ИСП Российская Академия Наук
Институт Системного Программирования

И.Б.БУРДОНОВ & А.С.КОСАЧЕВ
В.В.КУЛЯМИН

ТЕОРИЯ СООТВЕТСТВИЯ ДЛЯ СИСТЕМ С БЛОКИРОВКАМИ И РАЗРУШЕНИЕМ

Москва 2006

ТЕОРИЯ СООТВЕТСТВИЯ ДЛЯ СИСТЕМ С БЛОКИРОВКАМИ И РАЗРУШЕНИЕМ

И.Б.Бурдонов, А.С.Косачев, В.В.Кулямин
Institute for System Programming of Russian Academy of Sciences (ISPRAS),
B. Communisticheskaya, 25, Moscow, Russia
igor@ispras.ru, kos@ispras.ru, kuliamin@ispras.ru
http://www.ispras.ru

Аннотация. В работе изучается тестирование соответствия систем, в которых возможна блокировка (приёма) стимулов и разрушение системы. Дивергенция также моделируется разрушением. В качестве соответствия предлагается отношение ***ioco***$_{\beta\gamma\delta}$ – обобщение отношения ***ioco*** (**I**nput-**O**utput **CO**nformance). Для того, чтобы избегать разрушения реализации при тестировании, отношение строится только на безопасных трассах, которые не могут привести к разрушению. Предлагается гипотеза о безопасности, определяющая класс реализаций, которые можно тестировать на соответствие заданной спецификации. Рассматриваются два вида моделей: трассовые модели и система переходов (Labelled Transition System), и показывается их эквивалентность. Описывается генерация тестов и её алгоритмизация. Рассматриваются различные виды пополнения частично-определённых по стимулам спецификаций. Сравниваются семантики отношений ***ioco*** и ***ioco***$_{\beta\gamma\delta}$. Рассматривается проблема несохранения соответствия при композиции и предлагается её решение с помощью монотонного преобразования спецификаций. Излагается общая теория монотонности соответствия и определяются достаточные условия монотонности. Предлагаются монотонные преобразования для общего случая и для подклассов без блокировок и/или разрушения. Рассматриваются проблемы алгоритмизации преобразований и композиции и описываются соответствующие алгоритмы.

Keywords: test machines, conformance testing, trace models, labelled transition system, ioco relation, partially input-enabled, refused inputs, forbidden actions, safe testing, test generation, specification completion, asynchronous testing, composition verification, algorithmization.

Оглавление

Содержание

Предисловие

В течение последних 50-ти лет можно было наблюдать несколько существенных изменений в технологиях промышленной разработки программного обеспечения (ПО). Начиналась она как работа небольших групп программистов, переквалифицировавшихся из математиков и электротехников, затем прошла стадии огромных забюрократизированных коллективов разработчиков в крупных организациях и небольших, но фонтанирующих идеями команд компаний-стартапов.

Многие из программистов «первого призыва» имели математическое или инженерно-техническое образование и склад ума, способствующий точному мышлению. Они стремились к точности и аккуратности и в построении программ. Однако, в мире рыночных отношений успех того или иного программного продукта определяется не только и не столько его правильностью и надежностью. Огромную роль в этом играют и факторы моды, революционности продукта или, наоборот, привычек пользователей, умелая реклама, соглашения с крупными партнерами и прочие составляющие любой экономической деятельности. Поэтому пока бурно развивались технологии разработки, позволяющие создавать все более и более сложные программные комплексы, востребованные рынком как часть инфраструктуры современной экономики, методы обеспечения корректности и надежности создаваемого ПО оставались в тени, поскольку требовали существенного увеличения вложений в его производство без ясно видимой прибыли в будущем. Программисты-математики при этом либо приспосабливались к таким обстоятельствам, либо переходили из индустрии в образовательные и исследовательские учреждения, где их увлечение построением математически правильных программ не вызывало у начальства ассоциаций с упущенной выгодой и необоснованными трудозатратами.

Кризис в индустрии производства ПО в 2000 году не только отрезвил участников рынка, развеяв множество грандиозных, хотя и несбыточных планов, но и вскрыл возникший перекос в технологиях разработки программ. Они позволяли достаточно быстро и с небольшими затратами создавать программные комплексы такой сложности, которая и не снилась даже большим коллективам разработчиков еще в середине 80-х годов. Однако для проверки их надежности и корректности работы использовались те же методы, которые были изобретены еще в 60-х годах. Если до кризиса клиента еще можно было убедить отдать деньги за сырой продукт, распаляя его надежды на потрясающие результаты внедрения: снижение издержек, повышение эффективности и пр., то после ситуация резко изменилась. Теперь руководители проектов должны были, скрепя сердце, тратить

драгоценное время и не менее дорогие усилия разработчиков не только на то, чтобы создать нечто новое и невиданное, но и продемонстрировать заказчику, что оно действительно решает нужные ему задачи и делает это правильно. Неожиданно высокая трудоемкость этой второй части работ, на которую раньше и внимания-то особо не обращали, вынудила заняться поисками технологий, позволяющих сделать это более эффективно.

В новых обстоятельствах внезапно пригодилась «нерыночная» увлеченность программистов первой волны, уже ставших классиками Computer Science, корректностью работы программ. Стали востребованными созданные на основе их идей техники разработки ПО, изначально работающего правильно, и методы проверки правильности функционирования произвольно взятой программы. Даже компании, чьи продукты 10 лет назад славились далеко не идеальной стабильностью и надежностью работы, провозглашают курс на повышение надежности и безопасности работы ПО и проводят для своих разработчиков и партнеров тренинги по техникам написания корректных и безопасных программ.

Прямо на наших глазах складывается новая дисциплина, стоящая на границе между теоретической информатикой и такими инженерными науками как инженерия ПО и системная инженерия, — *тестирование на основе моделей*. Во многом она опирается на работы различных исследователей, выполненные в 80-х и в 90-х годах прошлого века и посвященные проблемам тестирования телекоммуникационных протоколов. Самые первые же статьи, которые с полным правом можно отнести с этой области, появились еще в конце 60-х–начале 70-х годов [Henn64,Vasil73]. Тем не менее, серьезный практический интерес к ее результатам возник только в начале XXI века, в связи с востребованностью эффективных методов контроля и обеспечения качества сложных программных и программно-аппаратных систем в индустрии.

1. Формальные модели и соответствия между ними

Основная идея тестирования на основе моделей вполне прозрачна: поскольку сложная система не поддается проверке «в лоб», создайте сначала более простую ее *модель*, описав в ней *только то, что для вас важно* на данном этапе, а затем стройте тесты *только* на основании полученной модели. Такой подход хорошо работает на практике, если саму модель и набор тестов для нее получается строить по частям, инкрементально, постепенно наращивая набор свойств реальной системы, отражаемых в модели, и, соответственно, охват тестами различных аспектов ее работы.

Как возможность такой инкрементальной разработки моделей и тестов, так и конкретные техники, используемые при этом, существенно зависят от

самого *вида* используемых моделей, и от того, что может быть использовано при тестировании, какие имеются *возможности по управлению* тестируемой системой и *наблюдению* различных элементов ее поведения. Чаще всего на данный момент используются модели в виде различных разновидностей *конечных автоматов* (finite automata, finite state machines, FSM) и близких к ним *систем помеченных переходов* (labeled transition systems, LTS). Модели обоих этих видов построены из состояний и помеченных переходов, но в автоматах меткой перехода служит пара <воздействие на систему, ответная реакция системы>, а в системе переходов такой меткой является только что-то одно — либо воздействие, либо реакция.

Такие модели одновременно достаточно просты, чтобы поддаваться строгому анализу, и достаточно универсальны, чтобы описывать большой класс практически важных систем. Возможность строгого анализа свойств модели становится существенной, если необходимы определенные гарантии того, что все важные аспекты ее поведения покрываются построенными тестами. Поэтому такие модели *формальны*, т.е. описаны на языке математики. Само же тестирование стоится на некоторой математической теории, предписывающей определенные способы построения тестов, которые обеспечат им нужные свойства полноты, а тестированию придадут необходимую глубину и надежность.

Тестирование всегда проверяет соответствие тестируемой системы требованиям, поэтому в его контексте рассматриваются обычно две модели — модель, описывающая требования, и модель, описывающая реальное поведение тестируемой системы. Первая называется *спецификацией*, а вторая — *реализацией*. При этом спецификация обычно задана, а реализация — нет. Мы просто предполагаем, что существует определенная модель данного вида, абсолютно адекватно описывающая все нужные нам нюансы поведения реальной системы. Эта гипотеза дает нам возможность строить теорию тестирования на строгой математической основе. Если же она не выполнена, нам необходимо искать другой вид моделей, который позволит выразить все важные элементы поведения реальной системы.

Что же касается используемого набора возможностей по управлению и наблюдению за тестируемой системой, при рассмотрении формальных моделей он также формализуется в виде понятия *формального соответствия* между спецификацией и реализацией. Определенный набор возможностей позволяет проверить определенное соответствие между двумя моделями, и, наоборот, соответствие говорит о том, какие модели можно отличить друг от друга при помощи заданного набора возможностей, а какие — нет.

В данной работе проводится исследование свойств определенного соответствия между системами помеченных переходов и излагается теория построения тестов для его проверки. Если даже по поводу вида моделей, которые нужно использовать для описания сложных систем, большинство исследователей сейчас еще не пришли к общему мнению, то по поводу используемого при построении тестов соответствия имеются десятки различных точек зрения. Поэтому необходимо аргументировать сделанный в данной работе выбор.

2. Отношение ioco и его проблемы

Мы старались идти от наиболее естественного набора возможностей, используемого при тестировании: *возможности оказывать* некоторые *воздействия на тестируемую систему*, и *возможности наблюдать ее* ответные *реакции*. Это базовые возможности, на которых основаны любые тесты.

Есть, однако, более тонкие возможности, которые часто встречаются на практике, например, *возможность наблюдать отсутствие реакции* системы. Например, если мы бросили в автомат с газировкой монетку, нажали нужную кнопку, а он ничего не выдает, через некоторое время мы обычно догадываемся, что, сколько ни жди, ничего больше не произойдет (а дальше по обстоятельствам — кто-то стучит по автомату и ругается, кто-то просто уходит, а кто-то вызывает на помощь техников). Причем, чтобы сделать такой вывод, мы чаще всего ждем не слишком долго — по опыту, если автомат не реагирует в течение десятка секунд, то он и дальше ничего делать не будет.

Этот важный новый тип наблюдений, вообще говоря, не сводится к наблюдению какой-то реакции тестируемой системы — он позволяет наблюдать отсутствие всякой реакции. Вместе с двумя базовыми возможностями он был использован в определении формального соответствия ***ioco*** между системами помеченных переходов в работах Тритманса [Tret92,Tret96] в начале 90-х годов прошлого столетия. С тех пор соответствие ***ioco*** считается одним из классических соответствий, используемых при тестировании на основе формальных моделей.

Однако, проанализировав более детально соответствие ***ioco***, можно заметить следующие используемые при тестировании на его основе допущения.

- Во-первых, реализация должна быть всюду определенной, т.е. всегда принимать любое оказанное на нее воздействие.

Такое допущение обычно оправдано для компонентов телекоммуникационных протоколов, или, более широко, для компонентов, которые должны взаимодействовать с почти произвольным окружением.

Но для внутренних компонентов сложных систем, взаимодействующих только с другими компонентами той же системы, такое ограничение кажется чересчур жестким. Практика показывает, что в этом случае более эффективно использовать не защитный стиль разработки, не делающий никаких предположений о входных данных, а кооперативный. При этом взаимодействующие компоненты должны подчиняться дополнительным ограничениям при вызовах операций — нужно обеспечивать некоторую согласованность входных данных, но за счет этого вызываемый компонент может рассчитывать на выполнение этих ограничений и реализовывать более эффективные алгоритмы работы. В целом кооперативная разработка не понижает общей надежности системы, но часто значительно повышает ее эффективность и позволяет более четко определить области ответственности различных компонентов.

Тем не менее, протестировать компонент, созданный в кооперативном стиле, с помощью ***ioco*** зачастую невозможно.

- Во-вторых, ***ioco*** предполагает, что во время тестирования можно не дать тестируемой системе выдать реакцию и даже можно вместо этого заставить ее принять другое воздействие. Все выглядит так, будто система находится под очень сильным контролем тестировщика — начинает проявлять реакцию только тогда, когда тестировщик разрешает ей сделать это.

  Это свойство следует из семантики взаимодействия между системами помеченных переходов основанной на механизме рандеву. Такая семантика возникает, если взаимодействие систем моделируется при помощи параллельной композиции в смысле процессных исчислений.

  Это ограничение также слишком жестко. Оно может быть оправданным лишь в тех случаях, когда возможности управления тестируемой системой очень велики, например, во время отладки, когда можно приостановить выполнение тестируемой системы и сделать к ней в это время еще одно обращение извне.

Оба эти допущения отмечались многими авторами, и самим Тритмансом. Второе служит причиной наиболее серьезных споров, поскольку на практике необходимая для его выполнения степень контроля над тестируемой системой встречается достаточно редко. Тем не менее, если продолжать

моделировать реальные системы системами помеченных переходов (LTS), а взаимодействие между ними — параллельной композицией, другого выхода нет. Наоборот, если при этом мы лишены возможности «приостановить» выдачу реакции системой, с точки зрения такой семантики взаимодействия это значит, что тестировщик и тестируемая система взаимодействуют не напрямую, а через некоторую среду (или контекст), не дающую такой возможности. Поведение такой среды также можно описать при помощи системы переходов. При этом мы тестируем не исходную систему, а ее композицию со средой.

Таким образом, выделяются случаи *синхронного тестирования*, при котором мы имеем необходимые возможности «приостановки» реакций, и, значит, можем использовать тесты на основе ***ioco***, и *асинхронного тестирования*, при котором между тестировщиком и тестируемой системой находится промежуточная среда, определенным образом искажающая их взаимодействие. Во втором случае, если поведение среды точно известно, логично было бы использовать в качестве спецификации композицию исходной спецификации и среды.

Тут то и обнаруживается еще один существенный недостаток ***ioco***. Это соответствие не сохраняется при параллельной композиции. Если исходная реализация соответствовала спецификации, то ее композиция со средой может уже не соответствовать композиции спецификации со средой. Примеры подобных реализаций и спецификаций можно найти далее в тексте Введения. Эта проблема является следствием того, что в процессных исчислениях более естественным соответствием между спецификацией и реализацией считается отношение бимоделирования между системами переходов, которое не может быть проверено при помощи тестирования в подавляющем большинстве случаев. Кроме того, оно слишком жестко привязывает реализацию к спецификации и иногда не является даже желательным, поскольку спецификация часто описывает альтернативы поведения, а в конкретной реализации выбирается одна из них.

Из сложившейся ситуации возможны несколько выходов.

1) Совсем забыть про ***ioco***, поскольку во многих практически важных ситуациях его допущения слишком сильны. Вместо этого строить математическую теорию тестирования для других соответствий, с более реалистичными допущениями об имеющихся возможностях. Этот путь прослеживается в работах Петренко и Евтушенко [PYH03,HP04], которые исследуют соответствие, возникающее при отделении тестируемой системы от тестировщика при помощи очередей.

2) Добавить новые возможности наблюдения, которые помогут различать ситуации, неразличимые для ***ioco*** при асинхронном тестировании, тем самым приблизив его по возможностям к синхронному и забыв об идее использования композиции для моделирования асинхронного тестирования.

   Такой путь рассматривается, например, в работе [JJTV99], где все события снабжаются временными метками, что позволяет в ряде случаев избавиться от ограничений асинхронного тестирования. Правда, в [JJTV99], этот подход исследуется по отношению не к ***ioco***, а к ***ioconf***, более простому соответствию.

3) Поправить определение параллельной композиции так, чтобы отношение ***ioco*** сохранялось новой композицией.

   Этот путь рассматривается в работах Тритманса с рядом соавторов [BRT03r,BRT03]. В этих работах им не удалось разрешить возникшую проблему, однако удалось нащупать ее возможную причину. Оказывается, в том случае, если спецификация полностью определена, никаких проблем не возникает — для таких спецификаций отношение ***ioco*** сохраняется параллельной композицией. Это позволяет предположить, что его несохранение связано с неопределенным поведением в спецификации.

3. Предлагаемый в данной работе подход

Здесь мы постараемся в общих чертах описать, что же мы пытались сделать в этой работе.

Мы выбрали третий путь из перечисленных выше, т.е. постарались понять, как можно «исправить» параллельную композицию с тем, чтобы сделать возможной общую теорию построения тестов для синхронного и асинхронного тестирования. Два первых пути практически не оставляют шансов на это, порождая десятки специальных теорий для каждой комбинации из набора используемых возможностей и промежуточной среды при тестировании.

Кроме того, мы попытались более внимательно изучить «камень преткновения» ***ioco*** — неопределенное в спецификации поведение. Видно, что эта проблема как-то связана и с указанным выше допущением о полной определенности реализации. При использовании ***ioco*** несимметрия между спецификацией и реализацией заложена сразу в двух местах — мы не только считаем, что можем приостановить ее работу, не принимая очередную ее

реакцию, но и отказываем в таком праве ей — она-то должна все всегда принимать.

Анализ возможного смысла неопределенного поведения дает следующие варианты (первый, третий и последний из них указаны в [BP94]).

1) *Запрещенное воздействие*.
   Неопределенность результатов воздействия может означать, что оказывать его в этом состоянии нельзя или нежелательно. В ходе теста этого нужно избегать. Это может быть связано как с возможностью разрушения системы при оказании такого воздействия (например, кнопка немедленного саморазрушения для систем оборонного назначения, или вызов функции освобождения памяти для незанятого ранее ее участка в операционной системе, где поддерживается единственный процесс). Такое воздействие может привести систему в нежелательное состояние, например, прекратить реагировать на внешние воздействия, или же его обработка может быть просто пока не реализована в данной версии системы. В любом случае смысл один — такие воздействия просто не должны оказываться во время тестирования.

2) *Отвергаемое* или *блокируемое воздействие.*
   Может оказаться, что такое воздействие можно оказывать, но система его не принимает, более того, явно демонстрирует каким-то образом его неприятие и всегда остается после этого в прежнем состоянии. Такого рода воздействия встречаются на практике, например, так можно моделировать неактивные элементы пользовательского интерфейса. Более того, в ходе тестирования чаще всего необходимо проверять, что такое воздействие *действительно отвергается*, а не принимается системой в данном состоянии.

3) *Ошибочное воздействие.*
   В этом случае считается, что воздействие может быть оказано, но его прием системой означает ошибку.

4) *Ничего.*
   Иногда неспецифицированное воздействие считается не оказывающим никакого влияния на поведение системы. Нам кажется, что в этом случае лучше (и честнее) прямо задать такое поведение в спецификации, поскольку первые три варианта представляют собой более серьезные поводы для неопределенности.

Таким образом, семантика неспецифицированного поведения может быть уточнена разными способами, и, в зависимости от выбранного уточнения,

будет демонстрироваться несколько различное поведение и наше отношение к такому поведению в ходе тестирования будет различным.

Это приводит к мысли рассматривать модели в виде систем переходов, в которых различные уточнения неопределенного поведения зафиксированы явно. Третий и четвертый случаи могут быть промоделированы при помощи достаточно обычных меток, а вот первый и второй нуждаются в специальной трактовке. Мы моделируем первый вариант при помощи специальной метки *разрушения*, а второй — считая отсутствие перехода по данному воздействию в данном состоянии его *блокировкой*.

Дальнейшее исследование посвящено анализу свойств таких моделей, обобщению соответствия ***ioco*** для них — соответствию $\boldsymbol{ioco}_{\beta\gamma\delta}$, а также решению проблемы перенесения соответствия при композиции. Разработанные техники позволили нам, в частности, решить и исходную задачу — поправить определение параллельной композиции, обеспечив с ее помощью перенесения оригинального ***ioco***.

В данной работе мы рассматривали, в основном, теоретические вопросы построения полных наборов тестов для проверки выбранных соответствий в разных ситуациях и алгоритмизации этого построения. Возникающие при этом практические трудности и особенности использования данных методов на практике оставлены, по большей части, за рамками этого исследования.

4. Благодарности

Данная работа была выполнена в рамках развития технологии тестирования UniTESK [KKPPB03,KPKB03]. Отдельные части этой работы выполнялись как части исследований, финансируемых грантами РФФИ и контрактами с Президиумом РАН.

Авторы выражают благодарность коллективу группы RedVerst[1] ИСП РАН[2] во главе с А.К. Петренко, а также В.З.Шнитману, А.Н.Томилину и В.П.Иванникову, обеспечившим нам возможность провести это исследование.

[1] **RE**search and **D**evelopment: **VER**ification, **S**pecification, **T**esting
[2] www.ispras.ru

Как читать эту книгу.

1. Краткое содержание книги

Общематические понятия и обозначения

Вводятся встречающиеся в тексте общематематические понятия и обозначения. Чтение этого материала можно отложить до тех пор, пока не возникнет потребности в нём.

Часть 1. Введение

В этой части вводятся основные понятия теории соответствия и рассматриваются важнейшие проблемы.

Глава 1.1. Формализация понятия «правильности»

Правильность реализации определяется как её соответствие спецификации, понимаемой как описание требований к реализации. Формализация правильности опирается на представление реализации и спецификации в виде математических моделей, когда соответствие – это отношение двух моделей. Выбор типа модели и соответствия определяется тестовыми возможностями и гипотезами о реализации.

Глава 1.2. Формализация тестового эксперимента

Целью тестового эксперимента является проверка правильности реализации. Для формализации вводится машина тестирования – «чёрный ящик», внутри которого размещается реализация, и который снабжается средствами наблюдения за поведением реализации и управления им. Определяются трассы наблюдений, включающие как совершаемые реализацией действия, так и отсутствие таковых (отказы). Дополнительно вводится разрушение как абстракция недопустимого (нежелательного) поведения. Обсуждается проблема приоритетов – зависимость действия реализации от множества разрешённых действий.

Глава 1.3. Дивергенция и недетерминизм

Две важнейшие проблемы возникают в тестовых экспериментах: дивергенция («зацикливание» реализации) и недетерминизм поведения.

Глава 1.4. Трассовая и автоматная модели

Трассовая модель опирается на трассы наблюдений в процессе эксперимента. Автоматная модель основана на состояниях и помеченных переходах между ними, а наблюдаемая трасса понимается как последовательность меток цепочки переходов.

Глава 1.5. Тесты

Определяются тесты для трассовой и автоматной моделей. Рассматриваются два вида тестирования: синхронное, когда тестер и реализация взаимодействуют «напрямую», и асинхронное, когда при котором взаимодействие опосредуется промежуточной средой – тестовым контекстом, в который погружена реализация. Также рассматриваются проблемы безопасности (неразрушаемости реализации) и полноты тестирования.

Глава 1.6. Модели ввода-вывода

Для моделей ввода-вывода взаимодействие реализации с окружением (тестом) понимается как обмен информацией: стимулами, принимаемыми реализацией, и реакциями, выдаваемыми реализацией. Вводится отношение $\boldsymbol{ioco}_{\beta\gamma\delta}$.

Глава 1.7. Проблемы взаимодействия тестера и реализации

Обсуждаются четыре проблемы, связанные с тем, что при взаимодействии тестера и реализации 1) стимулы или реакции могут отвергаться (блокироваться, «тормозиться») принимающей стороной, или 2) принимающая сторона может «выбирать» один стимул или одну реакцию из нескольких, предлагаемых передающей стороной.

Глава 1.8. Пополнение спецификации

Отсутствие перехода по некоторому стимулу в некотором состоянии спецификационной автоматной модели может трактоваться как «недоспецификация». Рассматривается пополнение спецификации, основанное на том или ином допущении полноты и преследующее цель «доспецифицировать» модель. В связи с этим обсуждается проблема рефлексивности (самоприменимости) спецификации.

Глава 1.9. Асинхронное тестирование и верификация композиции

Обсуждаются две проблемы асинхронного тестирования: 1) не ловятся ошибки, обнаруживаемые при синхронном тестировании, 2) ловятся ложные ошибки. Эти проблемы – частный случай общих проблем композиции, когда сложная система собирается по определённым правилам из нескольких компонентов. Главной из

них является 2-ая проблема, формулируемая как проблема монотонности соответствия: как добиться того, чтобы композиция реализаций, конформных своим спецификациям, была конформна композиции спецификаций.

Часть 2. Синхронное тестирование

Излагается общая теория соответствия при синхронном тестировании.

Глава 2.1. Машина тестирования

Формализация тестового эксперимента с помощью машины тестирования. Описываются формальные правила работы машины тестирования общего вида. Кроме того, рассматривается разновидность машины тестирования – βγδ-машина, порождающая βγδ-трассы и используемая для тестирования по соответствию $\boldsymbol{ioco}_{\beta\gamma\delta}$.

Глава 2.2. Трассовые модели

Излагается формальная трассовая теория, в которой моделью системы является дерево βγδ-трасс – трасс, включающих стационарность, блокировки и разрушение. Также рассматриваются модели безопасных и финальных трасс. Определяется гипотеза о безопасности и соответствие $\boldsymbol{ioco}_{\beta\gamma\delta}$. Описывается генерация тестов и обсуждаются вопросы алгоритмизации, имея в виду практическое тестирование.

Глава 2.3. LTS-модель

Здесь модель определяется как система переходов (LTS – labelled transition system), а взаимодействие моделируется параллельной композицией LTS. Показывается эквивалентность трассовой и LTS-моделей, что даёт возможность использования трассовой теории для LTS, в частности, для генерации LTS-тестов. Обсуждаются специфические для LTS вопросы алгоритмизации.

Глава 2.4. Сравнение моделей

Различные трассовые модели сравниваются как между собой, так и с LTS-моделью. Обсуждается вопрос о связи дивергенции с разрушением.

Глава 2.5. Пополнение спецификации

Изучаются различные виды пополнения (устранения неопределённости в трактовке неспецифицированного стимула) частично-определённых спецификаций на основе того или иного

допущения полноты. Пополнение используется для решения проблемы монотонности соответствия. Мы используем его также для прояснения классической семантики отношения ***ioco***, в частности, сравнением с семантикой отношения $\boldsymbol{ioco}_{\beta\gamma\delta}$.

Часть 3. Верификация композиции

Излагается теория верификации композиции для отношения $\boldsymbol{ioco}_{\beta\gamma\delta}$ как решение проблемы монотонности композиции. Проблема сохранения соответствия при асинхронном тестирование трактуется как частный случай монотонности соответствия (левомонотонность), когда композиция состоит ровно из двух компонентов: реализации и известного тестового контекста.

Глава 3.1. Общая теория монотонности

Вводятся понятия корректной спецификации системы, косой композиции, монотонного и левомонотонного преобразований. Определяются восемь достаточных условий монотонности. В оставшихся главах конкретизируются понятия общей теории монотонности для отношения $\boldsymbol{ioco}_{\beta\gamma\delta}$ и доказывается выполнение достаточных условий монотонности.

Глава 3.2. Мажорирование βγδ-трасс и отношение $ioco_{\beta\gamma\delta}$

Вводится отношение частичного порядка βγδ-трасс (мажорирование) и показывается эквивалентность этого мажорирования отношению $\boldsymbol{ioco}_{\beta\gamma\delta}$.

Глава 3.3. ϕ-трассы

Глава 3.4. Общий случай: Мажорирование ϕ-трасс

Определяется новый вид трасс, названных ϕ-трассами. Эти трассы, вообще говоря, не являются трассами наблюдений, но отражают ту часть структуры LTS, которая необходима и достаточна для определения композиции. Доказывается выполнение соответствующих условий монотонности.

Глава 3.5. Общий случай: Преобразование $\mathcal{T}_{\beta\gamma\delta}$

Определяется преобразование спецификации $\boldsymbol{\mathcal{T}}_{\beta\gamma\delta}$ и доказывается его монотонность.

Глава 3.6. Общий случай: Алгоритмизация

Обсуждаются вопросы алгоритмизации преобразования $\boldsymbol{\mathcal{T}}_{\beta\gamma\delta}$ и вводится новое монотонное преобразование $\boldsymbol{\mathcal{W}}_{\beta\gamma\delta}$, позволяющее

ослабить требования к исходным спецификациям. Также рассматриваются вопросы алгоритмической композиции преобразованных LTS.

Часть 4. Верификация композиции в частных случаях

Рассматривается верификация композиции для важных частных подклассов LTS и предлагаются упрощённые версии монотонного преобразования для этих случаев:

Глава 4.1. Спецификации без разрушения
Глава 4.2. Спецификации без блокировок
Глава 4.3. Спецификации без блокировок и без разрушения

Итоги и перспективы

В заключение перечисляются основные результаты и намечаются направления дальнейших исследований.

2. Нумерация и ссылки

Главы нумеруются двойным индексом `i.j`, где `i` номер части, а `j` номер главы в пределах части.
Разделы нумеруются тройным индексом `i.j.k`, где `i` номер части, `j` номер главы, а `k` номер раздела в пределах главы.
Определения и утверждения имеют сквозную нумерацию.
Формулировка утверждения заканчивается □`p`, где `p` номер страницы, с которой начинается доказательство утверждения в Приложении.

3. Порядок чтения книги

Если Вы хотите ознакомиться только с основными понятиями и проблемами соответствия и Вас не интересует формальная теория, достаточно пропустить «Общематематические понятия и обозначения» и прочитать «Часть 1. Введение».

Если Вас интересует формальная теория синхронного тестирования соответствия и пополнение спецификаций, но не интересуют асинхронное тестирование и верификация композиции, достаточно прочитать «Общематематические понятия и обозначения» и «Часть 2. Синхронное тестирование».

«Часть 3. Верификация композиции» опирается на все предыдущие части.

Если Вас не интересуют доказательства утверждений, не заглядывайте в Приложение.

Если Вас не интересуют проблемы алгоритмизации генерации тестов, пополнения и преобразования спецификаций, а также их композиции, пропустите разделы 2.2.9, 2.3.7 (за исключением подраздела «*S*-ветвящиеся LTS.»), 2.5.4, Глава 3.6 и разделы 4.1.3, 4.2.9 и 4.3.2.

Взаимосвязь разделов показана на следующей таблице. Здесь для каждого раздела `i` указаны те разделы `j`, определения, обозначения или утверждения из которых используются в данном разделе. Учитывая транзитивность ссылок, указывается минимально необходимое число разделов: если `i`→`j`→`k`, то для раздела `i` раздел `k` не указывается.

раздел	**используемые разделы**
Часть 1. Введение	
0. Общематематические понятия и обозначения	
Часть 2. Синхронное тестирование	
Глава 2.1. Машина тестирования	
2.1.1. Машина общего вида	0
2.1.2. Машина для *ioco*$\beta\gamma\delta$*-тестирования* – βγδ-`машина`	2.1.1
Глава 2.2. Трассовые модели	
2.2.1. βγδ-модель	2.1.2
2.2.2. Разложение βγδ-модели на поддеревья	2.2.1
2.2.3. Модель безопасных трасс	2.2.1
2.2.4. Модель финальных трасс	2.2.3
2.2.5. Соотношение трассовых моделей	2.2.4
2.2.6. Гипотеза о безопасности	2.2.3
2.2.7. Соответствие *ioco*$\beta\gamma\delta$	2.2.4, 2.2.6
2.2.8. Тест и полный набор тестов	2.2.7
2.2.9. Алгоритмизация	2.2.5, 2.2.8
Глава 2.3. LTS-модель	
2.3.1. Определение LTS	0
2.3.2. LTS-модель	2.3.1, 2.2.6
2.3.3. LTS-модель как βγδ-`машина`	2.3.2
2.3.4. Эквивалентность βγδ-`моделей` и LTS-моделей	2.2.2, 2.3.2
2.3.5. Гипотеза о безопасности и соответствие *ioco*$\beta\gamma\delta$	2.2.7, 2.3.4
2.3.6. Композиция LTS и тесты	2.2.8, 2.3.5
2.3.7. Алгоритмизация	2.2.9, 2.3.6
Глава 2.4. Сравнение моделей	2.2.5, 2.3.4
Глава 2.5. Пополнение спецификации	
2.5.1. Пополнение состояний	2.3.2
2.5.2. Трассовое пополнение	2.5.1
2.5.3. Классическая семантика *ioco*	2.3.5, 2.5.2
2.5.4. Алгоритмизация	2.3.7, 2.5.3

Часть 3. Верификация композиции	
Глава 3.1. Общая теория монотонности	
3.1.1. Косая композиция и монотонное соответствие	0
3.1.2. Восемь достаточных условий монотонности соответствия	3.1.1
3.1.3. План доказательства достаточных условий	2.3.7, 3.1.2
Глава 3.2. Мажорирование βγδ-трасс и отношение *iocoβγδ*	
3.2.1. Мажорирование βγδ-`трасс`.	3.1.3
3.2.2. Эквивалентность *iocoβγδ* мажорированию βγδ-`трасс` (Монотонность 1:)	3.2.1
Глава 3.3. ϕ-трассы	
3.3.1. ϕ-символы, ϕ-трассы и ϕ-маршруты LTS	3.1.3
3.3.2. ξ-оператор. Генеративность ϕ-трасс (Монотонность 2:)	3.3.1
3.3.3. Композиция ϕ-трасс. Аддитивность (Монотонность 3:)	3.3.1
Глава 3.4. Общий случай: Мажорирование ϕ-трасс	
3.4.1. Определение мажорирования ϕ-трасс	3.3.1
3.4.2. Мажорирование ϕ-`трасс` предпорядок (Монотонность 8:)	3.4.1
3.4.3. Генеративность мажорирования ϕ-`трасс` (Монотонность 4:)	3.3.2, 3.4.1
3.4.4. Композиционность мажорирования ϕ-`трасс` (Монотонность 5:)	3.3.3, 3.4.2
Глава 3.5. Общий случай: Преобразование *Tβγδ*	
3.5.1. Примеры и общая идея преобразования	3.4.1
3.5.2. Формальное определение преобразования	3.5.1
3.5.3. Конформность преобразования (Монотонность 6:)	3.2.2, 3.5.2
3.5.4. *S-ветвящиеся* и локально-конечно-ветвящиеся LTS.	3.1.3
3.5.5. Мажорантность преобразования (Монотонность 7:)	3.3.2, 3.5.3, 3.5.4,
3.5.6. Монотонность преобразования	3.4.3, 3.4.4, 3.5.5
3.5.7. Оптимизация преобразования	3.5.6
Глава 3.6. Общий случай: Алгоритмизация	
3.6.1. Замкнутость по алгоритмическому преобразованию *Tβγδ*	3.5.5
3.6.2. Алгоритмизация преобразования *Tβγδ* для конечного алфавита	3.5.6
3.6.3. Алгоритмическое преобразование *Tβγδ* при многократной композиции	3.5.7, 3.6.2
3.6.4. Алфавит и ветвление при композиции LTS	3.6.1
3.6.5. Проблема дивергенции	3.6.4
3.6.6. Слабо-регулярность и *Tβγδ-преобразование.* Сильно-регулярность	3.6.5
3.6.7. Двойная LTS и *Wβγδ-преобразование.*	3.6.2, 3.6.6
3.6.8. Алгоритмизация композиции *Tβγδ-преобразованных* LTS	3.6.2, 3.6.6
3.6.9. Алгоритмизация композиции *Wβγδ-преобразованных* LTS	3.6.7, 3.6.8
Часть 4. Верификация композиции в частных случаях	
Глава 4.1.Спецификации без разрушения	
4.1.1. Мажорирование ϕ-`трасс` без разрушения	3.4.3, 3.4.4
4.1.2. Преобразование *Tβδ*	3.5.6
4.1.3. Алгоритмизация	3.6.8, 4.1.2

Глава 4.2. Спецификации без блокировок	
4.2.1. Определение мажорирования ϕ-трасс	3.2.1, 3.3.2
4.2.2. Мажорирование ϕ-`трасс` предпорядок (Монотонность 8:)	4.2.1
4.2.3. Генеративность мажорирования ϕ-`трасс` (Монотонность 4:)	4.2.1
4.2.4. Композиционность мажорирования ϕ-`трасс` (Монотонность 5:)	4.2.1
4.2.5. Преобразование $\mathcal{T}\gamma\delta$	3.5.1
4.2.6. Конформность преобразования $\mathcal{T}\gamma\delta$ (Монотонность 6:)	4.2.5.
4.2.7. Мажорантность преобразования $\mathcal{T}\gamma\delta$ (Монотонность 7:)	3.5.4, 4.2.5.
4.2.8. Монотонность преобразования $\mathcal{T}\gamma\delta$	3.3.3, 4.2.3, 4.2.4, 4.2.6, 4.2.7
4.2.9. Алгоритмизация	3.6.9, 4.2.2, 4.2.6, 4.2.7
Глава 4.3. Спецификации без блокировок и без разрушения	
4.3.1. Монотонность	4.2.8
4.3.2. Алгоритмизация	4.1.3

ОБЩЕМАТЕМАТИЧЕСКИЕ ПОНЯТИЯ И ОБОЗНАЧЕНИЯ

Общематематические понятия и обозначения

Структура раздела:
1. Множества, числа и соответствия
2. Последовательности
3. Деревья последовательностей
4. Порождающий граф

Здесь мы вводим некоторые встречающиеся в тексте общематематические понятия и обозначения. Чтение этого материала можно отложить до тех пор, пока не возникнет потребности в нём.

1. Множества, числа и соответствия

Определение 1:

- $\mathcal{N}$ – множество натуральных чисел;
- $\mathcal{N}_0 =_{def} \mathcal{N}\cup\{0\}$ – множество целых неотрицательных чисел.
- Введём символ бесконечности ∞, и определим $\mathcal{N}_{0_\infty} =_{def} \mathcal{N}_0\cup\{\infty\}$ и $\forall n\in\mathcal{N}_0\ n<\infty,\ \infty+n=n+\infty=\infty,\ \infty-n=\infty$.
- Определим отрезок множества $\mathcal{N}_0$: для $n\in\mathcal{N}_0$ и $k\in\mathcal{N}_{0_\infty}$:
  $[n..k] =_{def} \{i\in\mathcal{N}|n\le i\le k\}$.
- Для множества C:
  - $\wp(C) =_{def} \{A|A\subseteq C\}$ – множество всех подмножеств C,
  - $|C|$ – мощность множества C;
    для бесконечного множества C будем обозначать $|C|=\infty$.
- Для соответствий $f\subseteq A\times B$ и $g\subseteq C\times A$:
  - $afb =_{def} f(a,b) =_{def} (a,b)\in f$;
  - $Dom(f) =_{def} \{a\in A|\exists b\in B\ afb\}$;
  - $Im(f) =_{def} \{b\in B|\exists a\in A\ afb\}$;
  - $f^{-1} =_{def} \{(b,a)|afb\}$;
  - $f\circ g =_{def} \{(c,b)|c\in C\ \&\ b\in B\ \&\ \exists a\in A\ cga\ \&\ afb\}$;
- *Отображением* называется однозначное соответствие. Мы будем рассматривать, как правило, всюду определённые отображения:
  $f:A\to B =_{def} f\subseteq A\times B\ \&\ \forall a\in A\ |\{b\in B|afb\}|=1$.
- Для отображения $f:A\to B$, элемента $a\in A$ и подмножества $C\subseteq A$:
  - $f(a) =_{def} b\in B$ такое, что afb;
  - $f(C) =_{def} \{f(a)|a\in C\}$.
- $Bool =_{def} \{true, false\}$.
- *Предикатом* называется отображение в $Bool$.

- Если задано отображение $f:A\to B$, то, по умолчанию, будем считать также заданным отображение $f:\wp(A)\to\wp(B)$, определяемое следующим образом: $\forall U\subseteq A\ f(U) =_{def} \{f(u)\mid u\in U\}$.
- Кроме бинарных операций объединения и пересечения классов, мы будем использовать также унарные операции, операндами которых являются классы.
- При отсутствии скобок операции (отображения) выполняются в порядке следования их идентификаторов слева направо. Исключение составляют арифметические и логические операции, для которых используется обычная система приоритетов: возведение в степень приоритетнее умножения, умножение приоритетнее сложения, отрицание приоритетнее конъюнкции, конъюнкция приоритетнее дизъюнкции и т.д.

□

2. Последовательности

Определение 2:

- Последовательностью длины $n\in\mathcal{N}_{0\infty}$ в алфавите C называют инъекцию $\sigma:[1..n]\to C$.
  - $|\sigma| = |Dom(\sigma)| = n$ – длина последовательности.
  - ϵ – пустая последовательность (нулевой длины).
- Множества последовательностей в алфавите C:
  - C^{∞} – бесконечные последовательности;
  - C^{*} – конечные ($n\neq\infty$) последовательности;
    будем считать, что всегда (в том числе для пустого C) $\epsilon\in C^{*}$.
  - $C^{n} =_{def} \{\sigma\in C^{*}\mid |\sigma|=n\}$ – последовательности длины $n\in\mathcal{N}_0$;
  - $C^{\omega} =_{def} C^{*}\cup C^{\infty}$ – конечные и бесконечные последовательности.
- Для конечного $I\subseteq\mathcal{N}$ обозначим ν_I последовательность, нумерующую элементы I в порядке возрастания:
  $$Dom(\nu_I)=[1..|I|]\ \&\ Im(\nu_I)=I$$
  $$\&\ \forall i,j\in Dom(\nu_I)\ (i<j \Rightarrow \nu_I(i)<\nu_I(j)).$$
- Пусть заданы предикат $p:Dom(p)\to Bool$ и функция $f:Dom(f)\to C$ с пересечением их доменов $Dom(p)\cap Dom(f)\subseteq\mathcal{N}$. Определим конструктор последовательности, состоящей из значений функции от возрастающих индексов, на которых функция и предикат определены, и предикат истинен (если таких индексов нет, конструируется пустая последовательность):
  $$\langle f(i)|p(i)\rangle =_{def} \{(i,f(j)) \mid i\in[1..|I|]\ \&\ j=\nu_I(i)\},$$

где $I = \boldsymbol{p}^{-1}(\boldsymbol{true}) \cap \boldsymbol{Dom}(\boldsymbol{f})$.

- Для $c_1, \ldots c_n \in C$: $\langle c_1, \ldots c_n \rangle =_{def} \sigma$ такая, что
  $\boldsymbol{Dom}(\sigma)=[1..n]$ и $\forall i \in \boldsymbol{Dom}(\sigma)\ \sigma(i)=c_i$.
- Для $\sigma \in C^\omega$ последовательность $\langle \sigma(i) | \boldsymbol{p}(i) \rangle$ называется *подпоследовательностью*.
- Для последовательности $\sigma \in C^\omega$ и множества A определим две подпоследовательности, называемые *проекциями*:
  $\sigma{\downarrow}A = \langle \sigma(i) | \sigma(i) \in A \rangle$,
  $\sigma{\uparrow}A = \langle \sigma(i) | \sigma(i) \notin A \rangle$.
- *Отрезком* последовательности называют подпоследовательность
  $\sigma[n..k] =_{def} \langle \sigma(i) | n \le i \le k \rangle$, где $n \in \mathcal{N}_0$ и $k \in \mathcal{N}_{0\infty}$.
  Отрезок называется *префиксом*, если $n \le 1$, *постфиксом*, если $k \ge |\sigma|$.
- Если задано отображение $\boldsymbol{f}: A \to B$, то, по умолчанию, будем считать также заданным отображение $\boldsymbol{f}: A^\omega \to B^\omega$, определяемое следующим образом: $\forall \sigma \in A^\omega\ \boldsymbol{f}(\sigma) =_{def} \langle \boldsymbol{f} \circ \sigma(i) | i \in [1..|\sigma|] \rangle$.
- Конкатенация последовательностей $\cdot\cdot : C^* \times C^\omega \to C^\omega$.
  Для $\mu \in C^*$, $\lambda \in C^\omega$: $\mu \cdot \lambda =_{def} \sigma$ такая, что
  $|\sigma| = |\mu| + |\lambda|$ и $\sigma[1..|\mu|] = \mu$, $\sigma[|\mu|+1..|\sigma|] = \lambda$,
  то есть μ префикс σ, а λ следующий за μ постфикс σ.
  Будем говорить, что λ *продолжает* (является *продолжением*) μ, μ *продолжается* (имеет *продолжение*) λ, μ *продолжается до* $\mu \cdot \lambda$.
- Как обычно, конкатенация распространяется на операнды, которые являются множествами последовательностей
  $\cdot\cdot : \wp(C^*) \times \wp(C^\omega) \to \wp(C^\omega)$:
  для $\mathbf{M} \subseteq C^*$, $\mathbf{\Lambda} \subseteq C^\omega$ $\mathbf{M} \cdot \mathbf{\Lambda} =_{def} \{\mu \cdot \lambda | \mu \in \mathbf{M}\ \&\ \lambda \in \mathbf{\Lambda}\}$;
- Для (не обязательно коммутативного) бинарного оператора $\cdot \otimes \cdot : C^* \times C^* \to C^*$ естественным образом определяется унарный оператор, применяемый к последовательности (конечных последовательностей, множеств конечных последовательностей и т.д.) $\otimes\cdot : C^{*\omega} \to C^\omega$, $\otimes\cdot : \wp(C^*)^\omega \to \wp(C^\omega)$ и т.д.
  При этом считается, что для одноэлементной последовательности $\oplus(\langle \sigma \rangle) = \sigma$, где $\sigma \in C^*$, а для пустой последовательности $\oplus(\epsilon) = \epsilon$.
  Таким оператором является, в частности, конкатенация $\cdot$.
- Для $\mu, \sigma \in C^\omega$: $\mu \le \sigma =_{def} \mu = \sigma[1..|\mu|]$, то есть μ префикс σ,
  $\mu < \sigma =_{def} \mu \le \sigma\ \&\ \mu \ne \sigma$.
  Очевидно, что отношение $\le$ является частичным порядком.
  Если σ бесконечна, то из $\mu < \sigma$ следует конечность μ.

3. Деревья последовательностей

Определение 3:

- *Деревом последовательностей* в алфавите C назовём такое множество последовательностей $\Sigma \subseteq C^{\omega}$, которое вместе с каждой последовательностью содержит любой её префикс: $\forall \sigma \in \Sigma \ \forall \mu \leq \sigma \ \mu \in \Sigma$. Такое множество последовательностей называют ещё замкнутым по взятию префикса (prefix-closed [Glab93]).[3]
- Дерево конечно, если оно конечно как множество (последовательностей).
- Дерево последовательностей будем называть *нётеровым*, если в нём нет бесконечно возрастающей цепочки последовательностей (в частности, бесконечной последовательности).
- Множество всех деревьев последовательностей в алфавите C обозначим $\boldsymbol{Trees}^{\omega}(C)$, множество всех деревьев *конечных* последовательностей в алфавите C обозначим $\boldsymbol{Trees}(C)$.
- Для последовательности $\sigma \in C^{\omega}$ множество её префиксов, очевидно, является деревом, и мы будем обозначать это множество (очевидно, включающее исходную последовательность)

  $\boldsymbol{Tree}(\sigma) =_{\text{def}} \{\mu \in C^{\omega} \mid \mu \leq \sigma\}$.

