И. Бурдонов, А. Косачев
Тестирование соответствия (conformance testing)
ИСПРАН
239 слайдов

ПОСТАНОВКА ПРОБЛЕМЫ

Основная проблема композиционного тестирования. 3
Монотонность. 4
Косая композиция. 5
Компонуемость спецификаций. 6
Формальные определения. 6
Преобразование спецификаций. 7
Цели косой композиции. 7

БЛОКИРОВКА И РАЗРУШЕНИЕ

Пример немонотонности из-за блокировки. 9
Спецификации без блокировок. 9
Пример немонотонности из-за разрушения. 9
Гамма-нормализованные спецификации без блокировок в безопасных трассах. 10

ОБЩИЙ СЛУЧАЙ

Достаточные условия монотонности. 12
Условия монотонности 1 и 2: Гамма-расширение $\Gamma: \wp(\Psi) \to \wp(\Psi)$. 14
Разрушение. 15
ϕ-трассы $\phi: A A \to \wp(\Phi)$. 15
Условие монотонности 3: ϕ-трассы и βγδ-трассы: $\psi = \cup \circ \pi \circ \phi$. 16
Условие монотонности 4: Аддитивность ϕ-трасс: $\phi(A \| B) = \cup(\phi(A) \| \phi(B))$. 17
Условие 5: Существование максимальной реализации. 18

БАЗОВЫЕ И НОРМАЛЬНЫЕ ϕ-ТРАССЫ

Базовые ϕ-трассы. 19
Финальные ϕ-трассы. 20
Релевантные ϕ-трассы. 21
Сингулярные ϕ-трассы. 22
Нормальные ϕ-трассы. 25

АЛГОРИТМ $\mathcal{A}$(S)

Состояния $\mathcal{A}$(S) – это нормальные базовые ϕ-трассы максимальной реализации. 26
Нормальные ϕ-трассы $\mathcal{A}$(S) – это нормальные базовые ϕ-трассы максимальной реализации. 27
Ограничения на спецификацию. 28
Максимальная βγδ-трасса. 28
Бесконечное ветвление. 29
τ-замкнутое множество состояний спецификации. 29
Состояния автомата $\mathcal{A}$(S). 30
Формальные правила вывода. 32
Спецификация безопасных стимулов. 32

КОСАЯ КОМПОЗИЦИЯ

Цели построения *косой* композиции. 33
Ограничение на спецификации компонентов. 33
Итеративное построение. 34
Безопасность и безопасно-конвергентность. 35
Корректность системной спецификации. 35
Генерация системных тестов по косой композиции. 36

Основная проблема композиционного тестирования.

Композиционное тестирование – это тестирование системы, собранной из компонентов по известной схеме компоновки с помощью применения определённых правил композиции.

Асинхронное тестирование является частным случаем композиционного тестирования. Здесь система состоит из двух компонентов: реализация и среда. При этом предполагается, что модель среды известна.

Основная проблема композиционных систем звучит так: если компоненты работают правильно, то почему система в целом работает неправильно?

В тестировании соответствия правильность реализации компонента определяется как её соответствие спецификации компонента, а правильность системы – как ее соответствие спецификации системы. Разработчик компонента руководствуется единственным документом – техническим заданием, формальной моделью которого является спецификация компонента. Правильность реализации компонента проверяется автономным тестированием компонента. Очевидно, одной из причин неправильной работы системы является то, что, на самом деле, ее компоненты работают неправильно: при автономном тестировании они не были полностью проверены. С учетом бесконечности полного набора тестов такая ситуация вполне возможна и действительно часто встречается на практике.

Однако проблема композиционных систем этим не исчерпывается. Может оказаться, что реализации всех компонентов соответствуют своим спецификациям, в то время как собранная из этих компонентов система не соответствует спецификации системы в целом. В силу вышесказанного претензий к разработчикам компонентов быть не может.

Тогда кто же виноват, и что делать?

Очевидно, проблема лежит в иной плоскости: само соотношение спецификаций компонентов и спецификации системы неправильно. А это уже ошибка архитектора, который неправильно декомпозировал спецификацию системы на спецификации ее компонентов. По вполне понятным причинам такие ошибки гораздо хуже ошибок разработчиков, поскольку их труднее обнаруживать, и они имеют более печальные последствия.

С этой точки зрения, более сложное системное или комплексное тестирование не просто продолжает более простое, но незаконченное, автономное тестирование, но предназначено также для обнаружения ошибок архитектурного уровня, которые не обнаруживаются автономными тестами. Последствиями таких ошибок является изменение спецификаций, иногда достаточно радикальное, что требует модификации или даже повторной реализации всех или части компонентов.

Монотонность.

Какое соотношение спецификаций компонентов и спецификации системы следует признать правильным? Очевидно, такое, которое удовлетворяет следующему условию монотонности: любые реализации компонентов, соответствующие своим спецификациям, образуют систему, соответствующую спецификации системы.

Замечу, что отсюда вовсе не следует, что, если реализации компонентов образуют систему, соответствующую спецификации системы, то сами эти реализации компонентов соответствуют спецификациям компонентов. Мы этого не требуем, да нас это и не должно волновать: важно лишь, чтобы система отвечала функциональным требованиям, зафиксированным в системной спецификации, а как она это делает – не имеет значения. Это общая ситуация, частным случаем которой является то, что мы называли вседозволенностью асинхронного тестирования.

Корректной спецификацией системы можно назвать такую спецификацию системы, которая удовлетворяет условию монотонности при заданных спецификациях компонентов.

Самая сильная (то есть, предъявляющая максимальные функциональные требования) корректная спецификация системы определяется самим условием монотонности при заданных спецификациях компонентов и схеме компоновки.

Если задана спецификация системы, то верификация её согласованности со спецификациями компонентов заключается в проверке того, что она корректна, то есть, эквивалентна или слабее самой сильной корректной спецификации. Отношение определяется соответствием $\boldsymbol{ioco}_{\beta\gamma\delta}$: самая сильная корректная спецификация системы должна быть конформна заданной спецификации системы.

Если у нас вообще нет спецификации системы, то мы могли бы её построить как самую сильную корректную спецификацию.

Здесь, однако, возникают три проблемы:
1) Определение самой сильной корректной спецификации неявно.
2) Существует ли такая самая сильная корректная спецификация?
3) Если она существует, то как её построить по спецификациям компонентов и схеме компоновки?

Косая композиция.

Спецификационные модели компонентов – это безопасные и безопасно-конвергентные асинхронные автоматы без θ-переходов. Реализации компонентов – это асинхронные автоматы, конформные соответствующим спецификационным моделям. Правила композиции реализаций компонентов моделируются оператором $\upharpoonleft\downharpoonright$ параллельной композиции.

Первой неожиданностью стало то, что самая сильная корректная спецификация системы, если она существует, может оказаться слабее композиции спецификаций компонентов, если эту композицию проводить по тем же правилам, что и композицию реализаций компонентов при сборке системы.

Несохранение соответствия при асинхронном тестировании можно рассматривать как частный случай этой проблемы. Здесь система состоит из двух компонентов: реализация и среда, которые компонуются друг с другом.

Если композицию с помощью оператора $\upharpoonleft\downharpoonright$ называть реализационной, то оказалось, что требуется иная – *спецификационная* – композиция спецификаций, вычисляющая самую сильную корректную спецификацию системы, если она существует, как функцию спецификаций двух компонентов. Спецификационную композицию будем обозначать косым знаком $/\!/$. Для удобства терминологии такую композицию будем называть *косой композицией* асинхронных автоматов.

Причиной такого положения, то есть несовпадения прямой и косой композиции, является разный уровень абстракции, используемый в определении соответствия и в определении прямой композиции. Соответствия основаны на наблюдаемых трассах – последовательностях наблюдаемых действий реализационной модели, а композиция – на полной модели реализации. Отношение $\boldsymbol{ioco}_{\beta\gamma\delta}$ основано на следующих наблюдаемых действиях: прием или блокировка стимула, выдача реакции или отсутствие выдачи какой-либо реакции (стационарность). Прямая композиция дополнительно учитывает состояния, ненаблюдаемые действия (τ-переходы) и соотношение состояний и действий.

Компонуемость спецификаций.

Частный случай общей проблемы композиции – это то, что прямая композиция конформных реализаций может оказаться не безопасно-тестируемой относительно косой композиции спецификаций компонентов.

Это может быть в том случае, когда косая композиция спецификаций не безопасна или не безопасно-конвергентна. Поэтому после построения косой композиции её нужно верифицировать на безопасность и безопасно-конвергентность.

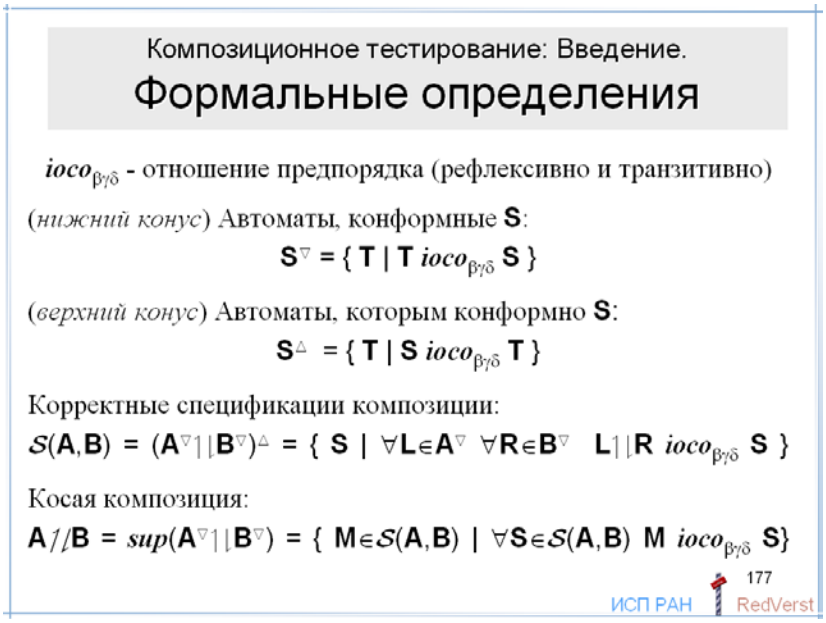
Спецификации будем называть *компонуемыми*, если их косая композиция безопасна и безопасно-конвергентна.

Формальные определения.

Дадим формальные определения корректной спецификации композиции и косой композиции.

Соответствие $\boldsymbol{ioco}_{\beta\gamma\delta}$ является отношением предпорядка: рефлексивно и транзитивно. Поэтому для него можно определить нижний и верхний конусы.

