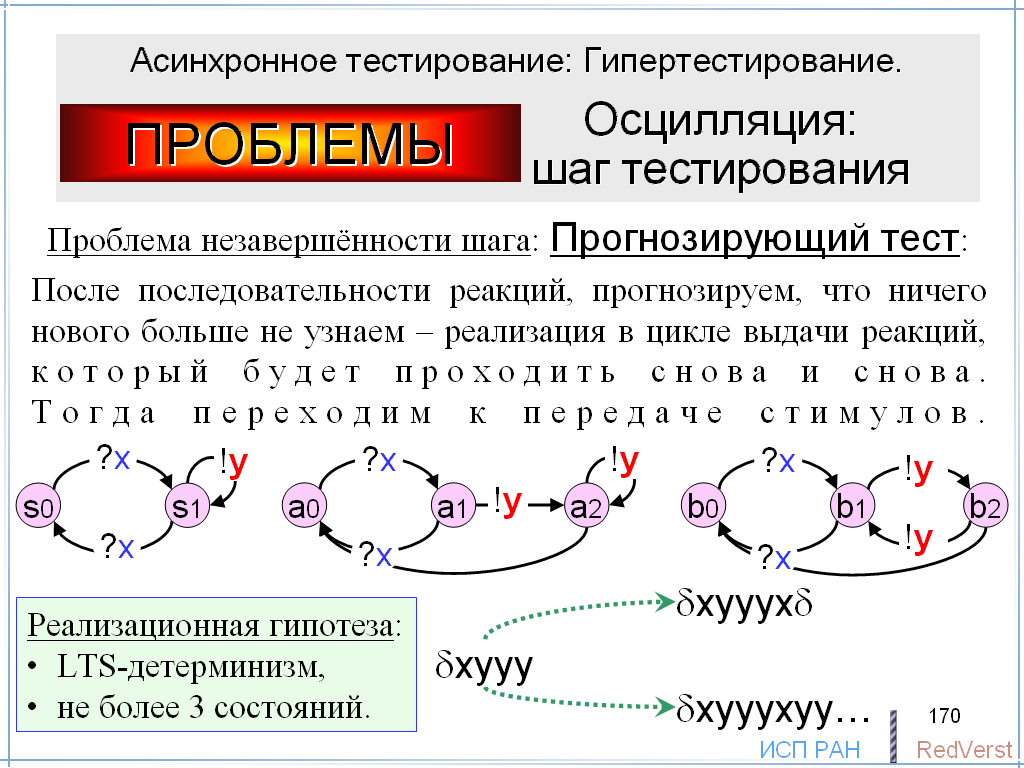
Фактически, речь идёт об изменении работы мистического синхронизатора, реализующего оператор параллельной композиции автоматов. Сейчас при наличии вариантов он выбирает любой из них полностью недетерминированно. Проблема в том, что такое решение выходит за рамки обычной модели асинхронного автомата и параллельной композиции автоматов. Фактически, нам нужно описывать такие приоритеты в спецификации или, по крайней мере, иметь соответствующие умолчания. Оператор параллельной композиции должен эти приоритеты учитывать. Об этом мы поговорим позже.

### ПРОБЛЕМЫ: Осциллирующая реализация. Проблема незавершённости шага тестирования.

Предположим теперь, что проблема дивергенции тем или иным способом решена, то есть при тестировании композиция реализации и среды не впадает в дивергенцию. Тогда остаётся проблема незавершённости шага тестирования. Тест может непрерывно принимать реакции из среды и никогда не обнаружит стационарность просто потому, что в реализации есть цикл по выдаче реакций.

В настоящее время есть только идея решения этой проблемы, которую можно назвать идеей *прогнозирующего теста*. Эту идею высказал Виктор Кулямин. Получив некоторую последовательность реакций, мы делаем прогноз, что ничего нового мы уже не узнаем: реализация прошла некоторый цикл выдачи реакций и дальше будет только повторять его снова и снова. В этом случае тест переключается на выдачу стимула, пытаясь таким образом столкнуть реализацию с этого цикла. Такой прогноз мы можем сделать только на основании спецификации, в которой есть такой цикл, и реализационной гипотезе, ограничивающей недетерминизм реализации и число её состояний.

Объясним эту идею на следующем простом примере. Пусть в спецификации только два состояния и три перехода. В начальном состоянии спецификация принимает один стимул и переходит во второе состояние, в котором выдаёт в цикле-петле одну реакцию и принимает стимул, возвращаясь в начальное состояние: s0xs1ys1xs0.



Предположим, что реализация детерминирована в смысле детерминизма LTS: каждое её состояние стабильно и в нём есть не более одного перехода по каждому стимулу и каждой реакции. Предположим также, что в реализации не более трёх состояний.

Тогда, обнаружив стационарность в начальном состоянии, послав один стимул и получив три одинаковые реакции, то есть пройдя трассу δxyyy, мы можем заключить, что никаких других реакций мы больше не получим и реализация не перейдёт в стационарное состояние.

Здесь возможны три реализации, конформные спецификации, если не учитывать возможность приёма стимула не в начальном состоянии:

1. s0xs1ys1,
2. a0xa1ya2ya2,
3. b0xb1yb2yb1.

После такого прогноза мы перестаём принимать реакции, и посылаем в реализацию стимул, пытаясь столкнуть реализацию с цикла выдачи реакций. Если это удалось, мы должны обнаружить стационарность в принимающем состоянии теста. Тем самым, мы проверим трассу δxyyxδ.

Разумеется, реализация не обязана сходить с цикла выдачи реакций и принимать стимул. Вместо трассы δxyyxδ мы можем получить трассу δxyyxууу....

Для того чтобы гарантированно стимулом столкнуть реализацию с цикла выдачи реакций, нам опять нужны приоритеты. На этот раз в реализации следует установить приоритет приёма стимулов над выдачей реакций.

В общем, для тестирования осциллирующих реализаций нам нужно, во-первых, уметь сталкивать реализацию с цикла выдачи реакций с помощью приоритетов переходов в автомате или каким-то ещё способом ограничения недетерминированного поведения и, во-вторых, разработать алгоритм генерации прогнозирующих тестов по спецификации, основанный на соответствующей реализационной гипотезе.