  Соответственно, пополнение произвольного множества последовательностей $T \subseteq C^{*}$ префиксами всех его последовательностей также является деревом $\cup_{\circ} \boldsymbol{Tree}(T)$.
- Для дерева $\Sigma \in \boldsymbol{Trees}^{\omega}(C)$:
  - *высотой* дерева будем называть длину его максимальной последовательности;
  - Σ *ограниченное*, если его высота конечна;
  - множество максимальных (по отношению $\leq$) последовательностей дерева:

    $\boldsymbol{max}(\Sigma) =_{\text{def}} \{\sigma \in \Sigma \mid \neg \exists \sigma` \in \Sigma \ \sigma < \sigma`\}$;
  - *началом* дерева назовём множество символов, с которых могут начинаться последовательности дерева:

    $\boldsymbol{head}(\Sigma) =_{\text{def}} \{c \in C \mid \langle c \rangle \in \Sigma\}$;
  - для $\sigma \in C^{*}$: $\Sigma \ \boldsymbol{after} \ \sigma =_{\text{def}} \{\lambda \in C^{\omega} \mid \sigma \cdot \lambda \in \Sigma\}$;
    очевидно, $\Sigma \ \boldsymbol{after} \ \sigma \in \boldsymbol{Trees}^{\omega}(C)$;

---

[3] Дерево последовательностей является частным случаем дерева как частично-упорядоченного множества, но не любое множество последовательностей, являющееся деревом во втором смысле, является деревом в первом смысле.

- подмножество $\mathbf{T}\subseteq\mathbf{\Sigma}$, являющееся деревом, то есть $\mathbf{T}\in \boldsymbol{Trees}^{\omega}(C)$, называется *поддеревом* дерева $\mathbf{\Sigma}$;
- $\mathbf{\Sigma}$ *конечно-ветвящееся*, если каждая его конечная последовательность продолжается конечным числом символов: $\forall\sigma\in\mathbf{\Sigma}\cap C^{*}\ \ |\boldsymbol{head}(\mathbf{\Sigma}\ \boldsymbol{after}\ \sigma)|\neq\infty$.
- $\mathbf{\Sigma}$ *перечислимо-ветвящееся*, если каждая его конечная последовательность продолжается перечислимым множеством символов: $\forall\sigma\in\mathbf{\Sigma}\cap C^{*}\ \boldsymbol{head}(\mathbf{\Sigma}\ \boldsymbol{after}\ \sigma)$ перечислимо.
- $\mathbf{\Sigma}$ *разрешимо-ветвящееся*, если каждая его конечная последовательность продолжается разрешимым (относительно алфавита) множеством символов: $\forall\sigma\in\mathbf{\Sigma}\cap C^{*}\ \boldsymbol{head}(\mathbf{\Sigma}\ \boldsymbol{after}\ \sigma)$ разрешимо относительно C.

□

4. Порождающий граф

Для наглядности мы будем использовать в примерах представление множества последовательностей порождающим графом. Порождающим графом называется ориентированный граф, дуги которого могут быть помечены символами алфавита. Допускаются непомеченные (пустые) дуги. В графе выделяют начальные и конечные вершины. Последовательность, порождаемая графом, – это последовательность символов алфавита, которыми помечены непустые дуги маршрута, начинающегося в начальной вершине и, если последовательность конечная, заканчивающегося в конечной вершине. Дерево конечных последовательностей может порождаться графом, в котором все вершины конечные и одна начальная; мы будем помечать её серым кружком (см. Рис.1).

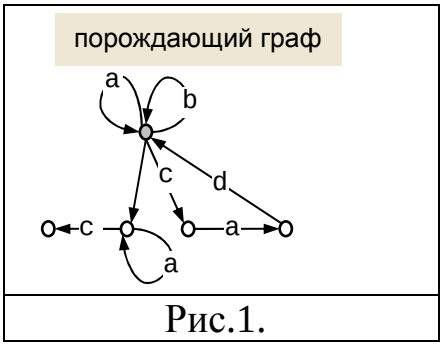


Рис.1.


ВВЕДЕНИЕ

Часть 1. Введение

Структура части:
1. Формализация понятия «правильности»
2. Формализация тестового эксперимента
3. Дивергенция и недетерминизм
4. Трассовая и автоматная модели
5. Тесты
6. Модели ввода-вывода
7. Проблемы взаимодействия тестера и реализации
8. Пополнение спецификации
9. Асинхронное тестирование и верификация композиции

Во Введении неформально вводятся базовые понятия и обсуждаются основные проблемы тестирования соответствия. Специально рассматриваются блокировки стимулов, разрушение и дивергенция.

Глава 1.1. Формализация понятия «правильности»

Структура главы:
1. Функциональность.
2. Взаимодействие.
3. Аналитическая верификация и тестирование.
4. Формальные спецификации.
5. Математическая модель.
6. Проблема извлечения модели.
7. Тестовые возможности и гипотезы о реализации.

Верификация программ – это проверка их правильности. То, правильность чего проверяется, называется реализацией. Что такое правильность? Идея теории соответствия простая и естественная: не бывает абсолютно правильной реализации, правильность понятие относительное и определяется сравнением с эталоном – спецификацией. Более строго, правильность трактуется как соответствие реализации функциональным требованиям, а спецификация понимается как описание этих требований.

Требования обычно записываются в виде формальных спецификаций на подходящем языке спецификаций или спецификационном расширении языка программирования. Считается, что спецификация описывает некоторую абстрактную систему – модель, и соответствие понимается как бинарное *отношение* «похожести» на классе моделей. Таким образом, целью верификации является проверка того, что пара реализация-спецификация связаны заданным отношением.

Отметим два ключевых слова: *функциональность* (требований) и *формальность* (спецификаций).

1.1.1. Функциональность.

Функциональность означает, что спецификация отвечает на вопрос «что делает программа», отвлекаясь от того «как она это делает».

В качестве примера можно привести спецификацию функции вычисления квадратного корня. Она может быть записана в двух формах:

$y^2=x$ или $y=$***алгоритм***$\_\sqrt{(x)}$.

Первую форму обычно называют *имплицитной*, а вторую – *эксплицитной*. Наиболее наглядно идею функциональности спецификации демонстрирует имплицитная форма. Спецификация утверждает, что реализация правильная, если значение, возвращаемое функцией, будучи возведённым в квадрат,

равно аргументу. Здесь нет никакого алгоритма вычисления квадратного корня, хотя именно по такому алгоритму должна работать реализация. Иными словами, спецификация говорит: не важно как это делает реализация, но результат должен быть вот таким.

Но даже если спецификация эксплицитна, и в ней записан тот же самый алгоритм вычисления квадратного корня, что и в тексте реализации, смысл этих записей разный. Почему? Потому что требованиям спецификации удовлетворяет не только та реализация, в которой используется такой же алгоритм, но и та, которая использует другой алгоритм, но возвращает такое же значение квадратного корня.

В этом месте *функциональность* – это слово, производное от слова *функция*. А математическая функция – это не то же самое, что алгоритм вычисления функции. Спецификация описывает саму функцию, неважно каким способом, а реализация выполняет алгоритм вычисления функции.

Поскольку противопоставление «что» и «как» относительно, функциональность также оказывается относительным понятием. Она зависит от того, что для нас важно в программе («что»), а что второстепенно («как»).

Характерным примером являются временны*е* характеристики реализации. Обычно они считаются нефункциональными. Нам не важно, сколько времени реализация вычисляет квадратный корень, лишь бы она делала это правильно. Но иногда время оказывается настолько важным, что ограничение на него следует рассматривать как функциональное требование. Отвлекаясь в сторону, скажем, что в этом случае используется, например, не обычная автоматная модель реализации, а так называемые временны*е* автоматы, которые умеют описывать время и, тем самым, позволяют генерировать тесты, проверяющие время исполнения.

Выбранная функциональность фиксирует уровень абстракции, при котором мы концентрируем внимание на одних (важных) свойствах программы и отвлекаемся от других (второстепенных) её свойств.

1.1.2. Взаимодействие.

В настоящей работе мы ограничиваемся такой функциональностью, когда всё «важное» может быть выражено в терминах внешнего поведения реализации («функциональность» от слова «функционирует»). Реализация правильна, если она правильно взаимодействует с окружением. Спецификация описывает требования, налагаемые на такое взаимодействие.

Теперь уже не любые требования функциональные в смысле *функции* остаются функциональными в смысле *взаимодействия*.

Вот лишь несколько примеров требований, которые часто предъявляют к программным продуктам, но которые не могут быть проверены при тестировании:

- лицензионная чистота,
- использование определённого языка программирования,
- хорошая самодокументированность программы.

Для функциональности, основанной на взаимодействии, становится возможным *тестирование* – проверка правильности реализации в *эксперименте* над нею, когда тестирующая система (*тестер*) подменяет собой окружение (или его часть). Этим тестирование отличается от *аналитической верификации*, основанной на методах доказательства правильности программы. Иногда тестирование называют *динамической верификацией*.

Широко распространённым частным случаем взаимодействия является обмен дискретными порциями информации (сообщениями) между реализацией и окружением. Такие системы называют *системами ввода-вывода*. Если сообщение передаётся из окружения в реализацию, оно называется *стимулом* (input); если сообщение передаётся в обратном направлении, из реализации в окружение, оно называется *реакцией* (output). Мы будем говорить, что окружение выдаёт или посылает стимул, а реализация принимает стимул. Соответственно, реализация выдаёт или посылает реакцию, а окружение принимает реакцию. Заметим, что при рассмотрении взаимодействия компонентов сложной системы сообщение, принимаемое одним компонентом, то есть его стимул, является одновременно сообщением, выдаваемым другим компонентом, то есть его реакцией.

1.1.3. Аналитическая верификация и тестирование.

Теоретически, аналитические методы имеют более широкую область применения, поскольку могут использоваться и тогда, когда нас интересуют свойства программы, не выражаемые в терминах взаимодействия. Однако на общей области применения (функциональность взаимодействия) тестирование имеет бо́льшую практическую применимость, поскольку тестирование требует меньше знаний о реализации, чем аналитическая верификация.

Аналитическая верификация оперирует на уровне некоторой абстрактной модели реализации. На практике такую модель часто можно получить только

из самой реализации, и такое извлечение модели из реализации становится первым этапом верификации. Другой вариант, когда модель реализации существует "сама по себе" без реализации (например, как её прототип). В обоих случаях под вопросом оказывается правильность соответствия реализации и её модели. Иными словами, фактически, мы верифицируем не реализацию, а её модель. Даже если ошибок не обнаружено в модели, это не означает, что их нет в реализации.

В отличие от этого, при тестировании реализация может рассматриваться как «чёрный ящик», о котором мы ничего не знаем, кроме интерфейса взаимодействия с ним и, быть может, некоторых гипотез, ограничивающих класс тестируемых реализаций.

Заметим, что при аналитической верификации и при тестировании не удаётся полностью избавиться от ручного труда. В обоих случаях без помощи человека невозможно извлечь эксплицитную модель спецификации из её имплицитного описания. В дальнейшем распределение ручного труда различно. Само доказательство правильности, как правило, не удаётся полностью автоматизировать: требуется помощь человека для формулировки промежуточных утверждений. При генерации тестов человек помогает выделять конечное подмножество тестов из, вообще говоря, бесконечного набора тестов, осуществляющего полную проверку.

1.1.4. Формальные спецификации.

Для чего спецификации должны быть *формальными*? Прежде всего, для того, чтобы требования, описываемые спецификацией, понимались однозначно: как человеком, так и компьютером. Можно выделить три области применения формальных спецификаций.

1) Для создания «правильных» реализаций: как инструкция разработчику программ или как входные данные для программ генерации программ.

2) Для верификации реализации и, особенно, для автоматической верификации с помощью компьютера: как при аналитических методах доказательства правильности, так и при генерации тестов.

3) Как инструкция, которой должны руководствоваться разработчики (или программы генерации) других программ, использующих данную.

Если спецификация неформальна, это часто служит источником недопонимания между спецификатором, формулирующим требования, и разработчиком, создающим реализацию. Или между разработчиком и тестировщиком. Или между заказчиком и тестировщиком (опережающее

создание тестов по спецификации, когда ещё нет реализации). И, наконец, между разработчиком системы и её пользователем.

В этой работе нас будет интересовать, прежде всего, вторая задача. Конечная цель здесь – автоматическая верификация. Поэтому независимо от того, применяются ли аналитические методы доказательства правильности, или генерируются тесты, спецификация должна быть достаточно формальной, чтобы её мог понимать компьютер. Как было сказано выше, верификация полностью автоматическая только в идеале, а на практике она всего лишь автоматизирована, то есть требует интеллектуальных усилий человека, хотя и не на всём пути, а лишь в некоторых критических точках. Поэтому для верификации также требуется понимание спецификации не только компьютером, но и человеком.

Что касается тестирования, то здесь можно выделить две вещи, которые создаются на основе спецификации: 1) *тесты*, которые должны проводить интересующие нас эксперименты над реализацией, и 2) *тестовые оракулы*, которые проверяют правильность поведения реализации в каждом таком эксперименте. Иногда выделяют третью составляющую, оценивающую *качество тестирования* или, другими словами, его *полноту*.

1.1.5. Математическая модель.

Для того чтобы спецификация была формальной, она должна быть сформулирована на формальном языке. В основе любого такого языка лежит математика как универсальный язык формализации. Иными словами, мы должны выбрать *математическую модель* спецификации и реализации, а также математически описать понятие правильности как соответствия между ними. Теория верификации соответствия базируется на выбранной математической метамодели, в терминах которой трактуются спецификация, реализация, их соответствие, тесты, тестовые оракулы, тестер и взаимодействие тестера с реализацией [ISO95].

Модель спецификации, или спецификационная модель, должна быть известна. Считается, что она задана самой спецификацией (на том или ином языке), то есть спецификация понимается как описание такой модели.

Понимание реализации как модели формулируется в виде *тестовой гипотезы* [Bern91]: «Существует такая (реализационная) модель, наблюдаемое поведение которой неотличимо от поведения реализации при любом взаимодействии с окружением в терминах выбранного уровня абстракции». С точки зрения тестирования это означает, что при любом тестовом эксперименте мы не можем различить, что находится в «чёрном

ящике»: реализация или её модель. Заметим, что существование реализационной модели не означает, что она известна.

Соответствие, тем самым, понимается как отношение между двумя моделями: спецификационной и реализационной. Такое соответствие – это математическое соответствие, то есть подмножество декартового произведения множества реализационных моделей и множества спецификационных моделей.

Тест также понимается как тестовая модель – модель тестера, которая генерируется по спецификационной модели и данному соответствию. Наблюдаемое во взаимодействии с тестером (в тестовом эксперименте) поведение реализации сравнивается с возможным поведением спецификации, то есть поведением спецификации, поставленной на место реализации.

В этой работе под спецификацией, реализацией и тестом мы будем, как правило, иметь в виду соответствующие модели.

1.1.6. Проблема извлечения модели.

Для верификации соответствия нужно, чтобы была явно задана математическая модель спецификации. К сожалению, на практике об этом можно только мечтать: существует проблема извлечения модели из того текста спецификации, который мы имеем. Здесь мы только обозначим эту проблему, а в остальной части работы будем предполагать, что эта проблема так или иначе решена.

Формальность спецификации, сама по себе, гарантирует лишь, что спецификационная модель задана, однако не обязательно явно. На практике спецификации пишутся не на языке математики, а на специальных языках спецификации или спецификационных расширениях языков программирования. И хотя в основе спецификации всегда лежит некоторая математическая модель, а спецификация понимается как описание этой модели, тем не менее, спецификация и спецификационная модель – не одно и то же.

Если спецификация имплицитна, например, определяется пред- и постусловиями, то модель задаётся, фактически, системой уравнений, которая не всегда может быть удовлетворительно решена алгоритмическим способом.

Может быть, для создания (человеком или компьютером) реализации по такой спецификации о спецификационной модели можно не вспоминать, но

методы проверки правильности (как доказательства, так и тестирования) опираются как раз на математическую модель.

Нужно также учесть, что одной и той же спецификации может соответствовать несколько моделей разных типов. В известном смысле модель – это способ математической интерпретации спецификации. Вообще говоря, человек, который пишет спецификацию, обычно имеет в голове модель некоторого типа, хотя часто это бывает неосознанно, а спецификация получается осмысленной просто потому, что хорошо продуман язык спецификации, не позволяющий человеку писать полную ерунду. Иными словами, тип модели заложен в самом языке спецификации. Однако в основе языка не обязательно лежит единственный тип модели. Чем язык универсальнее, тем больше свободы он предоставляет в выборе типа модели, которая подразумевается при создании спецификации.

Здесь приходится балансировать, если можно так выразиться, между желаниями человека и компьютера (или другого человека). Тот, кто пишет спецификации, хочет иметь универсальный язык, хотя бы потому, что ему лень изучать много разных языков. Ещё он хочет, чтобы спецификация была ему самому понятной. А для автоматического извлечения модели желательно, чтобы тип этой модели как можно более явно отражался в конструкциях языка, делая его, тем самым, менее универсальным. И здесь важна не понятность для человека, а формальное соответствие типу модели. На практике часто приходится делать выбор между понятностью спецификации и явностью отражения в ней модели. И всё равно самый первый этап генерации тестов делается вручную как раз потому, что на этом этапе извлекается модель. Кроме того, эта модель ещё и преобразуется для удобства тестирования (для того, чтобы полный тестовый набор был конечным). Это, так называемый, процесс факторизации модели, который мы в данной работе не рассматриваем.

В ИСП РАН разработан метод автоматического извлечения тестовой модели из спецификации в процессе тестирования. Спецификация предполагается заданной пред- и постусловиями, хотя важно здесь лишь то, что по такой спецификации можно генерировать тестовые оракулы и итераторы стимулов, определённых в состояниях спецификации. Метод основан на обходе графа переходов, общих в реализации и спецификации [ВКК00, ВКК03, ВКК03-1, КРКВ03].

Напомним, что для методов аналитической верификации, в отличие от тестирования, требуется явное задание не только спецификационной, но и реализационной модели. Поэтому возникает дополнительная проблема: как извлечь реализационную модель из «текста» реализации? А эта проблема на порядок сложнее. Понятно, что если у нас нет исходного кода реализации, то

извлекать модель просто не из чего, и проблема становится неразрешимой. Однако даже если такой программный код есть, он совсем не похож на код спецификации. Это и понятно: спецификация специально предназначена для описания функциональных требований, а реализация пишется на языке программирования, и её задача – «разворачивать» эти требования в конкретные алгоритмы, структуры данных и т.п. Иными словами, код реализации гораздо больше «замусорен» второстепенными деталями, чем код спецификации.

Это оказывается одной из главных причин, по которой область применения тестирования гораздо шире, чем область применения аналитической верификации. Правда, у этой медали, как обычно, есть и другая сторона: доказательство, если оно возможно, как правило, даёт гарантированно правильный результат за конечное время, в то время как тестирование – это процесс, про который почти никогда нельзя сказать, что оно закончено. Эту проблему мы рассмотрим позже.

1.1.7. Тестовые возможности и гипотезы о реализации.

Математическая модель – это абстракция, но абстракция полезная, при которой мы отвлекаемся от того, что считаем второстепенным (вопрос *как*?) и сосредоточиваем своё внимание на существенном (вопрос *что*?). Типов моделей может быть много, и выбор нужной модели – дело чрезвычайно важное.

Точно так же не любое математическое отношение пригодно в качестве соответствия, определяющего правильность реализации. В данном исследовании мы ограничиваемся такими отношениями, которые могут быть проверены в тестовом эксперименте. В литературе описаны десятки таких отношений, предлагаемых для тестирования [Glab93]. Выбор той или иной модели и того или иного отношения для практического тестирования определяется общими представлениями о "правильности", тестовыми возможностями и гипотезами о реализации.

Общие представления определяют трактовку самой спецификации: какое поведение реализации спецификация «требует», какое «разрешает», а к какому «безразлична».

Тестовые возможности – это возможности по управлению реализацией и наблюдению её поведения, которые определяются интерфейсом между тестером и реализацией. Тестовые возможности, в свою очередь, определяют класс отношений, которые могут быть проверены при таком тестировании.

Реализационные гипотезы – это гипотезы о реализации, дополнительно ограничивающие тестируемые реализации некоторым подклассом. Это даёт возможность применять методы тестирования соответствия, которые в общем случае не работают. В известном смысле гипотезы о реализации дополняют тестовые возможности, оставляя только те реализации, для которых эти возможности оказываются достаточными для проверки соответствия. Такие предположения не проверяются при тестировании, лишь в некоторых случаях они могут контролироваться, да и то частично. Реализационная гипотеза – это предусловие тестирования.

Глава 1.2. Формализация тестового эксперимента

Структура главы:
1. Машина тестирования.
2. Разрешение тупиков (θ-наблюдение). Отказы.
3. Разрушение (forbidden action).
4. Приоритеты.

1.2.1. Машина тестирования.

Теперь нам нужно формализовать само понятие тестового эксперимента. Для этого используется термин *сценарий тестирования*, то есть формальное описание того, как происходит взаимодействие теста и реализации. На самом деле такое описание ещё не вполне математическое, это как бы мостик между интуитивным представлением о том, что такое взаимодействие, и математическим формализмом. Поэтому, на первый взгляд, такое описание выглядит немного смешно, но оно очень полезно, если мы хотим понимать, что мы имеем в виду, употребляя слово «взаимодействие». Инструментом описания здесь служит, так называемая, *машина тестирования*. Такая машина формализует важную часть тестирования, которая, как мы уже говорили, во многом определяет то соответствие между реализацией и спецификацией, которое мы можем выбирать. А именно – *тестовые возможности*: возможности *по наблюдению* за поведением реализации и возможности *по управлению* этим поведением.

Машина тестирования представляет собой чёрный ящик, внутрь которого помещена реализация. Реализация нам не видна, но ящик снабжён разного рода индикаторами, дисплеем или печатающим устройством для наблюдения и кнопками или переключателями для управления. Тем самым, фиксируется *интерфейс* между реализацией и тестом.

Наблюдаемое поведение предполагается дискретным и может рассматриваться как последовательность *внешних, наблюдаемых действий* из некоторого алфавита. Кроме наблюдаемых действий, машина может иметь *внутреннюю активность*, то есть внутренние, ненаблюдаемые (и, тем самым, неразличимые одно от другого) действия. Набор возможных наблюдений как раз и определяет тестовые возможности по наблюдению.

Если тестирование ограничивается только наблюдением, то его называют *мониторингом* или *пассивным тестированием*. Оно тоже полезно, но мы будем рассматривать общий случай тестирования, которое позволяет как-то управлять поведением реализации.

Для управления машина снабжается клавиатурой: набором кнопок, разрешающих те или иные внешние действия машины. Внутренняя активность обычно считается всегда разрешённой. В этой работе мы предлагаем машину, в которой каждая кнопка разрешает некоторое непустое подмножество внешних действий, но нажимать можно одновременно только одну кнопку. Набор кнопок машины как раз и определяет тестовые возможности по управлению. Такую машину мы называем *машиной с ограниченным управлением*. Под ограничениями здесь имеются в виду ограничения, налагаемые на возможные действия оператора (набор кнопок). Она является развитием реактивной машины, введённой Milner [Miln81], и генеративной машины, предложенной Glabbeek [Glab90,Glab93].

Результат любого тестового эксперимента сводится к последовательности нажимаемых кнопок управления и наблюдаемого ответного поведения машины. Такую последовательность мы называем *(обобщённой) трассой (наблюдений)*[4]. Пока мы рассмотрели только один вид наблюдений – наблюдения внешних действий. Теперь рассмотрим другие наблюдения: отказы и разрушение.

1.2.2. Разрешение тупиков (θ-наблюдение). Отказы.

При работе машины тестирования возможно возникновение *тупиков* (deadlock), когда 1) внешние действия, разрешаемые нажатой кнопкой управления, машина выполнить не может, а внешние действия, которые она может выполнить, не разрешены, и 2) машина не имеет внутренней активности.

Для обнаружения тупика и его разрешения (продолжения тестирования после тупика) вводится специальное *θ-наблюдение*, которое происходит тогда и только тогда, когда невозможно (нет и не будет) наблюдение никакого внешнего действия [Tret96].

На практике θ-наблюдение может быть реализовано как код ответа «не выполнено никакого действия», а в случае отсутствия такой возможности, как *тайм-аут* в тестере. Концепция тайм-аута основана на допущении, что любое внешнее действие и любая конечная последовательность внутренних действий системы выполняются за время, не превосходящее некоторое фиксированное T (тайм-аут равен $2T$). Дополнительно предполагается, что не бывает бесконечной последовательности внутренних действий (*дивергенции*). Если допущение тайм-аута не верно, можно предложить

[4] Обычно в трассе нажимаемые кнопки не учитываются. Однако этого достаточно только для машин без приоритетов (см. ниже).

более слабое допущение: *одно* действие (как внешнее, так и внутреннее) системы выполняется за время, не превосходящее `T`. Ограничивая длину последовательности внутренних действий тестируемых реализаций числом `N` и устанавливая тайм-аут `(N+1)T`, мы можем обнаруживать и разрешать тупик в *этом* классе реализаций.

θ-наблюдение является, на самом деле, классом дополнительных наблюдений – *отказов* (refusal sets). Интерпретация θ-наблюдения как отказа определяется нажатой кнопкой: отказ – это множество внешних действий, разрешаемых нажатой кнопкой, при θ-наблюдении, то есть когда внутренней активности нет, а все разрешённые внешние действия машина выполнить не может. Соответственно, трассы наблюдений могут содержать не только внешние действия, но и отказы.

Мы будем считать, что тайм-ауты встроены в кнопки управления, хотя, может быть, не во все кнопки. Срабатывание тайм-аута приводит к θ-наблюдению (наблюдению тупика) и соответствующему отказу.

Иногда рассматривают машины, в которых возможно лишь *однократное* (в данном сеансе тестирования) наблюдение тупика: при тупике кнопка остаётся нажатой и заблокирована машиной. В этом случае отказ может быть только последним элементом трассы: трассы после отказа не имеют продолжений.

В данном исследовании, как правило, мы будем рассматривать два вида отказов: блокировки стимулов (данный стимул не принимается) и стационарность (отсутствие реакций). Прагматику этих отказов мы обсудим подробнее ниже.

1.2.3. Разрушение (forbidden action).

В этой работе нововведением является понятие *разрушения*. Под разрушением системы мы понимаем любое нежелательное её поведение. Семантика разрушения включает в себя и реальное разрушение системы, которого следует избегать при тестировании.

Иногда при обмене стимулами и реакциями такое разрушение предполагается возможным в результате приёма неспецифицированного стимула (см. для автоматов Мили [BP94,LY96]). В то же время исключение из рассмотрения разрушающих стимулов (стимулов, после приёма которых возможно разрушение) часто мотивируют тем, что реализация после приёма стимула должна проверять его корректность [Tret96,Heer98]. Если стимул некорректен, реализация либо просто игнорирует его, либо сообщает об ошибке (ответной реакцией). Такое требование к реализации естественно,

если это система «общего пользования»: в ней должна быть предусмотрена «защита от дурака».

Однако часто требуется тестировать внутренние компоненты или подсистемы, доступ к которым строго ограничен и взаимные проверки корректности обращений излишни. Такое часто встречается для стимулов со сложной внутренней структурой и нетривиальными условиями корректности, когда накладные расходы на проверку неоправданно увеличивают трудозатраты на создание системы, её объём и время выполнения. Альтернативой в этом случае является строгая спецификация предусловий взаимодействующих компонентов. Тестированию подлежит не поведение компонентов в ответ на некорректные стимулы, выдаваемые (как реакции) другими компонентами, а правильность обращения компонентов друг к другу. Последнее означает, фактически, проверку поведения каждого компонента (по его постусловию) только для таких стимулов, которые удовлетворяют предусловию компонента.

В настоящей работе мы допускаем разрушение не только после приёма стимула, но и после выдачи реакции. В последнем случае тестер не должен принимать реакции, если это может привести к разрушению.

Разрушение считается условно-наблюдаемым поведением: в отличие от внутренней активности оно может быть наблюдаемо, но при тестировании мы должны избегать такого наблюдения. В этом смысле наблюдаемость разрушения – чистая абстракция, смысл которой – обозначить нежелательное поведение как разрушение и избегать его.

Итак, мы вводим в рассмотрение новый вид трасс – трассы, которые могут заканчиваться разрушением. Мы будем обозначать разрушение символом γ. Предполагается, что продолжение после разрушения невозможно или недостоверно.

1.2.4. Приоритеты.

Говорят, что в реализации есть приоритеты, если выполнение внешнего действия зависит от того, какая нажимается кнопка, разрешающая это действие. Иными словами, выполнимость действия зависит от того, какие ещё внешние действия разрешает нажатая кнопка. Разумеется, приоритеты возможны, если одно и то же действие может разрешаться несколькими кнопками, на которых написаны разные множества действий.

Важно отметить, что сами приоритеты, если они есть, описываются в спецификации. Машина тестирования лишь формализует тестовую возможность наблюдения: обнаружение нарушения (или соблюдения)

приоритетов. Если такой возможности нет, то приоритеты ненаблюдаемы и тестируемое соответствие не может быть построено на них.

Другим видом приоритетов является приоритет внешнего действия над внутренней активностью. Тем самым, внутренняя активность может быть теперь не всегда разрешена: при нажатии некоторых кнопок она может запрещаться.

Последний вид приоритета решает проблему дивергенции: дивергенция может допускаться, если внешние действия имеют приоритет над внутренней активностью. Приоритет одних внешних действий над другими также часто используется на практике: например, приказ «*выполнить операцию x*» может запустить целую серию действий, а приказ «*отменить операцию x*» должен прервать эту серию в любой точке, то есть он должен быть более приоритетным, чем любое из действий в серии.

К сожалению, приоритеты не отражаются в современной теории тестирования. В обычно используемых моделях таких приоритетов нет, и отмена приказа выполняется равновероятно с продолжением выполнения приказа до самого конца независимо от того, в какой момент времени пришёл приказ об отмене. В ещё большей степени это относится к взаимным приоритетам внешних действий и внутренней активности: предполагается, что внутренняя активность может выполняться независимо от нажатия или не нажатия каких бы то ни было кнопок. Тот же van Glabbeek предполагает, что внутреннее действие всегда разрешено, а разрешение внешнего действия a определяется только переключателем “a” и не зависит от положения переключателя “b”[5]. Иными словами, нет никакого встроенного механизма приоритетов действий по отношению друг к другу, на что van Glabbeek`y в устном сообщении указал Jan Bergstra [Glab93].

В ИСП РАН разработан ряд моделей с приоритетами для выхода из цикла внутренней дивергенции, для выхода из цикла осцилляции (бесконечной цепочки реакций), для первоочередного приёма стимула или выдачи реакции [BKK03].

В настоящей работе мы рассматриваем реализации без приоритетов. Для такой реализации из трассы наблюдений можно удалить все кнопки, поскольку наблюдение внешнего действия не зависит от того, какая кнопка, разрешающая это действие, нажималась, а наблюдение отказа однозначно

[5] В генеративной машине van Glabbeek`a каждому внешнему действию соответствует один переключатель, разрешающий или запрещающий это действие, но в положение «разрешено» могут быть поставлены несколько переключателей.

определяет эту кнопку (отказ – это множество действий, написанное на кнопке).

Глава 1.3. Дивергенция и недетерминизм

Структура главы:
1. Проблема дивергенции.
2. Проблема недетерминизма реализации.
3. Недетерминизм спецификации.

1.3.1. Проблема дивергенции.

Как было сказано выше, наблюдаемость отказов предполагает отсутствие *дивергенции* – бесконечной цепочки внутренних действий. Понятно, что это является ограничением на тестируемую реализацию и должно быть чётко сформулировано в виде реализационной гипотезы. Это ограничение не означает, что мы не можем *определить* дивергенцию, а только то, что мы не можем *различить* дивергенцию как бесконечную внутреннюю активность и отказ как отсутствие вообще всякой активности.

Поэтому в настоящей работе мы будем рассматривать дивергенцию как нежелательное поведение и моделировать её разрушением.

Тем не менее, отметим, что проблема дивергенции не так проста, как может показаться. Дело в том, что дивергенция – это вовсе не обязательно ошибка «зацикливания» программы.

Мы уже говорили, что при тестировании тест может подменять собой не всё окружение, а только его часть. В этом случае дивергенция может быть бесконечным взаимодействием реализации с остальной частью окружения. А такое взаимодействие может быть не только не ошибочное, но, напротив, предписываемое требованиями поведение реализации. Например, другая часть окружения имеет просто б*о*льший приоритет, чем тест, и реализация, учитывая приоритеты, обрабатывает в первую очередь неиссякающий поток обращений от более приоритетных клиентов. Эта ситуация весьма характерна для фоновых тестов, которые не только не должны нарушать работу системы, но и не должны существенно снижать её производительность.

Конечно, идеальным решением проблемы был бы полный перехват взаимодействия реализации с окружением. Чтобы не нарушать работу, тест только фиксировал бы взаимодействие с другой частью окружения и прозрачно пропускал бы это взаимодействие через себя. К сожалению, часто полный перехват взаимодействия с окружением оказывается невозможным, а если и возможным, то экономически невыгодным – на разработку перехватчиков всех видов взаимодействия большой реализации, например, операционной системы, требуется много труда.

Но даже если мы будем рассматривать только «внутреннюю» дивергенцию, без взаимодействия с окружением, то и такое зацикливание не обязательно ошибка. На самом деле проблема ведь не в том, что машина может долго или даже бесконечно долго думать о чём-то своём, машинном, а в том, что она предаётся пустым размышлениям тогда, когда извне от неё требуется выполнение обязательных действий. Если каждый раз, когда такое требование возникает, машина прерывает свои размышления и делает то, что от неё требуется, то это то поведение, какое надо, и наличие дивергенции не является ошибкой. Характерный пример – сборка мусора или статистики, которой реализация может заниматься, сколько её душе угодно, но только не тогда, когда поступает запрос на выполнение работы.

Как уже было сказано выше, такого рода свойства реализации могли бы описываться с помощью приоритетов. В настоящей работе мы ограничиваемся системами без приоритетов и поэтому будем рассматривать дивергенцию как нежелательное поведение, то есть моделировать её разрушением. Таким образом, мы ослабляем требование отсутствия дивергенции (конвергентность): дивергенция может быть, но при тестировании мы будем её избегать. Соответственно, и отношение конформности допускает (тест не проверяет) любое поведение в тех ситуациях, когда возможна дивергенция.

1.3.2. Проблема недетерминизма реализации.

В машине тестирования возможен *недетерминизм*: в одной и той же ситуации при нажатии одной и той же кнопки могут наблюдаться разные внешние действия машины или отказ. Иными словами трасса реализации может продолжаться различными внешними действиями, разрешаемыми данной нажимаемой кнопкой, или отказом. Такой недетерминизм называется наблюдаемым. Недетерминизм можно рассматривать как результат абстрагирования от не учитываемых внешних факторов – «*погодных условий*» [Glab93], которые определяют тот или иной выбор. Предполагается, что для каждого выбора существуют погодные условия, определяющие именно этот выбор.

Недетерминизм рождает сложную проблему полноты тестирования. В общем случае мы никогда не можем быть уверены, что провели тестовые испытания для всех возможных погодных условий. Следовательно, мы не уверены, что в реализации нет ошибки, проявляющейся при тех погодных условиях, при которых мы не прогоняли тот или иной тест. Возможны различные решения этой проблемы.

Одно из них – специальные тестовые возможности по управлению погодой. Для этого мы должны выйти за рамки модели, которая как раз и абстрагировалась от второстепенных деталей внешних факторов, то есть от погоды. Тем самым, тестирование становится зависящим не только от спецификации, но и от реализационных деталей, от того, что можно назвать операционной обстановкой, в которой работает реализация. Для каждого варианта такой операционной обстановки мы будем вынуждены создавать свой набор тестов. Тем не менее, в некоторых частных случаях на этом пути можно получить практические выгоды.

Другое решение – специальные реализационные гипотезы о том, что, если реализация ведёт себя правильно при одних погодных условиях, то она ведёт себя правильно при остальных погодных условиях. Здесь можно вводить эквивалентность погодных условий и осуществлять тестирование на базе факторизации погодных условий по этому отношению эквивалентности [BKK04]. Разумеется, такой подход правомерен, если соответствующие реализационные гипотезы обоснованы.

Третье решение основано на том, что нам известно распределение вероятностей тех или иных погодных условий. В этом случае тестирование оказывается полным с той или иной вероятностью [BGNV05].

Близкое к этому четвёртое решение предполагает, что в каждой ситуации (после трассы) возможно лишь конечное число погодных условий (с точностью до эквивалентности) и существует такое число N, что после N прогонов теста гарантированно будет проверено поведение реализации при всех возможных в этой ситуации погодных условиях [Miln80, FB92].

Наконец, существует и более радикальное решение – просто запретить недетерминизм реализации. То есть реализационная гипотеза ограничивает класс реализаций только детерминированными реализациями. При всей своей наивности, это достаточно распространённый практический приём [PYB96]. Обоснованием может служить то, что во многих случаях заранее известно, что интересующие нас реализации детерминированы.

В настоящей работе мы не касаемся этих вопросов, предполагая, что выбрано то или иное решение на более высоком (ближе к практике) уровне.

1.3.3. Недетерминизм спецификации.

Даже если мы требуем детерминизма реализации, отсюда не следует детерминизм спецификации. Например, спецификация квадратного корня в имплицитной форме $y^2=x$ ничего не говорит о знаке возвращаемого значения, допуская для аргумента $x=4$ как $y=+2$, так и $y=-2$.

Эксплицитная спецификация, если она не хочет вводить дополнительного ограничения, делает это ещё более явно с помощью дизъюнкции: $y=+$***алгоритм***$\_\sqrt{(x)} \vee y=-$***алгоритм***$\_\sqrt{(x)}$. Таким образом, спецификация недетерминирована, однако, этот недетерминизм не обязательно понимать так, что реализация недетерминирована. Мы можем потребовать дополнительно, чтобы реализация была детерминированной, однако спецификация всё равно останется недетерминированной.

Это происходит потому, что спецификация описывает, фактически, не одну реализацию, а класс всех реализаций, в данном случае класс всех детерминированных реализаций, удовлетворяющих сформулированным требованиям. Дизъюнкция не означает, что реализация должна уметь возвращать при аргументе 4 как +2, так и –2. Она всегда может возвращать что-то одно, но только не 3 и не 5.

Детерминизм или недетерминизм спецификации определяется формой требований, которые она предъявляет реализации. В общем случае реализация и спецификация связаны отношениями *may* & *must*, *может* и *должна*. Реализация *должна* выполнять только те действия, которые разрешены спецификацией, но *может* выполнять лишь некоторые действия из списка возможных вариантов, предлагаемых спецификацией. Грубо говоря, спецификация недетерминирована, если этот список состоит более чем из одного элемента. В нашем примере квадратного корня список состоит из двух результатов +2 и -2.

Этот пример показывает различное отношение к стимулам и реакциям. Обычно считается, что, если спецификация определяет приём (не блокировку) нескольких стимулов (1,2,3,4,5,...), то реализация *должна* принимать каждый стимул, однако, если спецификация определяет (например, для данного стимула 4) выдачу несколько реакций (+2 или -2), то реализация *может* выдавать лишь некоторые из них.

В нашей практике был случай далеко не такой тривиальный, как квадратный корень. Это была одна из программ распределения памяти – буфер для строк переменной длины. Написать спецификацию этой программы детерминированным образом было невозможно, не вводя лишних, то есть не функциональных, ограничений. Однако реализация, естественно, предполагалась детерминированной, как оно на самом деле и было. Более того, алгоритм тестирования был способен тестировать только детерминированные реализации. При этом он мог иногда обнаруживать проявление недетерминизма, и, если это случалось, фиксировалась ошибка. И всё прекрасно работало, даже была найдена какая-то ошибка.

Глава 1.4. Трассовая и автоматная модели

Структура главы:
1. Трассовая модель.
2. Автоматная модель.

1.4.1. Трассовая модель.

Машина тестирования определяет самую первую (наиболее абстрактную) математическую модель, которую можно использовать для соответствия, – трассовую модель. Трассовая модель реализации – это множество всех трасс, которые можно получить во всех тестовых экспериментах над данной реализацией (помещённой внутрь машины). Спецификация также является множеством трасс, а соответствие – это отношение между множествами трасс.

Возможные трассы определяются, во-первых, возможными наблюдениями и, во-вторых, наличием или отсутствием приоритетов. Если приоритеты есть, трасса должна включать нажимаемые кнопки для того, чтобы можно было проверять соблюдение реализацией приоритетов, требуемых спецификацией. Если приоритетов нет, трассы содержат только наблюдения некоторых типов.

van Glabbeek описал 30 таких типов наблюдений и 155 основанных на них соответствий [Glab93]. Это уже большой прогресс по сравнению с необъятным числом математических соответствий как подмножеств декартового произведения. Правда, van Glabbeek не рассматривал машины с разделением действий на ввод и вывод, а здесь есть свои особые соответствия, к которым мы ещё вернёмся.

Не все из рассматриваемых van Glabbeek`ом наблюдений легко реализуемы на практике, а про некоторые можно сказать, что их практическая реализация невозможна. Тем не менее, польза от них не только теоретическая. Хотя они не наблюдаемы на практике, зато позволяют ввести нужные ограничения на реализацию, чтобы тестирование было достоверным. Характерным примером является наблюдение дивергенции.

В общем случае выбор набора типов наблюдений определяется тестовыми возможностями и реализационными гипотезами. В каждом конкретном случае нужно добиться ясного понимания того, что мы можем делать при тестировании, и какими могут быть тестируемые реализации.

Набор типов наблюдений определяет алфавит – множество наблюдений этих типов. В этом алфавите рассматриваются трассы как последовательности

наблюдений. Поскольку всё, что мы имеем от эксперимента – это трассы наблюдений, нас будут интересовать только такие соответствия между реализацией и спецификацией, которые могут быть сформулированы в терминах трасс наблюдений. Более конкретно: сравниваются множество трасс наблюдений, которые могут быть получены в экспериментах над реализацией, и множество трасс наблюдений, задаваемое спецификационной моделью. Тем самым, трассовая модель достаточна для задания соответствия и генерации тестов.

Отметим, что мы не налагаем никаких ограничений на число символов алфавита и число трасс в модели. Они могут быть бесконечными и даже неперечислимыми. Точно также мы не ограничиваем длину трасс в модели. Такого рода ограничения нам понадобятся только при рассмотрении вопросов алгоритмизации: алгоритмического задания модели и генерации тестов.

Мы ввели машину тестирования с ограниченным управлением и разрушением. Ограниченное управление означает, что набор кнопок может быть не полон (не все возможные непустые подмножества внешних действий), и/или тайм-ауты могут быть встроены не во все кнопки. В терминах наблюдаемых действий ограниченное управление означает ограничение на отказы, которые может выдавать машина. Тем самым, множество возможных соответствий существенно увеличивается по сравнению с классификацией van Glabbeek`a.

Манипулируя набором кнопок и встроенными тайм-аутами, то есть определяя те или иные тестовые возможности, мы можем получить много разных семантик соответствующих трасс.

Например, если разрушения нет, а единственная кнопка разрешает все внешние действия и в неё не встроен тайм-аут (отказов нет), мы имеем трассовую семантику (trace semantics). Если для такой же машины имеется встроенный тайм-аут, мы получаем семантику завершённых трасс (completed trace semantics). Если разрушения и ограничений нет (возможны любые отказы), мы имеем семантику трасс с отказами (failure trace semantics). Для однократных отказов получается failure semantics. (См. [Glab90].)

Если та или иная семантика описывается с помощью трассовой модели, то такое описание называется ещё *явной моделью* (explicit model) [Glab90].

В настоящей работе мы вводим соответствие $\boldsymbol{ioco}_{\beta\gamma\delta}$, которое является обобщением известного соответствия ***ioco***, предложенного Jan Tretmans [Tret96]. Но об этом подробнее речь пойдёт ниже.

1.4.2. Автоматная модель.

Трассовая модель достаточна для описания соответствия и генерации тестов. Однако она не даёт возможности описывать композицию систем, составленных из нескольких взаимодействующих компонентов. Для этого используется более «подробная» автоматная модель, основанная на понятии состояния. Другое название: *система размеченных переходов – Labelled Transition System* (LTS).

В случае разделения внешних действий на стимулы и реакции иногда говорят об *Input-Output Labelled Transition System* (IOLTS) [JJTV99]. В литературе можно встретить и другие названия: *Input-Output Transition System* (IOTS) [Tret96], *Input-Output Automaton* (IOA) [LT87], *Input-Output State Machines* (IOSM) [Phal94] и т.п. Различия между этими понятиями маргинальны.

LTS определяется в том или ином алфавите – множестве внешних действий. Состояние предназначено для описания того, какие внешние и внутренние действия возможны в системе в тот или иной момент времени. Состояние является результатом абстрагирования от деталей внутреннего устройства и памяти системы: внимание концентрируется только на действиях, возможных в текущем состоянии, и на том, каким будет следующее состояние после того или иного возможного действия. В каждом состоянии определяются переходы в другие состояния, помеченные символами внешних действий или специальным символом τ для внутреннего действия. Внутренняя активность понимается как цепочка внутренних действий.

Отметим, что мы не налагаем никаких ограничений на число символов алфавита, число состояний и переходов в автоматной модели. Они могут быть бесконечными и даже неперечислимыми. Этим LTS отличается от обычных конечных автоматов (FSM – Finite State Machine). Некоторые ограничения будут нужны только при рассмотрении вопросов алгоритмизации: алгоритмического задания модели, генерации тестов и др.

Отказы могут наблюдаться только в *стабильных* состояниях – состояниях, в которых нет внутренней активности, то есть τ-переходов. Стабильное состояние s реализует отказ A (подмножество алфавита внешних действий), если ни по одному действию $a \in A$ нет переходов из s.

Говорят, что LTS полностью определена по стимулам (input-enabled), если в каждом её достижимом стабильном состоянии определён переход по

каждому стимулу. Под IOTS или IOA чаще всего понимают как раз такие LTS. В общем случае в LTS могут быть блокировки стимулов.

Между автоматной и трассовой моделями есть чёткая связь. Для LTS определяется множество её трасс, которое как раз и является соответствующей трассовой моделью. В этом смысле LTS можно рассматривать как неявную модель той или иной семантики, явной моделью которой является соответствующая трассовая модель.