Нижний конус автомата S – это множество автоматов, конформных автомату S. Верхний конус автомата S – это, наоборот, множество автоматов, которым конформен автомат S.



Корректная спецификация композиции – это автомат из верхнего конуса прямой композиции нижних конусов спецификаций: такой автомат, которому конформна композиция любых реализаций, конформных данным спецификациям.

Косая композиция – как самая сильная корректная спецификация – это супренум прямой композиции нижних конусов спецификаций: такая корректная спецификация композиции, которой конформна любая другая корректная спецификация композиции.

Заметим, что косая композиция спецификаций неоднозначно определяет асинхронный автомат. По сути, мы только наложили ограничения на такой автомат. Тем самым, косую композицию можно понимать как множество автоматов, удовлетворяющих этому ограничению. Наша задача – суметь построить по заданным спецификациям компонентов хотя бы один такой автомат.

Преобразование спецификаций.

Основная идея построения косой композиции заключается в следующем. Мы хотим найти такое преобразование $\mathcal{F}$ спецификаций компонентов, чтобы косая композиция исходных спецификаций была равна прямой композиции преобразованных спецификаций:

$$\mathbf{A} /\!/ \mathbf{B} = \mathcal{F}(\mathbf{A}) \,|\!|\, \mathcal{F}(\mathbf{B}).$$

В этом случае будем говорить, что соответствие *монотонно относительно преобразования* $\mathcal{F}$.

Вообще говоря, таких преобразований $\mathcal{F}$ может быть много. Существование хотя бы одного преобразования показывается легко. Для этого достаточно взять объединение всех реализаций, конформных данной спецификации:

$$\mathcal{M}(\mathbf{S}) = \cup(\mathbf{S}^{\triangledown}).$$

Объединение автоматов строится так. Добавляется новое начальное состояние, и из него проводятся τ-переходы в начальные состояния объединяемых автоматов.

Понятно, что такое преобразование неконструктивно: число конформных реализаций бесконечно. Поэтому ставится задача найти такое другое преобразование, которое можно было бы осуществлять алгоритмическим способом.

Построение может быть итеративным, что важно для бесконечных спецификаций. Итеративность будем понимать так, что для любого, наперёд заданного, числа n мы можем построить часть преобразованной спецификации, которая содержала бы все безопасные трассы длиной n.

Цели косой композиции.

Итак, построение косой композиции преследует три цели.

1. Проверка компонуемости, то есть проверка того, что косая композиция спецификаций компонентов безопасна и безопасно-конвергентна. Это гарантирует, что композиция любых конформных реализаций будет безопасно-тестируемой для любой корректной спецификации системы.
2. Проверка корректности системной спецификации, то есть проверка того, что косая композиция спецификаций компонентов конформна системной спецификации.
3. Генерация системных тестов по косой композиции спецификаций компонентов. Это полезно, когда у нас нет корректной системной спецификации, или эта спецификация недостаточно сильная для проверки интересующих нас свойств системы.

Сейчас мы более детально исследуем причины немонотонности соответствия. Таких причин две: блокировки стимулов и разрушение. Мы также рассмотрим спецификации, на которые наложены ограничения по блокировке и разрушению. Для таких спецификаций можно определить очень простые преобразования, относительно которых соответствие монотонно.

Пример немонотонности из-за блокировки.

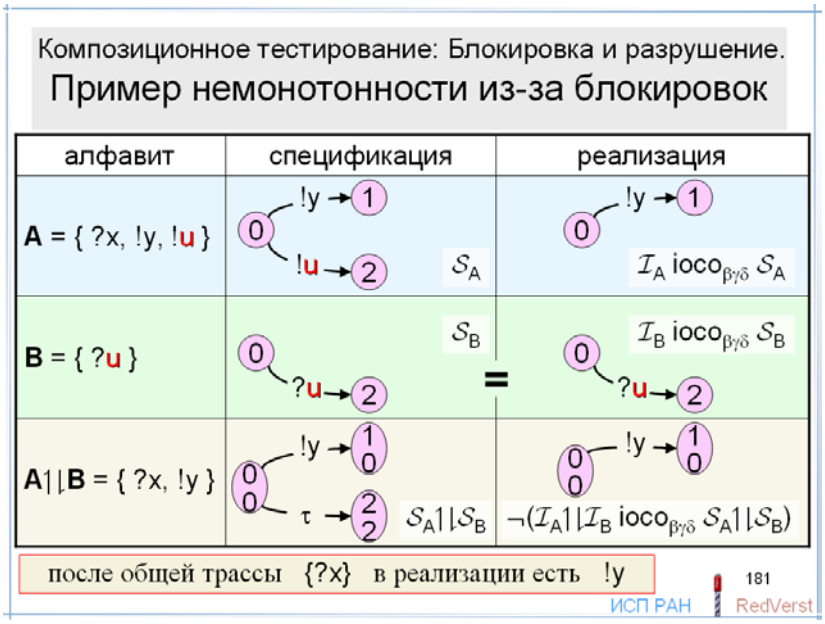
Сначала покажем, что, если в спецификации есть блокировки стимулов, то соответствие $\boldsymbol{ioco}_{\beta\gamma\delta}$ немонотонно относительно тождественного преобразования спецификаций. Иными словами, при наличии блокировок стимулов прямая композиция не является косой композицией.

Для этого достаточно рассмотреть пример на рисунке.

В первой строке показаны спецификация и конформная ей реализация.

Во второй строке спецификация и реализация совпадают.

В третьей строке показаны композиции спецификаций и реализаций.



Мы видим, что трасса {?x} является общей для прямой композиции спецификаций и прямой композиции реализаций.
Однако в композиции реализаций эта трасса продолжается реакцией !y, а в композиции спецификаций – не продолжается.

Спецификации без блокировок.

Если в спецификации нет блокировок, то соответствие $\boldsymbol{ioco}_{\beta\gamma\delta}$ можно называть соответствием $\boldsymbol{ioco}_{\gamma\delta}$. Для таких спецификаций оно монотонно относительно тождественного преобразования: $\mathcal{F}(\mathbf{S})=\mathbf{S}$. Иными словами, прямая композиция спецификаций без блокировок является также и косой композицией.

Именно поэтому при пополнении спецификаций стремятся избавиться от блокировок. В частности, пополнение спецификации, основанное на правиле приоритета стимула над его блокировкой и правиле сохранения поведения после стационарности и, после этого, демоническом пополнении, сохраняет классическую семантику отношения ***ioco*** и обеспечивает монотонность соответствия для пополненных спецификаций без дальнейших преобразований.

Пример немонотонности из-за разрушения.

При наличии разрушения довольно странно требовать отсутствия блокировок после приёма разрушающего стимула. В этом случае поведение автомата считается неопределённым с точки зрения соответствия ***ioco***$_{\beta\gamma\delta}$.

Поэтому мы ослабим требование отсутствия блокировок: потребуем, чтобы блокировок не было в безопасных βγδ-трассах.

Это, однако, уже не гарантирует монотонность соответствия при тождественном преобразовании спецификаций.

Рассмотрим пример на рисунке.

В первой строке показаны спецификация и конформная ей реализация.

Во второй строке спецификация и реализация совпадают.

В третьей строке показаны композиции спецификаций и реализаций.

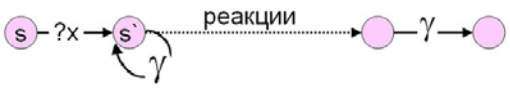
Мы видим здесь, что трасса ?x является общей для прямой композиции спецификаций и прямой композиции реализаций. Однако в композиции реализаций эта трасса продолжается тем же стимулом ?x, а композиция спецификаций – только блокировкой {?x}.

Если внимательно изучить этот пример, то можно увидеть, что немонотонность возникает из-за того, что в спецификации разрушение идёт не сразу после разрушающего стимула, а после некоторых реакций.

Гамма-нормализованные спецификации без блокировок в безопасных трассах.

Будем называть спецификацию *гамма-нормальной*, если в ней после перехода по разрушающему стимулу сразу идёт переход по разрушению.

Гамма-нормализацией назовём соответствующее преобразование $\mathcal{N}$ спецификаций. Это преобразование добавляет гамма-петлю в конце перехода по разрушающему стимулу. Такое состояние становится гамма-состоянием.



Прежде всего, отметим, что гамма-нормализованная спецификация ***ioco***$_{\beta\gamma\delta}$-эквивалентна исходной спецификации.

Если спецификации гамма-нормализованы и в их безопасных трассах нет блокировок, то соответствие $\boldsymbol{ioco}_{\beta\gamma\delta}$ также монотонно относительно тождественного преобразования спецификаций. Тем самым, для спецификаций, не содержащих блокировок в безопасных трассах, соответствие монотонно относительно гамма-нормализации: $\mathcal{F}(\mathbf{S})=\mathcal{N}(\mathbf{S})$.

Теперь при пополнении спецификаций после преобразований по правилам приоритета и сохранения мы применяем не демоническое пополнение, а пополнение разрушения. Такое преобразование также сохраняет классическую семантику отношения ***ioco***, но для более широкого класса спецификаций, поскольку в небезопасных трассах допускаются блокировка и дивергенция. Поскольку полученная спецификация гамма-нормальна, обеспечивается монотонность соответствия для пополненных спецификаций без дальнейших преобразований.

Теперь мы перейдём к общему случаю, когда на спецификации не накладывается никаких ограничений, кроме безопасности и безопасно-конвергентности.

Достаточные условия монотонности.

Введём обозначения:

Ψ – *множество всех βγδ-трасс;*

ψ: **AA**→ ℘(**Ψ**) – *отображение, которое каждому асинхронному автомату ставит в соответствие множество его βγδ-трасс;*

Φ – *множество трасс, которые мы будем называть ϕ-трассами; определение ϕ-трасс мы дадим позже, а здесь сформулируем только требования к ним;*

ϕ: **AA**→ ℘(**Φ**) – *отображение, которое каждому асинхронному автомату ставит в соответствие множество его ϕ-трасс;*

π: **Φ**→ ℘(**Ψ**) – *отображение, котороре преобразует ϕ-трассу во множество βγδ-трасс;*

Общий случай

Γ: $\wp(\Psi) \to \wp(\Psi)$ *– отображение, преобразующее множество βγδ-трасс в другое, расширенное, множество βγδ-трасс; мы будем называть его гамма-расширением δγβ-трасс; определение гамма-расширения мы дадим позже, а здесь сформулируем только требования к нему;*

$\downarrow\phi\downarrow$: $\Phi\times\Phi \to \wp(\Phi)$ *– отображение, которое по паре ϕ-трасс строит множество ϕ-трасс; мы будем называть его композицией ϕ-трасс; определение композиции ϕ-трасс мы дадим позже, а здесь сформулируем только требования к нему;*

$\mathcal{M}_\phi(\mathbf{S})$ *– максимальная ϕ-реализация для спецификации* ***S****: такой автомат, множество ϕ-трасс которого равно объединению множеств ϕ-трасс всех конформных реализаций:* $\phi\circ\mathcal{M}_\phi(\mathbf{S}) = \cup\circ\phi(\mathbf{S}^\triangledown)$.

Условия, которые нам нужны, следующие:

1. Эквивалентность соответствия вложенности гамма-расширений βγδ-трасс:
   $\mathbf{I}\ \mathit{ioco}_{\beta\gamma\delta}\ \mathbf{S} \Leftrightarrow \Gamma\circ\psi(\mathbf{I}) \subseteq \Gamma\circ\psi(\mathbf{S})$.
2. Свойства гамма-расширения:
   $\mathbf{A}\subseteq\Gamma(\mathbf{A})\ \&\ \Gamma\circ\Gamma=\Gamma\ \&\ (\ \mathbf{A}\subseteq\mathbf{B} \Rightarrow \Gamma(\mathbf{A})\subseteq\Gamma(\mathbf{B})\ )$.
   Левое свойство: гамма-расширение действительно является расширением, то есть, может долько добавлять βγδ-трассы, но не удалять их.
   Среднее свойство: гамма-расширение добавляет все нужные βγδ-трассы, то есть, его повторное применение ничего не меняет.
   Правое свойство: гамма-расширение сохраняет вложнность множеств βγδ-трасс.
3. Производность βγδ-трасс от ϕ-трасс: $\psi = \cup\circ\pi\circ\varphi$.
   βγδ-трассы автомата могут быть получены из его ϕ-трасс с помощью оператора π.
4. Аддитивность ϕ-трасс относительно композиции:
   $\phi(\mathbf{A}|\downarrow\mathbf{B}) = \cup(\phi(\mathbf{A})\downarrow\varphi\downarrow\phi(\mathbf{B}))$.
   ϕ-трассы композиции автоматов могут быть получены как объединение всех попарных композиций ϕ-трасс этих автоматов.
5. Существует максимальная ϕ-реализация: $\exists\mathcal{M}_\phi(\mathbf{S})$.
   То есть для каждой спецификации существует автомат, множество ϕ-трасс которого равно объединению множеств ϕ-трасс всех конформных реализаций.

Если эти условия выполнены, то в качестве преобразования спецификации можно взять преобразование $\mathcal{M}_\phi$, то есть соответствие $\mathit{ioco}_{\beta\gamma\delta}$ $\mathcal{M}_\phi$-монотонно. Отметим, что преобразование $\mathcal{M}_\phi$ так же, как преобразование $\mathcal{M}$, неоднозначно, то есть максимальных ϕ-реализаций может быть много. Нам нужно суметь построить хотя бы одну из них.

Условия монотонности 1 и 2: Гамма-расширение $\Gamma: \wp(\Psi) \to \wp(\Psi)$.

Определим гамма-расширение спецификации – это такое преобразование βγδ-трасс, при котором соответствие $\boldsymbol{ioco}_{\beta\gamma\delta}$ было бы эквивалентно вложенности гамма-расширенного множества βγδ-трасс реализации в гамма-расширенное множество βγδ-трасс спецификации.

Неявно гамма-расширение можно определить как объединение всех βγδ-трасс всех конформных реализаций.

Явное определение описывает способ построения βγδ-трасс гамма-расширения из βγδ-трасс исходного автомата. Интуиция подсказывает, что гамма-расширение – это добавление трасс, порождаемых трассами с разрушением. Если в спецификации есть стимул, разрушающий после трассы, то отношение $\boldsymbol{ioco}_{\beta\gamma\delta}$ не накладывает никаких ограничений на реализацию по поводу этого стимула: реализация может его принимать или блокировать с любым дальнейшим поведением, если такое дальнейшее поведение не индуцирует запрещённое поведение в безопасных трассах.

Мы рассмотрим два случая: приём и блокировка разрушающего стимула.

Приём. Приём разрушающего стимула может иметь в качестве продолжения любую возможную βγδ-трассу.

Блокировка. Если реализация блокирует стимул x после трассы μ, то в ней после трассы $\mu\cdot\langle\{x\}\rangle$ может быть, вообще говоря, не любая βγδ-трасса. Действительно, наличие трассы $\mu\cdot\langle\{x\}\rangle\cdot\lambda$ влечёт наличие трассы $\mu\cdot\lambda$, а на такую трассу конформность реализации может накладывать ограничения, если эта трасса безопасна в спецификации. Сформулируем условия, при которых гамма-расширение спецификации может для безопасной трассы $\mu\cdot\lambda$ вставить между μ и λ блокировку стимула $\{x\}$, разрушающего после μ, не нарушая конформности (сохраняя требования к реализации).

Первое условие: в трассе λ первый символ, отличный от блокировки или стационарности, не совпадает с x. Это следует из того, что после блокировки стимула не может следовать этот стимул.

Второе условие: вставку можно делать, если между трассами μ и λ до гамма-расширения существовали отказы, или трасса μ до гамма-расширения продолжалась всеми стимулами и хотя бы одной реакцией. Дело в том, что блокировка стимула возможна только в стабильном состоянии, и это условие означает, что между трассами μ и λ в конформной реализации может быть стабильное состояние.

Действительно, если второе условие нарушено, то до гамма-расширения трасса μ либо a) не продолжалась некоторым стимулом $x`$, либо b) не продолжалась никакой реакцией. Поскольку стимул, разрушающий после трассы, продолжает эту трассу, стимул $x`$ неразрушающий после трассы μ. Поэтому для сохранения конформности свойство a) или b) должно остаться и после гамма-расширения. С другой стороны, после гамма-расширения этим свойством должна обладать также трасса $\mu\cdot\langle\{x\}\rangle$, так как любое

продолжение после этой трассы есть и после трассы μ. Следовательно, поскольку после гамма-расширения существует трасса $\mu\cdot\langle\{x\}\rangle\cdot\lambda$, должна существовать либо a) трасса $\mu\cdot\langle\{x\}\rangle\cdot\langle\{x`\}\rangle\cdot\lambda$, либо b) трасса $\mu\cdot\langle\{x\}\rangle\cdot\langle\delta\rangle\cdot\lambda$. Но тогда после гамма-расширения существует либо a) трасса $\mu\cdot\langle\{x`\}\rangle\cdot\lambda$, либо b) трасса $\mu\cdot\langle\delta\rangle\cdot\lambda$. Поскольку второе условие не выполнено, между трассами μ и λ до гамма-расширения не существовали отказы. Следовательно, трасс $\mu\cdot\langle\{x`\}\rangle\cdot\lambda$ и $\mu\cdot\langle\delta\rangle\cdot\lambda$ до гамма-расширения не было, поэтому они должны появиться в результате гамма-расширения. Эти новые трассы не нарушают конформность только в том случае, когда они опасны. Однако, поскольку трасса $\mu\cdot\lambda$ безопасна, трасса $\mu\cdot\langle\delta\rangle\cdot\lambda$ также безопасна. Аналогично, трасса $\mu\cdot\langle\{x`\}\rangle\cdot\lambda$ опасна только, если стимул $x`$ разрушающий после трассы μ, что неверно.

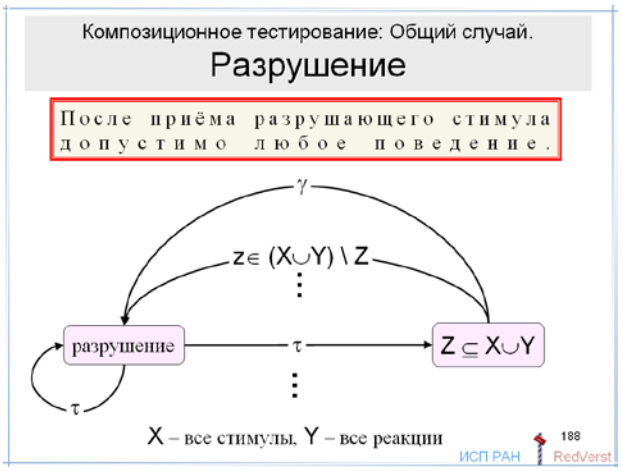
Разрушение.

Посмотрим, как можно реализовать произвольное поведение после приёма разрушающего стимула.

Состояние в конце перехода по разрушающему стимулу назовём состоянием «разрушение».

Определим в нём τ-переход в каждое стабильное состояние, определяемое подмножеством Z стимулов и реакций, по которым нет перехода из этого стабильного состояния.

Прежде всего, в этом стабильном состоянии определим гамма-переход.



Во-вторых, мы должны в этом стабильном состоянии определить переход по каждому символу, не входящему в Z. Тем самым, мы определим все возможные βγδ-трассы.

Наконец, поскольку произвольное поведение допускает дивергенцию после любой βγδ-трассы, определим в состоянии «разрушение» τ-петлю.

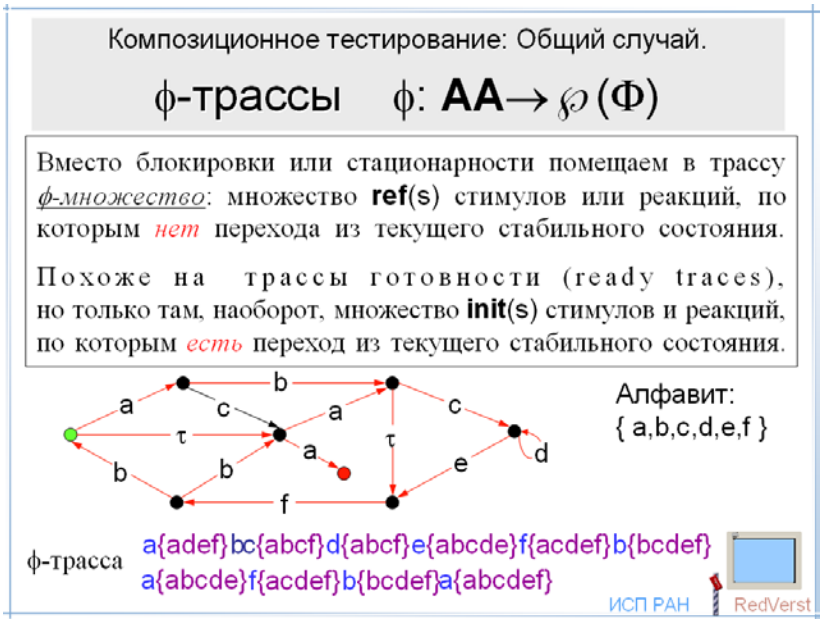
ϕ-трассы $\phi:AA\to\wp(\Phi)$.