Трассы LTS вычисляются по маршрутам – цепочкам переходов LTS. Каждый раз, когда проходится переход по внешнему действию a, в трассу помещается символ a; τ-переходы ничего не добавляют в трассу. Каждый раз, когда, двигаясь по маршруту, мы проходим стабильное состояние s, в трассу могут помещаться (но не обязательно!) отказы, реализуемые в s (добавлять можно несколько одинаковых или разных отказов). Важно отметить, что отказ не изменяет состояние: его можно понимать как виртуальный переход-петлю в состоянии, помеченный символом отказа, то есть множеством внешних действий.

Добавляя разрушение, мы обозначаем его символом γ, который добавляется в алфавит действий. Символ γ помещается в трассу при проходе по переходу, помеченному этим символом, аналогично переходу по внешнему действию с той лишь разницей, что на этом формирование трассы всегда заканчивается. Поскольку мы моделируем дивергенцию разрушением, трасса, заканчивающаяся в дивергентном состоянии, то есть в таком, в котором начинается бесконечная цепочка τ-переходов, продолжается символом γ независимо от того, есть в этом состоянии γ-переход или нет.

Композиция LTS определяется с помощью оператора параллельной композиции алгебры процессов. В литературе используются две основные нотации: CSP – Communicating Sequential Processes [Hoare85] и CCS – .

В CSP алфавит композиции двух LTS – это объединение алфавитов операндов, а состояния – это пары состояний операндов. Переходы делятся на асинхронные и синхронные. Синхронный переход – это переход по внешнему действию a из пересечения алфавитов (синхронное действие), которому соответствует пара переходов по a в операндах. Асинхронный переход – это либо τ-переход в одном из операндов, либо переход по действию a из разности алфавитов в соответствующем операнде (асинхронное действие). При синхронном переходе, вообще говоря, оба операнда меняют свои состояния, а при асинхронном переходе – только один. В дальнейшем применяется операция ***Hide***, «скрывающая» синхронные переходы, то есть превращающая их в τ-переходы.

В настоящей работе мы используем нотацию CCS, которая применяется к моделям ввода-вывода: стимулы обозначаются с префиксом вопросительный знак “?a”, реакции – с префиксом восклицательный знак “!a”. Синхронный переход сразу делается τ-переходом, но здесь он соответствует паре переходов в операндах по «противоположным» действиям: по стимулу ?a в одной LTS и по противоположной реакции !a в другой LTS. Асинхронный переход – это либо τ-переход в одном из операндов, либо переход по стимулу/реакции ?a/!a в одном операнде, для которого в алфавите другого операнда нет противоположного символа !a/?a.

Поскольку мы добавляем в LTS переходы по разрушению γ, мы должны доопределить параллельную композицию для таких γ-переходов. Будем считать, что они выполняются асинхронно так же, как τ-переходы. Такой оператор композиции мы обозначаем ⫴.

Функционирование композиционной системы слагается из внутренней работы одного из компонентов (асинхронные τ-переходы), взаимодействия компонентов между собой (синхронные τ-переходы) и с окружением (асинхронные переходы по внешним действиям), и разрушения того или иного компонента (асинхронный γ-переход в нём). Выбор возможного перехода осуществляется недетерминированным образом неким «мистическим синхронизатором». Недетерминизм синхронизатора рассматривается как результат абстрагирования от погодных условий, которые определяют тот или иной выбор.

Важной частной разновидностью IOLTS является конечный автомат (автомат Мили или Finite State Machine – FSM), в котором каждая реакция выдаётся в ответ на каждый стимул. Обычно переход конечного автомата помечается парой символов «стимул-реакция», что соответствует двум переходам LTS: по стимулу и, в промежуточном состоянии, по одной реакции. В трассе такого автомата стимулы и реакции строго чередуются. В промежуточном состоянии определён только один переход – по реакции, выдаваемой в ответ на предыдущий стимул. Тем самым, все промежуточные состояния имеют одни и те же отказы (есть блокировки всех стимулов и нет стационарности). В основных состояниях есть переходы только по стимулам и нет τ-переходов. Основные состояния могут отличаться блокировками. В некоторых случаях стимул, по которому нет перехода в основном состоянии интерпретируется не как блокируемый, а как запрещённый в этом состоянии. Подробнее о той или иной интерпретации отсутствия перехода по стимулу речь пойдёт ниже.

Глава 1.5. Тесты

Структура главы:
1. Трассовые тесты.
2. Автоматные тесты. Синхронное и асинхронное тестирование.
3. Безопасность тестирования.
4. Проблема полноты тестирования.

1.5.1. Трассовые тесты.

В общем случае под тестом можно понимать «инструкцию» по работе с машиной тестирования, в которой указывается, какую кнопку нужно нажимать в той или иной ситуации и что делать после наблюдения тех или иных внешних действий и отказов. Для трассовой спецификационной модели такая инструкция может задаваться множеством трасс (с кнопками). В терминах машины тестирования тестер – это оператор, выполняющий эту инструкцию.

После получения трассы σ, не заканчивающейся на кнопку, тестер может «нажать» кнопку “A”, если в тесте есть трасса $\sigma\cdot\langle\text{“A”}\rangle$. После получения трассы $\sigma\cdot\langle\text{“A”}\rangle$ тестер может наблюдать внешнее действие $a\in A$, если реализация выполняет действие a. Если в кнопку встроен тайм-аут, тестер также может иметь θ-наблюдение, когда реализация отказывается выполнять любое действие из A (истекает тайм-аут). В первом случае мы наблюдаем трассу $\sigma\cdot\langle\text{“A”}\rangle\cdot\langle a\rangle$, заканчивающуюся действием a, а во втором – трассу $\sigma\cdot\langle\text{“A”}\rangle\cdot\langle A\rangle$, заканчивающуюся отказом A.

Далее тестер анализирует полученную трассу $\sigma\cdot\langle\text{“A”}\rangle\cdot\langle a\rangle$ или $\sigma\cdot\langle\text{“A”}\rangle\cdot\langle A\rangle$. Если такой трассы нет в тесте, тестер «не знает», правильная ли такая трасса или нет, и что ему делать дальше. Поэтому в правильном тесте каждая немаксимальная (имеющая продолжение) трасса $\sigma\cdot\langle\text{“A”}\rangle$ продолжается всеми внешними действиями $a\in A$, то есть действиями, разрешаемыми кнопкой “A”, и отказом A (если в кнопку встроен тайм-аут). Максимальная трасса означает окончание работы теста и должна быть снабжена вердиктом ***pass***, если, с точки зрения теста[6], не обнаружено несоответствие, или ***fail***, если найдена ошибка (несоответствие). Таким образом, тест – это множество трасс с вердиктами ***pass*** и ***fail*** на максимальных трассах, которые проверяются на наличие в реализации.

---

[6] Ту часть теста, которая определяет вердикт ***pass*** или ***fail*** называют тестовым оракулом.

Говорят, что реализация *проходит* тест, если при *любом* прогоне этого теста тестер, выполняющий тест, не выдаёт вердикт ***fail***. Тест либо заканчивает свою работу за конечное время с вердиктом ***pass***, либо работает бесконечно, если в нём есть бесконечные трассы. Реализация проходит конечный тест, если тест всегда заканчивается свою работу с вердиктом ***pass***. Набор тестов *значимый* (sound), если любая реализация, соответствующая спецификации, проходит каждый тест из этого набора. Набор тестов *исчерпывающий* (exhaustive), если, наоборот, любая реализация, проходящая каждый тест из набора тестов, соответствует спецификации. Значимый и исчерпывающий набор тестов называется *полным* (complete).

Для практического применения на тесты налагают дополнительные ограничения: 1) тестер, выполняя тест, должен закончить свою работу за конечное время; 2) тестирование должно быть настолько управляемым, насколько это возможно, то есть избегать излишнего недетерминизма в тестере. Условие конечности предполагает, во-первых, отсутствие тупиков, не разрешаемых с помощью θ-наблюдения, и, во-вторых, отсутствие бесконечных трасс. Управляемость означает, что тестовая трасса, не заканчивающаяся на кнопку, продолжается только одной кнопкой, а не несколькими кнопками.

Если приоритетов нет, в управляемом тесте можно удалить кнопки из тестовых трасс, поскольку после такого удаления кнопка, которую нужно нажимать после немаксимальной трассы σ, однозначно вычисляется по имеющимся продолжениям этой трассы: это кнопка, на которой написано множество всех внешних действий, которыми в тесте продолжается трасса σ. Таким образом, для реализаций без приоритетов тестовые трассы управляемого теста не содержат кнопок так же, как трассы реализаций и спецификаций.

Основная задача генерации тестов – автоматически построить полный набор тестов для данного отношения по заданной спецификации или, если полный набор тестов бесконечен, построить алгоритм, перечисляющий тесты полного набора. Такой тест строится по спецификации и выбранному отношению. Говоря упрощённо, тест воспроизводит некоторые значимые с точки зрения отношения трассы спецификации, добавляя для ловли возможных ошибок внешние действия и отказы, недопустимые в спецификации, но возможные в реализации.

1.5.2. Автоматные тесты. Синхронное и асинхронное тестирование.

В автоматной модели тест – это LTS с θ-переходами, компонуемая с LTS-реализацией, в которой θ-переходов нет. Предполагается, что θ-переходы теста выполняются асинхронно в такой композиции. Такое тестирование, когда реализация и тестер взаимодействуют друг с другом непосредственно, называется синхронным тестированием. Для того, чтобы можно было проводить синхронное тестирование мы должны располагать соответствующими тестовыми возможностями.

На практике такие тестовые возможности часто отсутствуют: взаимодействие тестера и реализации опосредуется, так называемым, *тестовым контекстом*. Считается, что реализация «погружена» в тестовый контекст. В автоматной модели тестовый контекст обычно моделируется средой передачи – LTS, которая компонуется с LTS-реализацией. Тестер непосредственно, то есть синхронно, взаимодействует (компонуется) не с реализацией, а с композицией реализации и среды. Такое тестирование называют асинхронным.

Наиболее часто встречающаяся на практике и в теории, можно сказать, *стандартная* среда передачи для моделей ввода-вывода – это две неограниченные FIFO-очереди: входная очередь для стимулов, принимаемых реализацией, и выходная очередь для реакций, выдаваемых реализацией. В этом случае стимул, посылаемый тестером, попадает сначала не в реализацию, а во входную очередь, из которой реализация может принять этот стимул позже. Аналогично, реакция, выдаваемая реализацией, попадает не в тестер, а в выходную очередь, из которой тестер может её выбрать позже. Именно из-за такого разделения во времени момента выдачи стимула/реакции тестером/реализацией и момента его приёма реализацией/тестером и возникло название «асинхроннное тестирование».

Ниже мы более подробно рассмотрим проблемы, связанные с композиционными системами, то есть системами построенным из компонентов с помощью оператора композиции, в том числе специально проблемы асинхронного тестирования.

1.5.3. Безопасность тестирования.

Семантика разрушения предполагает, что при тестировании следует избегать разрушения реализации: нельзя получать трассы с разрушением. Если в реализации есть разрушающее внешнее действие, мы не должны нажимать кнопку, разрешающую это действие. Кнопка безопасна или опасна в зависимости от полученной к этому моменту времени трассы наблюдений.

Поскольку о безопасности мы можем судить только по спецификации и предыстории взаимодействия с тестером, предлагается следующая *гипотеза о безопасности*:

> кнопка, безопасная в спецификации, безопасна в реализации в такой же ситуации, то есть после той же самой предыстории взаимодействия с тестером (после той же самой трассы).

Таким образом, реализации, тестируемые на соответствие данной спецификации, ограничиваются классом реализаций, удовлетворяющих гипотезе о безопасности для этой спецификации.

В данной работе мы моделируем дивергенцию разрушением. Поэтому, в частности, гипотеза о безопасности гарантирует нам, что, избегая дивергенции в спецификации, мы тем самым избегаем дивергенции и в реализации.

1.5.4. Проблема полноты тестирования.

Тестирование полное, если оно правильно отвечает на вопрос, соответствует реализация спецификации или нет.

На одну из причин неполноты любого конечного тестирования мы уже указывали – это недетерминизм реализации, из-за которого в общем случае нельзя быть уверенным в том, что выполненное число прогонов данного теста достаточно.

Другая причина связана с бесконечным числом тестов, образующих полный набор тестов. К сожалению, почти всегда полный набор тестов оказывается бесконечным. Причина – в бесконечном числе трасс, которые должны быть протестированы для проверки соответствия. Системы, в которых число трасс конечно, называются системами с конечным поведением. В автоматной модели конечное поведение имеют только LTS с конечным алфавитом, конечным числом состояний и без циклов. Большинство практических реализаций, конечно, не такое.

Поэтому на практике приходится рассматривать конечные подмножества бесконечного полного набора тестов, которые только значимы, но не полны. Для того, чтобы можно было построить конечный набор тестов по заранее заданному критерию (набор, обладающий заранее заданным свойством), необходима перечислимость полного набора тестов: выбирая тесты один за другим, проверяем наш критерий на всех подмножествах уже выбранных тестов до тех пор, пока не получим искомый набор тестов. Разумеется,

предполагается, что в выбранном полном наборе тестов существует конечное подмножество, удовлетворяющее критерию отбора.

Такова теоретическая база практического тестирования. Конечно, в конкретных случаях такой значимый набор тестов может строиться «напрямую» – без перечисления бесконечного полного набора тестов.

Примером может служить тестирование конечного автомата по спецификации, заданной также в виде конечного автомата. Если у нас есть специальная операция, позволяющая достоверно и напрямую «читать» состояния реализационного автомата (status message), то, как известно, полное тестирование сводится к обходу графа переходов спецификационного автомата и применению status message в каждом проходимом состоянии [LY96]. Обход графа переходов используется также в случае тестирования методом «чёрного ящика», когда состояния реализации не видны. Но здесь для полноты тестирования требуются специальные реализационные гипотезы, компенсирующие отсутствие тестовой возможности достоверного status message. Метод обхода графа переходов является одним из основных в ИСП РАН [BKK03-1, BKK04, KPKB03].

Кроме того, для уменьшения объёма набора тестов часто вместо исходной спецификационной модели используется более грубая, так называемая, тестовая модель. Такая модель является результатом факторизации исходной спецификационной модели по отношению эквивалентности переходов, которое обычно сводится к эквивалентности состояний и/или действий [BKK00]. Разумеется, для того, чтобы такой подход был оправданным, нужны мотивированные реализационные гипотезы.

В настоящей работе мы не рассматриваем такого рода «оптимизации» и занимаемся только базовой проблемой перечисления полного набора тестов.

Глава 1.6. Модели ввода-вывода

Структура главы:
1. Минимальные тестовые возможности.
2. Модели без блокировок стимулов.
3. Отношение $ioco_{\beta\gamma\delta}$.

В данном исследовании мы ограничиваемся таким взаимодействием реализации с окружением (при тестировании – тестом), которое сводится к обмену стимулами и реакциями между реализацией и окружением, и все виды взаимодействий равноприоритетны между собой и с внутренней активностью. Иными словами, мы будем рассматривать системы ввода-вывода без приоритетов[7].

1.6.1. Минимальные тестовые возможности.

Мы предполагаем, что тестовые возможности «минимальны» и ограничены следующими:
1) для каждого стимула есть своя кнопка посылки стимула: тестер может посылать в реализацию любой стимул, не посылая одновременно других стимулов и не принимая реакций;
2) есть общая кнопка приёма всех реакций: тестер может принимать из реализации все реакции, не посылая одновременно стимулов, но и не выбирая, какую реакцию принимать, а какую нет.

Если во все кнопки встроены тайм-ауты, мы можем наблюдать отказы двух видов:
a) *Стационарность* как отсутствие реакций (quiescent), обозначается символом δ [Tret96,Vaan91]. Стабильное состояние автоматной модели, в котором не выдаются реакции, называется стационарным.
b) *Блокировка стимула* (input refusal) как отсутствие приёма данного стимула.

Ниже мы обсудим прагматику таких тестовых возможностей: насколько они реалистичны и когда их можно использовать.

1.6.2. Модели без блокировок стимулов.

В литературе можно найти четыре специальных отношения для моделей ввода-вывода без блокировок стимулов и без приоритетов [Tret96]:

---

[7] В то же время авторы считают, что предлагаемый в данной работе подход применим с соответствующими модификациями для машины тестирования общего вида.

- ***iot*** – **I**nput-**O**utput **T**esting relation или ***quiescent trace preorder*** [Vaan91];
- ***ioconf*** – **I**nput-**O**utput **CONF**ormance;
- ***ior*** – **I**nput-**O**utput **R**efusal relation или ***repetitive quiescence relation***;
- ***ioco*** – **I**nput-**O**utput **CO**nformance.

Вместо тестовой возможности по наблюдению блокировок стимулов во всех этих отношениях предполагается следующая реализационная гипотеза: реализация – это IOTS, то есть полностью определена по стимулам (input-enabled). Что касается спецификации, то она IOTS для ***quiescent trace preorder***, но может быть обычной LTS для ***iot***, ***ioconf***, ***ior*** и ***ioco***.

Эти четыре отношения различаются:

a) однократной (***iot*** и ***ioconf***) или повторяющейся (***ior*** и ***ioco***) стационарностью;
b) общими требованиями: все трассы реализации должны быть в спецификации (***iot*** и ***ior***) или общая трасса реализации и спецификации может в реализации продолжаться стимулом, которого нет после этой трассы в спецификации (***ioconf*** и ***ioco***).

Эти отношения формализуют следующее интуитивное понимание соответствия:

1) реализация может выдавать только те реакции, которые может выдать спецификация в такой же ситуации, то есть после той же самой трассы;
2) реализация может не выдавать никаких реакций (стационарность) только тогда, когда для спецификации это возможно в такой же ситуации, то есть после той же самой трассы.

Под трассами понимаются:

- для ***iot*** : трассы без отказов (traces);
- для ***ioconf*** : трассы без отказов (traces), но только те, которые есть в спецификации;
- для ***ior*** : трассы с задержками (Straces – suspension traces) – трассы, которые могут содержать, кроме внешних действий, стационарность; мы такие трассы называем δ-трассами;
- для ***ioco*** : трассы с задержками (Straces, δ-трассы), но только те, которые есть в спецификации.

Наиболее удачным следует признать отношение ***ioco***. ***Ioco***-теория достаточно хорошо разработана, включая генерацию тестов. Однако она, как и остальные соответствия, опирается на следующее ограничение: в реализации нет блокировок и разрушения (в том числе, дивергенции). Если тестер посылает хотя бы один стимул, реализация его принимает и не разрушается. Для приёма реакций тестер определяет приём всех реакций и

действие по θ-наблюдению (θ-действие). Если реализация посылает хотя бы одну реакцию, она будет принята тестером, а если реализация не посылает ни одной реакции, и в ней нет внутреннего действия, в тестере выполнится θ-действие, что соответствует стационарности. Таким образом, при указанном ограничении реализация никогда не разрушается, не «зацикливается», а тупик возможен только в стационарном состоянии реализации и разрешается θ-действием в тестере.

Заметим, что, если в спецификации трасса не продолжается некоторым стимулом, то поведение реализации после приёма такого стимула безразлично для отношения ***ioco***. Такой стимул не посылается тестом в реализацию. Поэтому формально реализация могла бы такой стимул не принимать, а блокировать, или и то и другое (естественно, в разных состояниях после трассы). Тем самым, требование всюду определённости реализации по стимулам оказывается избыточным даже с точки зрения отношения ***ioco***, основанном на трассах без блокировок и не проверяющем блокировки после трассы.

1.6.3. Отношение $ioco_{\beta\gamma\delta}$.

Целью настоящего исследования является снятие или ослабление указанных ограничений: в реализации допускается разрушение и блокировка стимулов. Приоритетов нет. Соответствующие трассы мы называем βγδ-трассами. Предлагается отношение $ioco_{\beta\gamma\delta}$ для таких βγδ-трасс со следующим правилом:

> реализация может ... только тогда, когда это возможно в спецификации после той же самой безопасной βγδ-трассы, где многоточие означает:
> - выдать реакцию,
> - не выдавать никаких реакций (стационарность),
> - принять стимул,
> - не принимать стимул (блокировка стимула).

Минимальные тестовые возможности, необходимые для тестирования отношения $ioco_{\beta\gamma\delta}$ указаны выше. Обсудим прагматику этих тестовых возможностей.

Глава 1.7. Проблемы взаимодействия тестера и реализации

Структура главы:
1. Проблема блокировки стимулов.
2. Проблема стимулов «на выбор».
3. Проблема реакций «на выбор».
4. Проблема «торможения» реакций.

В этой главе обсуждаются четыре проблемы, связанные со способом передачи стимулов из тестера в реализацию и реакций из реализации в тестер. То или иное решение этих проблем определяется тестовыми возможностями и реализационными гипотезами и налагает соответствующие ограничения на интерфейс реализации и тестера.

1.7.1. Проблема блокировки стимулов.

Требование отсутствия блокировок и разрушения сужает класс тестируемых реализаций, оставляя за его границами многие реальные системы. О семантике разрушения мы уже упоминали выше. Что касается блокировки, то запрет на блокировку делает невозможным тестирование систем, в функциональность которых входит блокировка. На это уже указывали некоторые авторы [Seg93, LG05] и мы [BK05-1,BKK06]. Рассмотрим три примера.

1) Воспользуемся излюбленным примером Jan Tretmans`a – машиной «чай-кофе». Эта машина имеет две кнопки для чая и для кофе, а также щель, куда нужно опускать монету. Эта щель может иметь заслонку, которая после опускания монеты закрывает щель до тех пор, пока не будет нажата какая-нибудь кнопка. Если мы опустим монету в щель и, не нажимая кнопок, попытаемся опустить вторую монету, она не будет принята – блокируется приём второй монеты. В [LG05] такой тип блокировки называется «физическим» (physical).
2) Математически машина «чай-кофе» – это пример FIFO-очереди (для монет) длиной один. В общем случае для ограниченной FIFO-очереди передача стимула – это помещение символа в конец очереди, а передача реакции – выборка символа из головы очереди. Спецификация очереди требует, чтобы стимул блокировался тогда и только тогда, когда очередь полностью заполнена.
3) Более практический пример – взаимодействие через пару симплексных каналов. Если связь через канал передачи стимулов разорвана по инициативе принимающей стороны, то о невозможности передачи сообщит канал, а не принимающая сторона, что интерпретируется как блокировка стимула.

4) Ещё один практический пример – управление графическим интерфейсом (GUI). В каждый момент времени на экране могут быть кнопки и поля, работа с которыми может быть разрешена или запрещена. Последнее соответствует блокировке стимулов. В [LG05] такой тип блокировки называется «программным» (software).

Запрет на блокировку стимулов в реализации иногда мотивируют тем, что блокировку стимула можно понимать как приём стимула реализацией с последующей ответной реакцией, извещающей о том, что стимул в данный момент не может быть «обработан». Однако примеры показывают нереалистичность такой мотивировки. Для «чая-кофе» это означало бы, что автомат сначала «глотает» моменту, а потом тут же её «выплёвывает» в окошко «для сдачи». Но тогда это был бы совсем другой автомат! Для взаимодействия через каналы связи это означало бы выдачу другой стороной реакции по обратному каналу. Для GUI это означало бы, что мы можем нажимать кнопку, но в ответ должно было бы всплывать окошко с надписью «не принято». Такой способ работы иногда реализуется, но он существенно отличается от описанного, когда кнопка «бледная» и её нажать не получается.

Только для FIFO-очереди мы могли бы считать, что операция поместить символ в очередь имеет код ответа (реакцию) «не принято». Однако такой код ответа существенно отличается от других реакций (выборка из очереди), он идёт «по другой линии». Это особенно заметно, когда очередь реализована как очередь сообщений по каналу связи (пример 3).

Исторически представление о реализациях, всегда готовых принимать любые стимулы, возникло из протоколов взаимодействия по постоянно установленным каналам. Стимулы помещаются в этот канал, который, как правило, всегда готов их принять и передать дальше (неограниченная FIFO-очередь). В этом смысле композиция реализации и входного (для реализации) неограниченного канала, конечно, всюду определена по стимулам. Однако программа «за каналом» вовсе не всегда обязана принимать стимулы из канала. Иными словами, реализация может блокировать стимулы и это можно проверить при синхронном тестировании, но при асинхронном тестировании картина «смазана», поскольку композиция реализации с неограниченной входной очередью не блокирует стимулы.

Jan Tretmans в своей диссертации [Tret92] одну из глав посвятил различного рода блокировкам, возникающим в ситуации асинхронного тестирования. В частности, выделил *временные* и *постоянные* блокировки стимулов, когда в голове входной очереди находится стимул, который реализация не может принять в данный момент времени. При *временной* блокировке реализация

может выдавать реакции и через некоторое время такая блокировка либо разрешается, либо становится постоянной. *Постоянная* блокировка возникает в ситуации, когда в реализации нет внутренней активности и она не выдаёт никаких реакций, а может только принимать стимулы, но не тот, что в голове входной очереди. К сожалению, в своих дальнейших исследованиях Jan Tretmans больше не возвращался к проблеме блокировок стимулов (за исключением совместных работ с Heerink`ом), считая, что реализация всегда input-enabled.

В работах Heerink и Tretmans по Multi Input-Output Transition Systems (MIOTS) также частично допускается блокировка стимулов [HT97,Heer98]. Здесь множество стимулов разбивается на подмножества – входные каналы, и для каждого входного канала либо должен приниматься любой стимул из этого канала, либо ни одного (отказ по стимулам). Множество реакций также разбивается на подмножества – выходные каналы, но в отличие от стимулов можно выдавать любое подмножество реакций выходного канала; отказ по реакциям возникает, если это подмножество пусто (не выдаётся ни одна реакция этого канала). Каждому каналу соответствует своё θ-наблюдение; оно означает: для входного канала – отказ по всем стимулам этого канала, для выходного канала – отсутствие реакций этого канала («частичная» стационарность). Соответствующие трассы, включающие наблюдения таких отказов по каналам, называются ***f-traces*** (failure-traces, в которых отказы – это подмножества стимулов или реакций, определяемые входными или выходными каналами).

Здесь блокировка оказывается временной также и в том случае, когда реализация не выдаёт реакций, но остаётся хотя бы один входной канал, из которого принимаются все стимулы, но в данный момент времени канал пустой (стимулы не предлагаются). Реализация ждёт, когда в таком канале появится стимул.

Если каждому стимулу в MIOTS будет соответствовать ровно один канал, то все возможные отказы по стимулам – это все блокировки стимулов. Если все реакции направляются в один выходной канал, то имеется только один отказ по реакциям – стационарность. Соответствующее отношение конформности ***mioco*** разрешает реализации блокировать стимул только в том случае, когда это возможно в спецификации. В то же время реализация может всегда принимать стимул независимо от того, принимается стимул в спецификации или только блокируется. По-видимому, такая «либеральность» объясняется происхождением ***mioco*** от ***ioco***, в которой реализация считается всюду определённой по стимулам. Для ***mioco*** это уже не так – реализация может блокировать стимул, если это разрешает спецификация, но по-прежнему нельзя потребовать, чтобы реализация *не* принимала стимул, то есть только

блокировала его. Однако, в примерах, рассмотренных выше, к реализации должно предъявляться как раз такое требование.

В то же время стоит отметить, что отношение ***mioco*** представляет собой существенный шаг вперёд по сравнению с ***ioco***, поскольку вводятся в рассмотрение трассы с отказами по стимулам. Именно после таких трасс, общих для спецификации и реализации, выполняется проверка правильности поведения реализации, а не только после δ-трасс (suspension traces) как в ***ioco***. Для случая «входной канал – один стимул, выходной канал – все реакции» ***mioco*** работает с трассами, которые, кроме стимулов и реакций, могут содержать блокировки стимулов и стационарность (мы такие трассы называем βδ-трассами). Это усиливает способность различения реализаций (конформные - не конформные).

В частности, хотя реализация имеет право как принимать, так и блокировать стимул, который в спецификации только блокируется после трассы, мы должны проверять, что именно делает реализация. Если реализация принимает стимул, её дальнейшее поведение может быть любым и не проверяется. Но если реализация блокирует стимул, то её дальнейшее поведение должно соответствовать поведению спецификации после трассы, продолженной блокировкой, а не просто после трассы. Тем самым, в отличие от ***ioco***, после каждой трассы проверяется каждый стимул (в общем случае хотя бы один стимул из входного канала).

Gregory Lestiennes и Marie-Claude Gaudel в [LG05] предлагают отношение ***rioco*** для, так называемых, Restrictive Input-Output Labelled Transition System (RIOLTS) которое, в противоположность ***mioco***, требует от реализации блокировки стимула, если в спецификации стимул не принимается (только блокируется). Однако, с другой стороны, отношение ***rioco*** основано всё ещё на δ-трассах, а не на βδ-трассах: блокировки проверяются после трассы, но сама трасса блокировок не содержит. Это, естественно, ослабляет способность различения реализацией.

Отношение ***rioco*** имеет ещё одну особенность для спецификации: стимул в состоянии может не приниматься и не блокироваться, такой стимул интерпретируется как *неспецифицированный* в состоянии. По поводу таких стимулов спецификация ничего не требует от реализации: они могут как приниматься, так и блокироваться. При этом в реализации все стимулы должны быть специфицированы, то есть реализация не отличается от обычной LTS. Посмотрим, какие требования отношение ***rioco*** налагает на реализацию, то есть что оно запрещает в реализации по поводу стимулов после некоторой трассы, общей для спецификации и реализации:

1) Как сказано выше, ***rioco*** требует блокировки в реализации и рассматривает приём стимула как ошибку, если в спецификации стимул блокируется во всех состояниях после трассы.
2) Наоборот, ***rioco*** трактует как ошибку множество блокировок стимулов, если, согласно спецификации, должен быть принят хотя бы один из них. Это проверяется посылкой стимулов из множества один за другим в любом порядке до тех пор, пока какой-нибудь стимул не будет принят, либо окажется, что все стимулы блокируются, что означает ошибку в реализации. Иными словами, ***rioco*** требует, чтобы в реализации трасса не продолжалась данной последовательностью блокировок, если такого не может быть в спецификации. Заметим, что префикс последовательности блокировок (произвольно упорядоченное подмножество блокировок) может допускаться спецификацией. Такое несколько вычурное требование возникает именно потому, что ***rioco*** использует δ-трассы, а не βδ-трассы (иначе, речь шла бы об одной блокировке, а не об их множестве).

Для отношения ***rioco*** также возможна ситуация, когда поведение реализации по поводу некоторого стимула безразлично с точки зрения спецификации. Это может быть в том и только том случае, когда в спецификации в каждом состоянии после трассы стимул не принимается и хотя бы в одном состоянии не специфирован. В этом случае тест не посылает такой стимул в реализацию. Это аналогично тому, что этот стимул разрушающий. Отличие лишь в том, что такой где-то неспецифицированный и везде не принимаемый стимул посылать в реализацию бессмысленно, а разрушающий стимул – запрещено. В то же время нужно отметить, что сама по себе неспецифицированность стимула слабее свойства стимула быть разрушающим.

1.7.2. Проблема стимулов «на выбор».

Рассмотрим проблему передачи стимулов: может ли тестер предлагать реализации несколько стимулов «на выбор»? Если на этот вопрос отвечают положительно, то предполагается, что в одном атомарном взаимодействии реализация сама выбирает один стимул из «предлагаемого списка», а остальные стимулы не принимаются. Возможность предлагать несколько стимулов означает специальную тестовую возможность, которой в машине тестирования соответствует кнопка, на которой написано несколько стимулов.

С точки зрения самой реализации такая ситуация вполне реалистична: стимулы могут передаваться через несколько входных портов или FIFO-очередей, как в MIOTS [HT97,Heer98], а реализация в атомарном взаимодействии выбирает один стимул из одной очереди, тем самым не выбирая предлагаемые стимулы из других непустых очередей.

Необходимость в множестве стимулов возникает тогда, когда имеются взаимные приоритеты между стимулами. Тогда мы должны проверять, что приём одного стимула приоритетнее приёма другого стимула. А это можно сделать только предложив реализации на выбор оба эти стимула. Тем самым в машине тестирования должна быть кнопка с этими двумя стимулами и соответствующий отказ.

В настоящей работе мы ограничиваемся системами без приоритетов. В этом случае наличие кнопок с множественными стимулами не усиливает способность отношения конформности различать реализации и, соответственно, не увеличивает мощность тестирования, разумеется, при условии, что мы можем посылать каждый стимул отдельно (одиночная кнопка, на которой написан только один стимул). Для наблюдения отказа с несколькими стимулами достаточно посылать эти стимулы один за другим (одиночными кнопками), наблюдая блокировки этих стимулов. Здесь используется тот факт, что любой отказ, в частности блокировка, не меняет состояние реализации.

Поэтому для систем без приоритетов остаётся один вопрос: есть ли у нас тестовая возможность для каждого стимула послать в реализацию *один* этот стимул?

Во-первых, на практике системы без такой тестовой возможности до сих пор не встречались (или всегда находилось обходное решение).

Во-вторых, мы можем изменить уровень абстракции тестера, сменив его алфавит стимулов: ввести новые множественные стимулы, которые были бы множествами старых, элементарных стимулов (по крайней мере, тех, которым соответствовали кнопки машины тестрования). У нас получится новая машина тестирования, в которой на каждой кнопке будет написан только один множественный стимул. В этом случае реализация как бы окружается оболочкой, преобразующей один множественный стимул во множество предлагаемых элементарных стимулов, а реализация выбирает из них один элементарный стимул. С точки зрения тестера, работающего на уровне множественных стимулов, мы получаем лишь дополнительный недетерминизм в реализации, поскольку не знаем, какой именно элементарный стимул, соответствующий этому множественному стимулу, выбрала реализация. Это можно понимать и так, что в LTS переход по элементарному стимулу заменяется на кратные переходы по всем множественным стимулам, содержащим этот элементарный стимул.

Таким образом, для систем без приоритетов мы всегда можем считать, что тестер имеет возможность послать в реализацию один стимул.

1.7.3. Проблема реакций «на выбор».

Рассмотрим проблему передачи реакций: может ли тестер «выбирать», какие реакции принимать от реализации, а какие нет? Если на этот вопрос отвечают положительно, то предполагается, что в одном атомарном взаимодействии тестер сам выбирает одну реакцию из «предлагаемого списка», а остальные реакции не принимаются. Возможность принимать часть реакций означает специальную тестовую возможность, которой в машине тестирования соответствует кнопка, на которой написано одна или несколько, но не все, реакции.

Некоторые семантики общего вида ориентированы на такую тестовую возможность: failure semantics и failure trace semantics. Отношение ***mioco***, как сужение failure trace semantics, применяется для MIOTS, которые можно рассматривать как адекватные модели многих реальных реализаций. Такая реализация имеет несколько выходных FIFO-очередей (портов), в которые она посылает свои реакции. Тестер может принимать реакции из разных очередей независимым образом. В терминах машины тестирования, пару (реакция, номер очереди) можно понимать как новую реакцию; тогда на кнопке может быть написано множество новых реакций, соответствующих одной очереди. Тем самым, тестер может обнаруживать «частичную стационарность», то есть отказ, содержащий подмножество всех новых реакций, а именно множество новых реакций, соответствующих одной очереди.

Итак, если у нас есть тестовая возможность «выбирать» реакции, мы можем использовать отношения конформности, более сильные в смысле различения реализаций.

С другой стороны такая тестовая возможность необходима тогда, когда имеются взаимные приоритеты между реакциями, то есть не все реакции «находятся в равном положении», и мы хотим проверять соблюдение или нарушение приоритетов. В терминах машины тестирования это означает, что на некоторых кнопках должны быть написаны подмножества реакций, то есть тестер должен уметь принимать часть реакций. Поскольку соответствие должно выражаться в терминах наблюдаемых трасс, взаимные приоритеты реакцией выражаются в виде некоторого отношения между реакциями и/или стимулами после той или иной трассы.

Например, спецификация после трассы σ допускает четыре реакции `1,2,3,4` с приоритетатми `1<3<4` и `2<4`. Если тестер всегда принимает все реакции, то разрешённые множества реакций после σ только такие `{1},{2},{3},{4},{1,2},{2,3}`. При получении в двух прогонах теста

после трассы σ любой пары сравнимых реакций {1,3}, {1,4} или {2,4} фиксируется ошибка нарушения приоритета. Однако, если при приёме тестером всех реакций всегда принимается реакция 4, то это ещё ничего не говорит о взаимном приоритете реакций 1,2 и 3 в реализации. Для проверки этого следовало бы принимать только реакции 1,2 и 3: ошибка нарушения приоритета фиксируется, если в двух прогонах теста получается пара реакций {1,3}.

Таким образом, тестирование таких приоритетов неизбежно оказывается «тормозящим» реакции, выдаваемые реализацией (по крайней мере, часть их).

Заметим, что отсутствие приоритета между реакциями не означает отсутствие приоритетов вообще, в частности между стимулами. Это означает лишь, что, с точки зрения приоритетов, тестер принимает либо все реакции, либо ни одной. Если реакции написаны на разных кнопках, то эти кнопки отличаются только набором стимулов. Значит, выполнение реакций может зависеть от того, какие стимулы разрешаются нажатой кнопкой, но в этой зависимости, если она есть по спецификации, все реакции занимают одинаковое положение. Тем более сохраняется возможность описывать в спецификации приоритеты стимулов, если стимул может быть написан на разных кнопках.

В настоящей работе мы предполагаем, что приоритетов нет, а тестовые возможности минимальны: тестер принимает либо все реакции, либо ни одной.

1.7.4. Проблема «торможения» реакций.

Итак, тестирование приоритетов между реакциями неизбежно ведёт к «торможению» реакций. Но само такое «торможение» – более общая проблема. При отсутствии приоритетов между реакциями тестеру нет нужды принимать только часть реакций. На каждой кнопке машины тестирования написаны либо все реакции, либо ни одной. Теперь, в терминах машины тестирования, проблема формулируется так: может ли быть только одна кнопка, на которой написаны все реакции, а для каждого стимула на кнопке написан только этот стимул? Или, в терминах автоматной модели: может ли тестер в некотором своём состоянии не принимать никаких реакций, а только посылать стимул?

Прежде чем обсуждать допустимость торможения реакций, посмотрим какие выводы следует сделать из запрета на торможение. Иными словами, как такое ограничение тестовых возможностей влияет на соответствие.

В стационарном состоянии, где реализация не выдаёт никаких реакций и только ждёт стимулов, никакого торможения реакций не будет даже тогда, когда тестер никаких реакций не принимает – их всё равно нет. Стационарность обнаруживается как θ-наблюдение при приёме всех реакций. После этого тестер может послать один стимул, не принимая реакций. Этот стимул будет либо принят реализацией, либо блокирован, что обнаруживается θ-наблюдением. В последнем случае реализация не меняет своего состояния, то есть оно по-прежнему стационарно, и тестер может посылать другой стимул.

Другая картина складывается в нестационарном состоянии `s` реализации, когда тестер посылает стимул и одновременно принимает все реакции.

Тестер наблюдает либо приём стимула, либо выдачу той или иной реакции. Поскольку любой из этих вариантов выбирается недетерминированно (приоритетов нет), мы имеем возможность при некотором прогоне теста наблюдать именно приём стимула, если этот стимул принимается (не блокируется). Если состояние `s` нестабильно, то приём стимула или выдача той или иной реакции может произойти не только в состоянии `s`, но и в любом другом состоянии вдоль цепочки τ-переходов, начинающейся в `s`. Если такая цепочка может закончиться в стационарном состоянии `s``, то возможно θ-наблюдение. Кроме стационарности, это θ-наблюдение будет обозначать также блокировку стимула. Этот вариант, если он есть в реализации, также может быть выбран наравне с приёмом стимула и выдачей той или иной реакции.

Если состояние `s` стабильно и стимул в нём не принимается, то стимул блокируется. Однако мы не сможем обнаружить эту блокировку: вместо θ-наблюдения каждый раз будет приниматься та или иная реакция. Блокировка в этом случае вычисляется не по θ-наблюдению, а как наблюдение-отрицание, означающее, что среди всех возможных наблюдений нет наблюдения приёма стимула. Такое наблюдение-отрицание введено van Glabbeek`ом [Glab93], но оно представляет собой чисто теоретическую конструкцию и невозможно на практике.

Таким образом, запрет на торможение реакций приводит к ограничению на наблюдение: мы можем наблюдать блокировки только в стационарных состояниях. Иначе говоря, блокировка наблюдается только одновременно со стационарностью; отдельная от стационарности блокировка не наблюдаема. Тем самым, блокировка в трассе одновременно означает стационарность, хотя стационарность без блокировки по-прежнему остаётся.

В терминах машины тестирования, запрет на торможение реакций означает, что у нас есть кнопка для приёма всех реакций и, для каждого стимула, кнопка передачи этого стимула с одновременным приёмом всех реакций. Соответствующие отказы: стационарность и блокировка+стационарность. Соответствующее отношение в духе $\boldsymbol{ioco}_{\beta\gamma\delta}$ для таких βγδ-трасс определяется следующим правилом:

> реализация может ... только тогда, когда это возможно в спецификации после той же самой трассы, где многоточие означает:
> - выдать реакцию,
> - не выдавать никаких реакций (стационарность),
> - принять стимул,
> - не принимать стимул и не выдавать никаких реакций (блокировка+стационарность).

Под трассой можно понимать либо δ-трассы как в ***rioco***, либо βδ-трассы, в которых блокировка берётся только в стационарных состояниях.

Последний вариант был бы наиболее естественным расширением отношения ***rioco***, как раз ориентированного на запрет торможения реакций. Частным случаем является разновидность спецификаций без смешанных состояний, то есть в состояниях не может быть смеси переходов по стимулам и реакциям (exclusive RIOLTS в [LG05]). В таких спецификациях βδ-трассы содержат как раз только такие блокировки, которые совмещены со стационарностью. Однако представляется, что лучшим решением было бы рассматривать общий вид RIOLTS, но при формировании адекватной трассовой модели блокировки в нестационарных состояниях не учитывать.

Итак, при отсутствии приоритетов между реакциями нам нужно тормозить реакции только в том случае, если мы хотим наблюдать блокировки в нестационарных стабильных состояниях для того, чтобы учитывать их в отношении конформности.

Теперь обсудим допустимость торможения реакций.

Идея о том, что тестер не должен тормозить реакции, выдаваемые реализацией, имеет те же исторические корни, что и запрет на блокировку стимулов – речь идёт о реализации протоколов. Здесь реализация оказывается окружённой не только входной FIFO-очередью стимулов, но и выходной FIFO-очередью реакций. Иными словами, тестирование такой реализации всегда асинхронное.

Тестер посылает стимулы не в реализацию, а во входную очередь, и потому тестер «не видит» блокировок стимулов самой реализацией, находящейся на «той стороне» очереди. Но реализация принимает стимулы из входной

очереди тогда, когда она этого хочет, то есть в самой реализации блокировки стимулов допускаются. С другой стороны, тестер принимает реакции из выходной очереди, а не из реализации непосредственно. Если очередь неограниченная, то тестер, не принимающий реакций, никак не тормозит саму реализацию, а только её выходную очередь. Вместе с тем, реализация выдаёт реакции в выходную очередь «без задержек».

При таком асинхронном тестировании никаких проблем с торможением реакций, выдаваемых реализацией, не возникает просто потому, что реакции выдаются в неограниченную выходную очередь, а не в тестер непосредственно. В этом кроется некоторая двойственность отношения к тому, что мы называем реализацией. Получается, что реализация «за очередью» не может тормозиться, а совокупность этой реализации и выходной очереди – может, то есть такая совокупность уже как бы «не реализация». Фактически, речь идёт о том, что реализации бывают разные: одни тормозить можно, а другие нельзя. Здесь «можно» или «нельзя» означает наличие или отсутствие соответствующей тестовой возможности.

Тем не менее, даже отмечая эту двойственность, мы не избавляемся от самой проблемы.

В связи с этим отметим, что математические модели взаимодействия часто разрешают торможение реакций. Определение непосредственного взаимодействия тестера и реализации с помощью оператора параллельной композиции алгебры процессов позволяет не только не принимать реакции, но и делать произвольный выбор: какие реакции принимать, а какие нет. В упоминавшейся выше модели Multi Input-Output Transition Systems (MIOTS) [HT97,Heer98] запрещается делать выбор между реакциями одного канала, но не между каналами, то есть те или иные каналы могут тормозиться тестером. В модели взаимодействующих автоматов – *Communicating Finite State Machines* (CFSM) [BZ83] компоненты системы могут взаимодействовать не непосредственно, а через FIFO-очереди сообщений. Такая очередь не может выбирать, какие реакции она принимает от реализации, а какие нет, но может быть ограниченной [Holz91]. Если компонент не принимает реакции из очереди, которая уже полна, возникает торможение реакций, выдаваемых другим компонентом в эту очередь.

Вопрос в том, насколько такие модели взаимодействия адекватны практике?

Прежде всего, отметим, что даже непосредственное взаимодействие тестера и реализации в духе алгебры процессов встречается на практике и широко распространено (не считая указанного выше случая композиции реализации с неограниченной выходной FIFO-очередью). Это обычная отладка, когда реализация находится под управлением тестера настолько полным, что

тормозить можно не только выдаваемые реализацией реакции, но и её внутренние переходы. Другое дело, что такой вид тестирования не единственный и, как правило, речь идёт о другом тестировании, когда реализация не столь управляема, то есть тестовые возможности намного скромнее.

Проблема торможения реакций, на наш взгляд, состоит не столько в том, что тестер может не принять реакции, выдаваемые реализацией, сколько в том, что из-за этого возникает задержка в выполнении реализации. Мы уже видели, что в стационарном состоянии реализации тестер может не принимать никаких реакций, поскольку их всё равно нет и не будет до тех пор, пока тест не пошлёт в реализацию какой-нибудь стимул, который реализация примет.

Также не будет задержки, если реализация одновременно с выдачей реакций готова принимать стимул и такой стимул ей посылает тестер. Иными словами, если реализация объявляет, что она готова принять стимул и готова выдать реакцию, то (при отсутствии приоритетов) она не накладывает никаких ограничений на то, какой из этих двух вариантов произойдёт: приём стимула или выдача реакции. Здесь вряд ли можно говорить о том, что реализация «неудержимо стремится» эти реакции выдать. В случае, когда возможны оба варианта, выбор одного из них выполняется вне реализации. Тогда, если стимул послан в реализацию, а реакции тормозятся (не принимаются тестером), остаётся один вариант и реализация принимает стимул, «забывая» о выдаче реакции до окончания перехода по этому стимулу. Никакой задержки в поведении реализации не возникает. Аналогично задержка не возникает в нестабильном состоянии, поскольку у реализации здесь всегда остаётся возможность выполнить τ-переход.