Множества βγδ-трасс компонентов неоднозначно определяют множество βγδ-трасс композиции. Теперь мы рассмотрим разновидность трасс, для которых такая однозначность есть. Мы будем называть их ϕ-трассами.

При построении βγδ-трасс проход через стабильное состояние добавляет в трассу блокировки стимулов, означающие отсутствие переходов по этим стимулам в этом состоянии, и стационарность, означающую отсутствие переходов по реакциям в этом состоянии.

При построении ϕ-трасс проход через стабильное состояние добавляет в трассу сразу всё множество отвергаемых стимулов и реакций, то есть стимулов и реакций, по которым не определены переходы из данного состояния.

Это множество мы будем называть ϕ-множеством, мы уже обозначали его как ***ref***(s), где s – стабильное состояние.



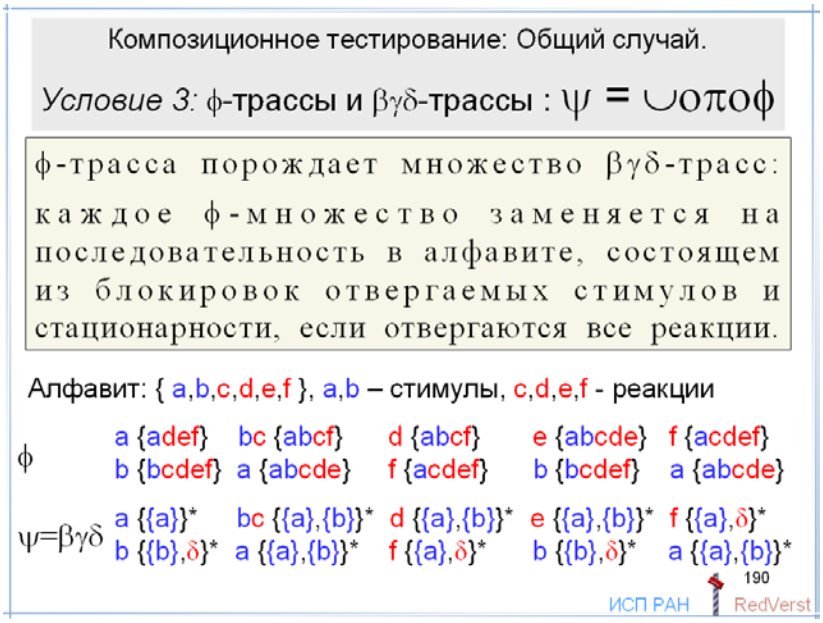
Более строго, проход стабильного состояния добавляет в трассу любое (в том числе нулевое) число ϕ-множеств этого состояния. Это можно понимать так, что в каждом стабильном состоянии определён переход-петля, помеченный ϕ-множеством состояния.

ϕ-трассы похожи на трассы готовности (ready traces), но только в последних используется не множество ***ref***(s) отвергаемых символов, а, наоборот, множество ***init***(s) символов, по которым есть переходы из состояния.

Условие монотонности 3: ϕ-трассы и βγδ-трассы: ψ = ∪∘π∘ϕ.

ϕ-трасса порождает множество βγδ-трасс. Соответствующее отображение – это оператор π.

Оператор π каждое ϕ-множество заменяет последовательностью в алфавите, состоящем из блокировок отвергаемых стимулов, то есть стимулов, принадлежащих ϕ-множеству, и стационарности, если отвергаются все реакции, то есть ϕ-множество содержит все реакции.

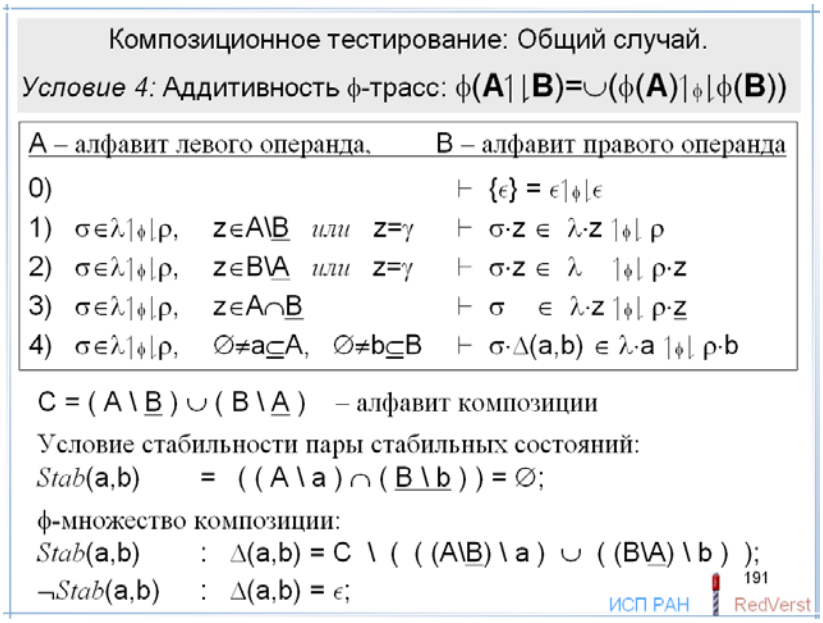


Очевидно, что для любой цепочки переходов каждая βγδ-трасса, соответствующая этой цепочке, получается с помощью оператора π из некоторой ϕ-трассы, соответствующей этой цепочке.

ϕ-трассу будем называть безопасной, если она порождает хотя бы одну безопасную βγδ-трассу.

Условие монотонности 4: Аддитивность ϕ-трасс: $\phi(A \upharpoonright\downarrow B) = \cup(\phi(A) \upharpoonright_\phi \downarrow \phi(B))$.

Теперь мы определим композицию ϕ-трасс.



0) Композиция двух пустых ϕ-трасс состоит из одной пустой ϕ-трассы.
1) 2) Если одна из ϕ-трасс продолжается внешним стимулом или реакций, или расширением γ, то есть символом, переход по которому выполняется асинхронно, то композиционная ϕ-трасса также продолжается этим символом.
3) Если обе ϕ-трассы продолжаются противоположными символами, то есть выполняется синхронный переход, который в композиционном автомате изображается как τ-переход, то композиционная ϕ-трасса остаётся той же.
4) Если обе ϕ-трассы продолжаются ϕ-множествами a и b, то композиционная ϕ-трасса продолжается значением функции Δ от a и b.

Рассмотрим подробнее последний, 4-ый случай. Сформулируем условие стабильности композиционного состояния. Очевидно, оно будет стабильным только в том случае, когда оба состояния компонентов стабильны. Однако это необходимое, но не достаточное условие. Если состояния обоих компонентов стабильны, то для стабильности композиционного состояния дополнительно требуется, чтобы не было ни одного синхронного перехода. A\a – это множество символов переходов из состояния левого операнда, B\b – это множество символов переходов из состояния правого операнда. Синхронного перехода не будет, если для каждого символа из одного множества в другом множестве нет противоположного символа: $(A\backslash a) \cap (\underline{B\backslash b}) = \varnothing$.

Если условие стабильности нарушено, композиционная ϕ-трасса остаётся той же, то есть не продолжается никаким ϕ-множеством. В этом случае функция Δ возвращает пустую трассу.

Если же условие стабильности выполнено, то композиционная ϕ-трасса продолжается ϕ-множеством, которое состоит из тех внешних символов, по которым нет переходов ни в

одном из компонентов. A\B – это внешние символы левого операнда. (A\B)\a – это внешние символы левого операнда, по которым в нём есть переходы. Соответственно, (B\A)\b – это внешние символы правого операнда, по которым в нём есть переходы. Объединение этих множеств – это все внешние символы, по которым есть переходы из композиционного состояния. Алфавит композиции – это C=A\B ∪ B\A. Вычитая из него символы, по которым есть переходы их композиционного состояния, получаем ϕ-множество композиционного состояния: C\((A\B)\a)∪(B\A)\b).

Две ϕ-трассы компонуемы, если их композиция не пуста.

Какие ϕ-трассы компонуемы? Возьмём в каждой ϕ-трассе подпоследовательность, получающуюся удалением внешних стимулов и реакций, и символа разрушения, то есть тех символов, переходы по которым выполняются асинхронно. Тогда эти подтрассы компонентов должны иметь одинаковую длину и на каждой i-ой позиции должны находиться либо 1) противоположные внутренние символы, что соответствует синхронному переходу, либо 2) ϕ-символы. Это второе условие (соответствие ϕ-символов) всегда можно выполнить, если выполнено первое условие (соответствие противоположных символов), подбором нужного количества идущих подряд ϕ-символов. Поэтому-то для удобства определения композиции ϕ-трасс мы разрешили ϕ-символ повторять в ϕ-трасе любое число раз.

Определённая таким образом композиция ϕ-трасс оказывается аддитивной: множество ϕ-трасс композиции равно объединению всех попарных композиций ϕ-трасс компонентов.

Условие 5: Существование максимальной реализации.

Максимальная реализация, как объединение всех конформных реализаций, очевидно, содержит те, и только те ϕ-трассы, которые есть в конформных реализациях. Тем самым существование максимальной ϕ-реализации доказано:

$$\phi\circ\boldsymbol{\mathcal{M}}(\mathbf{S}) = \phi\circ\cup(\mathbf{S}^{\triangledown}) = \cup\circ\phi(\mathbf{S}^{\triangledown})=\phi\circ\boldsymbol{\mathcal{M}}_{\phi}(\mathbf{S}).$$

Определение максимальной ϕ-реализации через ϕ-трассы всех конформных реализаций всё ещё неконструктивно. Вопрос стоит так: как можно получить ϕ-трассы конформных реализаций, глядя на ϕ-трассы спецификации?

Максимальных ϕ-реализаций может быть много. Мы ставим задачу найти способ *построения* одной такой максимальной ϕ-реализации по заданной спецификации.

Базовые ϕ-трассы.

Среди ϕ-трасс максимальной ϕ-реализации, на самом деле, есть лишние с точки зрения косой композиции.

Вполне достаточно ограничиться только некоторыми из них, которые мы будем называть базовыми ϕ-трассами.

Формально базовые ϕ-трассы – это ϕ-трассы, производные от ϕ-трасс автомата. Они задаются отображением множеств ϕ-трасс ***Base***.

Максимальная реализация, однако, содержит все свои базовые ϕ-трассы.

Главное свойство базовых ϕ-трасс состоит в том, что композиция множеств ϕ-трасс максимальных реализаций $\boldsymbol{ioco}_{\beta\gamma\delta}$-эквивалентна композиции подмножеств базовых ϕ-трасс:

$\boldsymbol{Base}\circ\phi\circ\mathcal{M}_\phi(\mathbf{A})\ \wr\varphi\downarrow\ \boldsymbol{Base}\circ\phi\circ\mathcal{M}_\phi(\mathbf{B})\ \approx\boldsymbol{ioco}_{\beta\gamma\delta}\ \phi\circ\mathcal{M}_\phi(\mathbf{A})\ \wr\downarrow\ \phi\circ\mathcal{M}_\phi(\mathbf{B})$.

Поэтому вместо того, чтобы строить автомат, ϕ-трассы которого – это все ϕ-трассы максимальной реализации, нам достаточно построить автомат, все базовые ϕ-трассы которого – это все базовые ϕ-трассы максимальной реализации.

На самом деле мы построим автомат, все ϕ-трассы которого – это ровно все базовые ϕ-трассы максимальной реализации:

$$\phi\circ\mathcal{M}_{\mathsf{base}\phi}(\mathbf{S}) \;=\; \textbf{\textit{Base}}\circ\phi\circ\mathcal{M}_{\phi}(\mathbf{B}).$$

Мы рассмотрим три типа ϕ-трасс: финальные, релевантные и сингулярные ϕ-трассы. Базовые ϕ-трассы – это ϕ-трассы, которые одновременно финальные, релевантные и сингулярные.

Финальные ϕ-трассы.

Финальные ϕ-трассы определяются на основе правила: «после разрушающего стимула произвольное поведение».

Финальные ϕ-трассы – это ϕ-трассы вида σ, σ·?x и σ·?x·γ, где σ - безопасная ϕ-трасса, а стимул ?x разрушающий после σ.

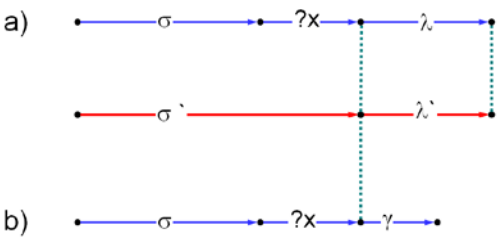
Пусть ϕ-трасса максимальной реализации не финальна и имеет вид **a)** σ·?x·λ, где ϕ-трасса σ безопасна, стимул ?x разрушающий после σ.

Поскольку после разрушающего стимула поведением конформной реализации может быть произвольным, есть такая конформная реализация, которая сразу разрушается.

Значит, максимальная реализация имеет ϕ-трассу **b)** σ·?x·γ.

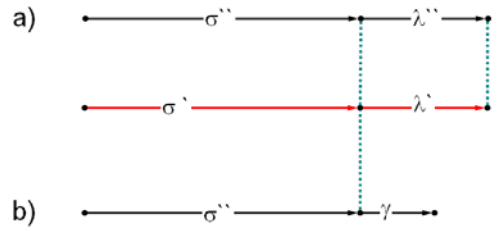
При композиции ϕ-трасса **a)** σ·?x·λ компонуется с некоторой ϕ-трассой σ`·λ`, где σ` компонуется с σ·?x и даёт множество ϕ-трасс σ``, а λ` с λ и даёт множество ϕ-трасс λ``.

Тогда ϕ-трасса **b)** σ·?x·γ компонуется с ϕ-трассой σ` и даёт множество ϕ-трасс σ``·γ.



Таким образом, в композиции любая из ϕ-трасс σ`` продолжается двумя способами: для **a)** ϕ-трассой из λ`` и для **b)** разрушением γ.

При гамма-расширении композиции всё, что порождают композиционные ϕ-трассы для случая **a)**, порождается композиционными ϕ-трассами для случая **b)**. А это и означает, что достаточно ограничиться только финальными ϕ-трассами.



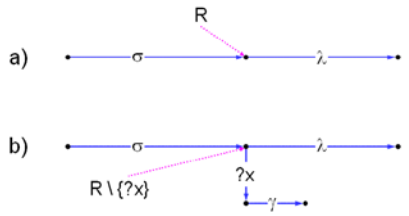
Релевантные ϕ-трассы.

Релевантные ϕ-трассы определяются на основе правила: «разрушающий стимул можно как принимать, так и блокировать».

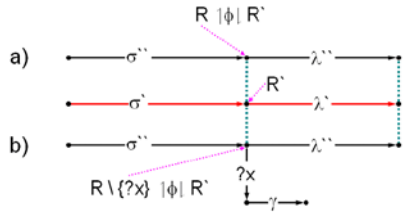
Релевантные ϕ-трассы – это ϕ-трассы, ϕ-множества которых содержат только такие стимулы, блокировки которые входят в порождаемые безопасные βγδ-трассы.

Пусть ϕ-трасса максимальной реализации не релевантна и имеет вид **a)** σ·R·λ, где ϕ-трасса σ безопасна, а R – ϕ-множество, в которое входит стимул x, не используемый при порождении безопасных βγδ-трасс.

Тогда есть такая конформная реализация, в которой есть ϕ-трасса, отличающаяся только тем, что в это ϕ-множество не входит стимул x, то есть ϕ-трасса **b)** σ·R\{x}·λ. Тогда после ϕ-трассы σ этот стимул x принимается и является разрушающим. В максимальной реализации после приёма такого стимула сразу должно быть разрушение.

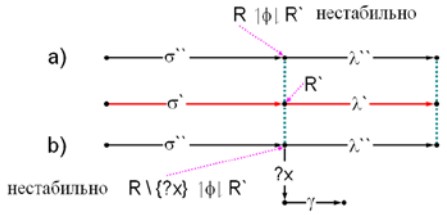


При композиции ϕ-трасса **a)** σ·R·λ компонуется с некоторой ϕ-трассой σ`·R`·λ`, где σ` компонуется с σ и даёт σ``, R` компонуется с R, а λ` с λ и даёт λ``.
Тогда с этой же ϕ-трассой σ`·R`·λ` компонуется ϕ-трасса **b)** σ·R\{x}·λ, причём σ` компонуется с σ и даёт ту же самую трассу σ``, R` компонуется с R\{x}, а λ` компонуется с λ и даёт ту же самую трассу λ``.



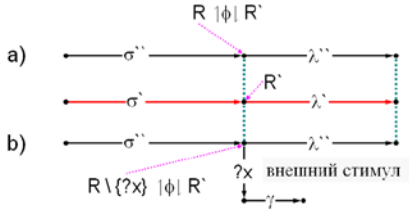
Рассмотрим два варианта в зависимости от того, является ли композиционное состояние в случае **a)** стабильным или нет.

<u>Композиционное состояние после ϕ-трассы σ`` нестабильно в случае **a)**</u>. Тогда композиционное состояние после ϕ-трассы σ`` нестабильно и в случае **b)**. Поэтому все βγδ-трассы, порождаемые в композиции в случае **a)**, порождаются также в случае **b)**.



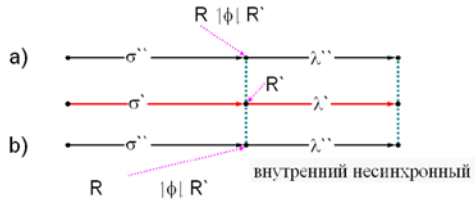
<u>Композиционное состояние после ϕ-трассы σ`` стабильно в случае **a)**</u>. Здесь возможны три подварианта в зависимости от стимула x: он внешний или внутренний и, если он внутренний, происходит по нему синхронизация или нет в случае **b)**.

<u>Стимул x внешний</u>. После композиционной ϕ-трассы σ`` композиционное состояние стабильно и в случае **b)**, но стимул x будет приниматься и далее идёт разрушение. Поэтому при гамма-расширении композиции всё, что порождают

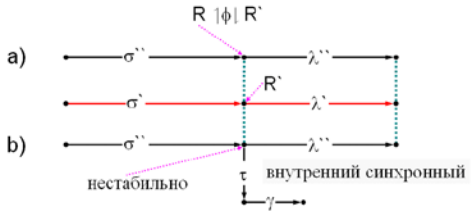


композиционные ϕ-трассы для случая **a)**, порождается композиционными ϕ-трассами для случая **b)**.

Стимул x внутренний, но не синхронизируемый. Случаи **a)** и **b)** в композиции вообще не различаются. Поэтому и в этом варианте при гамма-расширении композиции всё, что порождают композиционные ϕ-трассы для случая **a)**, порождается композиционными ϕ-трассами для случая **b)**.



Стимул x внутренний и происходит синхронизация по передаче этого стимула. В случае **b)** композиционная ϕ-трасса σ`` становится разрушающей. Поэтому и в этом варианте при гамма-расширении композиции всё, что порождают композиционные ϕ-трассы для случая **a)**, порождается композиционными ϕ-трассами для случая **b)**.



Таким образом, во всех вариантах можно оставить только такие ϕ-трассы, которые «более» релевантны. В конечном счёте, остаются только релевантные ϕ-трассы.

Сингулярные ϕ-трассы.

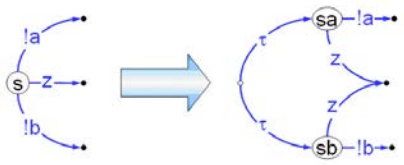
Сингулярные ϕ-трассы определяются на основе правила *may&must*: «можно выдавать не все реакции, допускаемые спецификацией, но стационарность должна сохраняться, то есть вообще не выдавать реакции можно только тогда, когда в спецификации нет реакций».

Для того чтобы два ϕ-множества порождали одинаковые последовательности отказов (блокировок и стационарностей), они должны содержать одни и те же стимулы. Однако реакции учитываются «все скопом» как стационарность. Поэтому эти ϕ-множества должны либо оба содержать все реакции (стационарность), либо оба не содержать всех реакций (нестационарность). В последнем случае они могут содержать разные подмножества реакций.

Сингулярные ϕ-трассы – это ϕ-трассы, в которых ϕ-множества содержат либо все реакции (стационарность), либо все реакции, кроме одной. В сингулярном состоянии, то есть состоянии с сингулярным ϕ-множеством, либо не определены переходы по реакциям, либо определены переходы ровно по одной реакции.

Пусть в конформной реализации есть стабильное состояние s, которое достижимо по безопасной βγδ-трассе, и в котором определены переходы по двум реакциям a и b.

Преобразуем реализацию, сдублировав это состояние так, что в одном из двух состояний s_a удаляются переходы по реакции b, а в другом s_b – по реакции a.