Таким образом, задержка возможна лишь в стабильных нестационарных состояниях, когда реализация либо вообще не принимает стимулов, либо не принимает те стимулы, которые ей посылает тестер, а тестер не принимает реакций, выдаваемых реализацией.

Теперь зададимся простыми вопросами: а почему, собственно, нельзя задерживать выполнение реализации? Что в этом страшного? Что происходит, когда такая задержка возникает? Разве реализация не может подождать? Происходит же такое ожидание в стационарных состояниях, в которых никто не требует от тестера обязательно посылать стимул, который реализация ждёт.

Здесь проходит разграничительная линия между семантикой стимулов и реакций. Стимулы могут ожидаться сколь угодно долго (примитив WAIT), а

вот реакции должны посылаться немедленно (примитив SEND) или не посылаться вообще, если пришёл ожидаемый стимул (примитив SENDorWAIT). Задержка критична только при выдаче реакций, когда возврат управления из примитива SEND или SENDorWAIT происходит не мгновенно, а через какое-то, быть может, значительное время. Здесь речь идёт именно о задержке, а не о выборе какую реакцию передавать.

На наш взгляд, интуитивно здесь имеется в виду некое «нарушение» поведения реализации в том случае, когда возникает такая задержка. То есть, если задержки нет, поведение реализации одно, стандартное, а при задержке – другое, альтернативное. Но в таком случае возникают два вопроса:

1) Не должно ли это альтернативное поведение также регламентироваться спецификацией и, тем самым, проверяться при тестировании? Тогда мы не должны избегать торможения реакций, но, напротив, специально создавать ситуацию торможения для проверки альтернативного поведения. Более точно, тестер должен проверять поведение реализации как при отсутствии торможения, так и при торможении.
2) Как эти два поведения могли бы изображаться в математической модели?

По поводу первого вопроса нужно сделать важное замечание. Альтернативное поведение реализации может означать её разрушение. Тогда мы, конечно, должны его избегать при тестировании. Но это не единственный вариант.

По поводу второго вопроса: на самом деле способ изображения в модели альтернативного поведения существует и давно известен. Только применяется этот способ не в спецификации, а в тесте. Это θ-действие, которое как раз и означает действие в случае возникновения тупика. Этот тупик реализация может разрешить с помощью θ-действия. Если бы в реализации был определён θ-переход, то вместо тупика «сработал» бы этот переход с дальнейшим альтернативным поведением.

Заметим, что такое θ-действие полезно не только для моделирования альтернативного поведения при торможении реакций. В стационарном состоянии θ-действие позволяет распознать отсутствие стимулов в данный момент времени. Этот вариант мы исследовали в [BKK03], где такое θ-действие называли ε-действием. Это исследование показало, что указанная выше «разграничительная черта» между стимулами и реакциями не так уж очевидна. Задержка в стационарном состоянии тоже может рассматриваться как критическая (хотя, возможно, с другим тайм-аутом), и спецификация может требовать от реализации альтернативного поведения в случае такой задержки. Заметим, что альтернативное поведение в ситуации, когда тестер посылает стимул, не принимаемый реализацией, можно было бы понимать

как переход по стимулу с этим последующим альтернативным поведнием. Однако остаётся случай, когда в состоянии реализации определены переходы по всем стимулам, но альтернативное поведение всё равно существует в случае, когда тестер вообще не посылает никаких стимулов в реализацию. И опять, как и для реакций, альтернативное поведение может означать разрушение и тогда мы должны избегать его при тестировании.

Можно показать, что модели с θ-действиями являются разновидностью моделей с приоритетами. Действительно, под θ-переходом мы могли бы понимать переход по любому внешнему действию или даже τ-переход, имеющий наинизший приоритет среди всех переходов из данного состояния. Такой переход срабатывал бы тогда и только тогда, когда все остальные переходы невозможны, то есть в ситуации тупика – задержки.

К сожалению, в литературе теория соответствия и тестирования для реализаций и спецификаций с приоритетами и, в частности, с θ-действиями не разработана. Авторы готовы описать соответствующие трассовую и автоматную модели, ***ioco***-подобное отношение конформности и генерацию тестов. Однако это далеко выходит за пределы настоящей работы.

Поэтому для моделей без приоритетов (и без θ-действий) остаётся проблема торможения реакций. Как мы видели выше, эта проблема проявляется в стабильных нестационарных состояниях реализации, когда тестер не посылает стимул в реализацию или посылает такой стимул, который реализация не принимает. В этой ситуации тестер должен принимать все реакции. Понятно, что тестер не знает, какие стимулы реализация принимает, а какие нет. Поэтому гарантированно задержка не возникает, если тестер не принимает реакции только в стационарных состояниях реализации. Последнее обнаруживается тестером по истечению тайм-аута ожидания реакций (θ-наблюдение).

A. Petrenko, N. Yevtushenko и J.L. Huo в [PYH03] предложили интересное решение: разделить тестер на два процесса: один всегда принимает реакции, а другой посылает стимулы. Тогда передающий процесс может посылать стимулы, а θ-наблюдение в принимающем процессе означает только стационарность. Иными словами, в совокупности этих процессов тупика не возникает, но стационарность наблюдается.

Правда, взаимодействие этих процессов практически отсутствует. Фактически, сравниваются словарные функции реализации и спецификации с учётом стационарности в конце трасс. Трассе LTS, заканчивающейся в стационарном состоянии, соответствуют входное слово и выходное слово как проекции на алфавиты стимулов и реакций, соответственно. Входному

слову в реализации и спецификации соответствуют множества возможных выходных слов, которые и сравниваются на вложенность. Получается отношение, названное ***queued-quiescent trace-included***. Реализация предполагается окружённой ограниченными входной и выходной FIFO-очередями. Ограниченность выходной очереди приводит к тому, что запрещается бесконечная трасса реакций после любой конечной трассы (*осцилляция*). Это похоже на наш подход в [BKK03] с тем отличием, что мы рассматривали неограниченные очереди и бесконечные трассы, поэтому не учитывали стационарности и разрешали осцилляцию, а также допускали разрушение.

На наш взгляд, решение в [PYH03] показывает, как можно *реализовать* тестирование без торможения реакций с посылкой стимулов в нестационарных состояниях реализации. В этой работе блокировки стимулов не рассматриваются: реализация и спецификация всюду определены по стимулам. Но для LTS общего вида их наблюдение вполне можно реализовать в передающем процессе: в нём θ-наблюдение означало бы блокировку. Здесь важно, что такая блокировка обнаруживается только в том случае, когда реализация не выдаёт реакций. Соответственно, если передающий процесс посылает стимул, принимающий процесс наблюдает стационарность только в том случае, когда реализация этот стимул блокирует. Иными словами, мы имеем θ-наблюдение в конце теста, которое означает либо стационарность, либо блокировку+стационарность. В первом случае входная очередь пуста, а во втором содержит блокируемый и все последующие стимулы.

Можно также отметить, что проблема торможения реакций снимается, если стимулы посылаются только в стационарных состояниях реализации. Это, так называемое, стационарное тестирование [BKK03], которое применяется для «грубой» проверки конформности.

Резюмируя, можно сказать, что проблема торможения реакций имеет две стороны.

Первая – отсутствие тестовой возможности тормозить реакции. Это означает, что что-то «мешает» тесту непосредственно взаимодействовать с реализацией по выдаче реакций, то есть между тестом и реализацией имеется некоторая среда передачи. В этом случае мы не можем производить синхронное тестирование и вынуждены ограничиваться более грубым асинхронным тестированием. На наш взгляд, отношение конформности всё равно удобнее определять для синхронного тестирования как некую базовую синхронную конформность. Тогда для асинхронного тестирования соответствующую асинхронную конформность можно «вычислить», заменяя

спецификацию на её композицию с неограниченной выходной очередью (или какой-то другой средой передачи, не тормозящей реакции реализации). Здесь, однако, возникает проблема несохранения соответствия, которую мы обсудим ниже.

Вторая сторона – неспособность современных моделей изображать приоритеты, в частности, θ-переходы в спецификации как средство описания альтернативного поведения при задержке выполнения, вызванной торможением реакций. Это может стать препятствием даже в том случае, когда у нас есть тестовая возможность тормозить реакции реализации. Нужно развивать теорию тестирования, распространяя её на случай моделей с приоритетами.

Особый случай – когда альтернативное поведение, по умолчанию, нежелательно, в нашей терминологии – разрушает реализацию. Здесь приходится использовать тестирование без торможения реакций, совмещая передачу стимула с приёмом всех реакций, из-за чего отношения конформности становятся несколько слабее. Примером служат ***rioco*** и ***queued-quiescent trace-included***.

В настоящей работе мы рассматриваем модели без приоритетов и определяем базовое синхронное соответствие. Тем самым, мы допускаем торможение реакций, предполагая, что возможная при этом задержка выполнения реализации некритична, то есть не вызывает альтернативного поведения. Мы будем генерировать тесты без лишнего недетерминизма, то есть без совмещения приёма реакций и выдачи стимулов. В то же время, представляется, что основные результаты нашей работы можно распространить на тестеры, не вызывающие торможения реакций и проверяющие соответствующее ослабленное отношение конформности с блокировками, совмещёнными со стационарностью.

Глава 1.8. Пополнение спецификации

Структура главы:
1. Неспецифицированный стимул.
2. Допущение полноты (подразумеваемое пополнение).
3. Проблема рефлексивности соответствия.

В LTS-спецификации в некоторых состояниях могут быть не определены переходы по некоторым стимулам или реакциям. Неспецифицированная реакция всегда означает, что реализация не должна выдавать эту реакцию (если реакция не специфицирована в каждом состоянии после трассы наблюдений). Однако семантика неспецифицированных стимулов не определена столь однозначно. В этой главе обсуждаются разные подходы к пониманию поведения системы по поводу неспецифицированных стимулов.

1.8.1. Неспецифицированный стимул.

Отсутствие перехода по стимулу из некоторого стабильного состояния LTS-спецификации не всегда понимается как блокировка. Для соответствий без блокировок ***iot***, ***ior***, ***ioconf*** и ***ioco*** такой стимул следует считать *неспецифицированным*. При этом, если трасса (трасса без отказов для ***iot*** и ***ioconf*** или δ-трасса для ***ior*** и ***ioco***) продолжается стимулом в одних состояниях, а в других состояниях стимул не специфицирован, то последние состояния просто игнорируются, когда мы определяем требуемое поведение по поводу этого стимула. Если же после трассы стимул не специфицирован во всех состояниях, то это означает отсутствие каких-бы то ни было требований к реализации по поводу этого стимула после этой трассы.

Для соответствий, учитывающих блокировки, тем не менее, в спецификации также могут быть неспецифицированные стимулы.

Для ***mioco*** спецификация – это обычная LTS (не обязательно MIOTS). Поэтому, если в конце трассы (***f-trace***) отсутствуют переходы по каждому стимулу из входного канала, то это понимается как отказ. Если же трасса одним стимулом из канала продолжается, а другим нет, то второй стимул понимается как неспецифицированный, что также не накладывает никаких ограничений на поведение реализации по поводу этого стимула, за исключением того, что стимул должен приниматься.

Для ***rioco*** спецификация – это RIOLTS, в которой блокировка стимула явно специфицируется, а неспецифицированный стимул – это стимул, по которому в состоянии не определён переход и не определена блокировка. Если после трассы стимул не принимается ни в каком состоянии и в некотором состоянии неспецифицирован, то разрешается любое поведение:

приём стимула с любым продолжением или блокировка. Если же после трассы стимул в одном состоянии принимается, а в другом не специфицирован, то разрешается блокировка или приём стимула, но уже не с любым поведением после приёма стимула, а с тем, которое спецификация разрешает после трассы, продолженной стимулом.

И только для вводимого нами отношения $\boldsymbol{ioco}_{\beta\gamma\delta}$ любая LTS-спецификация не содержит неспецифицированных стимулов: в любом стабильном состоянии отсутствие перехода по стимулу трактуется как его блокировка.

В отличие от спецификации, в реализации не может быть неспецифицированных стимулов: в каждом состоянии стимул либо принимается, либо блокируется. Поэтому возникает естественная задача такого преобразования спецификации, после которого в ней не остаётся неспецифицированных стимулов. Такое преобразование называется *пополнением спецификации*.

Различают пополнение состояний и трассовое пополнение в зависимости от того, о каких стимулах идёт речь: неспецифицированных в состоянии или после трассы.

Пополнение спецификации может быть подразумеваемым (допущение полноты) или преследовать цель «улучшения» спецификации. Можно выделить две такие цели, которые мы рассмотрим ниже: 1) рефлексивность и 2) монотонность соответствия.

1.8.2. Допущение полноты (подразумеваемое пополнение).

Если поведение реализации по поводу неспецифицированного стимула не регламентируется, то такое отношение конформности называется *слабым* [BP94]. Соотвественно, тестирование, не проверяющее это поведение реализации, называется *слабым тестированием* [LY96].

Вместе с тем, иногда неспецифицированность стимула означает не отсутствие требований к поведению реализации, а «поведение по умолчанию». Предполагается, что существует некоторая трактовка, определяющая такое поведение по умолчанию. Её обычно называют допущением полноты – completeness assumption. Для таких стимулов отношение и тестирование будут *сильными*. Пополнений на основе тех или иных допущений полноты предложено довольно много. Обзор различных видов пополнений можно найти в [BP94, LY96]. Если неспецифицированный стимул понимается как блокируемый, то мы будем считать это допущением полноты по умолчанию, что также предполагает сильное отношение и сильное тестирование.

Заметим, что то или иное допущение полноты делается ещё до того, как встаёт вопрос об отношении конформности. Везде, где речь идёт о спецификации, теперь мы должны иметь в виду спецификацию пополненную в соответствии с тем или иным допущением полноты.

1.8.3. Проблема рефлексивности соответствия.

Наличие в спецификации неспецифицированных стимулов (явно, как в ***rioco*** или неявно как в других отношениях) выделяет в классе моделей класс спецификаций ***SPEC*** и не совпадающий с ним класс реализаций ***IMPL***. В частности, для ***iot***, ***ior***, ***ioconf*** и ***ioco*** ***IMPL=IOTS*** и ***SPEC=LTS***. Поэтому отношение конформности, вообще говоря, оказывается нерефлексивным.

Заметим, что для отношений ***quiescent trace preorder*** и $\boldsymbol{ioco}_{\beta\gamma\delta}$ классы спецификаций и реализаций совпадают: ***IOTS*** и ***LTS***, соответственно.

Нерефлексивность означает, что спецификация не может быть понята как конформная ей самой реализация. Мы считаем это недостатком. Во-первых, это противоречит интуитивному пониманию спецификации как набору требований, которые можно реализовать «как есть». Во-вторых, это делает невозможным или затрудняет решение первой из трёх задач, указанных выше как области применения спецификаций: спецификацию нельзя использовать как прототип реализации. Если реализацию написать как эксплицитное выражение спецификации, то мы получим неконформную реализацию.

Решением этой проблемы могло бы быть использование эквивалентной, но конформной самой себе, спецификации. Здесь эквивалентность спецификаций понимается как совпадение множеств реализаций, конформных этим спецификациям. Модель $S \in \boldsymbol{SPEC} \setminus \boldsymbol{IMPL}$ не конформна сама себе, но модель $S` \in \boldsymbol{SPEC} \cap \boldsymbol{IMPL}$ для всех «естественных» отношений всегда конформна сама себе. Поэтому задача ставится так: найти спецификацию-реализацию $\boldsymbol{Comp}(S) \in \boldsymbol{SPEC} \cap \boldsymbol{IMPL}$, которая была бы эквивалентна заданной спецификации $S \in \boldsymbol{SPEC} \setminus \boldsymbol{IMPL}$. Эта задача решается соответствующим пополнением спецификации S.

Глава 1.9. Асинхронное тестирование и верификация композиции

Структура главы:
1. Две проблемы асинхронного тестирования.
2. Безопасность при асинхронном тестировании.
3. Дивергенция при асинхронном тестировании.
4. Проблема монотонности соответствия при композиции.

1.9.1. Две проблемы асинхронного тестирования.

Как было сказано выше, при асинхронном тестировании реализация погружена в тестовый контекст. Для LTS-модели тестовый контекст понимается как среда передачи стимулов и реакций, которая также моделируется LTS и располагается между тестом и реализацией.

При асинхронном тестировании отношение конформности **I** ***rel*** **S** между реализацией **I** и спецификацией **S** заменяется на отношение между композицией реализации и среды **I⌈⌊Q** и композицией спецификации и среды **S⌈⌊Q**, которое можно понимать как новое отношение между реализацией и спецификацией: **I** ***Qrel*** **S** = **I⌈⌊Q** ***rel*** **S⌈⌊Q**.

С асинхронным тестированием связаны две основные проблемы. Мы рассмотрим их на примере стандартной среды как совокупности двух неограниченных FIFO-очередей: входной для стимулов и выходной для реакций. Эти проблемы поставлены в [JJTV99], где они называются: «вседозволенность» и «несохранение соответствия».

Вседозволенность (Permissiveness, Рис.2 слева) означает ослабление соответствия: некоторые ошибки, которые ловятся при синхронном тестировании, не ловятся асинхронными тестами. Эта проблема является неизбежным следствием «удалённости» теста от реализации: среда вносит дополнительный недетерминизм, не управляемый тестом. Это неизбежное ослабление.

Несохранение соответствия (Non preservation of conformance, Рис.2 справа) означает, наоборот, усиление соответствия: асинхронный тест ловит «ложные» ошибки, не обнаруживаемые при синхронном тестировании. Эта проблема более серьёзна и с ней нужно бороться. Несохранение соответствия возникает при слабом тестировании, когда стимул, неспецифицированный после трассы, не посылается при синхронном тестировании, но может быть послан при асинхронном тестировании,

поскольку неограниченная входная очередь не блокирует стимулы от тестера.

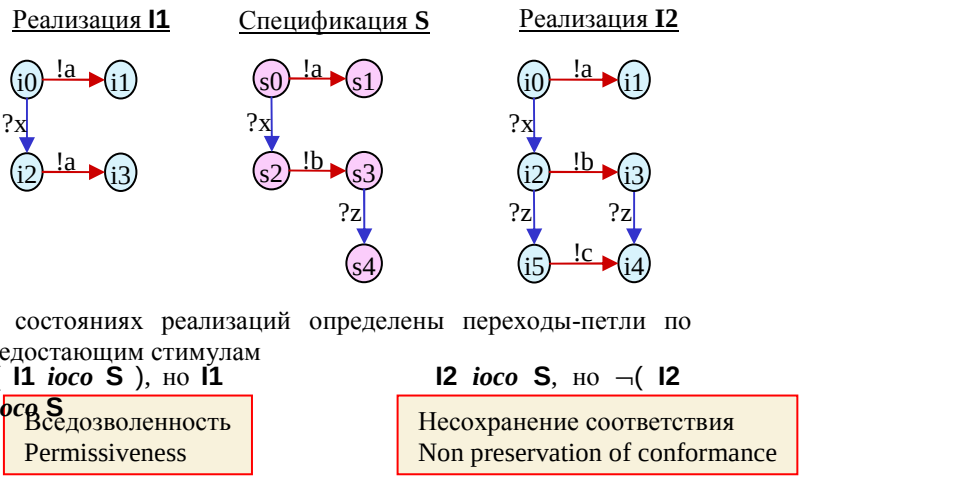


Рис.2.

Для решения проблемы несохранения соответствия предложено несколько подходов.

В [JJTV99] для отношения ***ioconf***, когда трассы не продолжаются после стационарности, применяется трассовое пополнение: стимул, неспецифицированный после трассы, принимается с произвольным дальнейшим поведением (кроме разрушения, которое в этой работе не рассматривается). Введение такого произвольного поведения называют ещё *демоническим пополнением*. В [BRT03] для отношения ***ioco*** предлагается демоническое пополнение состояний, но это приводит к ослаблению соответствия не только при асинхронном, но и при синхронном тестировании. Оба варианта пополнения (трассового и в состояниях) имеют общий недостаток: «лишнее» тестирование в асинхронном случае, поскольку теряется информация о неспецифицированности стимулов.

В настоящей работе мы описываем для отношения ***ioco*** трассовое *гамма-пополнение*, которое не меняет соответствие при синхронном тестировании, не усиливает его при асинхронном тестировании, а ослабление даёт только «неизбежное». В отличие от демонического пополнения, информация о неспецифицированных стимулах сохраняется: такие стимулы становятся разрушающими. Это даёт возможность избавиться от «лишнего» тестирования в асинхронном случае[8].

[8] Для реализаций без блокировок и разрушения асинхронное тестирование при демоническом пополнении ловит больше ошибок, чем при гамма-пополнении. Причина этого в том, что домен гамма-пополнения шире, и допускается разрушение после

Во всех этих вариантах пополнения спецификации проблема несохранения соответствия снимается: у нас при любом (синхронном или асинхронном) безопасном тестировании неспецифицированный стимул нельзя посылать, поскольку он может разрушить реализацию, а в вариантах [JJTV99,BRT03] – посылать стимул бессмысленно, поскольку допустимо любое поведение, хотя из-за потери информации при асинхронном тестировании стимул всё равно посылается и соответствующие проверки выполняются.

Важно отметить, что для отношений, не учитывающих блокировок, пополнение, удаляющее все блокировки, решает проблему несохранения соответствия не только для стандартной среды. Оно годится также для любой среды без блокировок, то есть не тормозящей реакции, выдаваемые реализацией, и стимулы, посылаемые тестером.

1.9.2. Безопасность при асинхронном тестировании.

При наличии разрушения для стандартной среды передачи отношение ***Qrel*** отличается от отношения ***rel*** условиями безопасности. Они перераспределяются «от реакций к стимулам»:

1) приём реакций в тестере становится всегда безопасным, поскольку реакции принимаются не из реализации непосредственно, а из неограниченной выходной очереди, которая, в свою очередь, всегда принимает реакции от реализации;
2) стимул становится разрушающим не только в том случае, когда после его приёма возможно разрушение, но и в том случае, когда между стимулом и разрушением оказывается любая конечная последовательность реакций, которая всегда может быть выбрана выходной очередью независимо от поведения тестера.

Соответствующим образом модифицируется понятие безопасной трассы и гипотеза о безопасности. В целом теория сохраняется и при асинхронном тестировании: аналогичным образом можно определить тесты и генерацию полного набора тестов.

1.9.3. Дивергенция при асинхронном тестировании.

Как было сказано выше, дивергенция – нежелательное поведение реализации, поскольку она неотличима от отказа. При асинхронном тестировании композиция реализации и среды наследует дивергенцию

стимула. Если отсутствие блокировок и разрушения специфицировать явно, то гамма-пополнение сведётся к демоническому пополнению.

компонентов, поскольку τ-переходы выполняются асинхронно. Обычно среда не имеет дивергенции, так что речь идёт о дивергенции реализации.

В то же время может появиться «новая» дивергенция, в частности, когда в реализации дивергенции не было. Например, бесконечная трасса реакций в реализации (осцилляция) приводит к дивергенции в композиции реализации с неограниченной выходной FIFO-очередью. Поэтому обычно налагают запрет на осцилляцию в реализации [PYH03]. Заметим, что решение – считать осцилляцию нежелательным поведением и моделировать её разрушением – не годится. Дело в том, что осцилляция нежелательна только при асинхронном тестировании, а спецификация остаётся той же самой и при синхронном тестировании.

1.9.4. Проблема монотонности соответствия при композиции.

Проблема несохранения соответствия при асинхронном тестировании является частным случаем проблемы композиции системы. В самом общем виде она 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 её соответствие спецификации системы. Разработчик компонента руководствуется единственным документом – техническим заданием, формальной моделью которого является спецификация компонента. Правильность реализации компонента проверяется синхронным тестированием компонента. Очевидно, одной из причин неправильной работы системы является то, что, на самом деле, её компоненты работают неправильно, а 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 её 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.

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. *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, но, во-первых, это определение неявно, во-вторых, не ясно, существует ли такая самая сильная корректная спецификация, и, в-третьих, неизвестно, как такую спецификацию построить.

При «наивном» рассмотрении первой неожиданностью оказывается то, что самая сильная корректная спецификация системы, если она существует, оказывается слабее композиции спецификаций, если эту композицию проводить по тем же правилам, что и композицию реализаций компонентов при сборке системы. Такая композиция спецификаций часто оказывается слишком сильной: она предъявляет лишние требования к реализации системы. В результате композиция конформных реализаций компонентов оказывается неконформной композиции спецификаций. При тестировании ловятся «ложные» ошибки. Это и есть несохранение соответствия, в частности, проявляющееся при асинхронном тестировании.

Следовательно, если обычную композицию называть *прямой*, то оказалось, что требуется иная композиция спецификаций (мы её называем *косой*). Она должна вычислять самую сильную корректную спецификацию системы, если она существует, как функцию спецификаций компонентов при заданной схеме компоновки (сборки) системы.

Причиной такого положения является разный уровень абстракции, используемый в определении соответствия и в определении прямой композиции. Соответствия основаны на наблюдаемых действиях, то есть, трассовой модели, а композиция – на полной, автоматной модели,

дополнительно учитывающей состояния, ненаблюдаемые действия и соотношение состояний и действий.

Таким образом, можно поставить общую задачу: определить косую композицию для выбранного отношения конформности.

Пополнение спецификаций – это самый первый подход к решению этой проблемы. Он основан на интуитивном понимании того, что для отношений, не учитывающих блокировки, причина несохранения соответствия – в наличии блокировок в спецификации. Поэтому от блокировок стремятся избавиться, пополняя спецификацию, но сохраняя композицию спецификаций по тем же правилам, что и композицию реализаций.

Одна из целей данной работы – определение косой композиции для отношения $\boldsymbol{ioco}_{\beta\gamma\delta}$. Как следствие, мы получим косую композицию для отношения $\boldsymbol{ioco}$ (отсутствуют блокировки и разрушение), применимую при различных пополнениях не полностью определенных спецификаций. Более того, мы покажем, что, при определённых условиях, косая композиция совпадает с прямой композицией *преобразованных* спецификаций. Если такое преобразование существует, то соответствие будем называть монотонным относительно этого преобразования, а преобразование – монотонным для этого соответствия. Таким образом, задача сводится к нахождению такого (хотя бы одного) преобразования спецификаций.

Асинхронное тестирование – это тестирование системы из двух компонентов, один из которых – реализация, а другой – фиксированная и известная среда передачи. В этом случае монотонное преобразование применяется только к спецификации реализации, а среда остаётся неизменной. В этом случае мы будем говорить о *левомонотонности* преобразования.

В дальнейшем, по умолчанию, рассматриваются только конечные последовательности.


СИНХРОННОЕ
ТЕСТИРОВАНИЕ

Часть 2. Синхронное тестирование

Структура части:

1. Машина тестирования

   Описывается формализация тестового эксперимента с помощью машины тестирования. Кроме общей машины тестирования, рассматривается её разновидность – βγδ-машина, порождающая βγδ-трассы и используемая для тестирования по соответствию $ioco_{\beta\gamma\delta}$.

2. Трассовые модели

   Излагается формальная трассовая теория, в которой моделью системы является дерево βγδ-трасс – трасс, включающих стационарность и блокировки. Также рассматриваются модели безопасных и финальных трасс. Определяется гипотеза о безопасности и соответствие $ioco_{\beta\gamma\delta}$. Описывается генерация тестов и обсуждаются вопросы алгоритмизации, имея в виду практическое тестирование.

3. LTS-модель

   Модель определяется как система размеченных переходов (LTS – labelled transition system), а взаимодействие моделируется параллельной композицией LTS. Показывается эквивалентность трассовой и LTS-моделей, что даёт возможность использования трассовой теории для LTS, в частности, для генерации LTS-тестов. Обсуждаются специфические для LTS вопросы алгоритмизации.

4. Сравнение моделей

5. Пополнение спецификации

   Изучаются различные виды пополнения частично-определённых спецификаций. Задача такого пополнения – устранить неопределённость в трактовке неспецифицированного стимула на основе того или иного допущения полноты. Пополнение используется для решения проблемы несохранения соответствия при асинхронном и композиционном тестировании. Мы используем его также для прояснения классической семантики отношения $ioco$, в частности, сравнением с семантикой отношения $ioco_{\beta\gamma\delta}$.

Синхронное тестирование – это такое тестирование, когда тестер и реализация взаимодействуют друг с другом непосредственно. Тестовый контекст отсутствует. В этом случае тестер может обладать максимальными тестовыми возможностями, допускаемыми реализацией и операционной средой, в которой выполняются реализация и тестер.

Такое непосредственное взаимодействие встречается на практике и широко распространено. Это обычная отладка, когда реализация находится под управлением тестера настолько полным, насколько это возможно. Вообще говоря, разработчик может выбирать различные уровни абстракции при

взгляде на реализацию, поскольку он знает не только спецификацию реализации, но и устройство самой реализации. Однако в этом исследовании мы будем рассматривать уровень абстракции, задаваемый спецификацией. Иными словами, мы будем считать, что единственная цель отладки – проверка того, что реализация делает то, что от неё требуется, то есть проверка соответствия реализации спецификации.

При отладке разработчик, обычно, имеет возможность наблюдать, разрешать или запрещать (приостанавливать) отдельные действия реализации. Когда эти действия выражены в терминах спецификации, такая отладка как раз и соответствует понятию синхронного тестирования.

Глава 2.1. Машина тестирования

Структура главы:
1. Машина общего вида
2. Машина для $ioco_{\beta\gamma\delta}$-тестирования – βγδ-машина

Теория тестирования соответствия базируется на выбранной математической метамодели, в терминах которой трактуются спецификация, реализация, их соответствие, тестер и взаимодействие тестера с реализацией [ISO95].

Спецификационная модель считается известной. При этом она может быть задана как явно, так и неявно, например, пред- и постусловиями. В последнем случае модель задаётся, фактически, системой уравнений, которая не всегда может быть удовлетворительно решена алгоритмическим способом. Тем не менее, спецификационная модель и в этом случае считается известной. В настоящей работе мы предполагаем такое задание спецификационной модели, которое позволяет её «строить» итеративно.

Понимание реализации как модели формулируется в виде тестовой гипотезы [Bern91]: «Существует такая (реализационная) модель, наблюдаемое поведение которой неотличимо от поведения реализации в терминах выбранного уровня абстракции при любом функциональном тестировании». Заметим, что, в отличие от спецификационной модели, тестовая гипотеза декларирует только существование реализационной модели, и это не означает, что модель известна.

Выбор модели и соответствия определяется тем способом, каким может вестись тестирование, понимаемое как взаимодействие тестера и реализации. Это способ задаётся, так называемым, *тестовым сценарием*. Тестовый сценарий обычно определяется описанием некоей машины тестирования, которая, фактически, задаёт тот или иной интерфейс с реализацией. Реализация как бы помещена внутрь этой машины и взаимодействие с ней осуществляется в терминах машины (Рис.3).

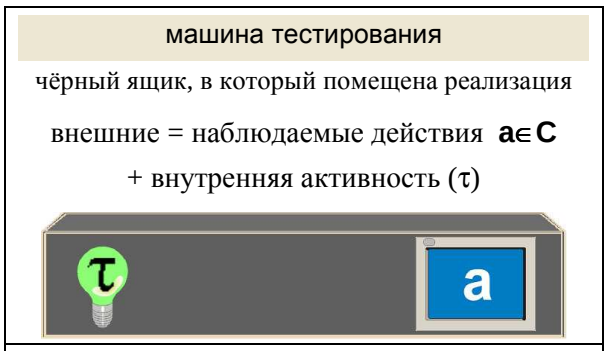


Рис.3.

Различают реактивные машины, введённые Milner [Miln81], и генеративные машины, введённые Glabbeek [Glab90,Glab93] (Рис.4). Реактивная машина функционирует «по приказу», реагируя на команды оператора, подаваемые с

помощью нажатия кнопок (одна кнопка соответствует одному действию). Генеративная машина выполняется как бы «сама по себе», но оператор имеет возможность разрешать или запрещать те или иные её действия с помощью переключателей (один переключатель соответствует одному действию). Различие между этими машинами, как показано в [Glab93], относится скорее к способу описания, чем к существу дела[9].

Рис.4.

В обоих видах машины предполагается возможным поведение машины, которое называется внутренней активностью. Оно означает, что машина «что-то делает», но что именно, неизвестно и ненаблюдаемо. Часто машина снабжается «зелёной лампочкой», которая показывает сам факт наличия внутренней активности и гаснет, когда машина останавливается.

В этой главе мы опишем *машину с ограниченным управлением*, из которой впоследствии выведем различные, но в каком-то смысле эквивалентные, модели. Нашу машину можно понимать как разновидность генеративной машины, в которой оператор может устанавливать не любое положение переключателей, или как реактивную машину, в которой можно нажимать несколько кнопок сразу, но не любое множество кнопок. Вместо зелёной лампочки мы используем реакцию машины, которую называем *отказом*, и которая возникает в той же ситуации, в которой гаснет зелёная лампочка при наличии разрешённых действий. Кроме того, вместо дисплея, на который машина выводит информацию для оператора, мы используем печатающее устройство (выбор дисплея или печати, очевидно, дело вкуса).

Нововведением является специальное поведение машины, которое мы будем называть *разрушением*. Разрушение считается условно-наблюдаемым

[9] Если в реактивной машине разрешить нажимать сразу несколько кнопок, предоставляя машине самой выбирать выполняемое действие.

поведением: в отличие от внутренней активности, оно наблюдаемо[10], но при тестировании мы должны избегать такого наблюдения.

Сначала мы опишем общий вид такой машины, реализующей семантику трасс с ограниченными отказами. Под ограничениями здесь имеются в виду ограничения, налагаемые на возможные действия оператора, то есть на положение переключателей в генеративной машине или на множество одновременно нажимаемых кнопок в реактивной машине. В терминах наблюдаемых действий такие ограничения означают ограничения на отказы, которые может выдавать машина. При отсутствии разрушений и ограничений мы имеем чистую семантику трасс с отказами (failure trace semantics). При отсутствии разрушений и ограничении, заключающемся в том, что все действия всегда разрешены, мы получаем семантику завершённых трасс (completed trace semantics). (см. [Glab90])

Далее мы рассмотрим разновидность машин, наблюдаемые действия которых трактуются как передача информации в машину (стимул) или из машины (реакция). В настоящей работе мы вводим обобщение соответствия ***ioco*** для систем, в которых возможно разрушение и блокировка передачи стимула. Такое обобщённое соответствие мы будем называть соответствием $\boldsymbol{ioco}_{\beta\gamma\delta}$ и опишем специальную βγδ-машину, предназначенную для его тестирования.

2.1.1. Машина общего вида

Структура раздела:
- Внешние действия и внутренняя активность.
- Разрушение.
- Отказы.
- Управление.
- Приоритеты.
- Наблюдение.
- Недетерминизм и погодные условия.
- Принципы работы машины.
- Дивергенция.
- Правило сбывающихся ожиданий.
- Безопасность.
- Конвергентность.
- Гипотеза о безопасности.

---

[10] Наблюдаемость разрушения – чистая абстракция, проскольку мы требуем, чтобы в любом тестовом эксперименте машина не разрушалась. Смысл этой абстракции – обозначить нежелательное поведение как разрушение и избегать его.

· Формальные правила, определяющие машину тестирования.

Внешние действия и внутренняя активность.

Машина представляет собой «чёрный ящик»: оператор не видит устройства машины, её внутреннего состояния и внутренней работы, но может взаимодействовать с машиной, воздействуя на неё извне и наблюдая ответное внешнее поведение машины. Для машины фиксируется множество C *внешних*, *наблюдаемых* дискретных действий. Внешнее действие $a \in C$ является обозначением для множества внешних действий машины, выполнение каждого из которых наблюдаемо оператором как “a”, то есть для него они неразличимы между собой. Кроме наблюдаемых действий, машина может иметь *внутреннюю активность*, которая внешне не наблюдаема. Мы будем обозначать её символом τ. Внутренняя активность может быть как конечной, то есть, исчезающей через какое-то время, так и бесконечной. Бесконечную внутреннюю активность ещё называют *дивергенцией*. Взаимодействие с машиной сводится к тому, что оператор указывает, какие внешние действия машине разрешено выполнять, а какие запрещено, и наблюдает те внешние действия, которые происходят.

Разрушение.

Мы вводим также дополнительное поведение машины, которое будем называть *разрушением* и обозначать символом γ. Разрушение можно понимать сколь угодно широко – это любое нежелательное поведение машины, которое мы должны избегать при тестировании. В частности, семантика разрушения включает реальное разрушение машины, после которого любые наблюдения невозможны или недостоверны. Поэтому будем считать, что после разрушения никакие наблюдения невозможны. В настоящей работе по причинам, которые мы укажем ниже, к такому нежелательному поведению относится также дивергенция. Разрушение считается условно-наблюдаемым поведением: в отличие от внутренней активности, оно наблюдаемо, но при тестировании мы должны избегать такого наблюдения.

Отказы.

В машине может возникнуть deadlock, когда она ничего не может делать. Это означает, что в данный момент времени в машине нет внутренней активности, нет разрушения и нет выполнимых разрешённых действий. В состоянии deadlock`а машина сообщает оператору о невозможности выполнить разрешённые действия, если такие действия есть. Такое сообщение мы будем называть *отказом*.

Управление.

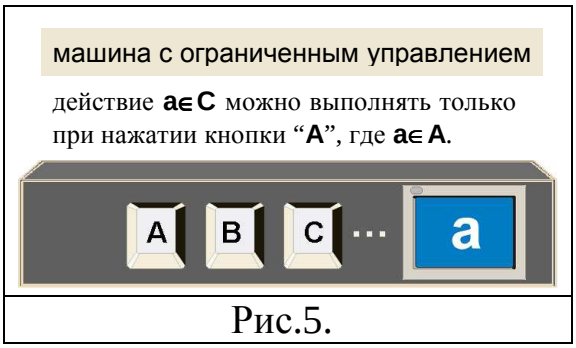


Рис.5.

Для разрешения и запрещения действий машина снабжена клавиатурой – набором кнопок с написанными на них символами действий (Рис.5); на одной кнопке может быть написано множество символов действий. (Очевидно, это эквивалентно набору кнопок, на каждой из которых написано только одно действие, но нажимать разрешается несколько кнопок, хотя, может быть, не любое множество кнопок.) Нажатая кнопка означает разрешение действий, символы которых написаны на кнопке. Оператор имеет право нажать не более одной кнопки. Нажатая кнопка блокирует клавиатуру до тех пор, пока машина не выполнит одно действие, символ которого написан на кнопке, или не сообщит об отказе выполнить какое бы то ни было из разрешённых действий. При выполнении такого действия или при отказе машина сама отжимает кнопку и клавиатура разблокируется (оператор получает возможность нажать ту же самую или другую кнопку).

Кроме того, поскольку нам требуется прогонять различные тесты, нужна возможность «перезапуска» машины тестирования. Для этого используется специальная кнопка *reset*, отжимающая все кнопки и заставляющая машину работать с начала.

Приоритеты.

В настоящей работе мы рассматриваем системы *без приоритетов*. Это означает, что разрешённые действия не влияют друг на друга: возможность выполнения действия a определяется только тем, что нажата кнопка, на которой написано множество символов A, содержащее a, и не зависит от того, какие ещё символы содержит это множество. Если в данной ситуации может наблюдаться действие a при нажатии одной кнопки, разрешающей a, то оно может наблюдаться также при нажатии любой другой кнопки, разрешающей a.

Более того, мы предполагаем равноприоритетность всех разрешённых действий, внутренней активности и разрушения машины.

Как было сказано в 1.7.4, отсутствие приоритетов, в частности, означает, что в реализации нет θ-действий. При возникновении тупика во взаимодействии с тестером реализация останавливается, то есть в ней нет альтернативного поведения. В этом случае срабатывает тайм-аут кнопки машины тестирования, что соответствует θ-наблюдению в тестере.

Наблюдение.

Для наблюдения за выполняемыми внешними действиями и отказами машина снабжается печатающим устройством. Каждый раз, когда выполняется наблюдаемое действие a, машина печатает символ a. Для систем без приоритетов это полная информация, поскольку неважно, какую именно кнопку мы нажимали среди тех кнопок, которые разрешают действие a. Сообщение об отказе также выводится на печать. Мы можем считать, что сообщение об отказе – это множество символов действий, написанных на нажатой кнопке (мы различаем символ и множество, состоящее из одного этого символа): машина отказывается выполнить каждое действие, символ которого написан на кнопке. Про действия, не разрешаемые нажатой кнопкой, мы ничего не знаем, то есть, не знаем, могла ли выполнить их машина, если бы они были разрешены, или не могла. Поскольку разрушение считается условно-наблюдаемым поведением машины, при его возникновении на печать выводится символ разрушения γ, после чего на печать больше ничего не выводится.

Последовательность напечатанных внешних действий и множеств действий (отказов), быть может, заканчивающаяся символом разрушения – это и есть последовательность наблюдаемых событий, которую мы называем *трассой*. Мы предполагаем, что, с точки зрения внешнего наблюдателя, о машине нельзя узнать ничего, кроме множества её конечных наблюдаемых трасс. Заметим, что формально мы могли бы запоминать не только наблюдаемые действия и отказы, но и нажимаемые кнопки. Так и нужно делать для машины общего вида. Однако для машины без приоритетов это не требуется, так как при наблюдении действия a не имеет значения, какую кнопку, разрешающую действие a, мы нажимали (отказ, как множество $A \subseteq C$ отвергаемых действий, однозначно указывает на то, что нажималась кнопка “A”).

Недетерминизм и погодные условия.

Предполагается, что оператор обладает достаточными возможностями для того, чтобы нажимать или не нажимать те или иные кнопки в любые моменты времени, когда клавиатура разблокирована. Кроме того, машина может быть недетерминированной. Это означает, что её работа зависит не только от работы оператора (нажатие тех или иных кнопок в те или иные моменты времени), но и от неких не учитываемых внешних факторов – погодных условий. Предполагается, что любые погодные условия могут быть воспроизведены в эксперименте. Это можно понимать так, что есть много кнопок *reset* – по одной для каждого возможного отображения линейного времени (множества моментов времени) во множество всех

возможных погодных условий. Другим вариантом является наличие кнопки *репликации*, которая позволяет оператору в любой момент времени создать множество копий машины с одним и тем же текущим состоянием – по одной для каждого варианта погодных условий – и далее продолжать эксперимент независимо с каждой из копий, начиная с привязанного к ней варианта погодных условий на данный момент времени [Glab90,Glab93]. Для той же цели может использоваться кнопка *откат* (*undo*), позволяющая вернуться на шаг назад и продолжить эксперимент с новым вариантом погодных условий для этого шага [Glab90].

Принципы работы машины.

Рассмотрим поведение машины в зависимости от наличия или отсутствия внутренней активности, выполнимых действий и разрушения.

1) Если в данный момент времени машина не имеет внутренней активности и не может разрушиться, будем говорить, что машина находится в *стабильном* состоянии. В этом состоянии возможны три случая:
   a) Если ни одна кнопка не нажата, машина «стоит»: ничего не делает.
   b) Если имеется нажатая кнопка, и хотя бы одно из разрешённых действий выполнимо, машина обязана выполнить любое из этих действий. Какое именно действие – определяется погодными условиями.
   c) Наконец, если имеется нажатая кнопка, но ни одно из разрешённых этой кнопкой действий не выполнимо, машина должна напечатать отказ.

   При печати отказа машина автоматически отжимает кнопку, после чего оператор имеет возможность нажать эту же или другую кнопку. Если в этой ситуации оператор не нажал кнопку действий, машины продолжает «стоять». Таким образом, в стабильном состоянии на каждое нажатие кнопки машина печатает либо символ одного из действий, разрешаемых этой кнопкой, либо отказ.

   Предполагается, что после печати отказа машина остаётся в том же самом стабильном состоянии. Это означает, что после печати отказа поведение машины точно такое же, как и до печати, за исключением того, что машина не будет повторно печатать отказ, пока не будет нажата кнопка действий, которую машина автоматически отжимает при печати отказа.

Опишем это формально (Рис.6).

Обозначим через $\Sigma(s)$ множество трасс, которые можно наблюдать после того, как машина попала в стабильное состояние s.

Пусть наблюдается отказ o, то есть $\langle o\rangle\in\Sigma$.

Обозначим через $\Sigma(s,o)$ множество трасс, которые можно наблюдать после того, как машина попала в стабильное состояние s, и далее после наблюдения отказа o: $\Sigma(s,o)=\{\lambda\,|\,\langle o\rangle\cdot\lambda\in\Sigma(s)\}$. Тогда $\Sigma(s,o)=\Sigma(s)$.

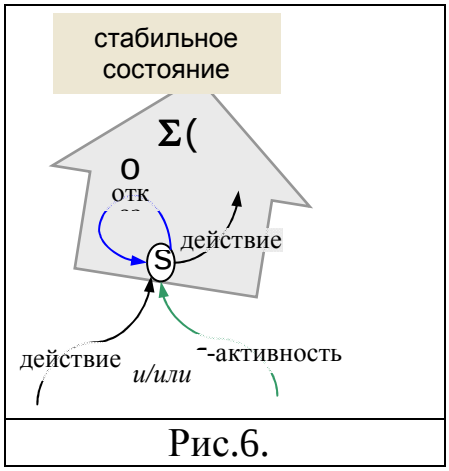


Рис.6.

2) Пусть в данный момент времени машина не имеет внутренней активности, но может разрушиться (Рис.7). При отсутствии выполнимых разрешённых действий, машина должна разрушиться. При наличии выполнимых разрешённых действий, машина должна выполнить одно из выполнимых разрешённых действий или разрушиться. Выбор зависит от погодных условий. Считается, что для каждого варианта возможны такие погодные условия, когда этот вариант реализуется.

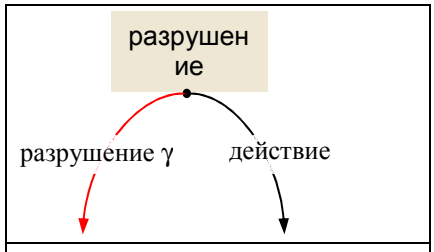


Рис.7.

3) Пусть в данный момент времени машина имеет внутреннюю активность (Рис.8). Тогда, если у машины есть возможность выполнить внешнее разрешённое действие, то она может, как выполнить, так и не выполнить его и остаться во внутренней активности. Точно также, если машина может разрушиться, то она может как разрушиться, так и не разрушиться и остаться во внутренней активности. Вместе с тем считается, что само наличие такой возможности означает, что возможны такие погодные условия, когда эта возможность реализуется, то есть, машина выполнит данное разрешённое действие или разрушится при наличии внутренней активности. Аналогично, возможны и такие погодные условия, когда эта возможность не реализуется: машина остаётся во внутренней активности. Что именно делает машина во внутренней активности, когда она не выполняет разрешённых действий, оператору неизвестно.