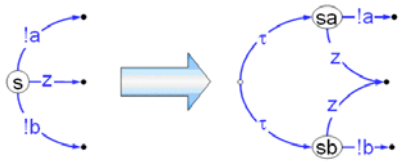
Очевидно, преобразованная реализация также конформна.

Поэтому максимальная реализация вместе с каждой ϕ-трассой содержит все сингулярные ϕ-трассы, которые из неё могут быть получены аналогичным расщеплением ϕ-множеств. Это означает, что, если такую процедуру расщепления стабильных состояний мы будем делать в максимальной ϕ-реализации, то мы не изменим её ϕ-трассы. Иными словами такое расщепление оставляет реализацию ϕ-максимальной.

Покажем, что для композиции можно оставить в максимальной ϕ-реализации только сингулярные ϕ-трассы. Для этого достаточно показать, что после расщепления стабильного состояния максимальной ϕ-реализации исходное состояние можно удалить. Если такую процедуру проделать по всем стабильным состояниям и всем реакциям, то у нас останутся только сингулярные состояния.

Мы покажем, что при композиции для любого композиционного маршрута, начинающегося в паре состояний ss`, есть маршрут с той же βγδ-трассой, начинающийся в паре состояний $s_a$s` или s_bs`.

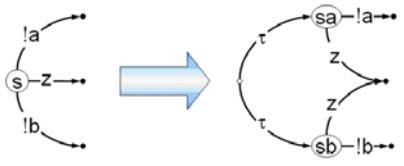
Пусть композиционное состояние ss` нестабильно, но не из-за переходов по реакциям a и b, то есть либо парное состояние s` нестабильно, либо есть внутренний символ, отличный от a и b, по которому есть переход в s, а в парном состоянии s` есть переход по противоположному символу. Тогда, очевидно, композиционные состояния $s_a$s` и s_bs` также будут нестабильны, и для каждого композиционного маршрута, начинающегося в состоянии ss`, есть маршрут с той же βγδ-трассой, начинающийся в состоянии s_as` или в состоянии $s_b$s`.



Пусть в композиции ss` нестабильно

Теперь пусть композиционное состояние ss` либо стабильно, либо нестабильно, но только из-за переходов по реакциям a и b, то есть один из этих переходов при композиции выполняется синхронно, что даёт τ-переход. Здесь возможны варианты в зависимости от реакций a и b: реакция внешняя или внутренняя и, если она внутренняя, происходит по ней синхронизация или нет. Всего получается 2^3=9 вариантов, но из-за симметрии достаточно рассмотреть 6 из них.

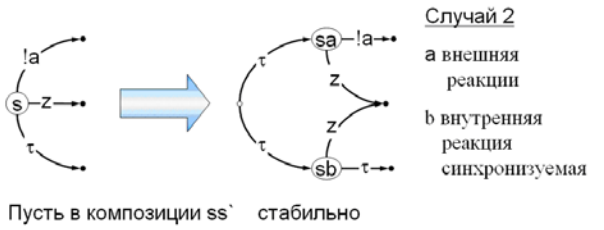
1) <u>Обе реакции внешние</u>. Все композиционные состояния ss`, $s_a$s` и s_bs` стабильные. Для любого композиционного маршрута, начинающегося в состоянии ss`, есть маршрут с той же βγδ-трассой, начинающийся: 1) в состояние s_as`, если в этой βγδ-трассе первый базовый символ – это реакция a, или 2) в состоянии $s_b$s`, если первый базовый символ – это реакция b, или 3) в каждом из этих состояний, если первый базовый символ отличен от a и b.



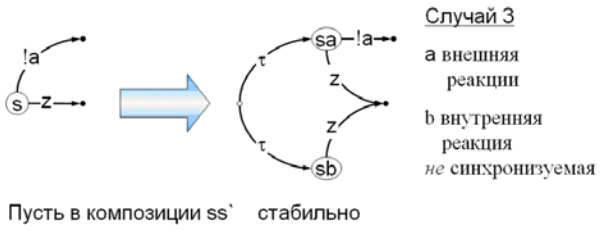
Пусть в композиции ss` стабильно

Случай 1
a и b внешние реакции

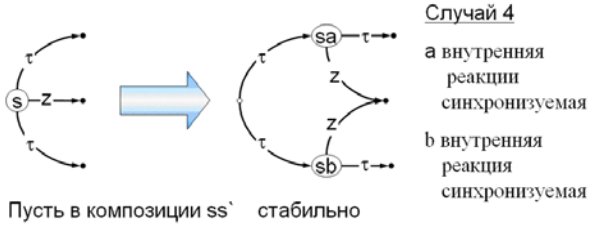
2) Реакция a внешняя, реакция b внутренняя синхронизуемая. Композиционные состояния ss` и $s_b$s` нестабильные, а состояние s_as` стабильное. Для любого композиционного маршрута, начинающегося в состоянии ss`, есть маршрут с такой же βγδ-трассой, начинающийся в состояние s_as` или $s_b$s`.



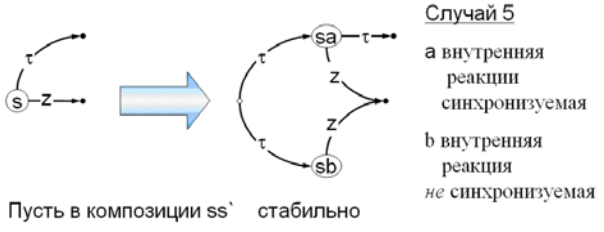
3) Реакция a внешняя, реакция b внутренняя несинхронизуемая. Все три композиционных состояния стабильны. Для любого композиционного маршрута, начинающегося в состоянии ss`, есть маршрут с такой же βγδ-трассой, начинающийся в состояние $s_a$s`.



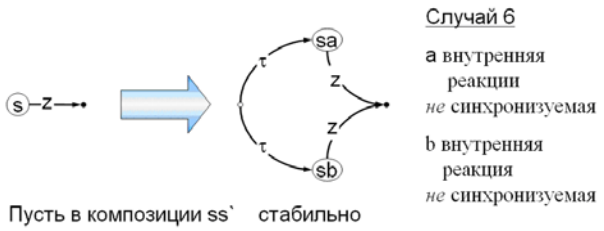
4) Реакции a и b внутренние синхронизуемые. Все три композиционных состояния нестабильны. Для любого композиционного маршрута, начинающегося в состоянии ss`, есть маршрут с такой же βγδ-трассой, начинающийся в состояние $s_a$s` или s_bs`.



5) Реакции a и b внутренние, но реакция a синхронизуемая, а реакция b несинхронизуемая. Композиционные состояния ss` и $s_a$s` нестабильные, а состояние s_bs` стабильное. Для любого композиционного маршрута, начинающегося в состоянии ss`, есть маршрут с такой же βγδ-трассой, начинающийся в состояние s_as`.



6) Реакции a и b внутренние и несинхронизуемые. Все три композиционных состояния стабильны. Для любого композиционного маршрута, начинающегося в состоянии ss`, есть маршрут с такой же βγδ-трассой, начинающийся в состояние $s_a$s` и в состоянии s_bs`.



Таким образом, во всех вариантах можно не только добавить состояния-дубли s_as` и $s_b$s`, но также можно удалить исходное состояние s. Если такую процедуру выполнять, пока можно, мы получим автомат, в котором все стабильные состояния, достижимые по безопасным трассам, сингулярны.

Нормальные ϕ-трассы.

Будем называть *нормальной* такую ϕ-трассу автомата, которая не может быть получена из другой ϕ-трассы автомата удалением единичных ϕ-символов, то есть ϕ-символов, до и после которых в ϕ-трассе нет ϕ-символов, или является начальным отрезком такой ϕ-трассы.

По подмножеству нормальных ϕ-трасс автоматически восстанавливается всё множество ϕ-трасс автомата. Поэтому из равенства подмножеств нормальных ϕ-трасс двух автоматов следует равенство всех ϕ-трасс этих автоматов.

$$\mathit{Norm}\circ\phi(\mathbf{A})=\mathit{Norm}\circ\phi(\mathbf{B}) \quad \Leftrightarrow \quad \phi(\mathbf{A})=\phi(\mathbf{B})$$

Таким образом, мы ставим задачу: найти алгоритм построения по спецификации такого автомата $\mathcal{A}(\mathbf{S})$, множество нормальных ϕ-трасс которого совпадало бы с множеством нормальных базовых (финальных+релевантных+сингулярных) ϕ-трасс максимальной реализации.

$$\mathit{Norm}\circ\phi(\mathcal{A}(\mathbf{S}))=\mathit{Norm}\circ\mathit{Base}\circ\phi(\mathbf{S})$$

Теперь мы рассмотрим идею алгоритма, который по спецификации S строит автомат $\mathcal{A}$(S).

Состояния $\mathcal{A}$(S) – это нормальные базовые ϕ-трассы максимальной реализации.

Сначала будем считать, что состояния автомата $\mathcal{A}$(S) – это все нормальные базовые ϕ-трассы максимальной реализации. Множество таких ϕ-трасс обозначим Σ.

Пусть ϕ-трасса $\sigma \in \Sigma$ безопасна и не заканчивается на ϕ-множество, то есть либо пуста, либо последний её символ – это стимул или реакция. В $\mathcal{A}$(S) этой трассе поставим в соответствие *нестабильное* состояние.

Сингулярным ϕ-множеством будем называть такое ϕ-множество, которое содержит либо все реакции, либо все, кроме одной. Множество сингулярных ϕ-множеств обозначим как $\mathcal{R}$.

Пусть ϕ-множество R сингулярно и может продолжать ϕ-трассу σ.

Тогда из состояния автомата $\mathcal{A}$(S), соответствующего ϕ-трассе σ, проведём τ-переход в состояние σ·R. Каждое состояние вида σ·R сделаем *стабильным*.

Далее, пусть стимул $?x_\gamma$ разрушающий после ϕ-трассы σ или после ϕ-трассы σ·R.

По каждому разрушающему стимулу проведём переход в состояние «разрушение».

Наконец, пусть символ (стимул или реакция) z безопасен после ϕ-трассы σ·R или после ϕ-трассы σ.

По символам z проведём переходы в соответствующие ϕ-трассы σ·z или σ·R·z.

Из состояния σ·R мы проводим переходы по тем символам z, которые не принадлежат R. А из состояния σ – по тем z, по которым мы не можем получить то же самое через переходы из стабильных состояний вида σ·R. Иными словами, ϕ-трасса σ·z должна быть нормальной: есть такое её продолжение λ, то есть имеется такая ϕ-трасса σ·z·λ, что для любого имеющегося ϕ-множества R нет ϕ-трассы σ·R·z·λ.

Нормальные ϕ-трассы $\mathcal{A}$(S) – это нормальные базовые ϕ-трассы максимальной реализации.

В построенном автомате нормальные ϕ-трассы – это нормальные базовые ϕ-трассы максимальной реализации.

Действительно, пусть ϕ-трасса σ безопасная нормальная базовая ϕ-трасса максимальной реализации. И пусть она также безопасна и нормальна в $\mathcal{A}$(S), а состояние σ – одно из тех состояний $\mathcal{A}$(S), в которых мы оказываемся после этой ϕ-трассы.

Тогда рассмотрим все ϕ-трассы, продолжающие ϕ-трассу σ в нашем построении: σ·R, σ·R·z, σ·z, σ·R·$?x_\gamma$ и σ·$?x_\gamma$. По построению, это все базовые нормальные ϕ-трассы максимальной реализации, которые минимально продолжают ϕ-трассу σ и не заканчиваются на ϕ-множество. Все эти ϕ-трассы будут и автомате $\mathcal{A}$(S). Можно показать, что каждая из них нормальна в $\mathcal{A}$(S) и заканчивается в нужном состоянии. При этом те ϕ-трассы, которые не заканчиваются в состоянии «разрушении», являются безопасными.

Ограничения на спецификацию.

Определение состояний автомата $\mathcal{A}$(**S**) как нормальных базовых ϕ-трасс максимальной реализации неконструктивно. Далее наша задача – попытаться определить эти состояния конструктивно. Такими состояниями у нас будут подмножества состояний спецификации **S**. Более точно: каждому состоянию $\mathcal{A}$(**S**) мы поставим в соответствие подмножество состояний **S**, хотя разным состояниям $\mathcal{A}$(**S**), вообще говоря, могут соответствовать одинаковые подмножества состояний **S**.

Но, прежде всего, сформулируем ограничения на спецификацию, которые позволят нам это сделать.

Мы будем рассматривать спецификации, в которых специфицируются стимулы, принимаемые в состоянии: как безопасные, так и разрушающие. Неспецифицированные в состоянии стимулы – это стимулы, блокируемые в этом состоянии.

Для генерации перечислимого полного тестового набора мы предъявляли к спецификации следующие требования:

1. Безопасна и безопасно-конвергентна.
2. Спецификация *принимаемых* стимулов (безопасных и разрушающих).
3. Регулярность по реакциям.
4. Конечно-безопасно-ветвящаяся.
   Итератор TI(s) конечного числа переходов из s.
5. Множество стимулов перечислимо.
   Итератор X перечислимого множества стимулов.

Все эти требования, кроме последнего, нужны также для построения $\mathcal{A}$(**S**).

Максимальная βγδ-трасса.

Пусть σ - безопасная базовая ϕ-трасса максимальной реализации

1. Назовём *амбивалентным стимулом* такой стимул, который принадлежит некоторому ϕ-множеству ϕ-трассы $?x \in \sigma(i)$ и существует βγδ-трасса, порождаемая соответствующим начальным отрезком ϕ-трассы $\mu \in \pi(\sigma[1..i])$, которая в спецификации продолжается не только блокировкой этого стимула {?x}, но и самим стимулом ?x.

Если стимул входит в ϕ-множество ϕ-трассы, но не амбивалентен, то он блокируется в любой βγδ-трассе, порождаемой предшествующей ϕ-трассой. Такой стимул, очевидно, безопасный, но «неинтересный».

2. Для конечно-безопасно-ветвящейся спецификации число амбивалентных стимулов ϕ-трассы конечно.

3. Кроме того, существует *максимальная* βγδ-трасса, порождаемая этой ϕ-трассой, $\sigma_0 \in \pi(\sigma)$, которая содержит блокировки всех амбивалентных стимулов (и только такие блокировки) и все возможные δ (для ϕ-множеств, содержащих все реакции).

4. Максимальная βγδ-трасса σ_0 заканчивается в спецификации в конечном числе состояний, и в каждом из них заканчиваются все порождаемые ϕ-трассой σ βγδ-трассы.

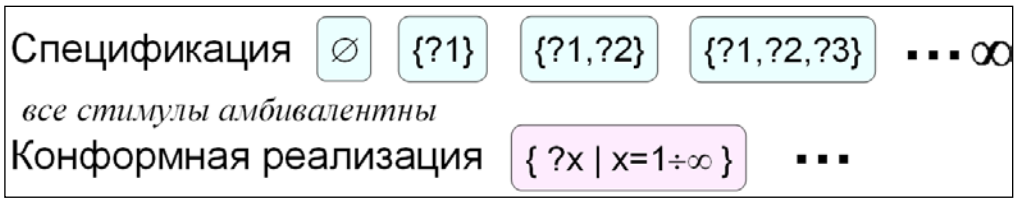
Бесконечное ветвление.

Если бы спецификация не была конечно-безопасно-ветвящейся, то такой максимальной βγδ-трассы могло и не быть.

Действительно, достаточно рассмотреть случай, когда некоторая ϕ-трасса есть как в максимальной реализации, так и в спецификации, и безопасна. Допустим, эта ϕ-трасса заканчивается в спецификации в бесконечном множестве состояний.

Рассмотрим пример. Пусть стимулы – это все натуральные числа, пусть задана безопасная ϕ-трасса в спецификации, и пусть после ϕ-трассы все стимулы безопасны.

Рассмотрим состояния после ϕ-трассы и блокируемые в них стимулы.
В 0-ом состоянии все стимулы принимаются, в 1-ом – блокируется стимул 1, во 2-ом – стимулы 1 и 2, в 3-ем – стимулы с 1-го по 3-ий и так далее.
Очевидно, все стимулы амбивалентны, поскольку каждый стимул в некотором состоянии спецификации принимается, а в некотором состоянии блокируется.



Тогда в некоторой конформной реализации эта ϕ-трасса может заканчиваться в состоянии, где блокируются все стимулы. Это объясняется тем, что любая конечная последовательность блокировок, порождаемая таким состоянием конформной реализации, порождается некоторым состоянием спецификации.

Число амбивалентных стимулов бесконечно – это все стимулы. Поскольку любая конечная βγδ-трасса содержит только конечное число блокировок, максимальной βγδ-трассы не существует.

τ-замкнутое множество состояний спецификации.

τ-замыканием множества состояний будем называть надмножество состояний, в которые можно попасть из состояний исходного множества по τ-переходам.

Через $\mathcal{T}$(**S**) обозначим семейство всех τ-замкнутых непустых подмножеств множества состояний спецификации **S**.

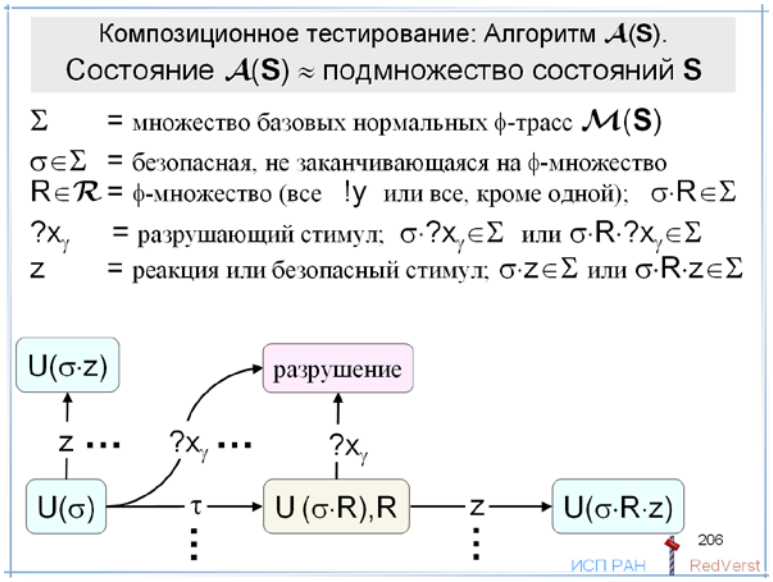
Множество состояний в конце любой βγδ-трассы τ-замкнуто.

Каждой безопасной базовой нормальной ϕ-трассе σ максимальной реализации, которая пуста или последний символ которой – это стимул или реакция (не ϕ-множество), поставим в соответствие конечное τ-замкнутое множество U(σ) = **S** ***after*** σ_0 состояний спецификации **S**, в которых заканчивается максимальная βγδ-трасса σ_0.

Состояния автомата $\mathcal{A}$(S).

Итак, вернёмся к построению автомата $\mathcal{A}$(**S**), состояния которого – это базовые нормальные ϕ-трассы максимальной реализации.

Прежде всего, заменим ϕ-трассу σ множеством состояний спецификации U(σ).



Соответственно, ϕ-трасса σ·z заменяется множеством состояний U(σ·z), а ϕ-трасса σ·R·z заменяется множеством состояний U(σ·R·z).

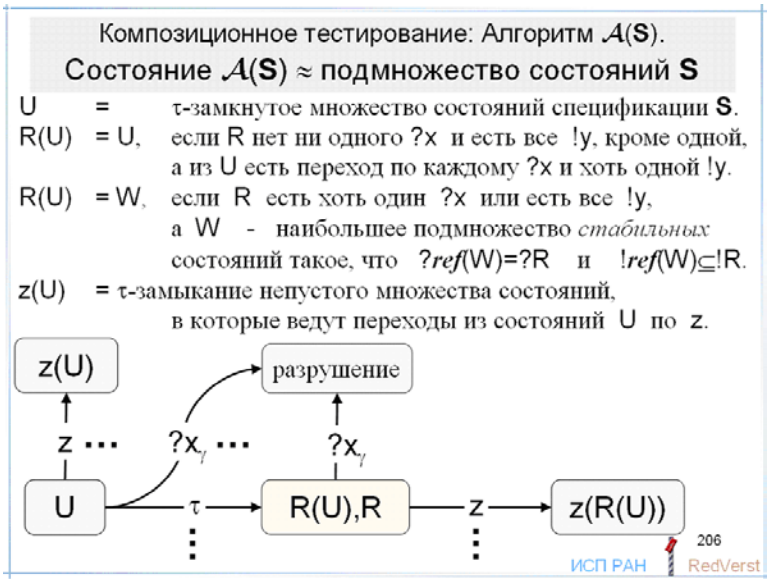
Однако, ϕ-трасса σ·R заменяется не просто множеством состояний U(σ·R), а парой: множество состояний U(σ·R) и ϕ-множество R. Почему?

1) Если для некоторой базовой нормальной ϕ-трассы σ` максимальной реализации, которая также безопасна и не заканчивается на ϕ-множество, оказалось то же самое множество состояний спецификации U(σ`)=U(σ·R), то в $\mathcal{A}$(**S**) ϕ-трассе σ` соответствует нестабильное состояние, а ϕ-трассе σ·R соответствует стабильное состояние. Поэтому мы имеем переход U(σ·R) —τ→U(σ·R),R.