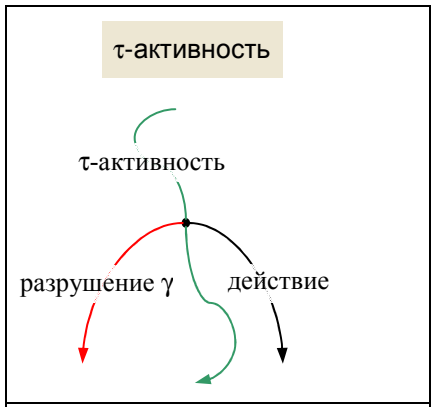


Рис.8.

Дивергенция.

Конечная внутренняя активность – это такая активность, которая при любых погодных условиях исчезнет «сама по себе»: машина через конечное время окажется в стабильном состоянии или разрушится. Бесконечная внутренняя активность (дивергенция) – это такая активность, которая при некоторых погодных условиях не исчезает «сама по себе»: машина сколь угодно долго не переходит в стабильное состояние. Заметим, что, если трактовать дивергенцию как частный случай разрушения, то машина при любых погодных условиях либо переходит в стабильное состояние, либо разрушается.

Правило сбывающихся ожиданий.

Предполагается, что, с точки зрения оператора, машина не меняется в процессе работы, меняться может только её «состояние». Это можно назвать *правилом сбывающихся ожиданий*, которое описывается следующим образом. Пусть кто-то, кто знает внутреннее устройство машины, в начале её работы ожидает, что при некоторых условиях будет наблюдаться трасса σ. Тогда, во-первых, любое продолжение λ трассы σ, которое можно ожидать после того, как трасса σ будет получена, ожидалось и с самого начала, то есть, ожидалась трасса $\sigma{\cdot}\lambda$. Во-вторых, для любой трассы $\sigma{\cdot}\lambda$, которая ожидалась с самого начала, возможно такое выполнение машины, при котором она будет получена (сначала σ, а после неё λ).

Опишем это формально. Обозначим через $\boldsymbol{\Sigma}$ множество трасс, которые могут ожидаться в начале работы при всех возможных действиях оператора и всех возможных погодных условиях. Для $\sigma\in\boldsymbol{\Sigma}$ через $\boldsymbol{W}(\sigma)$ обозначим множество всех возможных действий оператора и всех возможных погодных условий, которые обеспечивают получение к некоторому моменту времени трассы σ. Для $\boldsymbol{w}\in\boldsymbol{W}(\sigma)$ через $\boldsymbol{\Sigma}(\boldsymbol{w})$ обозначим множество трасс, которые могут наблюдаться после $\boldsymbol{w}$ при всех возможных дальнейших действиях оператора и погодных условиях. Неизменяемость машины означает, что любая трасса $\lambda\in\boldsymbol{\Sigma}(\boldsymbol{w})$, то есть, трасса, ожидаемая после $\boldsymbol{w}$, ожидалась в начале работы как продолжение трассы σ, то есть, $\lambda\in(\boldsymbol{\Sigma}\ \boldsymbol{after}\ \sigma)$, и, наоборот, любая трасса $\lambda\in(\boldsymbol{\Sigma}\ \boldsymbol{after}\ \sigma)$ ожидается после некоторого $\boldsymbol{w}\in\boldsymbol{W}(\sigma)$, то есть, $\lambda\in\boldsymbol{\Sigma}(\boldsymbol{w})$. Это эквивалентно $\forall\sigma\in\boldsymbol{\Sigma}\ \boldsymbol{\Sigma}\ \boldsymbol{after}\ \sigma = \cup(\{\boldsymbol{\Sigma}(\boldsymbol{w})\,|\,\boldsymbol{w}\in\boldsymbol{W}(\sigma)\})$.

Безопасность.

Семантика разрушения предполагает, что разрушение может быть только последним наблюдением в тестовом эксперименте: после разрушения ничего не может наблюдаться или, точнее, всё, что может быть после разрушения, нас не интересует. Мы будем считать, что оператор должен избегать разрушения, а такое тестирование будем называть *безопасным*.

Будем говорить, что машина *безопасна* (не саморазрушается), если в любом эксперименте, в котором не нажимаются никакие кнопки, она не разрушается. Очевидно, саморазрушающаяся машина не может тестироваться: такую машину нельзя «включать», так как она может разрушиться ещё до первого тестового воздействия (нажатия кнопки).

В безопасной машине разрушение может возникнуть после некоторой наблюдаемой трассы μ в результате выполнения действия a. Такое действие a будем называть *разрушающим* после трассы μ, то есть имеется трасса $\mu\cdot\langle a\rangle\cdot\langle\gamma\rangle$. Любую трассу вида $\mu\cdot\langle a\rangle\cdot\lambda$ будем называть *разрушающей трассой*, иными словами, разрушающая трасса – это трасса, которая заканчивается разрушением или имеет продолжение или ответвление разрушением (Рис.9).

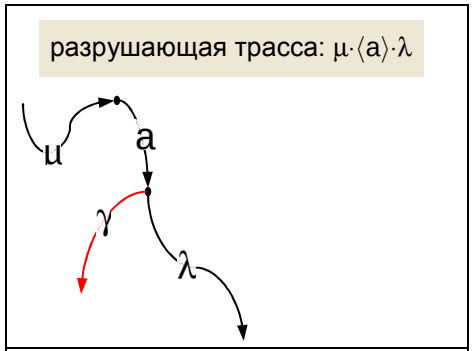


Рис.9.

Разрушающая трасса не должна наблюдаться в безопасном тестировании. Однако, если трасса неразрушающая, то по её виду ещё нельзя сказать, было тестирование безопасным или нет. Для того, чтобы машина выполнила разрушающее действие a, оператор должен нажать кнопку “A”, разрешающее это действие: $a\in A$. С другой стороны, нажатие такой кнопки после трассы μ не обязательно вызывает выполнение действия a, но может вызвать выполнение другого действия, разрешаемого этой кнопкой, $a`\in A$, которое не является разрушающим. Тем не менее, кнопка “A” *опасна* после трассы μ, поскольку её нажатие может вызвать разрушение. В безопасном тестировании мы не должны нажимать кнопку “A” после трассы μ.

Таким образом, безопасность тестирования определяется не только наблюдаемыми действиями, но и нажимаемыми кнопками. Мы можем говорить об опасности или безопасности кнопки после трассы. Заметим также, что, если символ каждого действия написан не более, чем на одной кнопке, то по действию однозначно восстанавливается нажатая кнопка. В

этом случае можно говорить об опасности или безопасности самих действий после трассы, независимо от нажимаемых кнопок.[11]

Конвергентность.

В данном исследовании мы ограничиваемся только таким тестированием, которое интерпретирует отказ как истечение тайм-аута при нажатой кнопке. Предполагается, что время между нажатием кнопки и выполнением машиной разрешённого действия или отказом ограничено сверху. С теоретической точки зрения можно считать, что разрешённое и выполнимое внешнее действие совершается мгновенно после нажатия кнопки, и тогда тайм-аут – любое ненулевое конечное время. Иными словами, если после нажатия кнопки на печать мгновенно не выведен символ действия, то он не будет выведен никогда (без изменения состояния кнопок), и поэтому машина печатает отказ. Поскольку дивергенция может вызвать сколь угодно длительную задержку перед выполнением разрешённого действия, или действие вообще не будет выполнено, интерпретация истечения тайм-аута как отказа может оказаться недостоверной.

Рис.10.

Поэтому при тестировании мы должны избегать дивергенции. Дивергенцию мы будем рассматривать как нежелательное поведение, то есть как частный случай разрушения, и мы её также будем обозначать символом γ (Рис.10). Тем самым, при безопасном тестировании не может возникнуть дивергенция.

Если в машине нет разрушения (в частности, дивергенции), то после любой трассы любая нажатая кнопка действий через конечное время будет отжата машиной и на печать выведено соответствующее действие или отказ. Отсутствие разрушения, конечно, достаточно для безопасного тестирования, но несколько избыточно. С учётом того, что дивергенция понимается как разрушение, необходимо и достаточно, чтобы тестовые трассы были безопасными.

---

[11] Для машины тестирования с приоритетами трассовая последовательность должна включать не только наблюдаемые действия и отказы, но также нажимаемые кнопки. Такая трасса безопасна, если каждая её кнопка безопасна после непосредственно предшествующего ей префикса трассы.

Гипотеза о безопасности.

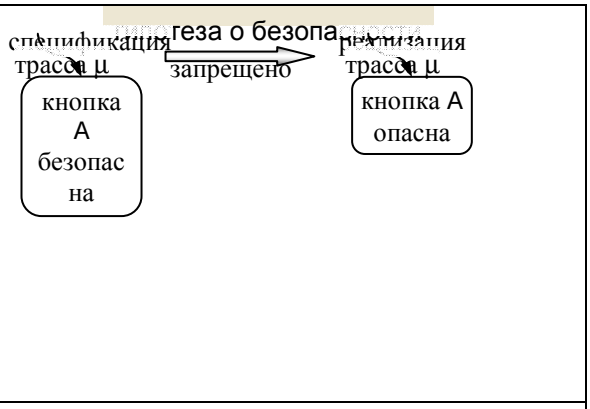


Рис.11.

Итак, при безопасном тестировании после каждой наблюдаемой трассы могут нажиматься только безопасные кнопки. Поскольку машина представляет собой «чёрный ящик», оператор может иметь лишь гипотезу о безопасности кнопок после трасс. Такую гипотезу мы будем называть *гипотезой о безопасности* и предполагать, что кнопка, безопасная после трассы согласно гипотезе, действительно безопасна в машине после этой трассы, если в машине эта трасса есть. Гипотеза формулируется как гипотеза о реализации для заданной спецификации. Спецификацию можно понимать как спецификационную машину, трассы которой определяют безопасность кнопок. Гипотеза, тем самым, утверждает, что любая кнопка, безопасная после трассы спецификации, безопасна после этой трассы в реализации, если в реализации есть эта трасса[12]. Иными словами: если кнопка безопасна после трассы в спецификации, то в реализации эта кнопка не может быть опасной после этой трассы (Рис.11). Такую реализацию будем называть *S-тестируемой*[13] (для данной спецификации).

Заметим, что для безопасности тестирования не требуется, чтобы кнопка, опасная в спецификации, действительно была опасной в реализации. Безопасное тестирование вообще не проверяет поведение машины при нажатии кнопки, если эта кнопка опасна в спецификации. Тем самым, реализация может выполнять любое действие, разрешаемое такой кнопкой, или печатать отказ, и не обязательно разрушаться после этого. Безопасное тестирование имеет целью получить (для проверки на «правильность») не все трассы реализации, а только такие, которые можно получить при безопасном тестировании согласно гипотезе о безопасности, то есть такие, которые можно безопасно получить в спецификации.

---

[12] Более строго, нужно говорить только о таких трассах спецификации, которые могут получаться при безопасном тестировании спецификации. Такую трассу можно назвать безопасной: существует такая трасса с кнопками, что данная трасса получается из неё удалением кнопок, и каждая кнопка безопасна после непосредственно предшествующего ей префикса трассы.

[13] *S-* от safe - безопасная.

Формальные правила, определяющие машину тестирования.

Определение 4: Пусть для алфавита действий $C \subseteq Z$ задана машина, снабжённая печатающим устройством и набором кнопок "A", где $A \subseteq C$, которые оператор может нажимать (но не более одной); кнопка отжимается только самой машиной. Пусть машина может печатать действия $a \in C$, символ разрушения γ и отказы $A \subseteq C$; в каждый момент времени машина может иметь внутреннюю активность, находиться в стабильном состоянии или быть разрушенной. Такую машину будем называть машиной тестирования, если она работает по следующим правилам:

(Машина.1) После печати символа разрушения γ машина разрушена: больше ничего не печатается.

(Машина.2) После печати отказа $A \subseteq C$ машина не может напечатать никакое из разрешённых действий $a \in A$.

(Машина.3) Машина может напечатать отказ $A \subseteq C$ только в том случае, когда нажата кнопка "A", а машина находится в стабильном состоянии.

(Машина.4) Если машина находится в стабильном состоянии, нажата кнопка "A", $A \subseteq C$, и хотя бы одно из разрешённых кнопкой действий $a \in A$ машина может напечатать, то машина обязана напечатать одно из этих разрешённых действий и отжать кнопку "A".

(Машина.5) Если машина находится в стабильном состоянии, нажата кнопка "A", $A \subseteq C$, но ни одно из разрешённых этой кнопкой действий $a \in A$ машина не может напечатать, то машина обязана напечатать отказ A и отжать кнопку "A".

(Машина.6) После печати отказа машина остаётся в том же самом стабильном состоянии: если машина перешла в стабильное состояние s, то любую трассу, которую она может напечатать после печати отказа $A \subseteq C$, она может напечатать и без печати отказа. Формально: $\Sigma(s,A)=\Sigma(s)$.

(Машина.7) Машина не изменяется в процессе работы, то есть, множество трасс, которые можно ожидать после печати любой трассы σ, равно Σ ***after*** σ, где Σ – множество трасс, ожидающихся с самого начала работы.

(Машина.8) Если машина не разрушена, она через конечное ограниченное время либо разрушается, либо переходит в стабильное состояние (моделирование дивергенции разрушением).

(Машина.9) Машина может печатать только те действия $a \in C$, которые разрешены нажатой кнопкой "A", то есть $a \in A$.

2.1.2. Машина для *ioco*$_{\beta\gamma\delta}$-тестирования – βγδ-машина

Здесь мы определим специальную машину, из которой будем выводить модели для тестирования обобщённого ***ioco***-соответствия – соответствия ***ioco***$_{\beta\gamma\delta}$. Такую машину будем называть βγδ-машиной.

В этой машине множество C действий разбивается на два непересекающихся подмножества: множество *стимулов*, и множество *реакций*. Наблюдаемые действия трактуются как передача информации в машину (стимул) или из машины (реакция). Действие-стимул мы будем также называть «приёмом стимула», а действие-реакцию – «выдачей реакции». Потребуем также, чтобы действия и отказы не перепутывались $C \cap \wp(C) = \varnothing$. Будем говорить, что машина задана в алфавите C (Рис.12).

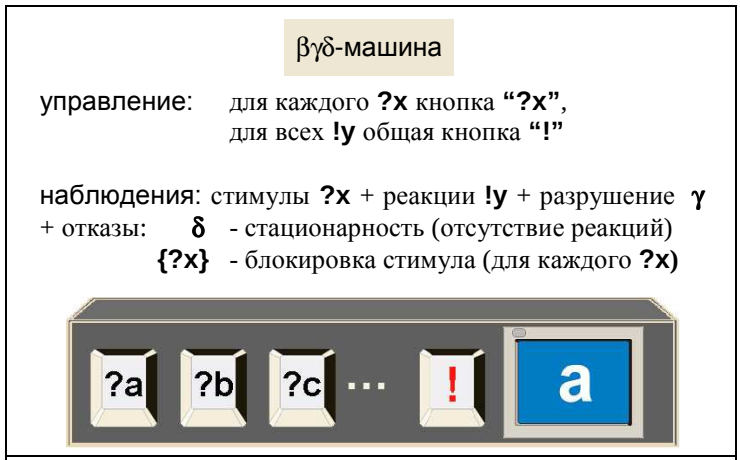


Рис.12.

Для каждого стимула имеется кнопка, на которой написан один этот стимул. Отказ {стимул} будем называть «блокировкой стимула». Кроме кнопок стимулов имеется ещё только одна кнопка, на которой написано множество всех реакций, которое мы будем также обозначать $\delta = \{$все реакции$\}$. Отказ δ будем называть «стационарностью» (отсутствие реакций в стабильном состоянии). Отказы βγδ-машины, то есть, блокировки стимулов и стационарность, мы будем называть βδ-отказами. Кроме этого, машина может разрушаться, разрушение обозначается символом γ. Напомним, что семантика разрушения включает также дивергенцию.

Если в машине нет разрушения и блокировок стимулов, то такая машина отвечает семантике соответствия ***ioco***.

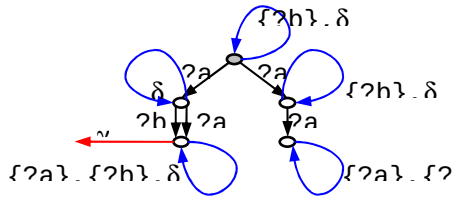
Для обозначения стимулов мы будем использовать (как это принято в CCS – Calculus of Communicating Systems [Miln89]) в качестве префикса вопросительный знак “?”, а для обозначения реакций – восклицательный знак “!”. Это гарантирует выполнение требований к алфавиту машины: множества стимулов и реакций не пересекаются и не содержат в качестве элементов множества (βδ-отказы) и γ.

В βγδ-машине каждый символ действия написан только на одной кнопке: стимул ?x – на кнопке “?x”, реакция !y – на кнопке “δ”. В любой машине, обладающей этим свойством, тестовая трасса с кнопками однозначно восстанавливается по подтрассе без кнопок. Поэтому мы будем рассматривать трассы без кнопок.

βγδ-машина безопасна, если в ней нет трассы $\langle\gamma\rangle$. В безопасной машине, по крайней мере, одна трасса безопасна – пустая трасса. Разрушение может возникнуть после некоторой безопасной трассы μ в результате либо приёма стимула ?x, либо выдачи реакции $!y_1$. Такой стимул ?x или такая реакция $!y_1$ разрушающие после трассы μ, а любая трасса вида $\mu\cdot\langle ?x\rangle\cdot\lambda$ или $\mu\cdot\langle !y_1\rangle\cdot\lambda$ – разрушающая трасса. Иными словами, разрушающая трасса – это трасса, которая заканчивается разрушением или имеет продолжение или ответвление разрушением.

Разрушающая трасса опасная, но существуют и другие опасные трассы (см. порождающий граф на Рис.13). Нажатие кнопки “?x” после трассы μ не обязательно вызывает приём разрушающего стимула ?x, но может вызвать блокировку {?x}. Поэтому *опасными символами* после трассы μ являются как стимул ?x, так и его блокировка {?x}, а любая трасса вида $\mu\cdot\langle ?x\rangle\cdot\lambda$ или $\mu\cdot\langle \{?x\}\rangle\cdot\lambda$ – *опасная трасса*. Аналогично, нажатие кнопки “δ” после трассы μ не обязательно вызывает выдачу разрушающей реакции $!y_1$, но может вызвать выдачу другой реакции $!y_2$ или стационарность δ. Поэтому любая реакция !y и стационарность δ являются *опасными символами* после трассы μ, а любая трасса вида $\mu\cdot\langle !y\rangle\cdot\lambda$ или $\mu\cdot\langle\delta\rangle\cdot\lambda$ – *опасная трасса*. Заметим, что, в отличие от стимула, после трассы реакции и стационарность либо все опасны (одна из реакций разрушающая), либо все безопасны (все реакции неразрушающие).

Таким образом, безопасная трасса – это такая трасса, которая не содержит разрушение и в которой все символы безопасны после непосредственно предшествующих им префиксов трассы.



разрушающие и опасные трассы

Алфавит: стимулы ?a,?b.

Трасса ⟨?a,{?b},?a⟩ неразрушающая, но опасная – содержит блокировку разрушающего стимула ?b.

Рис.13.

Как уже было сказано выше, мы ограничиваемся только таким тестированием, которое интерпретирует отказ как истечение тайм-аута при нажатой кнопке. Тем самым, при безопасном тестировании не может возникнуть дивергенция. С учётом того, что дивергенция понимается как разрушение, необходимо и достаточно, чтобы трассы были безопасными.

Гипотеза о безопасности переформулируется для трасс без кнопок: любая безопасная трасса спецификации либо отсутствует в реализации, либо безопасна в ней. Или, что то же самое: если трасса безопасна как в спецификации, так и в реализации, то её продолжение символом, оставляющим трассу безопасной в спецификации, либо отсутствует в реализации, либо оставляет трассу безопасной в реализации.

Глава 2.2. Трассовые модели

Структура главы:
1. βγδ-модель
2. Разложение βγδ-модели на поддеревья
3. Модель безопасных трасс
4. Модель финальных трасс
5. Соотношение трассовых моделей
6. Гипотеза о безопасности
7. Соответствие $ioco_{\beta\gamma\delta}$
8. Тест и полный набор тестов
9. Алгоритмизация

В качестве модели системы выбирается множество трасс βγδ-машины. Мы дадим независимое определение βγδ-модели и покажем, что множество трасс βγδ-машины удовлетворяет этому определению.

Обратное утверждение (любая βγδ-модель является множеством трасс некоторой βγδ-машины) будет доказано в Глава 2.3. Там будет введена другая модель – LTS (Labelled Transition System), для которой мы определим множество её βγδ-трасс. Будет показано, что 1) для каждой βγδ-модели существует LTS-модель с этим множеством βγδ-трасс и 2) любая LTS-модель может рассматриваться как βγδ-машина.

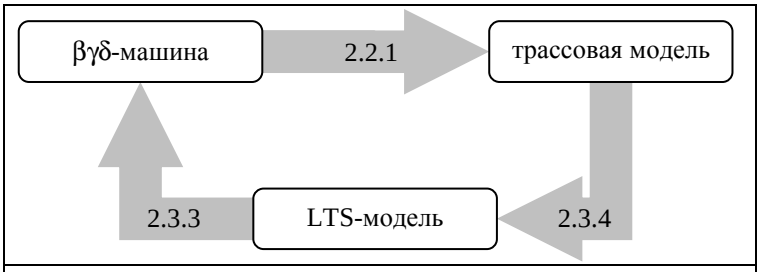


Рис.14.

В этой главе мы определим, в рамках βγδ-модели, спецификацию, реализацию, соответствие $\boldsymbol{ioco}_{\beta\gamma\delta}$, понятия теста и тестового набора, рассмотрим проблему генерации тестов и вопросы алгоритмизации.

Кроме этого, мы рассмотрим другие, производные от βγδ-модели, трассовые модели: модель безопасных трасс и модель финальных трасс, и соотношение между ними.

2.2.1. βγδ-модель

Структура раздела:
- Базовые βγδ-обозначения.
- βγδ-трассы.
- Определение βγδ-модели.
- βγδ-модели без блокировок и/или разрушения.
- βγδ-машина и βγδ-модель.
- Объединение и пересечение βγδ-моделей.

Базовые βγδ-обозначения.

Определение 5:
- Будем считать, что фиксирован универсум символов W. Обозначим:
  $Z =_{def} (\{?\}\times W)\cup(\{!\}\times W)$.
- Базовым алфавитом будем называть подмножество $C\subseteq Z$.
  Базовыми символами будем называть символы базового алфавита C, а также символ разрушения γ. Будем предполагать, что $\gamma\notin Z\cup\wp(Z)$.
- Для произвольного множества A обозначим:
  $?A \quad =_{def} \quad A\cap(\{?\}\times W)$ – стимулы,
  $!A \quad =_{def} \quad A\cap(\{!\}\times W)$ – реакции,
  $\beta(A) \quad =_{def} \quad \{\{?x\}\,|\,?x\in ?A\}$ – блокировки (β-отказы),
  $A_\beta \quad =_{def} \quad A\cup\beta(A)$ – множество плюс блокировки.
- Для фиксированного базового алфавита $C\subseteq Z$ и произвольного множества A обозначим:
  $\delta(A) \quad =_{def} \quad \{!C\,|\,!A=!C\}$ – стационарность (δ-отказ),[14]
  $\beta\delta(A) \quad =_{def} \quad \beta(A)\cup\delta(A)$ – βδ-отказы,[15]
  $A_\delta \quad =_{def} \quad A\cup\delta(A)$ – множество плюс δ-отказ.[16]
- Для фиксированного базового алфавита $C\subseteq Z$ обозначим:
  $\delta \quad =_{def} \quad !C$ – стационарность (все реакции алфавита)[17].

---

[14] $\delta(A)$ зависит от подразумеваемого алфавита C: $\delta(A)=\{!C\}$, если $!C=!A$, и $\delta(A)=\varnothing$, иначе. $\delta(C)=\{!C\}$.

[15] $\beta\delta(A)$ зависит от подразумеваемого алфавита C: $\beta\delta(A)=\beta(A)\cup\{!C\}$, если $!C=!A$, и $\beta\delta(A)=\beta(A)$, иначе. $\beta\delta(C)=\beta(C)\cup\{!C\}$.

[16] A_δ зависит от подразумеваемого алфавита C: $A_\delta=A\cup\{!C\}$, если $!C=!A$, и $A_\delta=A$, иначе. $C_\delta=C\cup\{!C\}$.

- Для любого множества A и символа $\alpha\in\{\tau,\gamma,\theta\}$ обозначим $A_\alpha =_{def} A\cup\{\alpha\}$.[18]
- Для алфавита $C\subseteq Z$ и стимула или реакции $z\in C$ через $\beta\delta(z)$ обозначим βδ-отказ, которому принадлежит z: для $?x,!y\in C$ $\beta\delta(?x) =_{def} \{?x\}$, $\beta\delta(!y) =_{def} \delta$.[19]
- βγδ-трассой в алфавите $C\subseteq Z$ будем называть последовательность стимулов, реакций, блокировок стимулов, стационарности и разрушения, то есть последовательность в алфавите $C_{\beta\gamma\delta}=?C\cup!C\cup\beta(C)\cup\{\gamma\}\cup\{\delta\}$.
- βγδ-деревом будем называть дерево βγδ-трасс в алфавите $C\subseteq Z$.
- Максимальные префикс и постфикс βγδ-трассы, состоящие только из βδ-отказов, будем называть βδ-префиксом и βδ-постфиксом, соответственно, и обозначать:

  $\boldsymbol{RefH}(\sigma)=_{def} \langle\ \sigma(i)\ |\ \sigma[1..i]\in\beta\delta(C)^*\ \rangle$ и

  $\boldsymbol{RefT}(\sigma)=_{def} \langle\ \sigma(i)\ |\ \sigma[i..|\sigma|]\in\beta\delta(C)^*\ \rangle$.

□

В этой главе под отказами, трассами и деревьями, если не оговорено противное, мы будем понимать βδ-отказы, βγδ-трассы и βγδ-деревья, соответственно.

βγδ-трассы.

Определение 6: Пусть задан базовый алфавит $C\subseteq Z$, дерево $\Sigma\in\boldsymbol{Trees}(C_{\beta\gamma\delta})$ и трасса $\sigma\in\Sigma$.

- Трассу σ будем называть *допустимой*, если разрушение γ либо не входит в трассу, либо входит только как последний символ трассы[20]: $\forall i\in[1..|\sigma|-1]\ \sigma(i)\neq\gamma$.
- Трассу σ будем называть *согласованной*, если в ней любая непустая последовательность βδ-отказов не продолжается ни разрушением, ни каким-либо символом, принадлежащим какому-либо отказу, входящему в эту последовательность[21]: $\forall i\in[2..|\sigma|]\ \forall o\ o\in\boldsymbol{Im}_\circ\boldsymbol{RefT}(\sigma[1..i-1])\ \Rightarrow\ \sigma(i)\neq\gamma\ \&\ \sigma(i)\notin o$.

---

[17] δ зависит от подразумеваемого алфавита C: $\{\delta\}=\delta(C)$.

[18] Семантика символа θ будет разъяснена в разделе 2.3.6.

[19] Здесь мы используем тот факт, что каждый стимул или реакция принадлежит только одному βδ-отказу: блокировке стимула или стационарности, соответственно. Кроме того, поскольку δ определена для данного базового алфавита, βδ-оператор зависит от подразумеваемого базового алфавита.

[20] Допустимость трассы σ не зависит от дерева Σ.

[21] Согласованность трассы σ не зависит от дерева Σ.

- Трассу σ будем называть *конвергентной по отказу* o, или o-*конвергентной*, и обозначать $\sigma{\downarrow}o$, если трасса продолжается в дереве этим отказом или каким-либо символом, принадлежащим отказу. Трассу будем называть *конвергентной по стимулу или реакции* z, или z-*конвергентной*, и обозначать $\sigma{\downarrow}z$, если трасса конвергентна по отказу, содержащему z.[22] Если трасса не u-конвергентна, будем называть её u-*дивергентной* и обозначать $\sigma{\uparrow}u$.

  $o\in\beta\delta(C)$: $\sigma{\downarrow}o =_{def} \sigma\cdot\langle o\rangle\in\mathbf{\Sigma} \vee \exists z\in o\ \sigma\cdot\langle z\rangle\in\mathbf{\Sigma}$,

  $?x\in C$: $\sigma{\downarrow}?x =_{def} \sigma{\downarrow}\{?x\}$,

  $!y\in C$: $\sigma{\downarrow}!y =_{def} \sigma{\downarrow}\delta$,

  $u\in C_{\beta\delta}$: $\sigma{\uparrow}u =_{def} \neg\sigma{\downarrow}u$.
- Трассу σ будем называть *конвергентной* и обозначать $\sigma{\downarrow}$, если она конвергентна по всем отказам[23]. В противном случае трассу будем называть *дивергентной*, и обозначать $\sigma{\uparrow}$.

  $\sigma{\downarrow} =_{def} \forall o\in\beta\delta(C)\ \sigma{\downarrow}o$,

  $\sigma{\uparrow} =_{def} \neg\sigma{\downarrow}$.
- Определим операции над трассами дерева $\mathbf{\Sigma}$:
  - ***D*** (***D***eletion) – удаление из трассы отказа o: $\mu\cdot\langle o\rangle\cdot\lambda\to\mu\cdot\lambda$.
  - ***R*** (***R***epetition) – повторение отказа o: $\mu\cdot\langle o\rangle\cdot\lambda\to\mu\cdot\langle o,o\rangle\cdot\lambda$.
  - ***T*** (***T***ransposition) – перестановка местами соседних отказов l,r: $\mu\cdot\langle l,r\rangle\cdot\lambda\to\mu\cdot\langle r,l\rangle\cdot\lambda$.
  - ***Ins*** (***Ins***ertion) – вставка отказа r в трассу $\mu\cdot\langle l\rangle\cdot\lambda$ после отказа l при условии, что $\mu\cdot\langle l\rangle$ не продолжается в $\mathbf{\Sigma}$ разрушением и не продолжается ни одним базовым символом из отказа r:

    $\mu\cdot\langle l,\gamma\rangle\notin\mathbf{\Sigma}$ & $\forall z\in r\ \mu\cdot\langle l,z\rangle\notin\mathbf{\Sigma}$: $\mu\cdot\langle l\rangle\cdot\lambda\to\mu\cdot\langle l,r\rangle\cdot\lambda$.
- Под замыканием по набору операций будем понимать отображение трассы σ во множество трасс, получаемых из σ с помощью всех возможных последовательностей применения операций из этого набора. Обозначим для трассы σ :
  - $\boldsymbol{D}(\sigma)$ – замыкание σ по операциям $\{\boldsymbol{D}\}$;
  - $\boldsymbol{R}(\sigma)$ – замыкание σ по операциям $\{\boldsymbol{R}\}$;
  - $\boldsymbol{T}(\sigma)$ – замыкание σ по операциям $\{\boldsymbol{T}\}$;

---

[22] В $\beta\gamma\delta$-машине каждый стимул $?x$ или реакция $!y$ принадлежат только одному отказу: блокировке $\{?x\}$ или стационарности δ, соответственно.

[23] Очевидно, трасса, конвергентная по всем отказам, конвергентна по всем стимулам и реакциям.

- $\boldsymbol{Ins}(\sigma)$ – замыкание σ по операциям $\{\boldsymbol{Ins}\}$;
- $\boldsymbol{DRT}(\sigma)$ – замыкание σ по операциям $\{\boldsymbol{D},\boldsymbol{R},\boldsymbol{T}\}$;
- $\boldsymbol{DRTIns}(\sigma)$ – замыкание σ по операциям $\{\boldsymbol{D},\boldsymbol{R},\boldsymbol{T},\boldsymbol{Ins}\}$.[24]

□

Определение βγδ-модели.

Определение 7: Пусть задан базовый алфавит $C\subseteq Z$. Непустое дерево $\boldsymbol{\Sigma}\in\boldsymbol{Trees}(C_{\beta\gamma\delta})$ будем называть *βγδ-моделью* в алфавите $C\subseteq Z$, если выполнены следующие пять требований:

(βγδ1) βγδ-допустимость: все трассы $\boldsymbol{\Sigma}$ допустимы;

(βγδ2) βγδ-согласованность: все трассы $\boldsymbol{\Sigma}$ согласованы;

(βγδ3) βγδ-конвергентность: все трассы $\boldsymbol{\Sigma}$, не содержащие и не продолжающиеся разрушением, конвергентны;

(βγδ4) βγδ-замкнутость (замкнутость по ***DRT***-операциям): $\boldsymbol{\Sigma}$ вместе с каждой трассой $\sigma\in\boldsymbol{\Sigma}$ содержит её замыкание по ***DRT***-операциям: $\boldsymbol{DRT}(\sigma)\subseteq\boldsymbol{\Sigma}$;

(βγδ5) βγδ-полнота (замкнутость по ***Ins***-операции): $\boldsymbol{\Sigma}$ вместе с каждой трассой $\sigma\in\boldsymbol{\Sigma}$ содержит её замыкание по ***Ins***-операции: $\boldsymbol{Ins}(\sigma)\subseteq\boldsymbol{\Sigma}$.

- Множество всех βγδ-моделей в алфавите C обозначим $\boldsymbol{MODEL}_{\beta\gamma\delta}(C)$, а множество всех βγδ-моделей $\boldsymbol{MODEL}_{\beta\gamma\delta}=\cup\{\boldsymbol{MODEL}_{\beta\gamma\delta}(C)\,|\,C\subseteq Z\}$.

□

Ортогональность свойств βγδ-модели показана контрпримерами на Рис.15. Здесь для каждого i-го свойства ($i=1\div5$) изображён порождающий граф βγδ-дерева, удовлетворяющего всем свойствам βγδ-модели, кроме i-го.

[24] Множества трасс $\boldsymbol{Ins}(\sigma)$ и $\boldsymbol{DRTIns}(\sigma)$ зависят от дерева $\boldsymbol{\Sigma}$. Там, где по контексту неясно, какое дерево имеется в виду, мы будем указывать его в нижнем индексе: $\boldsymbol{Ins}_{\boldsymbol{\Sigma}}(\sigma)$ и $\boldsymbol{DRTIns}_{\boldsymbol{\Sigma}}(\sigma)$.

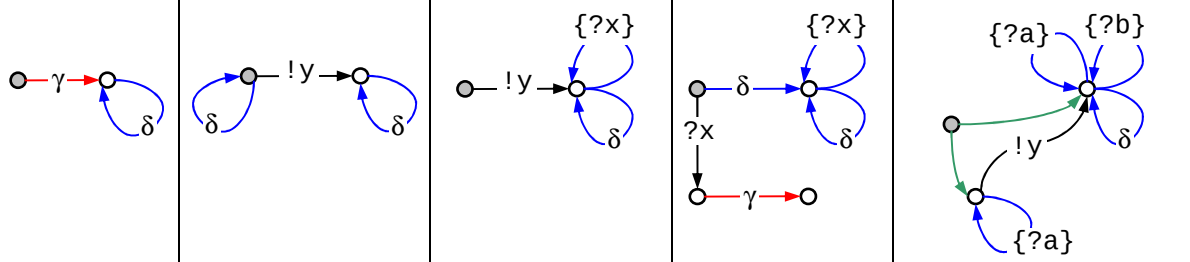
ортогональность свойств βγδ-модели **Σ**				
допустимость C=∅	согласованность C={!y}	конвергентность C={?x,!y}	замкнутость C={?x}	полнота C={?a,?b,!y}
				
⟨γ,δ⟩∈**Σ** не допустима	⟨δ,!y⟩∈**Σ** не согласована	ϵ∈**Σ** не ?x-конвергентна	μ=⟨δ,{?x}⟩∈**Σ**, σ=⟨{?x}⟩∉**Σ**, но σ∈***DRT***(μ)	μ=⟨{?a},!y⟩∈**Σ**, σ=⟨{?a},{?b},!y⟩∉**Σ**, но σ∈***Ins***(μ)

Рис.15.

(βγδ2α) При условии βγδ-замкнутости свойство βγδ-согласованности эквивалентно следующему свойству ослабленной βγδ-согласованности: $\forall i\in[2..|\sigma|]$
$\sigma(i-1)\in\beta\delta(C) \Rightarrow \sigma(i)\neq\gamma \;\&\; \forall z\in\sigma(i-1)\ \sigma(i)\neq z$.

βγδ-замкнутость и βγδ-полнота означают замкнутость по ***DRTIns***-операциям:
$\forall\sigma\in\mathbf{\Sigma}\ \ \boldsymbol{DRTIns}(\sigma)\subseteq\mathbf{\Sigma}$.
Поскольку для любой трассы $\sigma\cdot\lambda\in\mathbf{\Sigma}$ и любой трассы $\sigma`\in\boldsymbol{DRTIns}(\sigma)$ имеет место $\sigma`\cdot\lambda\in\boldsymbol{DRTIns}(\sigma\cdot\lambda)$, из βγδ-замкнутости и βγδ-полноты выводится очевидное следствие:
любое продолжение трассы продолжает каждую трассу, получаемую из данной трассы ***DRTIns***-операциями:
(βγδ4&5) $\forall\sigma\in\mathbf{\Sigma}\ \ \boldsymbol{DRTIns}(\sigma)\cdot(\mathbf{\Sigma}\ \textit{after}\ \sigma)\subseteq\mathbf{\Sigma}$.

βγδ-модели без блокировок и/или разрушения.

Определение 8: Пусть задан алфавит $C\subseteq Z$. Обозначим:
- модели без блокировок:
  $\boldsymbol{MODEL}_{\gamma\delta}(C)=_{def}\ \boldsymbol{MODEL}_{\beta\gamma\delta}(C)\cap\boldsymbol{Trees}(C_{\gamma\delta})$,
  $\boldsymbol{MODEL}_{\gamma\delta}\ =_{def}\ \cup\{\boldsymbol{MODEL}_{\gamma\delta}(C)\,|\,C\subseteq Z\}$;
- модели без разрушения:
  $\boldsymbol{MODEL}_{\beta\delta}(C)=_{def}\ \boldsymbol{MODEL}_{\beta\gamma\delta}(C)\cap\boldsymbol{Trees}(C_{\beta\delta})$,
  $\boldsymbol{MODEL}_{\beta\delta}\ =_{def}\ \cup\{\boldsymbol{MODEL}_{\beta\delta}(C)\,|\,C\subseteq Z\}$;
- модели без блокировок и разрушения:
  $\boldsymbol{MODEL}_{\delta}(C)\ =_{def}\ \boldsymbol{MODEL}_{\beta\gamma\delta}(C)\cap\boldsymbol{Trees}(C_{\delta})$,
  $\boldsymbol{MODEL}_{\delta}\ =_{def}\ \cup\{\boldsymbol{MODEL}_{\delta}(C)\,|\,C\subseteq Z\}$.

βγδ-машина и βγδ-модель.

Утверждение 1: Множество трасс βγδ-машины является βγδ-моделью (Рис.16).

□356

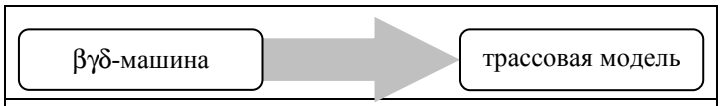


Рис.16.

Обратное утверждение (любая βγδ-модель является множеством трасс некоторой βγδ-машины) будет доказано в Глава 2.3.

Объединение и пересечение βγδ-моделей.

Вообще говоря, βγδ-модель в алфавите $C \subseteq Z$ не является βγδ-моделью в расширенном алфавите $C` \supset C$. Причина – в неконвергентности трасс по добавленным стимулам. Легко показать, что βγδ-модель можно расширить до βγδ-модели в расширенном алфавите, если выполнить её замыкание по ***DRTIns***-операциям в расширенном алфавите. ***Ins***-операция добавит блокировки добавленных стимулов после отказа l в трассу $\mu \cdot \langle l \rangle \cdot \lambda$, ***T***-операция разрешит начинать последовательность отказов с блокировок добавленных стимулов, а ***D***-операция позволит удалить старые отказы, оставив только блокировки добавленных стимулов.

Объединение βγδ-моделей, по той же причине (неконвергентность трасс одной модели по стимулам других моделей), вообще говоря, не является βγδ-моделью. Однако, объединение βγδ-моделей расширенных до βγδ-моделей в объединении алфавитов, очевидно, будет βγδ-моделью. В частности, объединение βγδ-моделей в одном алфавите является βγδ-моделью.

Пересечение деревьев является деревом. В то же время, пересечение βγδ-моделей может не быть βγδ-моделью. Свойства βγδ-допустимости, βγδ-согласованности и βγδ-замкнутости, очевидно, выполнены, но свойства βγδ-конвергентности и βγδ-полноты могут нарушиться. Пример в виде порождающих графов приведён на Рис.17: трасса $\langle \{?x\} \rangle$ заканчивается отказом, но в пересечении не продолжается ни реакцией, ни стационарностью.

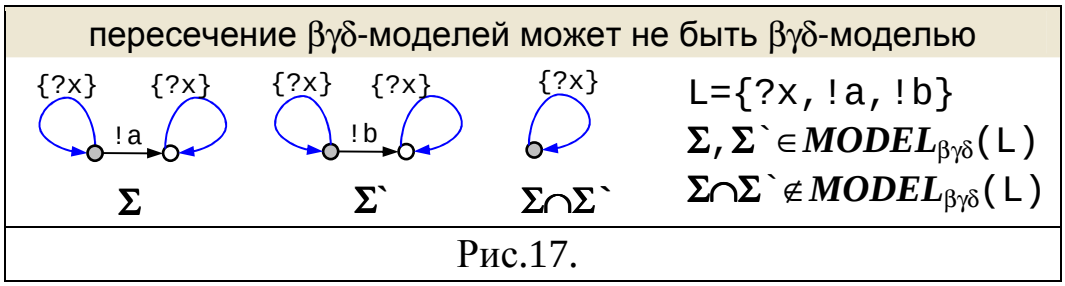


Рис.17.

2.2.2. Разложение βγδ-модели на поддеревья

Структура раздела:

- Конвергентные и дивергентные деревья.
- Стабильные и контрстабильные деревья и квази-βγδ-модели.
- Вспомогательные утверждения.
- Утверждение о разложении βγδ-модели.

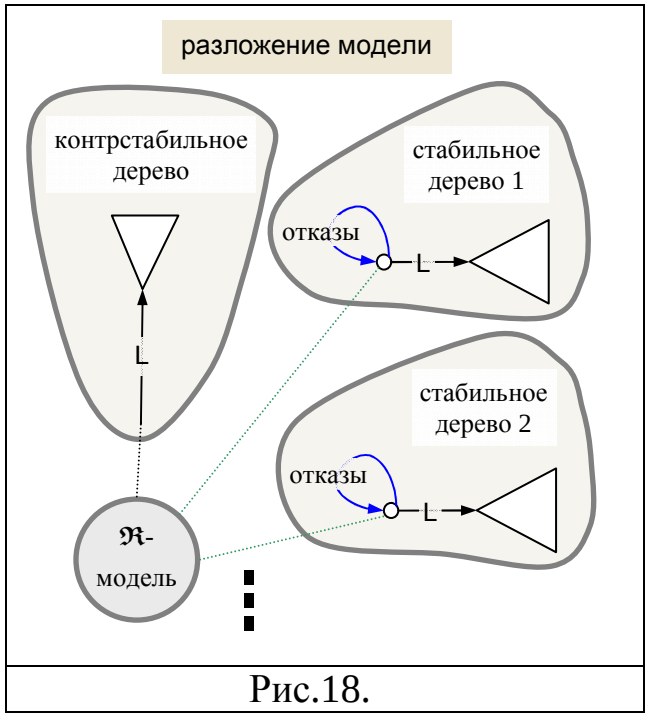


Рис.18.

В этом разделе мы покажем, что βγδ-модель раскладывается в объединение контрстабильного дерева и множества стабильных деревьев (Рис.18). Саму βγδ-модель можно рассматривать как аналог поведения βγδ-машины, в которой дивергенция заменена разрушением. Контрстабильное дерево описывает внешние действия (приём стимула и выдачу реакции), возможные в машине, вместе с продолжающим эти действия поведением машины, а также возможное разрушение. Стабильное дерево является аналогом стабильного состояния βγδ-машины, в котором конечная τ-активность может закончиться «сама по себе» без приёма стимулов и выдачи реакций. Подмодель, определяющая поведение машины после возможного действия (кроме разрушения) или после действия, возможного в данном стабильном состоянии, когда τ-активность закончилась «сама по себе», также раскладывается в свою очередь, на контрстабильное дерево и множество стабильных поддеревьев. И так далее.

Забегая вперёд, скажем, что целью такого разложения βγδ-модели является построение LTS-модели с тем же множеством βγδ-трасс (см. 2.3.4). Это будет сделано с помощью специального преобразования, создающего самую «компактную» LTS-модель: $T_{\beta\gamma\delta}2_{compact}L_{\gamma}$: βγδ-модель→LTS-модель. В

частности, если для данной βγδ-модели существует конечная LTS с таким же множеством βγδ-трасс, то таким преобразованием будет получена конечная LTS.

Конвергентные и дивергентные деревья.

Определение 9: Пусть $C \subseteq Z$ и $\Sigma \in \mathit{Trees}(C_{\beta\gamma\delta})$.

· Дерево Σ будем называть конвергентным и обозначать $\Sigma\downarrow$, если в нём нет трассы $\langle\gamma\rangle$, а его пустая трасса конвергентна: для каждого стимула есть трасса, начинающаяся с этого стимула или его блокировки, и есть трасса, начинающаяся с какой-нибудь реакции или со стационарности:

$$\Sigma\downarrow =_{def} \langle\gamma\rangle \notin \Sigma \;\&\; \epsilon\downarrow$$

$$= \langle\gamma\rangle \notin \Sigma \;\&\; (\forall ?x \in C\ ?x \in \mathit{head}(\Sigma) \vee \{?x\} \in \mathit{head}(\Sigma))$$

$$\& (\exists ! y \in C\ !y \in \mathit{head}(\Sigma) \vee \delta \in \mathit{head}(\Sigma)).$$

· Если дерево Σ не конвергентно, будем называть его *дивергентным*:

$$\Sigma\uparrow =_{def} \neg(\Sigma\downarrow).$$

□

Стабильные и контрстабильные деревья и квази-βγδ-модели.

Определение 10: Пусть $C \subseteq Z$.