2) Могут быть два разных ϕ-множества R1≠R2, которыми может продолжаться ϕ-трасса σ, но максимальные βγδ-трассы продолженных ϕ-трасс заканчиваются в одном и том же множестве состояний спецификации U(σ·R1)=U(σ·R2).

Условие проведения перехода из состояния U(σ) становится следующим: существует состояние спецификации, в которое мы можем попасть по максимальной βγδ-трассе ϕ-трассы σ·z, но не можем попасть по максимальной βγδ-трассе ни одной из существующих ϕ-трасс вида σ·R·z.

Теперь мы можем определить автомат $\mathcal{A}$(S), вообще не используя ϕ-трасс, но вычисляя соответствующие помножества состояний спецификации.



Пусть U – некоторое τ-замкнутое множество состояний спецификации, в которое мы попадаем по максимальной βγδ-трассе некоторой безопасной базовой нормальной ϕ-трассы максимальной реализации, не кончающейся на ϕ-множество.

Посмотрим, какими сингулярными ϕ-множествами R может продолжаться такая ϕ-трасса.

1) Пусть в R нет стимулов и есть все реакции, кроме одной реакции. Пусть по каждому стимулу есть переход из хотя бы одного состояния множества U, и есть переход по той реакции, которой нет в R, из хотя бы одного состояния множества U. Тогда через R(U) обозначим само множество U. В противном случае будем считать, что R(U) пусто.

2) Пусть R содержит хотя бы один стимул (блокировка) или содержит все реакции (стационарность). Пусть в U есть подмножество W, которое удовлетворяет следующим требованиям по стимулам и реакциям. Каждый стимул, который принадлежит R, блокируется в каждом состоянии W, а каждый стимул, который не принадлежит R, принимается хотя бы в одном состоянии W. Если R содержит все реакции, то все состояния W стационарны. Если же R не содержит одну реакцию, то эту реакцию выдаётся хотя бы в одном состоянии W. Тогда через

R(U) мы обозначим наибольшее такое подмножество W. Если существует хотя бы одно W, то существует и наибольшее, потому что объединение все подмножеств W, удовлетворяющим нашим требованиям по стимулам и реакциям, также удовлетворяет этим требованиям. Если не существует ни одного такого W, то будем считать, что R(U) пусто.

Через z(U) обозначим τ-замыкание непустого множества состояний, в которые можно попасть из состояний U с помощью переходов по символу z.

Формальные правила вывода.

Теперь мы можем записать формальные правила вывода автомата $\mathcal{A}(\mathbf{S})$ из спецификации **S**.

Пусть задана спецификация $\mathbf{S} = AA(V,C_\gamma,E,s_0)$.
Пусть $U \in \mathcal{T}(\mathbf{S})$ – τ-замкнутое множество состояний спецификации.
Обозначим через $\mathbf{X}_\gamma(U)$ – стимулы, разрушающие в состояниях U,
а через $U_{stab}(U)$ – подмножество всех стабильных состояний множества U.

Определим $\mathcal{A}(\mathbf{S}) = AA(V`,C_\gamma,E`,s_0`)$, где
$V` = \mathcal{T}(\mathbf{S}) \cup \mathcal{T}(\mathbf{S}) \times \mathcal{R} \cup \{\Gamma\}$ – множество состояний;
состояние – это либо τ-замкнутое множество состояний спецификации,
либо пара из такого множества и сингулярного ϕ-множества,
либо состояние «разрушение», которое мы обозначаем символом Γ;
$s_0` = \mathbf{S}\ \textit{after}\ \epsilon$ – начальное состояние,
а множество переходов – E` – это наименьшее множество, порождаемое правилами вывода:

$\forall U \in \mathcal{T}(\mathbf{S})\ \ \forall z \in C\ \ \forall R \in \mathcal{R}$

$z \notin \mathbf{X}_\gamma(U)\ \&\ z(U) \setminus z(U_{stab}(U)) \neq \varnothing$	$\vdash U \text{—} z \rightarrow z(U) \setminus z(U_{stab}(U))$
$?x \in \mathbf{X}_\gamma(U)$	$\vdash U \text{—} ?x \rightarrow \Gamma$
$R(U) \neq \varnothing$	$\vdash U \text{—} \tau \rightarrow (R(U),R)$
$z \notin \mathbf{X}_\gamma(U)\ \&\ z \notin R\ \&\ z(U) \neq \varnothing$	$\vdash (U,R) \text{—} z \rightarrow z(U)$
$x \in \mathbf{X}_\gamma(U)$	$\vdash (U,R) \text{—} x \rightarrow \Gamma$
	$\vdash \Gamma \text{—} \gamma \rightarrow \Gamma \text{—} \tau \rightarrow \Gamma$

Спецификация безопасных стимулов.

Мы рассмотрели алгоритм для автоматов, в которых специфицированы принимаемые стимулы (безопасные и разрушающие), и неспецифицированы блокируемые стимулы.

Нужна модификация алгоритма для автоматов, в которых специфицированы безопасные стимулы (принимаемые и блокируемые), и неспецифицированы разрушающие стимулы.

Цели построения *косой* композиции.

Напомним, что построение косой композиции преследует три цели.

1. Проверка компонуемости, то есть проверка того, что косая композиция спецификаций компонентов безопасна и безопасно-конвергентна. Это гарантирует, что композиция любых конформных реализаций будет безопасно-тестируемой для любой корректной спецификации системы.

2. Проверка корректности системной спецификации, то есть проверка того, что косая композиция спецификаций компонентов конформна системной спецификации.

3. Генерация системных тестов по косой композиции спецификаций компонентов. Это полезно, когда у нас нет корректной системной спецификации, или эта спецификация недостаточно сильная для проверки интересующих нас свойств системы.

Ограничение на спецификации компонентов.

Выше мы уже сформулировали требования к спецификации **S** для того, чтобы можно было построить преобразованную специкацию $\mathcal{A}(\mathbf{S})$.

Обратим внимание, что сама преобразованная спецификация $\mathcal{A}(\mathbf{S})$, по построению, гамма-нормальна.

Как строится композиция двух таких преобразованных спецификаций? Для каждой пары состояний, которая у нас может получиться в процессе построения, то есть достижимая из пары начальных состояний, мы должны построить все композиционные переходы. Что для этого нужно?

Во-первых, мы должны уметь перечислять переходы в каждом из компонентов. Это обеспечивается алгоритмом TI.

Во-вторых, для каждого символа – стимула или реакции, которым помечен переход в одном компоненте, мы должны определить: внешний этот символ или внутренний? То есть, принадлежит ли он подчёркнутому алфавиту другого компонента или нет.

Это эквивалентно разрешимости алфавита. Соответственно, нам нужен алгоритм, вычисляющий предикат принадлежности символа алфавиту.

Итеративное построение.

Прежде всего, заметим, что нас никогда не интересуют переходы в композиционных гамма-состояний, кроме, естественно, самих гамма-переходов. Поэтому мы при построении косой композиции мы не будет строить переходы из композиционных гамма-состояний, кроме, естественно, гамма-перехода.

При этих ограничениях на спецификации компонентов мы можем для любого, наперёд заданного, числа n построить подавтомат композиции, содержащий все композиционные *маршруты* длиной не более n.

Для каждой из объявленных трёх целей нам нужно большее: за конечное время построить подавтомат косой композиции, содержащий все *трассы* длиной не более n?

Для этого мы должны проверять все цепочки τ-переходов, которые у нас возникают.

При композиции τ-переход появляется либо асинхронно, наследуя τ-переход в одном из компонентов, либо синхронно, как результат соединения перехода по выдаче внутреннего символа в одном компоненте с приёмом этого символа в другом компоненте. Понятно, что, если у нас возникает бесконечная цепочка τ-переходов, проходящая через бесконечное число композиционных состояний, то мы эту задачу решить не сможем.

Поэтому выдвигается следующее требование внутренней регулярности компонентов:

> для состояния s, достижимого минуя гамма-состояния, должно быть конечно множество состояний, достижимых из s по машрутам без переходов по внешним стимулам и реакциям, которые не проходят, но могут заканчиваться в гамма-состояниях.

Безопасность и безопасно-конвергентность.

Для проверки безопасности композиции мы должны убедиться, что из пары *начальных* состояний разрушение не достижимо по τ-переходам и переходам по реакциям.

Для проверки безопасно-конвергентности нам нужно просматривать такие цепочки τ-переходов и переходов по реакциям из *каждого* композиционного состояния, достижимого по безопасной композиционной βγδ-трассе + стимул.

Ясно, что такие цепочки переходов должны проходить через конечное множество композиционных состояний. Для этого мы должны выдвинуть следующее требование внешней регулярности компонентов:

> для состояния s, достижимого по безопасной трассе [плюс, возможно, внешний стимул], должно быть конечно множество состояний, достижимых из s по машрутам без переходов по внешним стимулам (но, возможно, по внешним реакциям), которые не проходят, но могут заканчиваться в гамма-состояниях.

Этого достаточно для проверки безопасности косой композиции за конечное время.

Безопасно-конвергентность косой композиции проверяется *итеративно*: для всех безопасных трасс длиной не более заранее заданного числа n.

Этого достаточно для полной проверки безопасно-конвергентности косой композиции конечных спецификаций.

Иначе безопасно-конвергентность косой композиции обеспечивается условием внешней ограниченности компонентов:

> для состояния s, достижимого по безопасной трассе [плюс, возможно, внешний стимул], должны быть конечны все начинающиеся в s трассы без внешних стимулов.

Корректность системной спецификации.

Аналитическая проверка корректности системной спецификации выполняется *итеративно*: для всех безопасных трасс длиной не более заранее заданного числа n.

Этого достаточно в двух случаях:
1) Если системная спецификация имеет конечное поведение, то есть конечное число трасс.
2) Если и системная спецификация и косая композиция обе конечны, то есть содержат конечное число достижимых состояний и переходов.

В противном случае полная аналитическая проверка требует бесконечного числа конечных проверок (для трасс ограниченной длины) аналогично бесконечному полному тестовому набору.

Генерация системных тестов по косой композиции.

Условие внешней регулярности компонентов гарантирует регулярность по реакциям косой композиции.

Если в каждом компоненте задан итератор перечислимого множества стимулов и алгоритм разрешения алфавита, то можно построить итератор перечислимого множества внешних стимулов.

Если косая композиция безопасна и безопасно-конвергентна, то генерация полного набора системных тестов делается итеративно в процессе построения косой композиции.