· Дерево $T \in \mathit{Trees}(C_{\beta\gamma\delta})$ будем называть *стабильным* и обозначать T ***stable***, если выполнены следующие три условия:

(стаб.1) для любой трассы из T, содержащей только отказы, множество её продолжений в T совпадает с T:

$$\forall o \in T \cap (\beta\delta(C)^*)\ T\ \mathit{after}\ o = T;$$

(стаб.2) дерево T конвергентно: $T\downarrow$;

(стаб.3) в T нет двух трасс, одна из которых начиналась бы с отказа, а другая с символа, принадлежащего этому отказу:

$$\forall o \in \mathit{head}(T) \cap \beta\delta(C)\ \forall z \in o\ z \notin \mathit{head}(T).$$

· Если дерево T не стабильно, будем обозначать

$$T\ \mathit{unstable} =_{def} \neg\ T\ \mathit{stable}.$$

· Дерево $T \in \mathit{Trees}(C_{\beta\gamma\delta})$, в котором ни одна трасса не начинается с отказа, будем называть *контрстабильным*:

$$\mathit{head}(T) \subseteq C_\gamma.$$

· Для дерева $T \in \mathit{Trees}(C_{\beta\gamma\delta})$ через T_c обозначим его наибольшее (по вложенности) контрстабильное поддерево:

$$T_c =_{def} \{\lambda \in T \mid \lambda \neq \epsilon \Rightarrow \lambda(1) \in C_\gamma\}.$$

· Квази-βγδ-моделью будем называть такое контрстабильное дерево **T**, которое после каждого стимула или реакции является βγδ-моделью, а разрушение ничем не продолжается:

$$\forall z \in \mathit{head}(\mathbf{T}) \setminus \{\gamma\} \quad (\mathbf{T}\ \mathit{after}\ \langle z \rangle) \in \mathit{MODEL}_{\beta\gamma\delta}(C)$$
$$\& (\mathbf{T}\ \mathit{after}\ \langle \gamma \rangle) = \{\epsilon\}.$$

· Множество всех квази-βγδ-моделей в базовом алфавите C обозначим $\mathit{QMODEL}_{\beta\gamma\delta}(C)$.

□

Очевидно, контрстабильное дерево конвергентно тогда и только тогда, когда оно стабильно. Объединение контрстабильных деревьев является контрстабильным деревом, и любое его поддерево (подмножество, являющееся деревом) также является контрстабильным деревом. В отличие от этого, объединение стабильных деревьев не обязательно является стабильным деревом, хотя всегда конвергентно, и не любое поддерево стабильного дерева стабильно.

Вспомогательные утверждения.

Сначала сформулируем ряд вспомогательных утверждений, необходимых для доказательства основного утверждения о разложении βγδ-модели на поддеревья.

Квази-βγδ-модель после каждого стимула или реакции является βγδ-моделью. Поэтому для того, чтобы квази-βγδ-модель сама была βγδ-моделью, достаточно, чтобы свойства βγδ-модели выполнялись для пустой трассы и γ-трассы. При наличии γ-трассы эти свойства автоматически выполняются, а при её отсутствии требуется конвергентность пустой трассы, то есть конвергентность квази-βγδ-модели. Формально это показывается следующим утверждением.

Утверждение 2: Квази-βγδ-модель **T** является βγδ-моделью тогда и только тогда, когда $\langle\gamma\rangle \in \mathbf{T} \vee \mathbf{T}\!\downarrow$.

□360

Сохранение стабильного состояния машины тестирования после отказа означает, что в порождающем графе отказы – это петли. Это формально описывается свойством (стаб.1): после любой трассы отказов дерево не меняется. Следующее утверждение является, по существу, переформулировкой свойства (стаб.1): любая трасса дерева имеет префикс (быть может, пустой) из отказов, принадлежащих началу дерева, и любую последовательность таких отказов можно вставить перед любой трассой дерева (см. Рис.19).

Утверждение 3: Пусть $C \subseteq Z$, для дерева $T \in \mathit{Trees}(C_{\beta\gamma\delta})$ выполнено свойство (стаб.1), а O – множество отказов, с которых могут начинаться трассы дерева T: $O = \mathit{head}(T) \cap \beta\delta(C)$. Тогда $T = O^* \cdot T_c$ (T_c – наибольшее контрстабильное поддерево T).

□361

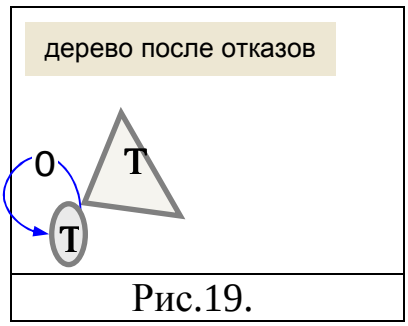


Рис.19.

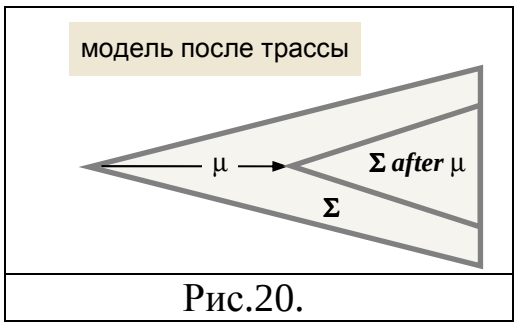


Рис.20.

Покажем, что после каждой трассы, не заканчивающейся на разрушение, βγδ-модель остаётся βγδ-моделью (см. Рис.20).

Утверждение 4: Пусть для $C \subseteq Z$ дерево $\Sigma \in \mathit{MODEL}_{\beta\gamma\delta}(C)$. Тогда для каждой трассы $\mu \in \Sigma$, не заканчивающейся на разрушение, дерево $(\Sigma \ \mathit{after}\ \mu) \in \mathit{MODEL}_{\beta\gamma\delta}(C)$.

□362

Утверждение 5: Наибольшее контрстабильное поддерево βγδ-модели является квази-βγδ-моделью.

□363

Квази-βγδ-модель, не содержащую трассу $\langle\gamma\rangle$, всегда можно расширить до βγδ-модели, сделав пустую трассу конвергентной: добавить блокировки стимулов, с которых не начинаются трассы, и добавить стационарность, если трассы не начинаются с реакций.

Утверждение 6: Пусть $C \subseteq Z$, $T \in \mathit{QMODEL}_{\beta\gamma\delta}(C)$ и $\gamma \notin \mathit{head}(T)$.
Тогда $O^* \cdot T \in \mathit{MODEL}_{\beta\gamma\delta}(C)$, где
$O = \{\{?x\} \mid ?x \in C \setminus \mathit{head}(T)\} \cup \{\delta \mid !C \cap \mathit{head}(T) = \varnothing\}$.

□364

Утверждение о разложении βγδ-модели.

Утверждение 7: Пусть для $C \subseteq Z$ дерево $\Sigma \in \mathit{MODEL}_{\beta\gamma\delta}(C)$. Тогда $\Sigma = \Sigma_c \cup (\cup(\Sigma_s))$, где Σ_c (наибольшее по вложенности контрстабильное поддерево Σ) является квази-βγδ-моделью, а Σ_s – множество стабильных βγδ-моделей.

□365

Для дальнейшего нам также понадобится следующее простое утверждение: замыкание непустой трассы отказов o по ***DRTIns***-операциям равно множеству всех конечных последовательностей в алфавите, содержащем блокировки всех стимулов, которыми не продолжается трасса o, и стационарность, если o не продолжается ни одной реакцией.

Утверждение 8: Пусть $C \subseteq Z$, $\Sigma \in \mathit{MODEL}_{\beta\gamma\delta}(C)$ и задана непустая трасса $\beta\delta$-отказов $o \in \Sigma \cap (\beta\delta(C)^*)$, $o \neq \epsilon$.

Тогда $\mathit{DRTIns}(o) = (\ \{\{?x\} \mid ?x \in C \setminus \mathit{head}(\Sigma\ \mathit{after}\ o)\}$
$\cup \{\ \delta \mid !C \cap \mathit{head}(\Sigma\ \mathit{after}\ o) = \varnothing\}\ )^*$.

□369

2.2.3. Модель безопасных трасс

Структура раздела:
- Безопасность.
- Определение модели безопасных трасс.
- Пополнение до конвергентности.
- *DRT*-замыкание.
- Модель безопасных трасс и $\beta\gamma\delta$-модель.

Повторим, что разрушение машины моделируется символом разрушения γ. Машина (потенциально) саморазрушается, если у неё есть трасса $\langle\gamma\rangle$. Если машина не саморазрушается, она безопасна. Стимул/реакция z разрушающие после трассы μ, если после трассы $\mu\cdot\langle z\rangle$ машина может разрушиться. Трасса разрушающая, если у неё есть префикс, продолжаемый разрушением во множестве трасс машины. Стимул $?x$ и его блокировка $\{?x\}$ безопасны после трассы μ, если после этой трассы стимул $?x$ не разрушающий. Каждая реакция и стационарность безопасны после трассы μ, если после этой трассы все реакции неразрушающие. Трасса безопасна в безопасной машине, если она не содержит символов, опасных после непосредственно предшествующих им префиксов трассы.

Безопасность.

Определение 11: Пусть $C \subseteq Z$ и $\Sigma \in \mathit{Trees}(C_{\beta\gamma\delta})$.

- *Безопасным* деревом будем называть такое дерево Σ, в котором нет трассы, состоящей из одного разрушения:

  $\Sigma\ \mathit{safe} =_{\text{def}} \langle\gamma\rangle \notin \Sigma$.

- Трасса $\sigma \in \Sigma$ *разрушающая*, если $\exists \mu \le \sigma\ \ \mu\cdot\langle\gamma\rangle \in \Sigma$.

· Множество безопасных символов:
$\mathit{safesymbols}(\Sigma) =_{\text{def}} \{?x\in C \mid \Sigma\ \mathit{after}\ \langle ?x\rangle\ \mathit{safe}\}_\beta \cup \{u\in !C_\delta \mid \forall !y\in C\ \Sigma\ \mathit{after}\ \langle !y\rangle\ \mathit{safe}\}$.[25]

· Безопасная часть начала дерева:
$\mathit{safehead}(\Sigma) =_{\text{def}} \mathit{safesymbols}(\Sigma) \cap \mathit{head}(\Sigma)$.

· Дерево безопасных трасс:
$\mathit{safe}(\Sigma) =_{\text{def}} \{\sigma\in\Sigma \mid \Sigma\ \mathit{safe}\ \&\ \forall\mu\cdot\langle u\rangle\le\sigma\ u\in\mathit{safesymbols}(\Sigma\ \mathit{after}\ \mu)\}$.

□

В отличие от βγδ-модели, множество безопасных трасс βγδ-модели не содержит трасс с разрушением. Мы дадим явное определение модели безопасных трасс, сформулировав ряд требований на дерево трасс. Потом покажем, что эти требования необходимы и достаточны для того, чтобы модель безопасных трасс была множеством безопасных трасс некоторой βγδ-модели. Также покажем, что модель безопасных трасс, вообще говоря, не является βγδ-моделью. В частности, в отличие от βγδ-модели, модель безопасных трасс может содержать неконвергентные трассы.

Определение модели безопасных трасс.

Определение 12: Пусть задан базовый алфавит $C\subseteq Z$. Дерево $T\in \mathit{Trees}(C_{\beta\delta})$ будем называть *моделью безопасных трасс* в алфавите $C\subseteq Z$, если выполнены следующие пять требований:

(βδ1) βδ-стационарность: если множество реакций пусто, то каждая трасса дерева должна продолжаться стационарностью:
$!C=\varnothing \Rightarrow \forall\sigma\in T\ \sigma\cdot\langle\delta\rangle\in T$.

(βδ2) βδ-согласованность: трассы T согласованы.[26]

(βδ3) βδ-конвергентность: если трасса $\sigma\in T$ o-конвергентна, где o отказ, то любая трасса $\mu\in T$, из которой σ может быть получена

---

[25] $\{?x\in C \mid \Sigma\ \mathit{after}\ \langle ?x\rangle\ \mathit{safe}\}_\beta = \{?x\in C \mid \Sigma\ \mathit{after}\ \langle ?x\rangle\ \mathit{safe}\} \cup \{\{?x\} \mid ?x\in C\ \&\ \Sigma\ \mathit{after}\ \langle ?x\rangle\ \mathit{safe}\}$;

$\mathit{safesymbols}(\Sigma) = \{?x\in C \mid \Sigma\ \mathit{after}\ \langle ?x\rangle\ \mathit{safe}\}_\beta$, если $\exists !y\in C\ \neg(\Sigma\ \mathit{after}\ \langle !y\rangle\ \mathit{safe})$,
$= \{?x\in C \mid \Sigma\ \mathit{after}\ \langle ?x\rangle\ \mathit{safe}\}_\beta \cup !C \cup \{\delta\}$, если $\forall !y\in C\ \Sigma\ \mathit{after}\ \langle !y\rangle\ \mathit{safe}$.

[26] Это свойство совпадает со свойством βγδ-согласованности βγδ-модели.

последовательностью применений ***DRT***-операций, также o-конвергентна:

$\forall\sigma,\mu\in\mathbf{T}\ \forall o\in\beta\delta(C)\ \ \sigma\in\boldsymbol{DRT}(\mu)\ \&\ \sigma{\downarrow}o\ \Rightarrow\ \mu{\downarrow}o.$

(βδ4) βδ-замкнутость (замкнутость по ***DRT***-операциям): каждая трасса μ, получаемая из некоторой трассы $\sigma\in\mathbf{T}$ с помощью последовательности применения ***DRT***-операций, либо принадлежит $\mathbf{T}$, либо её префикс $\lambda\in\mathbf{T}$ продолжается в μ символом u, по которому этот префикс дивергентен в $\mathbf{T}$:

$\forall\sigma\in\mathbf{T}\ \forall\mu\in\boldsymbol{DRT}(\sigma)\ \ \mu\in\mathbf{T}\ \vee\ \exists\lambda\ \exists u\ \ \lambda{\cdot}\langle u\rangle\le\mu\ \&\ \lambda\in\mathbf{T}\ \&\ \lambda{\uparrow}u.$

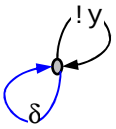
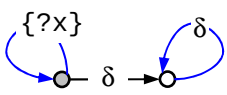
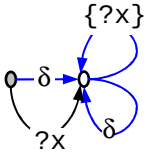
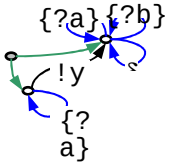
(βδ5) βδ-полнота (замкнутость по ***Ins***-операции): если трасса $\mu{\cdot}\langle l\rangle{\cdot}\lambda\in\mathbf{T}$ имеет префикс $\mu{\cdot}\langle l\rangle$, заканчивающийся отказом l, и этот префикс в $\mathbf{T}$ продолжается отказом r, но не продолжается каким-либо символом, принадлежащим r, то отказ r можно вставить в трассу после префикса $\mu{\cdot}\langle l\rangle$: $\forall\mu,\lambda\ \forall l,r\in\beta\delta(C)$

$\mu{\cdot}\langle l\rangle{\cdot}\lambda\in\mathbf{T}\ \&\ \mu{\cdot}\langle l,r\rangle\in\mathbf{T}\ \&\ \forall z\in r\ \ \mu{\cdot}\langle l,z\rangle\notin\mathbf{T}\ \Rightarrow\ \mu{\cdot}\langle l,r\rangle{\cdot}\lambda\in\mathbf{T}.$

Множество моделей безопасных трасс в алфавите C обозначим $\boldsymbol{MODEL}_{\text{safe}}(C)$, а множество всех моделей безопасных трасс обозначим $\boldsymbol{MODEL}_{\text{safe}}=\cup\{\boldsymbol{MODEL}_{\text{safe}}(C)\,|\,C\subseteq Z\}$.

□

Ортогональность свойств модели безопасных трасс показана контрпримерами на Рис.21. Здесь для каждого i-го свойства ($i=1\div5$) изображён порождающий граф βδ-дерева, удовлетворяющего всем свойствам модели безопасных трасс, кроме i-го.

ортогональность свойств модели безопасных трасс $\mathbf{T}$				
стационарность	согласованность	конвергентность	замкнутость	полнота
$C=\varnothing$	$C=\{!y\}$	$C=\{?x,!y\}$	$C=\{?x\}$	$C=\{?a,?b,!y\}$
$\mathbf{T}=\{\epsilon\}$				
$\epsilon\in\mathbf{T}$, но $\epsilon{\cdot}\langle\delta\rangle=\langle\delta\rangle\notin\mathbf{T}$	$\langle\delta,!y\rangle\in\mathbf{T}$ не согласована	$\epsilon,\langle\delta\rangle\in\mathbf{T}$, $\epsilon{\downarrow}\{?x\}$, $\epsilon\in\boldsymbol{DRT}(\langle\delta\rangle)$, но $\langle\delta\rangle{\uparrow}\{?x\}$	$\langle\delta,\{?x\}\rangle\in\mathbf{T}$, $\langle\{?x\}\rangle\in\boldsymbol{DRT}(\langle\delta,\{?x\}\rangle)$, но $\langle\{?x\}\rangle\notin\mathbf{T}$ и $\epsilon{\downarrow}\{?x\}$	$\langle\{?a\},!y\rangle\in\mathbf{T}$, $\langle\{?a\},\{?b\}\rangle\in\mathbf{T}$, $\langle\{?a\},?b\rangle\notin\mathbf{T}$, но $\langle\{?a\},\{?b\},!y\rangle\notin\mathbf{T}$
Рис.21.				

Пополнение до конвергентности.

Чем модель безопасных трасс отличается от βγδ-модели?

Поскольку в модели безопасных трасс нет разрушения, свойство βγδ-допустимости автоматически выполнено. Свойства βδ- и βγδ-согласованности совпадают. Тем самым, могут быть нарушены только три свойства: βγδ-конвергентность, βγδ-замкнутость и βγδ-полнота.

Для того, чтобы обеспечить свойство βγδ-конвергентности, определим пополнение модели безопасных трасс до конвергентного множества трасс. Если трасса $?x$-дивергентна, продолжим её стимулом $?x$ с последующим разрушением. Если множество реакций пусто, то, по βδ-стационарности, каждая трасса продолжается стационарностью δ и, тем самым, δ-конвергентна. Поэтому, если трасса δ-дивергентна, значит множество реакций не пусто. Тогда для того, чтобы сделать трассу δ-конвергентной, достаточно продолжить её какой-нибудь реакцией с последующим разрушением. Мы будем выбирать произвольное непустое множество таких реакций, в частности, множество всех реакций.

Определение 13: Пусть задан базовый алфавит $C \subseteq Z$ и модель безопасных трасс $\mathbf{T} \in \boldsymbol{MODEL}_{\text{safe}}(C)$. Продолжим каждую $?x$-дивергентную трассу $\sigma \in \mathbf{T}$ трассой $\langle ?x, \gamma \rangle$. Продолжим каждую δ-дивергентную трассу σ трассой $\langle !y, \gamma \rangle$ для одной или нескольких произвольно выбранных реакций $!y \in C$. Для трассы σ непустое множество выбранных реакций обозначим $\varnothing \neq Y_\sigma \subseteq !C$. Если трасса σ конвергентна по реакциям, доопределим $Y_\sigma = \varnothing$. Обозначим:

$\mathbf{\Sigma} = \mathbf{T} \cup \mathbf{X} \cup \mathbf{X} \cdot \{\langle \gamma \rangle\} \cup \mathbf{Y} \cup \mathbf{Y} \cdot \{\langle \gamma \rangle\}$, где

$\mathbf{X} = \{\sigma \cdot \langle ?x \rangle \mid \sigma \in \mathbf{T} \ \&\ \sigma \cdot \langle ?x \rangle \notin \mathbf{T} \ \&\ \sigma \cdot \langle \{?x\} \rangle \notin \mathbf{T}\}$

$\mathbf{Y} = \{\sigma \cdot \langle !y \rangle \mid \sigma \in \mathbf{T} \ \&\ !y \in Y_\sigma\}$.

Множество трасс $\mathbf{\Sigma}$ будем называть *пополнением до конвергентности* модели безопасных трасс $\mathbf{T}$. Если Y_σ для каждой δ-дивергентной трассы σ равно $!C$, то такое пополнение до конвергентности будем называть *γ-однородным* и обозначать $\mathbf{\Sigma} = \boldsymbol{CONV}_\gamma(\mathbf{T})$.

□

Утверждение 9: Пополнение до конвергентности модели безопасных трасс сохраняет множество безопасных трасс и обеспечивает выполнение всех свойств βγδ-модели, кроме, быть может, βγδ-замкнутости.

□369

***DRT*-замыкание.**

Теперь для того, чтобы из модели безопасных трасс получить βγδ-модель, нам нужно после пополнения до конвергентности обеспечить выполнение свойства βγδ-замкнутости. Для этого достаточно сделать ***DRT***-замыкание.

Утверждение 10: Если дерево трасс $\mathbf{\Sigma}$ обладает всеми свойствами βγδ-модели, кроме, быть может, βγδ-замкнутости, то его ***DRT***-замыкание $\cup_\circ \boldsymbol{DRT}(\mathbf{\Sigma})$ является βγδ-моделью с тем же множеством безопасных трасс: $\boldsymbol{safe}_\circ \cup_\circ \boldsymbol{DRT}(\mathbf{\Sigma}) = \boldsymbol{safe}(\mathbf{\Sigma})$.

□370

Из последних двух утверждений непосредственно следует, что каждую модель безопасных трасс можно достроить до βγδ-модели с сохранением безопасных трасс с помощью пополнения до конвергентности и ***DRT***-замыкания. Обратно, множество безопасных трасс каждой βγδ-модели является моделью безопасных трасс.

Модель безопасных трасс и βγδ-модель.

Утверждение 11: Пусть задан базовый алфавит $C \subseteq Z$. Для того, чтобы множество трасс $\mathbf{T}$ было подмножеством безопасных трасс некоторой βγδ-модели, необходимо и достаточно, чтобы $\mathbf{T}$ было моделью безопасных трасс: $\forall C \subseteq Z \ \ \boldsymbol{safe}_\circ \boldsymbol{MODEL}_{\beta\gamma\delta}(C) = \boldsymbol{MODEL}_{\text{safe}}(C)$.

□372

По определению, множество безопасных трасс дерева однозначно определяется этим деревом. Обратное, вообще говоря, не верно: могут существовать два разных дерева, имеющих одно и то же подмножество безопасных трасс. Это утверждение верно и для βγδ-моделей (примеры порождающих графов на Рис.22).

Кроме того, в отличие от βγδ-модели, для модели безопасных трасс мы не требуем непустоты дерева трасс. Пустая модель безопасных трасс соответствует βγδ-модели, которая не является безопасной, то есть содержит трассу $\langle\gamma\rangle$ (Рис.22 справа).

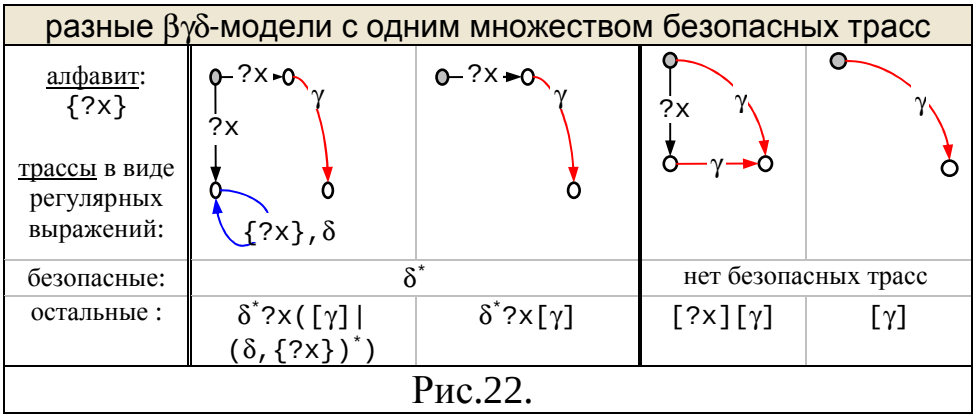


Рис.22.

Во всех примерах на Рис.22 подмодель безопасных трасс не является βγδ-моделью: она содержит дивергентные трассы или пуста.

В общем, соотношение моделей такое, как изображено на Рис.23.

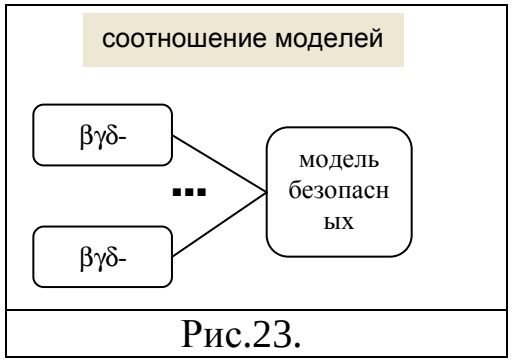


Рис.23.

Заметим также, что, хотя модели безопасных трасс $\boldsymbol{MODEL}_{\text{safe}}(C)$ и βδ-модели $\boldsymbol{MODEL}_{\beta\delta}(C)$ являются деревьями трасс в алфавите $C_{\beta\delta}$, они не совпадают. В βδ-модели все трассы конвергентны, а в модели безопасных трасс некоторые трассы дивергентны, что соответствует разрушающим стимулам или реакциям после этих трасс в соответствующих βγδ-моделях. С другой стороны, все трассы βδ-модели, очевидно, безопасны, так что подмножество безопасных трасс βδ-модели совпадает с ней самой. Поэтому любая βδ-модель является моделью безопасных трасс: $\boldsymbol{MODEL}_{\beta\delta}(C) \subseteq \boldsymbol{MODEL}_{\text{safe}}(C)$.

2.2.4. Модель финальных трасс

Структура раздела:

- Финальные трассы.
- Определение модели финальных трасс.
- Модель финальных трасс и βγδ-модель.

Финальные трассы.

Определение 14: Финальной трассой дерева трасс будем называть любой префикс трассы дерева, которая либо безопасна в дереве, либо является продолжением конечной безопасной трассы дерева символом и далее разрушением, либо состоит из одного разрушения. Множества всех финальных трасс дерева обозначим: для $C\subseteq Z$ и $T\in \mathit{Trees}(C_{\beta\gamma\delta})$

$$\mathit{final}(T) =_{def}$$
$$\{\ \sigma \mid \exists\mu\in \mathit{safe}(T)\ \exists\lambda\in T\ \exists z\in C\ \sigma\le\lambda\ \&\ \lambda\in\{\mu, \mu\cdot\langle z,\gamma\rangle, \langle\gamma\rangle\}\ \}.$$

□

Мы покажем, что трасса безопасна в поддереве финальных трасс тогда и только тогда, когда она безопасна в исходном дереве трасс. Мы дадим явное определение модели финальных трасс, сформулировав ряд требований на дерево трасс. Потом покажем, что эти требования необходимы и достаточны для того, чтобы модель финальных трасс была множеством финальных трасс некоторой βγδ-модели.

Определение модели финальных трасс.

Определение 15: Пусть задан базовый алфавит $C\subseteq Z$. Непустое дерево $T\in \mathit{Trees}(C_{\beta\gamma\delta})$ будем называть *моделью финальных трасс* в алфавите $C\subseteq Z$, если выполнены следующие следующие три требования:

(final1) финальность: все трассы T финальны $\mathit{final}(T)=T$;

(final2) конвергентность: все трассы T, не содержащие и не продолжающиеся разрушением, конвергентны[27];

(final3) безопасность: подмножество безопасных трасс T является моделью безопасных трасс $\mathit{safe}(T)\in \mathit{MODEL}_{safe}(C)$.

Множество моделей финальных трасс в алфавите C обозначим $\mathit{MODEL}_{final}(C)$, а множество всех моделей финальных трасс обозначим $\mathit{MODEL}_{final}=\cup\{\mathit{MODEL}_{final}(C)\mid C\subseteq Z\}$.

□

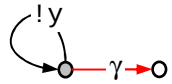
Прежде всего, заметим, что свойство βδ-стационарности подмножества безопасных трасс следует из свойства конвергентности: поскольку безопасная трасса должна быть конвергентна в T, то при отсутствии реакций она должна продолжаться стационарностью, а такое продолжение безопасно, поскольку реакций нет.

Ортогональность свойств модели финальных трасс показана контрпримерами на Рис.24, где выполнены все свойства, кроме указанного.

---

[27] Это свойство совпадает со свойством конвергентности βγδ-модели.

Для свойства `final3`:безопасность, включающем пять βδ-свойств модели безопасных трасс, контрпримеры приведены для каждого из этих βδ-свойств, кроме βδ-стационарности, которая следует из конвергентности всех трасс в дереве **T**.

ортогональность свойств модели финальных трасс **T**	
финальность	конвергентность
C={!y}	C={!y}
	**T**={ϵ} 
⟨!y,!y⟩ не финальна	ϵ не конвергентна: ⟨δ⟩∉**T**

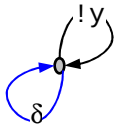
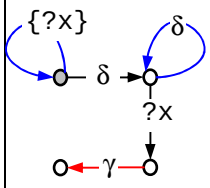
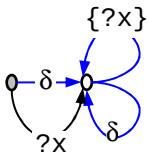
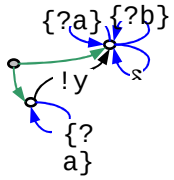
безопасность **Σ**=***safe***(**T**)			
βδ-согласованность	βδ-конвергентность	βδ-замкнутость	βδ-полнота
C={!y}	C={?x,!y}	C={?x}	C={?a,?b,!y}
			
⟨δ,!y⟩∈**Σ** не согласована	⟨δ⟩∈**Σ**, ϵ∈***DRT***(⟨δ⟩), ϵ↓{?x} в **Σ**, но ⟨δ⟩↑{?x} в **Σ**	⟨δ,{?x}⟩∈**Σ**, ⟨{?x}⟩∈***DRT***(⟨δ,{?x}⟩), но ⟨{?x}⟩∉**Σ** и ϵ↓{?x} в **Σ**	⟨{?a},!y⟩∈**Σ**, ⟨{?a},{?b}⟩∈**Σ**, ⟨{?a},?b⟩∉**Σ**, но ⟨{?a},{?b},!y⟩∉**Σ**

Рис.24.

Модель финальных трасс и βγδ-модель.

Переход от дерева трасс к его поддереву финальных трасс сохраняет безопасные трассы.

Утверждение 12: ***safe*∘*final*=*safe***.

□373

Поддерево финальных трасс любого дерева обладает свойством финальности.

Утверждение 13: ***final*∘*final*=*final***.

□374

Утверждение 14: Если **Σ** – модель финальных трасс, то её ***DRT***-замыкание ∪∘***DRT***(**Σ**) является βγδ-моделью с тем же множеством финальных трасс: ***final*∘∪∘*DRT*(Σ)=Σ**.

□374

Утверждение 15: Пусть задан базовый алфавит $C \subseteq Z$. Для того, чтобы множество трасс **T** было подмножеством финальных трасс некоторой βγδ-модели, необходимо и достаточно, чтобы **T** было моделью финальных трасс: $\forall C \subseteq Z \ \ \boldsymbol{final} \circ \boldsymbol{MODEL}_{\beta\gamma\delta}(C) = \boldsymbol{MODEL}_{final}(C)$.

□375

По определению, множество финальных трасс дерева однозначно определяется этим деревом. Обратное, вообще говоря, не верно: могут существовать два разных дерева, имеющих одно и то же подмножество финальных трасс. Это утверждение верно и для βγδ-моделей (примеры порождающих графов на Рис.25).

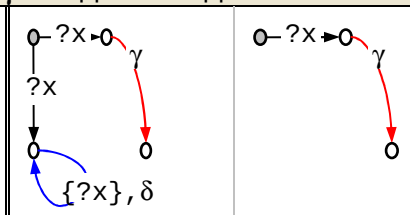
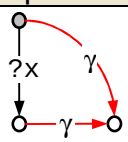
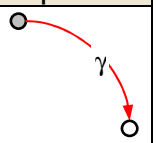
разные βγδ-модели с одним множеством финальных трасс				
алфавит: {?x} трассы в виде регулярных выражений:				
безопасные:	δ^*		нет безопасных трасс	
финальные:	$\delta^*[?x[\gamma]]$		$[\gamma]$	
остальные:	$\delta^*?x(\delta,\{?x\})^*$	нет	$?x[\gamma]$	нет

Рис.25.

В примерах на Рис.25 подмодель финальных трасс является βγδ-моделью. Но, вообще говоря, это не всегда так: модель финальных трасс может быть не замкнута по ***DRT***-операциям (Рис.26).

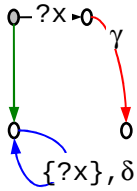
модель финальных трасс, но не βγδ-модель	
алфавит: {?x} трассы в виде регулярных выражений:	
безопасные:	$\delta(\delta,\{?x\})^*$
финальные:	$\delta(\delta,\{?x\})^* \mid \delta^*?x[\gamma]$
остальные:	$\{?x\}(\delta,\{?x\})^*$

Рис.26.

2.2.5. Соотношение трассовых моделей

Структура раздела:

- *F*-изоморфные и *F*-канонические βγδ-модели.
- *S*-изоморфные, γ-однородные и *S*-канонические модели финальных трасс и βγδ-модели.
- Соотношение выделенных моделей.

В целом, соотношение трассовых моделей такое, как изображено на Рис.27.

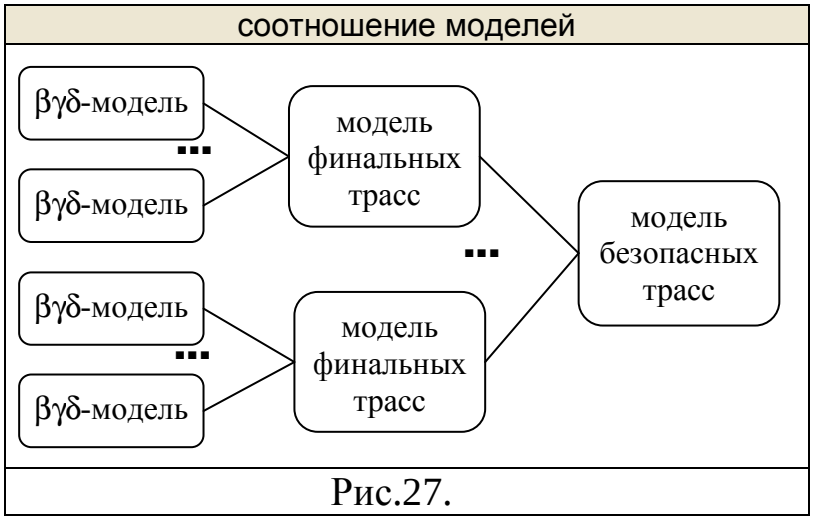


Рис.27.

***F*-изоморфные и *F*-канонические βγδ-модели.**

Определение 16: Две βγδ-модели $\boldsymbol{\Sigma}$ и $\boldsymbol{\Sigma}`$ будем называть *F-изоморфными*[28], если они имеют одно и то же множество финальных трасс: $\boldsymbol{final}(\boldsymbol{\Sigma})=\boldsymbol{final}(\boldsymbol{\Sigma}`)$.

□

Очевидно, *F*-изоморфизм является отношением эквивалентности. Среди класса *F*-изоморфных βγδ-моделей можно выделить "минимальную" βγδ-модель как пересечение моделей класса, которую будем называть *F*-канонической.

Утверждение 16: Пусть задан базовый алфавит $C\subseteq Z$. Пересечение βγδ-моделей класса *F*-изоморфных βγδ-моделей в алфавите C принадлежит этому классу и совпадает с ***DRT***-замыканием множества финальных трасс:

обозначая $\boldsymbol{F}(\boldsymbol{\Sigma})=\{T\in\boldsymbol{MODEL}_{\beta\gamma\delta}(C)\,|\,\boldsymbol{final}(T)=\boldsymbol{\Sigma}\}$,

$\forall C\subseteq Z\ \ \forall\boldsymbol{\Sigma}\in\boldsymbol{MODEL}_{\text{final}}(C)\ \ \cap_{\circ}\boldsymbol{F}(\boldsymbol{\Sigma})\in\boldsymbol{F}(\boldsymbol{\Sigma})\ \ \&\ \ \cap_{\circ}\boldsymbol{F}(\boldsymbol{\Sigma})=\cup_{\circ}\boldsymbol{DRT}(\boldsymbol{\Sigma})$.

□376

Определение 17: Пересечение βγδ-моделей класса *F*-изоморфных βγδ-моделей в алфавите C будем называть *F-канонической* βγδ-моделью.

□

Таким образом, мы имеем взаимно-однозначное соответствие между моделями финальных трасс и *F*-каноническими βγδ-моделями (Рис.28).

---

[28] ***F***- от final - финальная.

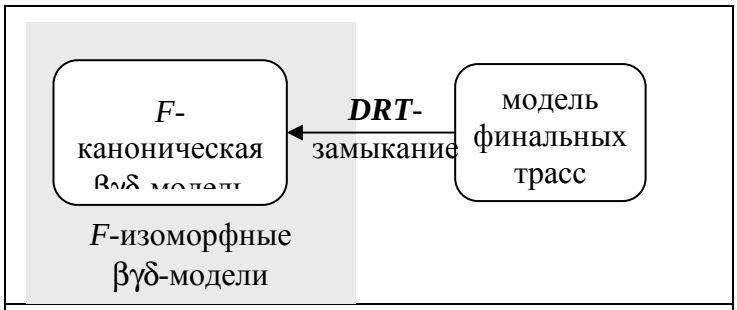


Рис.28.

S-изоморфные, γ-однородные и S-канонические модели финальных трасс и βγδ-модели.

Определение 18: Две модели финальных трасс (две βγδ-модели) Σ и Σ' будем называть *S-изоморфными*[29], если они имеют одно и то же множество безопасных трасс: $\mathit{safe}(\Sigma)=\mathit{safe}(\Sigma')$.

□

Среди множества *S*-изоморфных моделей финальных трасс (βγδ-моделей) можно выделить одну модель, в которой после каждой трассы все опасные реакции разрушающие. Такую модель будем называть γ-однородной.

Определение 19: Пусть задан базовый алфавит $C\subseteq Z$. Модель финальных трасс $\Sigma\in \mathit{MODEL}_{\text{final}}(C)$ будем называть *γ-однородной*, если

$\forall\sigma\ \exists!y\in C\ \sigma\cdot\langle!y,\gamma\rangle\in\Sigma \Rightarrow \forall!t\in C\ \sigma\cdot\langle!t,\gamma\rangle\in\Sigma$.

Соответственно, βγδ-модель будем называть γ-однородной, если её подмодель финальных трасс γ-однородна. *F*-каноническую γ-однородную βγδ-модель будем также называть *S-канонической*.

□

Заметим, что стимул, опасный после трассы, всегда разрушающий. Тем самым, в γ-однородной модели для каждой безопасной трассы и каждого опасного после неё базового символа (стимула или реакции) эта трасса продолжается этим символом с последующим разрушением:

$\forall\sigma\in\mathit{safe}(\Sigma)\ \ \forall z\in C\setminus\mathit{safesymbols}(\Sigma\ \mathit{after}\ \sigma)\ \ \sigma\cdot\langle z,\gamma\rangle\in\Sigma$.

Утверждение 17: Пусть задан базовый алфавит $C\subseteq Z$. В каждом классе *S*-изоморфных моделей финальных трасс в алфавите C существует и только одна γ-однородная модель, и она совпадает с объединением всех моделей класса и γ-однородным пополнением до конвергентности подмодели безопасных трасс:

---

[29] ***S***- от safe - безопасная.

обозначая $\boldsymbol{B}(\Sigma)=\{T\in \boldsymbol{MODEL}_{final}(C)\mid \boldsymbol{safe}(T)=\Sigma\}$,
$\forall C\subseteq Z\ \ \forall\Sigma\in \boldsymbol{MODEL}_{safe}(C)$
$\cup_{\circ}\boldsymbol{B}(\Sigma)\in\boldsymbol{B}(\Sigma)$ & $\cup_{\circ}\boldsymbol{B}(\Sigma)$ γ-однородна & $\cup_{\circ}\boldsymbol{B}(\Sigma)=\boldsymbol{CONV}_{\gamma}(\Sigma)$
& $\forall T\in\boldsymbol{B}(\Sigma)$ (T γ-однородна $\Rightarrow$ $T=\cup_{\circ}\boldsymbol{B}(\Sigma)$).

□376

Таким образом, мы имеем взаимно-однозначное соответствие между моделями безопасных трасс и γ-однородными моделями финальных трасс (Рис.29).

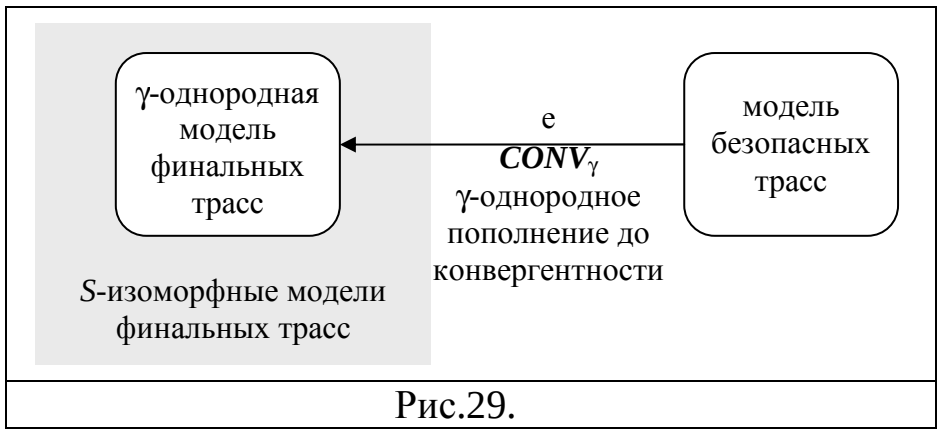


Рис.29.

Соотношение выделенных моделей.

γ-однородная βγδ-модель может не быть *F*-канонической. Однако, ***DRT***-замыкание γ-однородной модели финальных трасс является γ-однородной *F*-канонической, то есть *S*-канонической βγδ-моделью.

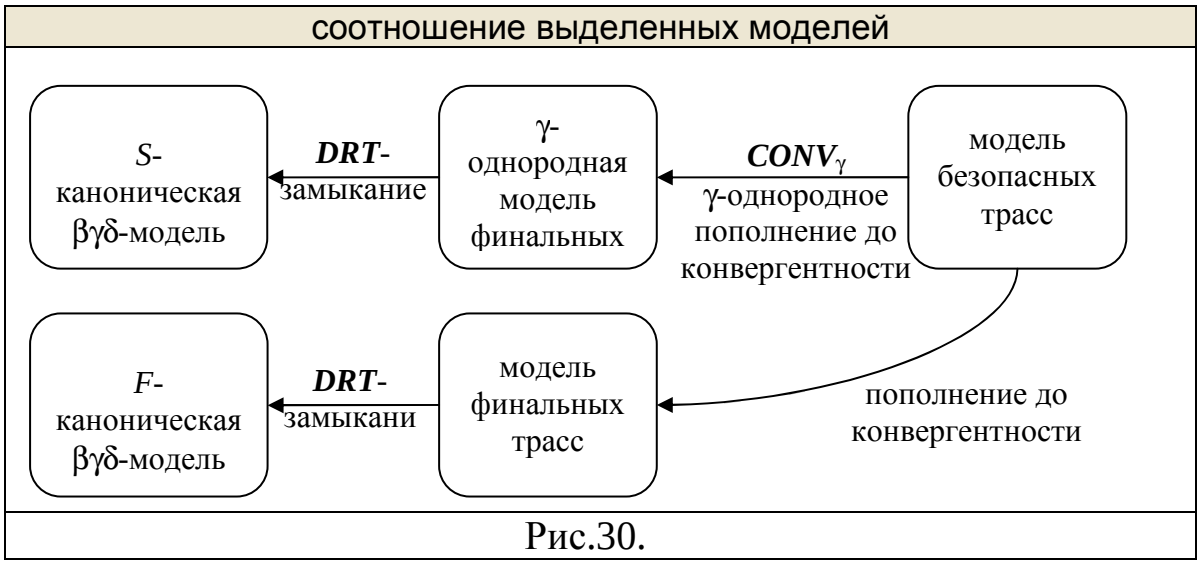


Рис.30.

2.2.6. Гипотеза о безопасности

Структура раздела:

- Безопасность тестирования.

- Тестовые трассы.
- Отношение *safe for*.

Безопасность тестирования.

Первое требование, предъявляемое к тестированию, – оно должно быть безопасным, то есть не разрушать реализацию. Если в реализации после некоторой трассы есть разрушающие стимулы, тестер не должен их посылать после этой трассы. Если в реализации есть разрушающие реакции после некоторой трассы, тестер не должен принимать реакций после этой трассы. При безопасном тестировании реализация может «пройти» только безопасную трассу. Если реализация разрушается «сама по себе», без посылки стимулов из тестера и приёма реакций в тестер, такую реализацию вообще нельзя тестировать. Это означает, что такая реализация не должна проходить даже пустую трассу, то есть её нельзя помещать внутрь «чёрного ящика» (или нельзя «включать» машину тестирования, если она начинает работать только при нажатии специальной кнопки включения).

Поскольку о безопасности трассы реализации мы можем судить только по спецификации, для безопасного тестирования нам требуется соответствующая гипотеза о реализации, которую будем называть *гипотезой о безопасности*. Эту гипотезу мы сформулируем в терминах тестовых трасс, то есть трасс, которые могут наблюдаться при безопасном тестировании.

Пусть при тестировании мы наблюдаем трассу, которая безопасна как в спецификации, так и в реализации. После такой трассы мы будем посылать в реализацию только такой стимул, который не может вызвать разрушение в спецификации, и будем требовать, чтобы в реализации такой стимул также не вызывал разрушение в этой ситуации. Аналогично, мы будем принимать реакции после трассы, если это не может вызвать разрушение в спецификации, и будем требовать, чтобы в реализации приём реакций также не вызывал разрушение в этой ситуации.

Заметим, что безопасность символа после трассы не означает, что эта трасса продолжается этим символом. Поведение спецификации и реализации при передаче стимула или реакции может быть различным; мы требуем только следующего: если поведение спецификации исключает разрушение, то и в реализации разрушение невозможно.

Тестирование заканчивается либо потому, что мы полностью проверили интересующую нас безопасную трассу спецификации и не хотим продолжать тестирование без рестарта, либо потому, что после некоторой безопасной трассы реализация повела себя не так, как это разрешает спецификация. В любом случае итоговая трасса будет в спецификации

безопасной трассой (в первом случае) или безопасной трассой, продолженной безопасным символом (в обоих случаях).

Таким образом, для заданной спецификации тестовой трассой будем считать безопасную трассу или продолжение безопасной трассы безопасным символом (Рис.31 слева).

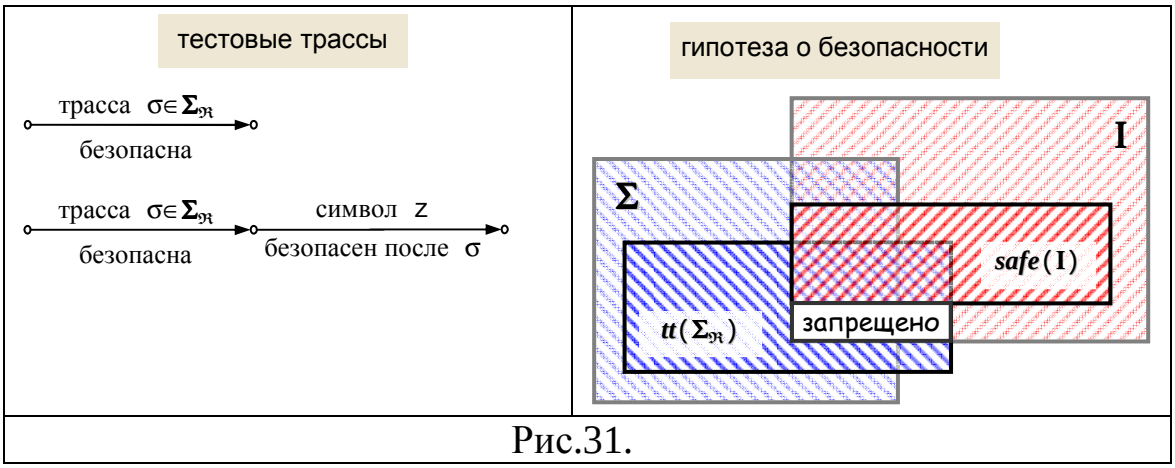


Рис.31.

Гипотеза о безопасности теперь может быть сформулирована следующим образом (Рис.31 справа):

любая тестовая трасса спецификации либо отсутствует в реализации, либо безопасна в ней.

Реализации, удовлетворяющих этой гипотезе, будем называть *S-тестируемыми*[30] для данной спецификации.

Тестовые трассы.

Определение 20: Пусть $C \subseteq Z$, $\Sigma \in \boldsymbol{MODEL}_{\beta\gamma\delta}(C)$.

- *Тестовой трассой* для βγδ-модели Σ будем называть такую трассу $\sigma \in C_{\beta\delta}^{*}$, которая удовлетворяет условию:

  $\sigma \in \boldsymbol{safe}(\Sigma) \vee \exists \mu \in \boldsymbol{safe}(\Sigma)\ \exists u \in \boldsymbol{safesymbols}(\Sigma\ \boldsymbol{after}\ \mu)\ \sigma = \mu \cdot \langle u \rangle$.

- Множество тестовых трасс для спецификации Σ обозначим $\boldsymbol{tt}(\Sigma)$.

□

Можно дать следующее эквивалентное определение тестовой трассы, используя оператор взятия тестовых трасс от заданной трассы.

[30] *S-* от safe - безопасная.

Определение 21: **Эквивалентное определение тестовой трассы.**

· Определим оператор взятия тестовых трасс от заданной трассы:

$\boldsymbol{t}(\epsilon) =_{def} \{\epsilon\}$,

$\boldsymbol{t}(\mu\cdot\langle ?x\rangle) =_{def} \boldsymbol{t}(\mu\cdot\langle\{?x\}\rangle) =_{def} \{\mu\cdot\langle ?x\rangle, \mu\cdot\langle\{?x\}\rangle\}$,

$\boldsymbol{t}(\mu\cdot\langle !y\rangle) =_{def} \boldsymbol{t}(\mu\cdot\langle\delta\rangle) =_{def} \{\mu\cdot\langle u\rangle \mid u\in !C_\delta\}$.

· Для $\Sigma\in \boldsymbol{MODEL}_{\beta\gamma\delta}(C)$ определим множество тестовых трасс:

$\boldsymbol{tt}(\Sigma) =_{def} \cup\circ\boldsymbol{t}\circ\boldsymbol{safe}(\Sigma) = \cup(\{\boldsymbol{t}(\mu)\mid\mu\in\boldsymbol{safe}(\Sigma)\})$.

□

Заметим, что тестовые трассы могут быть не согласованы. Поскольку трасса $\mu\in\boldsymbol{safe}(\Sigma)$ согласована, несогласованной может быть только тестовая трасса $\sigma=\mu\cdot\langle u\rangle$, когда $u=?x$ – стимул или реакция $u=!y$, но $\{?x\}\in\boldsymbol{RefT}(\mu)$ или $\delta\in\boldsymbol{RefT}(\mu)$, соответственно. Как мы увидим ниже, для целей тестирования достаточно только согласованных тестовых трасс. Однако для общности мы рассматриваем произвольные тестовые трассы, исходя только из требований безопасности тестирования, а не его эффективности.

Утверждение 18: Пусть $C\subseteq Z$, $\Sigma, I\in\boldsymbol{MODEL}_{\beta\gamma\delta}(C)$. Тогда

1) безопасные трассы βγδ-модели – это те её тестовые трассы, которые ей принадлежат: $\boldsymbol{safe}(\Sigma)=\boldsymbol{tt}(\Sigma)\cap\Sigma$;
2) тестовая трасса продолжается стимулом тогда и только тогда, когда она продолжается блокировкой этого стимула:
   $\forall\sigma\in\boldsymbol{tt}(\Sigma)\ \forall ?x\in C\ \sigma\cdot\langle ?x\rangle\in\boldsymbol{tt}(\Sigma) \Leftrightarrow \sigma\cdot\langle\{?x\}\rangle\in\boldsymbol{tt}(\Sigma)$;
3) тестовая трасса продолжается либо всеми реакциями и стационарностью, либо не продолжается ни одной реакцией и стационарностью:
   $\forall\sigma\in\boldsymbol{tt}(\Sigma)\ (\exists u\in !C_\delta\ \sigma\cdot\langle u\rangle\in\boldsymbol{tt}(\Sigma)) \Leftrightarrow \forall u\in !C_\delta\ \sigma\cdot\langle u\rangle\in\boldsymbol{tt}(\Sigma)$;
4) безопасные трассы однозначно определяют тестовые трассы:
   $\boldsymbol{safe}(\Sigma)=\boldsymbol{safe}(I) \Rightarrow \boldsymbol{tt}(\Sigma)=\boldsymbol{tt}(I)$;
5) тестовые трассы, вообще говоря, не однозначно определяют безопасные трассы.

□376

Отношение *safe for*.

Поскольку $\boldsymbol{safe}(I)=\boldsymbol{tt}(I)\cap I$, гипотеза о безопасности может быть переформулирована в следующем эквивалентном виде:

любая тестовая трасса спецификации, присутствующая в реализации, является её тестовой трассой.

Определение 22: Пусть $C\subseteq Z$, $\Sigma,I\in\boldsymbol{MODEL}_{\beta\gamma\delta}(C)$.

· Реализация **I** *S*-тестируема для данной спецификации **Σ**:
I ***safe for*** **Σ** $=_{def}$ $tt(\Sigma) \cap I \subseteq tt(I)$.
· Множество *S*-тестируемых реализаций для данной спецификации **Σ**:
$safeIMPL(\Sigma) =_{def} \{I \in MODEL_{\beta\gamma\delta}(C) \mid I \text{ safe for } \Sigma\}$.

□

Утверждение 19: Условие безопасности **I** ***safe for*** **Σ** эквивалентно конъюнкции следующих двух условий:
1. если спецификация безопасна, то реализация также безопасна:
   Σ ***safe*** $\Rightarrow$ **I** ***safe***;
2. после общей безопасной трассы символ, безопасный в спецификации, безопасен и в реализации (Рис.33):
   $\forall \sigma \in safe(\Sigma) \cap safe(I)$
   $safesymbols(\Sigma \text{ after } \sigma) \subseteq safesymbols(I \text{ after } \sigma)$.

□377

2.2.7. Соответствие $ioco_{\beta\gamma\delta}$

Структура раздела:
· Определение отношения $ioco_{\beta\gamma\delta}$.
· $ioco_{\beta\gamma\delta}$-предпорядок и $ioco_{\beta\gamma\delta}$-эквивалентность.
· Связь $ioco_{\beta\gamma\delta}$ с множествами трасс.

Соответствие моделируется бинарным отношением, связывающим спецификацию только с *S*-тестируемыми для этой спецификации реализациями, то есть, такими, которые удовлетворяют гипотезе о безопасности для этой спецификации. В настоящей работе мы рассматриваем отношение $\boldsymbol{ioco}_{\beta\gamma\delta}$ – обобщение отношения ***ioco*** [Tret96] для систем, в которых возможна блокировка стимулов и разрушение. Семантику отношения ***ioco*** и его сравнение с отношением $\boldsymbol{ioco}_{\beta\gamma\delta}$ мы рассмотрим ниже в специальном разделе 2.5.3.

Определение отношения $ioco_{\beta\gamma\delta}$.

Правила отношения $\boldsymbol{ioco}_{\beta\gamma\delta}$ имеют вид "*Реализация может … только тогда, когда для спецификации это возможно в такой же ситуации, то есть, после той же самой βγδ-трассы, безопасной в спецификации*", где многоточие означает *символ, оставляющий трассу безопасной в спецификации*, то есть одно из следующих:

Правило реакции: "выдать данную реакцию".
Правило стационарности: "не выдавать никаких реакций".
Правило стимула: "принять данный стимул".
Правило блокировки: "блокировать данный стимул".

Это условие для реализации $\mathbf{I}$ и спецификации $\mathbf{\Sigma}$ может быть записано также в трёх эквивалетных формах:

- $\forall \sigma \in \mathit{safe}(\mathbf{\Sigma})\ \ \forall u \in \mathit{safesymbols}(\mathbf{\Sigma}\ \mathit{after}\ \sigma)\ \ \sigma \cdot \langle u \rangle \in \mathbf{I} \Rightarrow \sigma \cdot \langle u \rangle \in \mathbf{\Sigma}$,
- $\forall \sigma \in \mathit{safe}(\mathbf{\Sigma})$
  $\mathit{safesymbols}(\mathbf{\Sigma}\ \mathit{after}\ \sigma) \cap \mathit{head}(\mathbf{I}\ \mathit{after}\ \sigma) \subseteq \mathit{head}(\mathbf{\Sigma}\ \mathit{after}\ \sigma)$,
- $\mathit{tt}(\mathbf{\Sigma}) \cap \mathbf{I} \subseteq \mathbf{\Sigma}$.

Определение 23: Для $C \subseteq Z$, $\mathbf{\Sigma}, \mathbf{I} \in \mathit{MODEL}_{\beta\gamma\delta}(C)$

- Реализация $\mathbf{I}$ конформна спецификации $\mathbf{\Sigma}$:
  $$\begin{aligned} \mathbf{I}\ \mathit{ioco}_{\beta\gamma\delta}\ \mathbf{\Sigma} &=_{\text{def}} \mathbf{I}\ \mathit{safe\ for}\ \mathbf{\Sigma} \ \&\ \mathit{tt}(\mathbf{\Sigma}) \cap \mathbf{I} \subseteq \mathbf{\Sigma} \\ &= \mathit{tt}(\mathbf{\Sigma}) \cap \mathbf{I} \subseteq \mathit{tt}(\mathbf{I}) \ \&\ \mathit{tt}(\mathbf{\Sigma}) \cap \mathbf{I} \subseteq \mathbf{\Sigma} \\ &= \mathit{tt}(\mathbf{\Sigma}) \cap \mathbf{I} \subseteq \mathbf{\Sigma} \cap \mathit{tt}(\mathbf{I}). \end{aligned}$$
- Множество реализаций, конформных данной спецификации $\mathbf{\Sigma}$:
  $\mathit{confIMPL}(\mathbf{\Sigma}) =_{\text{def}} \{\ \mathbf{I} \in \mathit{MODEL}_{\beta\gamma\delta}(C)\ \mid\ \mathbf{I}\ \mathit{ioco}_{\beta\gamma\delta}\ \mathbf{\Sigma}\ \}$.

□

Условие конформности есть конъюнкция двух условий: *S*-тестируемость реализации и тестируемое условие. *S*-тестируемость реализации есть предусловие тестирования, оно предполагается выполненным и не проверяется при тестировании. Тестируемое условие – это и есть то условие, которое проверяется при тестировании.

Соотношение трасс и тестовых трасс реализации и спецификации даёт 16 вариантов, проиллюстрированных таблицей на Рис.32. Каждый вариант определяет, каким их четырёх множеств трасс принадлежит данная трасса. В восьми вариантах (t0÷t7) трасса не является тестовой. В четырёх вариантах (t8÷tB) трассу можно тестировать, но её нет в реализации, и она не может наблюдаться при тестировании этой реализации. Два варианта (tC÷tD) запрещены гипотезой о безопасности. В остальных двух вариантах (tE÷tF) трассу можно тестировать и наблюдать для данной реализации, причём в варианте tE тестирование обнаруживает неконформность – нарушение тестируемого условия.

	$tt(\Sigma)$	I	$tt(I)$	Σ	гипотеза о безопасности $tt(\Sigma)\cap I \subseteq tt(I)$	тестируемое условие $tt(\Sigma)\cap I \subseteq \Sigma$	
t0	*нет*	*нет*	*нет*	*нет*	трассу нельзя тестировать		**трасса не тестовая**
t1	*нет*	*нет*	*нет*	**да**			
t2	*нет*	*нет*	**да**	*нет*			
t3	*нет*	*нет*	**да**	**да**			
t4	*нет*	**да**	*нет*	*нет*			
t5	*нет*	**да**	*нет*	**да**			
t6	*нет*	**да**	**да**	*нет*			
t7	*нет*	**да**	**да**	**да**			
t8	**да**	*нет*	*нет*	*нет*	трассу можно тестировать, но она не наблюдается для данной реализации		**тестовая трасса**
t9	**да**	*нет*	*нет*	**да**			
tA	**да**	*нет*	**да**	*нет*			
tB	**да**	*нет*	**да**	**да**			
tC	**да**	**да**	*нет*	*нет*	**запрещено:** трасса опасна в реализации	запрещено	
tD	**да**	**да**	*нет*	**да**			
tE	**да**	**да**	**да**	*нет*	трассу можно тестировать и наблюдать для данной реализации	**неконформность**	
tF	**да**	**да**	**да**	**да**		**конформность**	

Рис.32.

На Рис.33 слева показано соотношение безопасных после трассы символов спецификации и реализации, допускаемое гипотезой о безопасности. При условии, что эта гипотеза выполнена, справа показано соотношение безопасных и продолжающих символов в спецификации и в реализации, допускаемое тестируемым условием. Обозначения областей взяты из Рис.34.

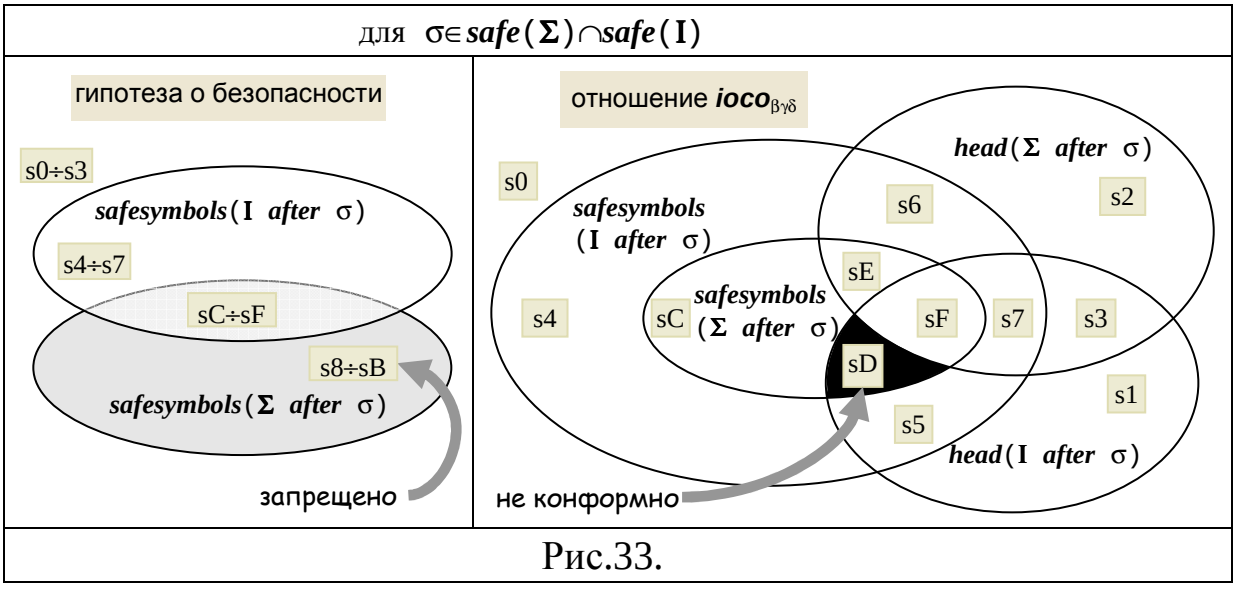


Рис.33.

Соотношение безопасных и продолжающих символов в спецификации и в реализации даёт 16 вариантов, проиллюстрированных таблицей на Рис.34. В этой таблице варианты расписаны для трассы σ и одного символа, в качестве примера выбрана блокировка стимула {?x}. В восьми вариантах

(s0÷s7) стимул ?x не проверяется после трассы, поскольку он разрушающий в спецификации. Четыре варианта (s8÷sB) запрещены гипотезой о безопасности. В остальных четырёх вариантах стимул ?x проверяется после трассы, и один вариант (sD) означает нарушение тестируемого условия, то есть неконформность, проверяемую при тестировании.

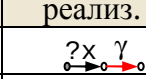
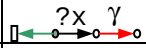
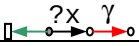
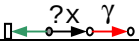
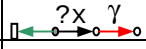
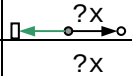
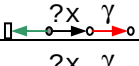
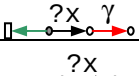
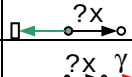
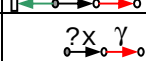
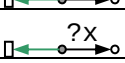
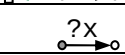
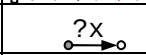
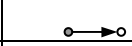
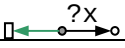
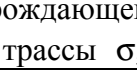
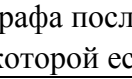
	блокировка {?x} после трассы σ				поведение после трассы σ в		
	безопасна в		имеется в				
	специф.	реализ.	специф.	реализ.	специф.	реализ.	
s0	*нет*	*нет*	*нет*	*нет*			конформно не проверяем стимул ?x
s1	*нет*	*нет*	*нет*	**да**			
s2	*нет*	*нет*	**да**	*нет*			
s3	*нет*	*нет*	**да**	**да**			
s4	*нет*	**да**	*нет*	*нет*			
s5	*нет*	**да**	*нет*	**да**			
s6	*нет*	**да**	**да**	*нет*			
s7	*нет*	**да**	**да**	**да**			
s8	**да**	*нет*	*нет*	*нет*			запрещено гипотезой безопасности
s9	**да**	*нет*	*нет*	**да**			
sA	**да**	*нет*	**да**	*нет*			
sB	**да**	*нет*	**да**	**да**			
sC	**да**	**да**	*нет*	*нет*			конформно
sD	**да**	**да**	*нет*	**да**			не конформно
sE	**да**	**да**	**да**	*нет*			конформно
sF	**да**	**да**	**да**	**да**			конформно
серый кружок – начальная вершина порождающего графа после трассы σ белый прямоугольник – вершина после трассы σ, в которой есть петля по {?x}							

Рис.34.

$ioco_{\beta\gamma\delta}$-предпорядок и $ioco_{\beta\gamma\delta}$-эквивалентность.

Утверждение 20: Отношение ***safe for*** рефлексивно, но не транзитивно, а отношение $ioco_{\beta\gamma\delta}$ рефлексивно и транзитивно (предпорядок).

□378

Отношение предпорядка индуцирует эквивалентность.

Определение 24: Две βγδ-модели $\mathbf{I}, \mathbf{\Sigma} \in MODEL_{\beta\gamma\delta}(C)$, где $C \subseteq Z$, будем называть $ioco_{\beta\gamma\delta}$-*эквивалентными* и обозначать $\mathbf{I} \sim_{ioco_{\beta\gamma\delta}} \mathbf{\Sigma}$,
если $\mathbf{I}\ ioco_{\beta\gamma\delta}\ \mathbf{\Sigma}$ и $\mathbf{\Sigma}\ ioco_{\beta\gamma\delta}\ \mathbf{I}$.

□

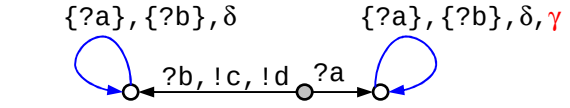
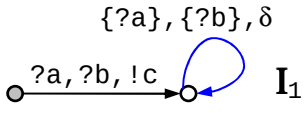
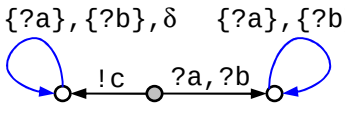
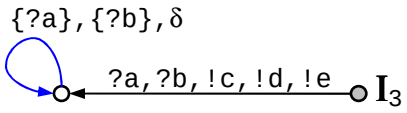
Связь $ioco_{\beta\gamma\delta}$ с множествами трасс.

Теперь мы будем исследовать связь $ioco_{\beta\gamma\delta}$-конформности и $ioco_{\beta\gamma\delta}$-эквивалентности с равенством или вложенностью множеств безопасных трасс, финальных трасс и всех трасс.

Утверждение 21: Для $C \subseteq Z$ и $I, \Sigma \in MODEL_{\beta\gamma\delta}(C)$
$I \sim_{ioco_{\beta\gamma\delta}} \Sigma \Leftrightarrow safe(I) = safe(\Sigma)$.

□380

Заметим, что сама по себе вложенность безопасных трасс не является ни необходимым, ни достаточным условием $ioco_{\beta\gamma\delta}$-конформности. Примеры в виде порождающих графов приведены на Рис.35.

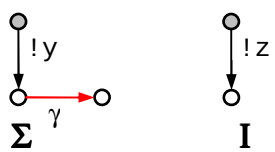
вложенность безопасных трасс и условие конформности		
		
		
I_1 $ioco_{\beta\gamma\delta}$ Σ $safe(I_1) \not\subseteq safe(\Sigma)$ $safe(I_1) \not\supseteq safe(\Sigma)$	I_2 $\neg ioco_{\beta\gamma\delta}$ Σ $safe(I_2) \subseteq safe(\Sigma)$ I_2 $\neg$*safe for* Σ	I_3 $\neg ioco_{\beta\gamma\delta}$ Σ $safe(I_3) \supseteq safe(\Sigma)$ I_3 *safe for* Σ
Рис.35.		

В то же время вложенность безопасных трасса вместе с выполнением гипотезы о безопасности влечёт $ioco_{\beta\gamma\delta}$-конформность. Вложенность множеств всех трасс также влечёт $ioco_{\beta\gamma\delta}$-конформность.

Утверждение 22: Пусть $C \subseteq Z$, $\Sigma, I \in MODEL_{\beta\gamma\delta}(C)$.
1) $safe(I) \subseteq safe(\Sigma)$ & I *safe for* $\Sigma \Rightarrow I$ $ioco_{\beta\gamma\delta}$ Σ.
2) $I \subseteq \Sigma \Rightarrow I$ $ioco_{\beta\gamma\delta}$ Σ.

□381

В то же время из $ioco_{\beta\gamma\delta}$-конформности не следует вложенность трасс. Пример на Рис.36.

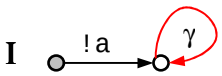
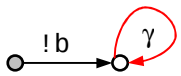
	алфавит {!y,!z} $I \not\subseteq \Sigma$ I $ioco_{\beta\gamma\delta}$ Σ
Рис.36.	

Утверждение 23: Совпадение множеств конечных финальных трасс двух βγδ-моделей влечёт $\boldsymbol{ioco}_{\beta\gamma\delta}$-эквивалентность этих моделей:

$\forall C \subseteq Z \quad \forall \mathbf{I}, \boldsymbol{\Sigma} \in \boldsymbol{MODEL}_{\beta\gamma\delta}(C) \quad \boldsymbol{final}(\mathbf{I}) = \boldsymbol{final}(\boldsymbol{\Sigma}) \Rightarrow \mathbf{I} \sim_{\boldsymbol{ioco}_{\beta\gamma\delta}} \boldsymbol{\Sigma}.$

□381

В то же время из $\boldsymbol{ioco}_{\beta\gamma\delta}$-эквивалентности не следует равенство финальных трасс. Пример на Рис.37.

$\boldsymbol{ioco}_{\beta\gamma\delta}$-эквивалентность не влечёт равенство финальных трасс		
		$\mathbf{I} \sim_{\boldsymbol{ioco}_{\beta\gamma\delta}} \boldsymbol{\Sigma}$ $\boldsymbol{final}(\mathbf{I}) \neq \boldsymbol{final}(\boldsymbol{\Sigma})$
Рис.37.		

Утверждение 24: Пусть $C \subseteq Z$.

1) Реализация без разрушения *S*-тестируема для любой спецификации:
   $\forall \boldsymbol{\Sigma} \in \boldsymbol{MODEL}_{\beta\gamma\delta}(C) \quad \boldsymbol{MODEL}_{\beta\delta}(C) \subseteq \boldsymbol{safeIMPL}(\boldsymbol{\Sigma})$.
2) Для спецификации без разрушения отношение $\boldsymbol{ioco}_{\beta\gamma\delta}$ эквивалентно вложенности βδ-трасс:
   $\forall \boldsymbol{\Sigma} \in \boldsymbol{MODEL}_{\beta\delta}(C) \quad \forall \mathbf{I} \in \boldsymbol{MODEL}_{\beta\gamma\delta}(C) \quad \mathbf{I} \; \boldsymbol{ioco}_{\beta\gamma\delta} \; \boldsymbol{\Sigma} \Leftrightarrow \mathbf{I} \subseteq \boldsymbol{\Sigma}$.[31]

□381

2.2.8. Тест и полный набор тестов

Структура раздела:

- Безопасность тестирования.
- Вердикт.
- Практические ограничения на тесты.
- Определение теста и тестового набора.
- Значимые, исчерпывающие и полные тестовые наборы.
- Редукция.
- Строго-значимые тесты.
- Полные строго-значимые наборы тестов: существование и построение.
- Тесты-«ёршики».

В βγδ-модели синхронное взаимодействие тестера и реализации означает прохождение ими одной и той же трассы. Тест, как «план» тестера, – это просто набор βγδ-трасс, которые мы хотим проверить этим тестом. Тест всегда является деревом, поскольку прохождение любой трассы означает

[31] Из этого утверждения следует, что реализация конформная спецификации без разрушения не имеет разрушения: $\mathbf{I} \in \boldsymbol{MODEL}_{\beta\delta}(C)$.

также прохождение каждого её префикса. Кроме того, на тест налагаются дополнительные ограничения.

Безопасность тестирования.

Прежде всего, тестирование должно быть безопасным, то есть, не разрушать реализацию. Поскольку мы не знаем, какую реализацию мы тестируем, мы должны предполагать, что это может быть любая реализация с единственным ограничением: она должна удовлетворять гипотезе о безопасности для данной спецификации. Это означает, что тест должен содержать только тестовые трассы спецификации, поскольку прохождение только таких трасс гарантирует неразрушение любой реализации, *S*-тестируемой для данной спецификации.

Вердикт.

Тестер в конце своей работы должен вынести вердикт ***fail***, если обнаружено несоответствие реализации и спецификации, или ***pass*** в противном случае. Заметим, что вердикт ***pass***, вообще говоря, не означает, что реализация конформна спецификации, а только то, что данный тест такого несоответствия не обнаруживает. В βγδ-модели окончанию работы тестера соответствует максимальная трасса в дереве тестовых трасс, и вердикт определяется на множестве максимальных трасс. Таким образом, тестер останавливается тогда и только тогда, когда он прошёл максимальную трассу.

Определение 25: Пусть для $C \subseteq Z$ и спецификации $\Sigma \in \boldsymbol{MODEL}_{\beta\gamma\delta}(C)$ задано дерево тестовых трасс $T \subseteq \boldsymbol{tt}(\Sigma)$ & $T \in \boldsymbol{Trees}(C_{\beta\delta})$.

- *Вердиктом* называется отображение $\boldsymbol{verdict} : \boldsymbol{max}(T) \to \{\boldsymbol{pass}, \boldsymbol{fail}\}$.
- ***pass***-/***fail***-трассами будем называть максимальные трассы с вердиктом ***pass***/***fail***:
  $\boldsymbol{pass}(T) =_{def} \boldsymbol{verdict}^{-1}(\boldsymbol{pass})$, $\boldsymbol{fail}(T) =_{def} \boldsymbol{verdict}^{-1}(\boldsymbol{fail})$.

□

Практические ограничения на тесты.

Для практического применения на тесты налагают дополнительные ограничения.

1) Тестер, выполняя тест, должен закончить свою работу за конечное время. Это означает, что каждая трасса теста не может продолжаться сколь угодно долго и рано или поздно должна закончиться вердиктом ***pass*** или ***fail***. Иными словами, в тестовом дереве не должно быть бесконечно возрастающей последовательности трасс, то есть оно должно быть нётеровым.

Заметим, что из нётеровости тестового дерева вовсе не следует ни конечность теста, ни ограниченность длин его трасс (см. Рис.38).

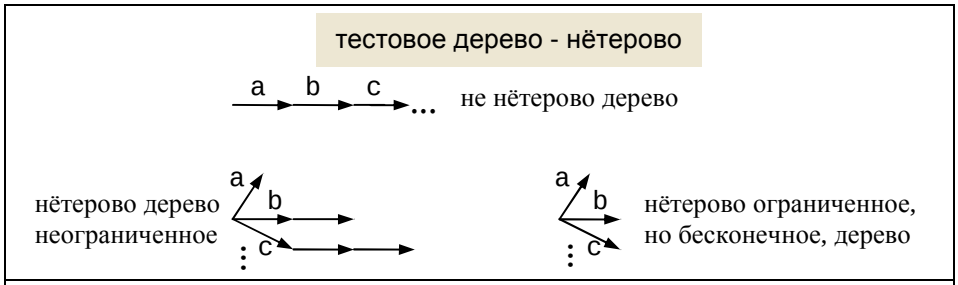


Рис.38.

Ограниченность теста означала бы не просто конечность времени выполнения: ещё до запуска теста мы заранее знали бы, что тест завершится до известного момента времени. Хотя свойство ограниченности также полезно, мы рассматриваем более общий случай нётеровых тестов. Тем не менее, как будет показано ниже, основные результаты этого раздела верны и для случая ограниченных тестов.

Можно также заметить, что ограниченность теста сама по себе также не означает его конечности: число трасс может быть бесконечным, если бесконечен алфавит.

2) Тестер не должен попадать в тупик: если трасса теста продолжается стимулом, она должна также продолжаться его блокировкой и наоборот; если трасса продолжается реакцией, она должна также продолжаться любой реакцией и символом δ и, наоборот, если трасса продолжается символом δ, то она должна также продолжаться любой реакцией.

3) Тестирование должно быть настолько управляемым, насколько это возможно, то есть, избегать излишнего недетерминизма в тестере. Это означает отсутствие в тесте смешанных продолжений трасс: трасса продолжается либо одним стимулом и его блокировкой, либо всеми реакциями и δ.

Ограничения 2 и 3 в совокупности соответствуют набору кнопок $\beta\gamma\delta$-машины тестирования: одна кнопка для каждого стимула с тайм-аутом ожидания приёма машиной стимула и одна кнопка для всех реакций с тайм-аутом ожидания выдачи машиной реакции (Рис.39).

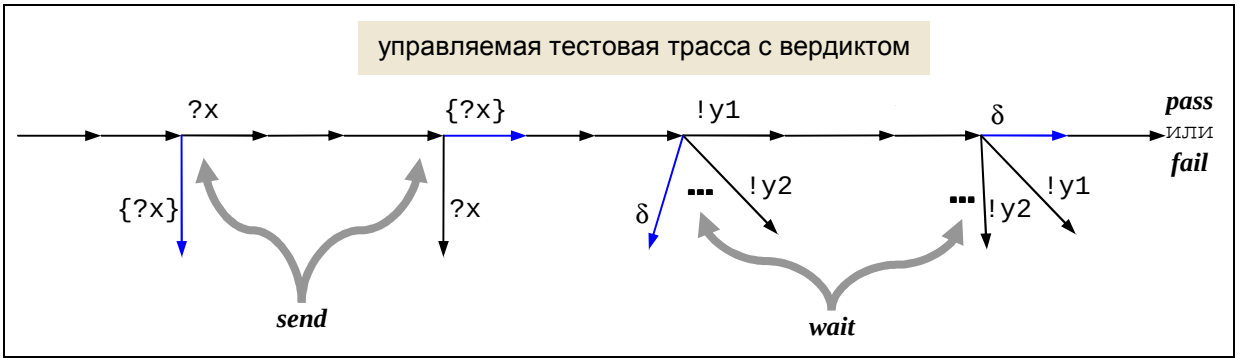


Рис.39.

Определение 26: Для $C \subseteq Z$, спецификации $\Sigma \in MODEL_{\beta\gamma\delta}(C)$, дерева тестовых трасс T и трассы $\sigma \in T$ определим:

- σ ***send in*** T $=_{def}$ $\exists ?x \in C$ ***head***$(T$ ***after*** $\sigma) = \{?x, \{?x\}\}$;
- σ ***wait in*** T $=_{def}$ ***head***$(T$ ***after*** $\sigma) = !C_\delta$;
- управляемое дерево:

  T ***controllable*** $=_{def}$ $\forall \sigma \in T$

  $\sigma \in max(T) \vee \sigma$ ***send in*** $T \vee \sigma$ ***wait in*** T.

□

Определение теста и тестового набора.

Определение 27: Для $C \subseteq Z$ и спецификации $\Sigma \in MODEL_{\beta\gamma\delta}(C)$

- *Тест или тестовый пример (test case)* – это управляемое (controllable) нётерово дерево тестовых трасс с заданным вердиктом.
- Множество всех тестовых примеров обозначим $TEST(\Sigma)$.
- *Тестовый набор (test suite)* TT – это множество тестовых примеров: $TT \subseteq TEST(\Sigma)$.

□

Тест как управляемое дерево тестовых трасс представляет собой «ворсистое» дерево выделенных безопасных ***pass***-трасс, от которых ответвляются «ворсинки» – отдельные символы, заканчивающие максимальные трассы теста (как ***pass***-, так и ***fail***-трассы).

Каждый тест T содержит только тестовые трассы спецификации, то есть только трассы, производные (с помощью ***t***-оператора) от безопасных трасс спецификации. С другой стороны, отсутствие тупиков в тесте означает, что вместе с каждой безопасной трассой спецификации он содержит *все* производные от неё тестовые трассы. Таким образом, трассы теста – это все трассы, производные от тех трасс теста, которые безопасны в спецификации: $T = \cup_{\circ} t(T \cap safe(\Sigma))$. При этом управляемость означает, что дерево трасс теста, безопасных в спецификации, ветвится в каждой точке либо по стимулу и его блокировке, либо по всем реакциям и стационарности.

Тест можно понимать как инструкцию оператору при работе с βγδ-машиной тестирования: после ***send***-трассы теста нужно нажимать кнопку передачи стимула, после ***wait***-трассы – кнопку приёма всех реакций, а после максимальной трассы выносится вердикт и тестирование заканчивается.

Значимые, исчерпывающие и полные тестовые наборы.

Реализация *проходит* тест, если при любом прогоне этого теста (при любых погодных условиях) выносится вердикт ***pass***. Поскольку в βγδ-модели синхронное взаимодействие тестера и реализации означает прохождение ими одной и той же трассы, реализация *проходит* тест, если все ***fail***-трассы теста отсутствуют в реализации. Набор тестов *значимый* (sound), если реализация, которая не проходит некоторый тест, не соответствует спецификации. Набор тестов *исчерпывающий* (exhaustive), если, наоборот, реализация, не соответствующая спецификации, не проходит некоторый тест из набора. Значимый и исчерпывающий набор называется *полным* (complete).

Определение 28: Для $C \subseteq Z$, $\mathbf{\Sigma} \in \boldsymbol{MODEL}_{\beta\gamma\delta}(C)$, $\mathbf{I} \in \boldsymbol{safeIMPL}(\mathbf{\Sigma})$, $\mathbf{T} \in \boldsymbol{TEST}(\mathbf{\Sigma})$, $\mathbf{TT} \subseteq \boldsymbol{TEST}(\mathbf{\Sigma})$:

- $\mathbf{I}\ \boldsymbol{passes}\ \mathbf{T} =_{def} \mathbf{I} \cap \boldsymbol{fail}(\mathbf{T}) = \varnothing$;
- $\mathbf{I}\ \boldsymbol{passes}\ \mathbf{TT} =_{def} \forall \mathbf{T} \in \mathbf{TT}\ \ \mathbf{I}\ \boldsymbol{passes}\ \mathbf{T}$;
- $\mathbf{TT}\ \boldsymbol{sound}\ \mathbf{\Sigma} =_{def} \forall \mathbf{I} \in \boldsymbol{safeIMPL}(\mathbf{\Sigma})$
  $(\ \mathbf{I}\ \boldsymbol{ioco}_{\beta\gamma\delta}\ \mathbf{\Sigma} \Rightarrow \mathbf{I}\ \boldsymbol{passes}\ \mathbf{TT}\ )$;
- $\mathbf{TT}\ \boldsymbol{exhaustive}\ \mathbf{\Sigma} =_{def} \forall \mathbf{I} \in \boldsymbol{safeIMPL}(\mathbf{\Sigma})$
  $(\ \mathbf{I}\ \boldsymbol{ioco}_{\beta\gamma\delta}\ \mathbf{\Sigma} \Leftarrow \mathbf{I}\ \boldsymbol{passes}\ \mathbf{TT}\ )$;
- $\mathbf{TT}\ \boldsymbol{complete}\ \mathbf{\Sigma} =_{def} \forall \mathbf{I} \in \boldsymbol{safeIMPL}(\mathbf{\Sigma})$
  $(\ \mathbf{I}\ \boldsymbol{ioco}_{\beta\gamma\delta}\ \mathbf{\Sigma} \Leftrightarrow \mathbf{I}\ \boldsymbol{passes}\ \mathbf{TT}\ )$.

□

Редукция.

Поскольку повторные отказы (одинаковые отказы в одной последовательности отказов) ничего не дают при тестировании (βγδ-замкнутость βγδ-модели), их можно удалить, и рассматривать только *редуцированные* трассы – трассы без повторных отказов.

Можно также потребовать от тестера, чтобы он заканчивал работу сразу, как только исчезла неопределённость в возможном вердикте. Это означает, что каждая немаксимальная трасса теста μ продолжается в нём как до ***pass***-трассы $\mu \cdot \lambda_p$, так и до ***fail***-трассы $\mu \cdot \lambda_f$.

Редуцированым тестом будем называть тест, в котором

1) все трассы редуцированы и

2) каждая немаксимальная трасса теста продолжается в нём как до ***pass***-трассы, так и до ***fail***-трассы.

Из любого теста можно получить редуцированный тест с помощью следующих операций редукции:

1) из каждой тестовой трассы μ удаляются повторные отказы, и
2) если все продолжения $\lambda_1, \lambda_2, \ldots$ трассы μ до максимальных трасс $\mu\cdot\lambda_1, \mu\cdot\lambda_2, \ldots$, имеют одинаковый вердикт, то эти продолжения $\lambda_1, \lambda_2, \ldots$ удаляются, трасса μ становится максимальной с тем вердиктом, который имели продолжающие её максимальные трассы.

Таким образом, можно ограничиться рассмотрением только редуцированных тестов. Однако, поскольку это является очевидной оптимизацией, мы в дальнейшем не будем на этом акцентировать внимание.

Строго-значимые тесты.

Определение 29: Для $C \subseteq Z$, $\Sigma \in \boldsymbol{MODEL}_{\beta\gamma\delta}(C)$ тест $T \in \boldsymbol{TEST}(\Sigma)$ будем называть *строго-значимым*, если выполняется следующее условие: $\forall \sigma \in T \;\; \sigma \in \boldsymbol{fail}(T) \;\Leftrightarrow\; \sigma \notin \Sigma$.
Тестовый набор будем называть строго-значимым, если все его тесты строго-значимые.

□

Очевидно, что для любого значимого теста верна импликация в одну сторону: $\forall \sigma \in T \;\; \sigma \in \boldsymbol{fail}(T) \;\Rightarrow\; \sigma \notin \Sigma$. Для любого значимого теста обратную импликацию легко сделать верной, если для каждой его ***pass***-трассы, отсутствующей в спецификации, заменить вердикт ***pass*** на вердикт ***fail***: $\forall \sigma \in \boldsymbol{pass}(T) \setminus \Sigma \;\; \boldsymbol{verdict}(\sigma) := \boldsymbol{fail}$. Тем самым, любой значимый тест тривиально превращается в строго-значимый тест, который обнаруживает не меньшее число ошибок. Очевидно, для тестирования можно ограничиться строго-значимыми тестами.

Заметим также, что, поскольку тест $T \subseteq \boldsymbol{tt}(\Sigma)$ – подмножество тестовых трасс спецификации, условие $\sigma \in \Sigma$ эквивалентно $\sigma \in \boldsymbol{safe}(\Sigma)$. Поэтому строго-значимость теста – это жёсткое правило назначения вердикта максимальным трассам теста: вердикт ***pass*** получают безопасные трассы спецификации, а вердикт ***fail*** – трассы, отсутствующие в спецификации.

В редуцированном строго-значимом тесте за отказом o, по первой операции редукции, не может следовать этот же отказ o. А тогда за отказом o не может следовать символ $z \in o$, так как, иначе, можно было бы применить

вторую операцию редукции: несогласованные трассы отсутствуют в спецификации и, следовательно, если максимальны, то (в строго-значимом тесте) имеют вердикт ***fail***. Тем самым, редуцированные строго-значимые тесты согласованы и не имеют повторных отказов.

Полные строго-значимые наборы тестов: существование и построение.

Утверждение 25: Для алфавита $C \subseteq Z$ и спецификации $\mathbf{\Sigma} \in \boldsymbol{MODEL}_{\beta\gamma\delta}(C)$ строго-значимый тестовый набор $\mathbf{TT} \subseteq \boldsymbol{TEST}(\mathbf{\Sigma})$, содержащий все безопасные трассы спецификации $\boldsymbol{safe}(\mathbf{\Sigma}) \subseteq \cup(\mathbf{TT})$, является полным.

□382

Из $\mathbf{T} = \cup_{\circ}\boldsymbol{t}(\mathbf{T} \cap \boldsymbol{safe}(\mathbf{\Sigma}))$ вытекает $\cup(\mathbf{TT}) = \cup_{\circ}\boldsymbol{t}((\cup(\mathbf{TT})) \cap \boldsymbol{safe}(\mathbf{\Sigma}))$. Поэтому из условия $\boldsymbol{safe}(\mathbf{\Sigma}) \subseteq \cup(\mathbf{TT})$ следует $\cup(\mathbf{TT}) \supseteq \cup_{\circ}\boldsymbol{t}_{\circ}\boldsymbol{safe}(\mathbf{\Sigma})$. Учитывая, что $\boldsymbol{tt}(\mathbf{\Sigma}) = \cup_{\circ}\boldsymbol{t}_{\circ}\boldsymbol{safe}(\mathbf{\Sigma})$, имеем: $\cup(\mathbf{TT}) \supseteq \boldsymbol{tt}(\mathbf{\Sigma})$. С другой стороны, все трассы тестов тестового набора – тестовые трассы спецификации, то есть $\cup(\mathbf{TT}) \subseteq \boldsymbol{tt}(\mathbf{\Sigma})$. Таким образом, условие $\boldsymbol{safe}(\mathbf{\Sigma}) \subseteq \cup(\mathbf{TT})$ влечёт $\cup(\mathbf{TT}) = \boldsymbol{tt}(\mathbf{\Sigma})$, то есть трассы тестов тестового набора – это все тестовые трассы данной спецификации.

Теперь покажем, что для каждой спецификации существуют строго-значимые полные наборы тестов.

Для строго-значимого теста поддерево ***pass***-трасс и их префиксов – это нётерово поддерево безопасных трасс, которое ветвится «полууправляемым» образом: либо по стимулу и/или его блокировке, либо по некоторым реакциям и/или стационарности (Рис.40). Наоборот, по каждому такому полууправляемому дереву можно построить строго-значимый тест, добавив нужные ответвления-«ворсинки», чтобы дерево стало полностью управляемым.

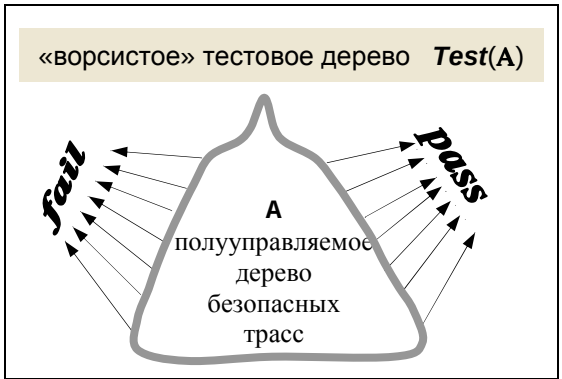


Рис.40.

Заметим, что одному строго-значимому тесту соответствует, вообще говоря, несколько полууправляемых деревьев безопасных трасс. Если семейство полууправляемых деревьев покрывает все безопасные трассы, то соответствующий тестовый набор будет полным.

Определение 30: Для $C \subseteq Z$, $\mathbf{A} \in \boldsymbol{Trees}(C_{\beta\gamma\delta})$ и трассы $\sigma \in \mathbf{A}$

· σ ***semi-send in*** $\mathbf{A}$ $=_{\text{def}}$ $\exists ?x \in C$

$\varnothing \neq \boldsymbol{head}(\mathbf{A}\ \boldsymbol{after}\ \sigma) \subseteq \{?x, \{?x\}\}$;

· σ ***semi-wait in*** $\mathbf{A}$ $=_{def}$ $\varnothing \neq \mathit{head}(\mathbf{A}\ \mathit{after}\ \sigma) \subseteq !C_\delta$;
· $\mathbf{A}$ ***semi-controllable*** $=_{def}$ $\forall \sigma \in \mathbf{A}$
$\sigma \in \mathit{max}(\mathbf{A}) \vee \sigma$ ***semi-send in*** $\mathbf{A} \vee \sigma$ ***semi-wait in*** $\mathbf{A}$.

□

Определение 31: Пусть $C \subseteq Z$, $\mathbf{\Sigma} \in \mathit{MODEL}_{\beta\gamma\delta}(C)$, а $\mathbf{A} \subseteq \mathit{safe}(\mathbf{\Sigma})$ нётерово полууправляемое (***semi-controllable***) дерево. Обозначим

$$\mathit{Test}(\mathbf{A}) =_{def} \mathbf{A} \cup \{\ \mu\cdot\langle u\rangle \mid \exists u`\ \ \mu\cdot\langle u`\rangle \in \mathbf{A}$$
$$\&\ (u`=?x \ \Rightarrow\ u=\{?x\})$$
$$\&\ (u`=\{?x\} \Rightarrow\ u=?x)$$
$$\&\ (u` \in !C_\delta \ \Rightarrow\ u \in !C_\delta)\ \}.$$

Определим вердикт:

$\lambda \in \mathit{max} \circ \mathit{Test}(\mathbf{A}) \cap \mathbf{\Sigma}$: ***verdict***(λ) = ***pass***,
$\lambda \in \mathit{max} \circ \mathit{Test}(\mathbf{A}) \setminus \mathbf{\Sigma}$: ***verdict***(λ) = ***fail***.

Для безопасной трассы $\sigma \in \mathit{safe}(\mathbf{\Sigma})$ множество $\mathit{Tree}(\sigma)$ нётерово и полууправляемое, и тестом-«ёршиком» будем называть тест $\mathit{Test}(\sigma) =_{def} \mathit{Test} \circ \mathit{Tree}(\sigma)$.

□

Тест $\mathit{Test}(\mathbf{A})$ работает следующим образом. При получении символа u после безопасной трассы $\sigma \in \mathbf{A}$ проверяется $\sigma\cdot\langle u\rangle \in \mathbf{\Sigma}$. Если это не так, выдаётся вердикт ***fail***. В противном случае выдаётся вердикт ***pass***, если $\sigma\cdot\langle u\rangle \in \mathit{max} \circ \mathit{Test}(\mathbf{A})$, или нажимается кнопка машины тестирования, определяемая немаксимальной трассой $\sigma\cdot\langle u\rangle$.

Утверждение 26: Пусть $C \subseteq Z$ и $\mathbf{\Sigma} \in \mathit{MODEL}_{\beta\gamma\delta}(C)$.

1) Для любого нётерова полууправлямого дерева $\mathbf{A}$ безопасных трасс $\mathit{Test}(\mathbf{A})$ является строго-значимым тестом.
2) Для любого строго-значимого теста $\mathbf{T}$ множество его ***pass***-трасс и их префиксов является нётеровым полууправляемым поддеревом безопасных трасс и $\mathbf{T} = \mathit{Test} \circ \cup \circ \mathit{Tree} \circ \mathit{pass}(\mathbf{T})$.

□383

Утверждение 27: Для $C \subseteq Z$, $\mathbf{\Sigma} \in \mathit{MODEL}_{\beta\gamma\delta}(C)$ и любого семейства $\mathbf{AA}$ нётеровых полууправляемых множеств безопасных трасс, покрывающего все безопасные трассы, $\cup(\mathbf{AA}) = \mathit{safe}(\mathbf{\Sigma})$, тестовый набор $\mathbf{TT}(\mathbf{AA}) = \{\mathit{Test}(\mathbf{A}) \mid \mathbf{A} \in \mathbf{AA}\}$ является полным.

□384

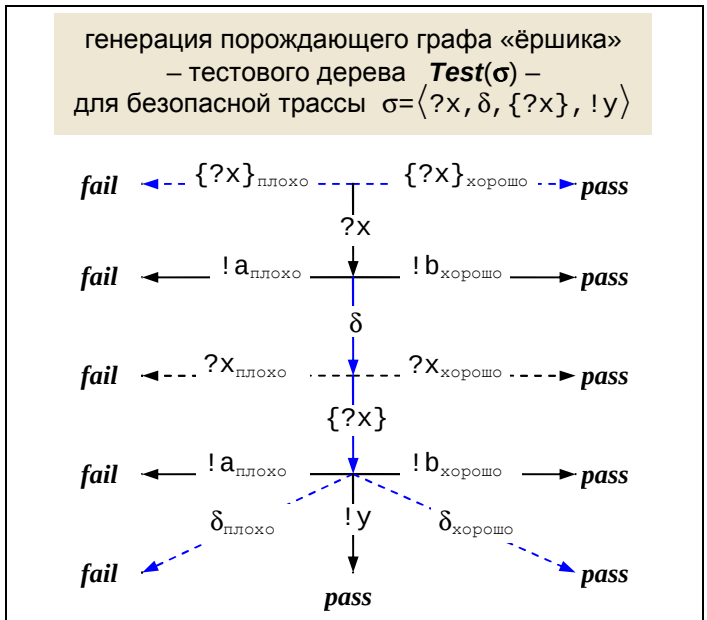
Тесты-«ёршики».

В качестве примера семейства $\mathbf{AA}$ можно взять множество всех деревьев, каждое из которых есть множество префиксов некоторой безопасной трассы:

AA=$\{$***Tree***$(\sigma)\,|\,\sigma\in$***safe***$(\Sigma)\}$. Это значит, что среди всех строго-значимых тестов – как «ворсистых» деревьев безопасных трасс – можно выделить множество тестов-«ёршиков» $\{$***Test***$(\sigma)\,|\,\sigma\in$***safe***$(\Sigma)\}$, в каждом из которых есть только одна выделенная ***pass***-трасса σ, от которой ответвляются «ворсинки»-символы (Рис.41).

Заметим, что полный тестовый набор «ёршиков» состоит только из ограниченных тестовых деревьев: все трассы «ёршика» ***Test***(σ) имеют длину не более длины выделенной ***pass***-трассы σ. Поэтому для каждого теста заранее известно максимальное время его выполнения.

Отметим также, что требование наличия среди трасс тестов полного тестового набора всех безопасных трасс спецификации является достаточным, но не необходимым условием. Мы можем редуцировать все тесты без изменения полноты тестового набора. После этого останутся только те безопасные трассы, в которых нет повторных отказов (первая операция редукции). Кроме того, если безопасная трасса безопасно продолжается в спецификации всеми возможными способами, то продолжающие её трассы также будут удалены (вторая операция редукции).



Символ снабжён индексом «$_{\text{хорошо}}$», если он есть в спецификации после предшествующей трассы; иначе – индексом «$_{\text{плохо}}$».

Пунктиром показаны взаимоисключающие продолжения, ведущие из одной вершины порождающего графа

Рис.41.

2.2.9. Алгоритмизация

Структура раздела:
- Конечное время выполнения теста и перечислимость тестовых наборов.
- Модель безопасных трасс (способ `T0`).
- Два способа задания трассовой модели (способы `T1` и `T2`).
- Модель финальных трасс (способ `T1`).
- Модель трансфинальных трасс (способ `T2`).
- Сравнение способов задания трассовой модели.

Конечное время выполнения теста и перечислимость тестовых наборов.

Для практического применения ***ioco***-теории требуется конечность времени выполнения теста и конечность тестового набора. К сожалению, для большинства спецификаций любой полный тестовый набор бесконечен, и приходится рассматривать его конечные подмножества, которые только значимы, но не полны.

Построение конечного тестового набора по заранее заданному критерию (набора, обладающего заранее заданным свойством), можно выполнять на основе перечисления некоторого полного тестового набора: выбирая тесты один за другим, проверяем наш критерий на всех подмножествах уже выбранных тестов до тех пор, пока не получим искомый тестовый набор. Разумеется, предполагается, что выбранный полный тестовый набор перечислим и в нём существует конечное подмножество, удовлетворяющее критерию отбора.

Мы ограничимся строго-значимыми тестами.

Будем считать, что задана спецификация `Σ` в алфавите `C`.

Мы будем предполагать, что все символы – стимулы, реакции, блокировки и стационарность, – задаются конечными представлениями (последовательностями конечной длины в конечном алфавите)[32]. Также будем считать, что мы можем алгоритмически определять, является ли символ стимулом (например, префикс “?”), реакцией (например, префикс “!”), блокировкой (например, префикс “{?” и постфикс “}”) или стационарностью (например, “δ”). Наконец, будем предполагать, что по стимулу `?x` мы можем вычислить его блокировку `{?x}`, и наоборот.

[32] Отсюда следует, что множество всех символов счётно.

Поскольку тест – нётерово дерево трасс, он выполняется за конечное время, если за конечное время выполняется каждый шаг теста: нажатие кнопки оператором и проверка правильности полученного наблюдения: стимула, блокировки стимула, реакции или стационарности. Для этого требуется разрешимость (и наличие алгоритма разрешения) множества правильных наблюдений, то есть наблюдений, допускаемых спецификацией Σ после уже полученной трассы σ, относительно множества наблюдений, возможных при безопасном тестировании после этой трассы σ. Запишем это условие формально.

Определение 32: Для $C \subseteq Z$ и $\Sigma \in MODEL_{\beta\gamma\delta}(C)$ **условием конечности времени выполнения $\beta\gamma\delta$-теста** назовём разрешимость множества $head(\Sigma\ after\ \sigma)$ продолжающих символов относительно множества $safesymbols(\Sigma\ after\ \sigma)$ безопасных символов для каждой безопасной трассы σ.

□

Как следствие, если множество $safesymbols(\Sigma\ after\ \sigma)$ перечислимо для каждой безопасной трассы σ, то перечислимо множество $safehead(\Sigma\ after\ \sigma) = head(\Sigma\ after\ \sigma) \cap safesymbols(\Sigma\ after\ \sigma)$.

Перечислимость множества $safehead(\Sigma\ after\ \sigma)$ гарантирует перечислимость множества всех безопасных трасс спецификации, а перечислимость множества $safesymbols(\Sigma\ after\ \sigma)$ вместе с разрешимостью множества $safehead(\Sigma\ after\ \sigma)$ относительно множества $safesymbols(\Sigma\ after\ \sigma)$ – перечислимость всех тестовых трасс.

В общем случае отсюда не следует перечислимость множества всех строго-значимых тестовых деревьев, которое может оказаться несчётным. Однако мы можем ограничиться тестами, построенными на основе конечного числа выделенных безопасных трасс. Множество таких тестов перечислимо.

Например, для каждой безопасной трассы σ можно построить «ёршик» $Test(\sigma)$, который содержит σ как выделенную ***pass***-трассу, а все остальные максимальные трассы теста являются ответвлениями от σ ровно на один символ. Множество строго-значимых «ёршиков» для всех безопасных трасс спецификации является полным тестовым набором (по Утверждение 27:). Любое «ворсистое» дерево с конечным числом выделенных трасс представляет собой объединение конечного числа «ёршиков». Алгоритм перечисления строго-значимых «ёршиков» тривиально строится из алгоритма перечисления безопасных трасс спецификации. Получая на каждом шаге конечное множество «ёршиков», мы можем построить конечное множество их объединений, отбраковывая те их них, которые не

являются тестами. Тем самым, перечисляя строго-значимые «ёршики», мы перечисляем и все строго-значимые тесты, из которых можем набирать конечное множество таких тестов, удовлетворяющее нужному критерию.

Утверждение 28: Пусть $C \subseteq Z$, $\Sigma \in MODEL_{\beta\gamma\delta}(C)$, множество безопасных трасс перечислимо и задан алгоритм их перечисления. Тогда множество всех конечных полууправляемых деревьев безопасных трасс (полууправляемых деревьев, в которых конечно число трасс и все эти трассы безопасны) перечислимо и можно построить алгоритм перечисления этого множества.
□384

Модель безопасных трасс (способ T0).

Сначала предположим, что мы умеем определять безопасность спецификации (отсутствие в ней трассы $\langle\gamma\rangle$) и для каждой безопасной трассы перечислять все продолжающие трассу безопасные символы (стимулы, реакции, блокировки и стационарность). Кроме того, для каждого безопасного символа мы можем проверять, продолжается трасса этим символом или нет.

Таким способом мы задаём не все трассы спецификации, а только её безопасные трассы, то есть модель безопасных трасс. Иными словами, этим способом задаётся класс *S*-изоморфных моделей финальных трасс или класс *S*-изоморфных βγδ-моделей. Можно также считать, что этим способом однозначно задаётся γ-однородная модель финальных трасс или *S*-каноническая βγδ-модель (Рис.42). По Утверждение 21:, любые две *S*-изоморфные βγδ-модели $ioco_{\beta\gamma\delta}$-эквивалентны.

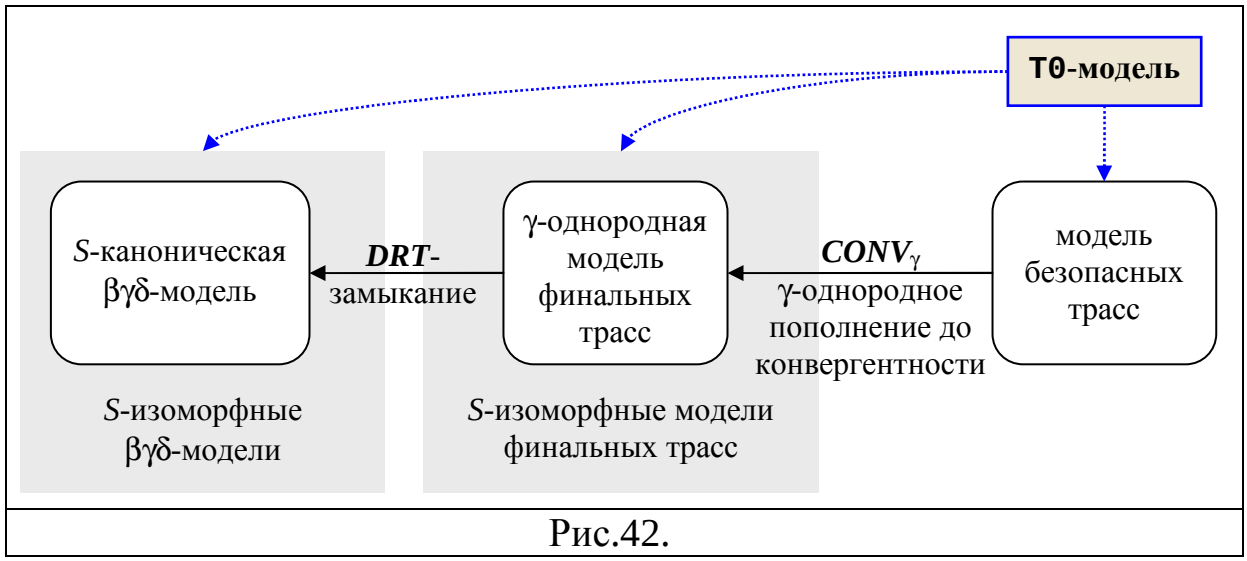


Рис.42.

Определение 33: Пусть $C \subseteq Z$. Будем говорить, что трассовая модель $\Sigma \in MODEL_{\beta\gamma\delta}(C)$ алгоритмически задана способом **T0** (**T0**-модель), если

1) задан оракул безопасности $\mathbf{Safe}_{\Sigma}()$, который возвращает ***true***, если модель безопасна: $\mathbf{Safe}_{\Sigma}() = \Sigma$ ***safe***;

и для каждой безопасной трассы $\sigma \in \mathit{safe}(\Sigma)$

2) *множество безопасных продолжающих символов перечислимо*[33]
и задан итератор (алгоритм перечисления) $\mathbf{Safehead}_{\Sigma}(\sigma)$, перечисляющий множество $\mathit{safehead}(\Sigma \ \mathit{after} \ \sigma)$;

3) выполнено условие конечности времени выполнения теста:
множество продолжающих символов разрешимо относительно множества безопасных символов для безопасной трассы,
и задан оракул (алгоритм разрешения) $\mathbf{SafeExtend}_{\Sigma}(\sigma, u)$, который для каждого безопасного символа $u \in \mathit{safesymbols}(\Sigma \ \mathit{after} \ \sigma)$ возвращает ***true***, если трасса продолжается этим символом $u \in \mathit{head}(\Sigma \ \mathit{after} \ \sigma)$.

Множество **T0**-моделей в алфавите C будем обозначать $\mathbf{T0}(C)$, а множество всех **T0**-моделей будем обозначать $\mathbf{T0} = \cup\{\mathbf{T0}(C) \mid C \subseteq Z\}$.

□

Утверждение 29: Для **T0**-спецификации множество всех строго-значимых тестов с конечным числом немаксимальных трасс перечислимо, и можно построить алгоритм его перечисления.

□384

Два способа задания трассовой модели (способы T1 и T2).

Теперь мы рассмотрим способы задания спецификации, когда у нас нет итератора безопасных символов, продолжающих безопасную трассу. Вместо этого мы можем перечислять все символы и для каждого из них проверять, являются ли он безопасным после трассы и продолжает ли он трассу.

Заметим, что блокировка стимула $\{?x\}$ безопасна тогда и только тогда, когда безопасен сам стимул $?x$, а стационарность δ безопасна тогда и только тогда, когда безопасна каждая реакция (реакции либо все безопасны, либо все опасны). Поэтому нам достаточно перечислять только стимулы и реакции и проверять безопасность каждого стимула и реакции.

Далее, мы будем предполагать, что у нас нет явного оракула безопасности стимула или реакции, но есть возможность проверить, являются ли стимул или реакция разрушающими после трассы или нет. Отсутствие разрушения подразумевает отсутствие дивергенции. Заметим, что стимул безопасен тогда и только тогда, когда он неразрушающий, а реакция безопасна тогда и

---

[33] Это эквивалентно тому, что дерево безопасных трасс перечислимо-ветвящееся.

только тогда, когда все реакции неразрушающие. Поэтому для проверки безопасности стимула достаточно проверить, что стимул неразрушающий. Однако для проверки безопасности реакции нужно проверить, что все реакции неразрушающие. Последнее возможно за конечное время только в том случае, если число реакций конечно.[34]

Мы рассмотрим два способа задания спецификации в зависимости от того, задаётся оракул разрушаемости явно или нет:

1) Задан *оракул разрушаемости* стимулов и реакций после безопасной трассы и *оракул продолжения* безопасной трассы *безопасными символами*.
2) Задан *оракул продолжения* безопасной трассы *любыми стимулами и реакциями*, а также *безопасными отказами*, и *оракул разрушения*, проверяющий наличие разрушения после безопасной трассы, продолженной одним стимулом или реакцией.

При задании спецификации в виде пред- и пост-условий эти два способа по-разному распределяют проверки между предикатами пред- и постусловия:

1) Предусловие запрещает разрушающие стимулы и реакции, а постусловие проверяет продолжаемость трассы неразрушающими стимулами и реакциями.
2) Предусловие запрещает стимулы и реакции, не продолжающие трассу, а постусловие проверяет разрушаемость продолжающих трассу стимулов и реакций.

Замечание о разрушающих и опасных стимулах и реакциях. Можно было бы рассматривать модифицированный способ 1a), когда вместо оракула разрушаемости используется оракул опасности стимулов и реакций после безопасной трассы. Соответственно, предусловие запрещает опасные стимулы и реакции, а постусловие проверяет продолжаемость трассы безопасными стимулами и реакциями. Для стимулов здесь нет никакой разницы: стимул безопасен тогда и только тогда, когда он неразрушающий. Однако для реакций есть разница: реакция безопасна, если *все* реакции неразрушающие. Способ 1a) задаёт трассовую модель с меньшей точностью, однако, достаточной для генерации тестов для соответствия $\boldsymbol{ioco}_{\beta\gamma\delta}$. В то же время способ 1) предпочтительнее для задания спецификации в пред- и

[34] Вместо конечности алфавита реакций можно было бы потребовать конечности множества реакций, продолжающих каждую безопасную трассу. Соответственно меняется алгоритмическое задание модели: вместо итератора реакций и оракула для определения продолжения трассы данной реакцией нужен итератор продолжающих трассу реакций (для первого способа ещё нужно определять разрушающая реакция или нет).

постусловиях безотносительно к выбору того или иного соответствия, поскольку в этом случае мы не закладываемся на те тестовые возможности, которые разрешают наблюдение лишь одного вида отказов, связанных с реакциями, а именно стационарности как отсутствия всех реакций. Если у нас есть возможность наблюдать, например, отсутствие указанной реакции (отдельная кнопка машины тестирования для приёма одной указанной реакции), то способ 1а) не годится: может оказаться, что указанная реакция неразрушающая, хотя другая реакция разрушающая.

В обоих способах задаются все безопасные трассы модели. Поскольку для тестирования достаточно безопасных трасс, оба способа задания спецификации обеспечивают возможность генерации полного тестового набора.

Модель финальных трасс (способ T1).

Первый способ задаёт модель финальных трасс. Иными словами, этот способ задаёт класс *F*-изоморфных βγδ-моделей. Можно также считать, что этим способом однозначно задаётся *F*-каноническая βγδ-модель (Рис.43).

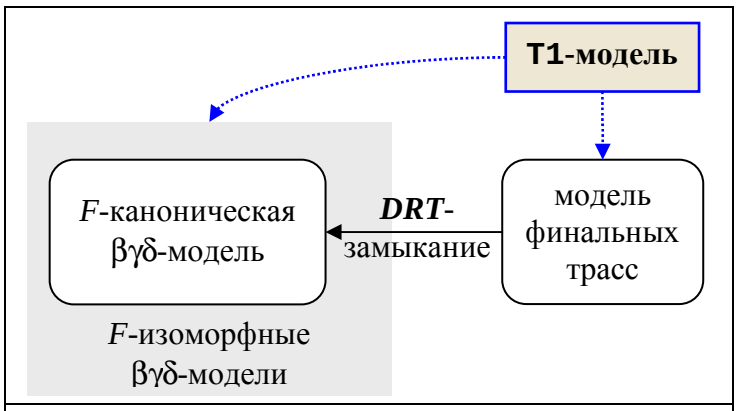


Рис.43.

Определение 34: Пусть $C \subseteq Z$. Будем говорить, что трассовая модель $\Sigma \in \boldsymbol{MODEL}_{\beta\gamma\delta}(C)$ алгоритмически задана способом **T1** (**T1**-модель), если

1) задан оракул безопасности $\mathbf{Safe}_{\Sigma}()$, который возвращает ***true***, если модель безопасна: $\mathbf{Safe}_{\Sigma}() = \Sigma$ ***safe***;
2) *множество стимулов перечислимо*, и задан итератор (алгоритм перечисления) $\mathbf{X}_C$, перечисляющий множество $?C$;
3) *множество реакций конечно*, и задан итератор (алгоритм перечисления) $\mathbf{Y}_C$, перечисляющий множество $!C$;

далее для каждой безопасной трассы $\sigma \in \boldsymbol{safe}(\Sigma)$:

4) *множество разрушающих стимулов и реакций разрешимо относительно множества всех стимулов и реакций*, и задан оракул (алгоритм разрешения) $\mathbf{Gamma1}_{\Sigma}(\sigma,z)$, который для каждого символа $z\in C$ возвращает ***true***, если $\langle z,\gamma\rangle\in(\Sigma\ \textit{after}\ \sigma)$;

5) выполнено условие конечности времени выполнения теста: *множество продолжающих символов разрешимо относительно множества безопасных символов*, и задан оракул (алгоритм разрешения) $\mathbf{Extend1}_{\Sigma}(\sigma,u)$, который для каждого безопасного символа $u\in\textit{safesymbols}(\Sigma\ \textit{after}\ \sigma)$ возвращает ***true***, если трасса продолжается этим символом $u\in\textit{head}(\Sigma\ \textit{after}\ \sigma)$.[35]

Множество **T1**-моделей в алфавите C будем обозначать $\mathbf{T1}(C)$, а множество всех **T1**-моделей будем обозначать $\mathbf{T1}=\cup\{\mathbf{T1}(C)\,|\,C\subseteq Z\}$.

□

Утверждение 30: Существует алгоритм преобразования $\mathbf{T1}2\mathbf{T0}:\mathbf{T1}\to\mathbf{T0}$, который по каждой **T1**-спецификации (модели финальных трасс) строит модель её безопасных трасс как **T0**-спецификацию. Как следствие, для **T1**-спецификации множество всех строго-значимых тестов с конечным числом немаксимальных трасс перечислимо, и можно построить алгоритм его перечисления.

□386

Модель трансфинальных трасс (способ T2).

При втором способе мы получаем все финальные трассы плюс некоторые нефинальные трассы вида $\sigma\cdot\langle!y\rangle$, где трасса σ безопасна, а реакция $!y$ неразрушающая, но опасная (то есть разрушающая другая реакция $!t$, дающая финальную трассу $\sigma\cdot\langle!t,\gamma\rangle$). Такие нефинальные трассы будем называть трансбезопасными трассами. Заметим, что трансбезопасная трасса ничем не продолжается во множестве трасс, которые мы можем получить при задании спецификации вторым способом. Финальную или трансбезопасную трассу будем называть трансфинальной. Определим явно класс βγδ-моделей, задаваемый этим способом.

Определение 35: Пусть $C\subseteq Z$. Для дерева $\Sigma\in\textit{Trees}(C_{\beta\gamma\delta})$ *трансбезопасной* трассой будем называть трассу вида $\sigma\cdot\langle!y\rangle$, где трасса σ безопасна, а реакция $!y$ неразрушающая, но опасная после σ:

$$\textit{tsafe}(\Sigma)\ =_{\text{def}}\ \Sigma\cap(\textit{safe}(\Sigma)\cdot!C^{1})\setminus\textit{final}(\Sigma).$$

---

[35] Деревья финальных и безопасных трасс **T1**-модели разрешимо-ветвящиеся. Для перечислимого алфавита это влечёт перечислимо-ветвимость деревьев.

Трансфинальной трассой будем называть финальную или трансбезопасную трассу. *Трансфинальным* поддеревом будем называть множество $\boldsymbol{tfinal}(\Sigma)$ его трансфинальных трасс:

$\boldsymbol{tfinal}(\Sigma) =_{def} \boldsymbol{final}(\Sigma) \cup \boldsymbol{tsafe}(\Sigma)$.

Если дерево Σ βγδ-модель, то его трансфинальное поддерево будем называть подмоделью трансфинальных трасс[36].

□

Определение 36: Пусть $C \subseteq Z$. Две βγδ-модели $\Sigma, \Sigma` \in \boldsymbol{MODEL}_{\beta\gamma\delta}(C)$ будем называть *TF-изоморфными*, если они имеют одну и ту же подмодель трансфинальных трасс: $\boldsymbol{tfinal}(\Sigma)=\boldsymbol{tfinal}(\Sigma`)$.

□

Среди множества *TF*-изоморфных βγδ-моделей можно выделить одну "минимальную" βγδ-модель $\Sigma \in \boldsymbol{MODEL}_{\beta\gamma\delta}$, которую будем называть *TF*-канонической.

Утверждение 31: Пусть задан базовый алфавит $C \subseteq Z$. Пересечение всех βγδ-моделей класса *TF*-изоморфных βγδ-моделей в алфавите C совпадает с ***DRT***-замыканием их подмодели трансфинальных трасс. Пополнение этого пересечения продолжениями всех трансбезопасных трасс[37] всеми возможными трассами отказов является βγδ-моделью.

Обозначая $\boldsymbol{E}(T_2) = \{T \in \boldsymbol{MODEL}_{\beta\gamma\delta}(C) \mid \boldsymbol{tfinal}(T)=T_2\}$ и

$\boldsymbol{CR}(T) = T \cup (\boldsymbol{tsafe}(T)\cdot\beta\delta(C)^*)$,

$\forall C \subseteq Z \ \forall T_2 \in \boldsymbol{tfinal}\circ\boldsymbol{MODEL}_{\beta\gamma\delta}(C)$

$\cap\circ\boldsymbol{E}(T_2)=\cup\circ\boldsymbol{DRT}(T_2) \ \& \ \boldsymbol{CR}\circ\cup\circ\boldsymbol{DRT}(T_2) \in \boldsymbol{E}(T_2)$.

□386

Определение 37: Пополнение пересечения всех βγδ-моделей класса *TF*-изоморфных βγδ-моделей в алфавите C продолжениями всех трансбезопасных трасс всеми возможными трассами отказов будем называть *TF*-канонической βγδ-моделью.

□

Таким образом, второй способ задаёт модель трансфинальных трасс. Иными словами, задаётся класс *TF*-изоморфных βγδ-моделей. Можно также считать, что этим способом однозначно задаётся *TF*-каноническая βγδ-модель (Рис.44).

---

[36] Можно было бы определить эксплицитно модель трансфинальных трасс аналогично тому, как это сделано для модели безопасных трасс и модели финальных трасс.

[37] то есть трасс, неконвергентных в этом пересечении.

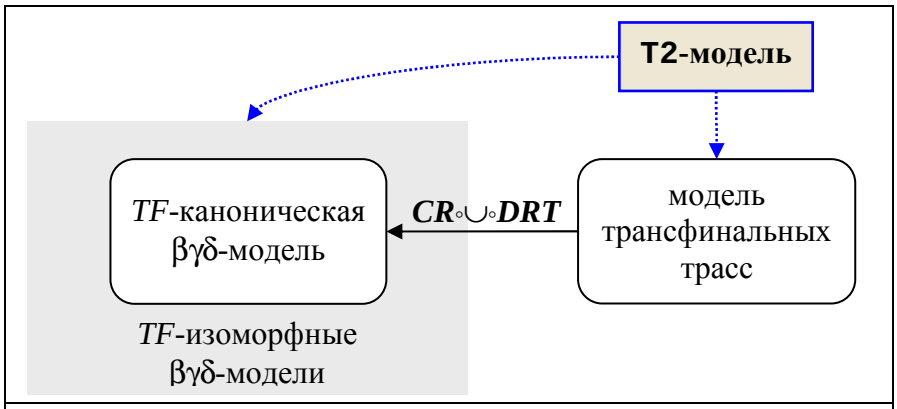


Рис.44.

Определение 38: Пусть $C \subseteq Z$. Будем говорить, что трассовая модель $\Sigma \in \boldsymbol{MODEL}_{\beta\gamma\delta}(C)$ алгоритмически задана способом **T2** (**T2**-модель), если

1) *множество стимулов перечислимо*, и задан итератор (алгоритм перечисления) $\mathbf{X}_C$, перечисляющий множество $?C$;
2) *множество реакций конечно*, и задан итератор (алгоритм перечисления) $\mathbf{Y}_C$, перечисляющий множество $!C$;

далее для каждой безопасной трассы $\sigma \in \boldsymbol{safe}(\Sigma)$:

3) *множество продолжающих символов разрешимо относительно множества всех стимулов и реакций, а также безопасных отказов*, и задан оракул (алгоритм разрешения) $\mathbf{Extend2}_\Sigma(\sigma, u)$, который для каждого символа $u \in \boldsymbol{safesymbols}(\Sigma\ \boldsymbol{after}\ \sigma) \cup C$ возвращает ***true***, если трасса продолжается этим символом $\sigma \cdot \langle u \rangle \in \Sigma$;
4) задан оракул $\mathbf{Gamma2}_\Sigma(\sigma_+)$, который для пустой трассы $\sigma_+ = \epsilon$ и каждой трассы $\sigma_+ = \sigma \cdot \langle z \rangle \in \Sigma$, где $\sigma \in \boldsymbol{safe}(\Sigma)$ и $z \in C$, возвращает ***true***, если $\langle \gamma \rangle \in (\Sigma\ \boldsymbol{after}\ \sigma_+)$.[38]

Множество **T2**-моделей в алфавите C будем обозначать $\mathbf{T2}(C)$, а множество всех **T2**-моделей будем обозначать $\mathbf{T2} = \cup\{\mathbf{T2}(C) \mid C \subseteq Z\}$.

□

Утверждение 32: Существует алгоритм преобразования $\mathbf{T2}2\mathbf{T1} : \mathbf{T2} \to \mathbf{T1}$, который по каждой **T2**-спецификации (модель трансфинальных трасс) строит модель её финальных трасс как **T1**-спецификацию. Как следствие, для **T2**-спецификации множество всех строго-значимых тестов с конечным числом немаксимальных трасс перечислимо, и можно построить алгоритм его перечисления.

---

[38] Деревья трансфинальных, финальных и безопасных трасс **T2**-модели разрешимо-ветвящиеся и, поскольку алфавит перечислим, перечислимо-ветвящиеся.

□388

Сравнение способов задания трассовой модели.

Из **T1**-модели финальных трасс можно получить **T2**-модель, задающую то же множество трасс, если при наличии разрушающей реакции после безопасной трассы, считать все неразрушающие реакции отсутствующими.

Утверждение 33: Существует алгоритм преобразования **T1**2**T2**:**T1**→**T2**, который по каждой модели финальных трасс, задаваемой **T1**-спецификацией, строит **T2**-спецификацию, задающую то же множество трасс.

□388

Разумеется, не любая **T2**-спецификация может быть получена из **T1**-спецификации, поскольку одной модели финальных трасс соответствует, вообще говоря, несколько моделей трансфинальных трасс. Заметим также, что, вообще говоря, **T1**2**T2**◦**T2**2**T1**(**Σ**)≠**Σ**, поскольку из множества трасс **Σ** будут удалены все трансбезопасные трассы. Однако, если в **Σ** нет трансбезопасных трасс, то способом **T2** будет задано только множество финальных трасс, и будет выполнено равенство **T1**2**T2**◦**T2**2**T1**(**Σ**)=**Σ**.

Ниже на Рис.45 приведена таблица сравнения различных способов задания трассовой спецификации. Стрелками показана выводимость алгоритмов, задающих спецификацию на конце стрелки, из алгоритмов, задающих спецификацию в начале стрелки.

T0	$\Sigma \in MODEL_{\beta\gamma\delta}(C)$
задан оракул безопасности $\mathbf{Safe}_\Sigma()$, который возвращает ***true***, если модель безопасна: $\mathbf{Safe}_\Sigma() = \Sigma$ ***safe***	
множество безопасных продолжающих символов перечислимо и задан итератор $\mathbf{Safehead}_\Sigma(\sigma)$: ***safehead***$(\Sigma$ ***after*** $\sigma)$	
множество продолжающих символов разрешимо относительно множества безопасных символов и задан оракул $\mathbf{SafeExtend}_\Sigma(\sigma,u) = u \in$ ***head***$(\Sigma$ ***after*** $\sigma)$ для $u \in$ ***safesymbols***$(\Sigma$ ***after*** $\sigma)$	

▲

T1	$\Sigma \in MODEL_{\beta\gamma\delta}(C)$
задан оракул безопасности $\mathbf{Safe}_\Sigma()$, который возвращает ***true***, если модель безопасна: $\mathbf{Safe}_\Sigma() =$ ***safe*** Σ	
множество стимулов перечислимо и задан итератор стимулов $\mathbf{X}_C$: ?C	
множество реакций конечно и задан итератор реакций $\mathbf{Y}_C$: !C	
для $\forall\sigma \in$ ***safe***(Σ):	
множество разрушающих стимулов и реакций разрешимо относительно множества всех стимулов и реакций и задан оракул $\mathbf{Gamma1}_\Sigma(\sigma,z) = \langle z,\gamma\rangle \in (\Sigma$ ***after*** $\sigma)$ для $z \in C$	
множество продолжающих символов разрешимо относительно множества безопасных символов и задан оракул $\mathbf{Extend1}_\Sigma(\sigma,u) = u \in$ ***head***$(\Sigma$ ***after*** $\sigma)$ для $u \in$ ***safesymbols***$(\Sigma$ ***after*** $\sigma)$	

▽▲

T2	$\Sigma \in MODEL_{\beta\gamma\delta}(C)$
множество стимулов перечислимо и задан итератор стимулов $\mathbf{X}_C$: ?C	
множество реакций конечно и задан итератор реакций $\mathbf{Y}_C$: !C	
для $\forall\sigma \in$ ***safe***(Σ):	
множество продолжающих символов разрешимо относительно множества всех стимулов и реакций, а также безопасных отказов и задан оракул $\mathbf{Extend2}_\Sigma(\sigma,u) = \sigma\cdot\langle u\rangle \in \Sigma$ для $u \in$ ***safesymbols***$(\Sigma$ ***after*** $\sigma) \cup C$	
задан оракул $\mathbf{Gamma2}_\Sigma(\epsilon) = \langle\gamma\rangle \in (\Sigma$ ***after*** $\epsilon)$, $\mathbf{Gamma2}_\Sigma(\sigma\cdot\langle z\rangle) = \langle\gamma\rangle \in (\Sigma$ ***after*** $\sigma\cdot\langle z\rangle)$ для $z \in C$ и $\sigma\cdot\langle z\rangle \in \Sigma$	

Рис.45.

Глава 2.3. LTS-модель

Структура главы:

1. Определение LTS
2. LTS-модель
3. LTS-модель как βγδ-машина
4. Эквивалентность βγδ-моделей и LTS-моделей
5. Гипотеза о безопасности и соответствие $ioco_{\beta\gamma\delta}$
6. Композиция LTS и тесты
7. Алгоритмизация

В этой главе в качестве модели рассматривается система размеченных переходов (LTS – Labelled Transition System), состоящая из состояний и переходов между состояниями, помеченных символами действий. В треминах теории графов, LTS – это ориентированный граф с помеченными дугами: вершина – это состояние, дуга – переход, а пометка на дуге – символ перехода. Внешние действия моделируются переходами, помеченными символами внешних действий, а внутренняя активность – переходами, помеченными символом τ. Переход, соответствующий внешнему действию, может выполниться, если он разрешён (нажата кнопка); τ-переходы могут выполняться независимо от нажатия кнопок действий. Состояние имеет конечную внутреннюю активность, если все цепочки τ-переходов, начинающиеся в этом состоянии, имеют конечную длину. В противном случае, если есть бесконечная цепочка τ-переходов, начинающаяся в этом состоянии, состояние имеет бесконечную внутреннюю активность, то есть дивергентно.

Развивая эту модель, мы также разрешаем в LTS разрушение – переход по символу γ, который может выполняться независимо от нажатия кнопок, но условно считается наблюдаемым переходом. Состояние будем называть стабильным, если нет τ- и γ-переходов, начинающихся в этом состоянии. Отказ может возникнуть в стабильном состоянии, если нажата кнопка, разрешающая только такие переходы, которые не определены в этом состоянии. Таким образом, LTS, помещённая внутрь чёрного ящика, функционирует как машина тестового сценария, описанная в 2.1.1. Формально это будет показано ниже (2.3.3).

Для βγδ-машины внешние переходы LTS разделяются на переходы по стимулам и по реакциям. Блокировка стимула `{?x}` может возникнуть в стабильном состоянии, в котором нет перехода по стимулу `?x`. Стационарность δ может возникнуть в стабильном состоянии, в котором нет переходов ни по одной реакции.

Если LTS строго-конвергентна, то есть в ней нет бесконечных цепочек τ-переходов, то мы можем распознавать отказ, если считать, что переходы совершаются мгновенно[39]. Для безопасного тестирования требование строго-конвергентности можно ослабить: дивергентным может быть только такое состояние, которое не достижимо по безопасным βγδ-трассам, то есть тем трассам, по которым будет вестись тестирование. Моделируя дивергенцию разрушением, можно сказать иначе: все безопасные βγδ-трассы заканчиваются в конвергентных состояниях.

2.3.1. Определение LTS

Определение 39: *Система размеченных переходов* (LTS - Labelled Transition System) – это совокупность $\mathbf{S}=LTS(V_S, C, E_S, s_0)$, где:

V_S – непустое множество *состояний*,
C – множество *символов* (алфавит символов),
$\tau \notin C \cup \wp(C)$ – символ внутреннего действия,
$E_S \subseteq V_S \times C_\tau \times V_S$ – множество переходов,
$s_0 \in V_S$ – начальное состояние (Рис.46).

- Множество LTS с заданным алфавитом C обозначим $\boldsymbol{LTS}(C)$.
- LTS будем обозначать заглавными буквами жирным шрифтом $\mathbf{S}, \mathbf{T}, \ldots$, множества их состояний и переходов буквами V и E, соответственно, с именем LTS в нижнем индексе – $V_S, V_T, \ldots$ и $E_S, E_T, \ldots$, а их начальные состояния соответствующими строчными буквами с нижним индексом "0" – $s_0, t_0, \ldots$.

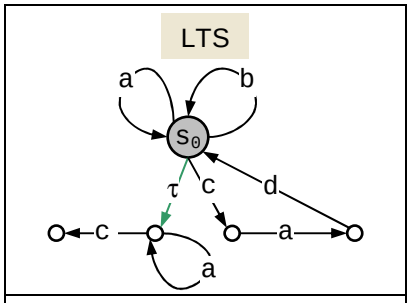


Рис.46.

□

Определение 40: Для $\mathbf{S}=LTS(V_S, C, E_S, s_0)$, $s, s` \in V_S$, $U \subseteq V_S$, $t \in C_\tau$, $z \in C$, $\sigma \in C^\omega$, $\mu \in C_\tau^\omega$ определим:

- $\boldsymbol{pre}(s,t,s`) =_{def} s$.
- $\boldsymbol{post}(s,t,s`) =_{def} s`$.
- $\boldsymbol{symbol}(s,t,s`) =_{def} t$.
- $S \in E_S^\omega$ будем называть маршрутом с началом в s, если $S=\epsilon$ или $S \neq \epsilon$ & $\boldsymbol{pre} \circ S(1)=s$ & $\forall i \in [2..|S|]$ $\boldsymbol{post} \circ S(i-1)=\boldsymbol{pre} \circ S(i)$.

---

[39] Достаточно, чтобы любая конечная цепочка переходов, все переходы которой, кроме, быть может, одного, это τ-переходы, выполнялась за время, ограниченное сверху конечным тайм-аутом.

- Начальное состояние подразумевается для любого маршрута, в том числе пустого. Для определённости, пустой маршрут, начинающийся в состоянии s, будем обозначать (s,ϵ).
- Множество маршрутов, начинающихся в одном состоянии, очевидно, является деревом. Допуская вольность речи, множество маршрутов, начинающихся в нескольких состояниях, также будем называть деревом маршрутов.
- Концом конечного маршрута $S \in E_S^*$ назовём пост-состояние его последнего перехода, если маршрут не пуст, или его начало, если маршрут пуст (а его начальное состояние подразумевается); конец маршрута обозначим $\omega(S)$:

  $: \omega((s,\epsilon)) =_{def} s;$

  $S \neq (s,\epsilon): \omega(S) =_{def} \mathit{post} \circ S(|S|).$
- $\mathit{runs}^{\omega}(s)$ – множество маршрутов с началом в s.
- $\mathit{runs}^{\infty}(s)$ – множество бесконечных маршрутов с началом в s.
- $\mathit{runs}(s)$ – множество конечных маршрутов с началом в s.[40]
- $\mathit{runs}^{\omega}(\mathbf{S}) =_{def} \mathit{runs}^{\omega}(s_0).$
- $\mathit{runs}^{\infty}(\mathbf{S}) =_{def} \mathit{runs}^{\infty}(s_0).$
- $\mathit{runs}(\mathbf{S}) =_{def} \mathit{runs}(s_0).$
- Для $S \in \mathit{runs}^{\omega}(s)$:
  - $\mathit{trace}(S) =_{def} (\mathit{symbol} \circ S)\uparrow\{\tau\};$
  - S τ-маршрут $=_{def} |\mathit{trace}(S)| = \epsilon.$
- $s \text{—}\mu\rightarrow =_{def} \exists S \in \mathit{runs}^{\omega}(s)\ \mu = \mathit{symbol} \circ S.$
- $s \text{—}\mu\rightarrow s` =_{def} \exists S \in \mathit{runs}(s)\ \mu = \mathit{symbol} \circ S \ \&\ s` = \omega(S).$
- $s \text{—}\mu\not\rightarrow =_{def} \neg\ s \text{—}\mu\rightarrow.$
- $s \text{—}\mu\not\rightarrow s` =_{def} \neg\ s \text{—}\mu\rightarrow s`.$
- $s = \sigma\Rightarrow =_{def} \exists S \in \mathit{runs}^{\omega}(s)\ \sigma = \mathit{trace}(S).$
- $s = \sigma\Rightarrow s` =_{def} \exists S \in \mathit{runs}(s)\ \sigma = \mathit{trace}(S) \ \&\ s` = \omega(S).$
- $s = \sigma\not\Rightarrow =_{def} \neg\ s = \sigma\Rightarrow.$
- $s = \sigma\not\Rightarrow s` =_{def} \neg\ s = \sigma\Rightarrow s`.$
- $s = \Rightarrow s` =_{def} s = \epsilon\Rightarrow s`.$
- $s = \not\Rightarrow s` =_{def} s = \epsilon\not\Rightarrow s`.$

---

[40] Очевидно, $\mathit{runs}^{\omega}(s)$ и $\mathit{runs}(s)$ деревья, а $\mathit{runs}^{\infty}(s)$ не дерево.

· $s\text{—}t\to \;=_{def}\; s\text{—}\langle t\rangle\to$.

· $s\text{—}t\to s' \;=_{def}\; s\text{—}\langle t\rangle\to s'$;[41]

если $t\neq\tau$, то говорят, что символ t *допустим* в состоянии s.

· $s\text{—}t\not\to \;=_{def}\; s\text{—}\langle t\rangle\not\to$.

· $s\text{—}t\not\to s' \;=_{def}\; s\text{—}\langle t\rangle\not\to s'$.

· $s{=}z{\Rightarrow} \;=_{def}\; s{=}\langle z\rangle{\Rightarrow}$.

· $s{=}z{\Rightarrow}s' \;=_{def}\; s{=}\langle z\rangle{\Rightarrow}s'$.

· $s{=}z{\not\Rightarrow} \;=_{def}\; s{=}\langle z\rangle{\not\Rightarrow}$.

· $s{=}z{\not\Rightarrow}s' \;=_{def}\; s{=}\langle z\rangle{\not\Rightarrow}s'$.

· s ***after*** $\sigma \;=_{def}\; \{s'\in V_s \mid s=\sigma\Rightarrow s'\}$.

· $\mathbf{S}$ ***after*** $\sigma \;=_{def}\; s_0$ ***after*** σ.

· ***der***$(s) \;=_{def}\; \{s'\in V_s \mid \exists\sigma\in C^*\ s=\sigma\Rightarrow s'\}$

– состояния, достижимые из s.

· ***der***$(\mathbf{S}) \;=_{def}$ ***der***(s_0) – достижимые состояния $\mathbf{S}$.

· ***init***$(s) \;=_{def}\; \{t\in C_\tau \mid s\text{—}t\to\}$.

· ***traces***$^{\omega}(s) \;=_{def}\; \{$***trace***$(S) \mid S\in$ ***runs***$^{\omega}(s)\}$.

· ***traces***$^{\infty}(s) \;=_{def}\; \{$***trace***$(S) \mid S\in$ ***runs***$^{\infty}(s)\}$.

· ***traces***$(s) \;=_{def}\; \{$***trace***$(S) \mid S\in$ ***runs***$(s)\}$.[42]

· ***traces***$^{\omega}(\mathbf{S}) \;=_{def}$ ***traces***$^{\omega}(s_0)$ – множество трасс $\mathbf{S}$.

· ***traces***$^{\infty}(\mathbf{S}) \;=_{def}$ ***traces***$^{\infty}(s_0)$ – множество бесконечных трасс $\mathbf{S}$.

· ***traces***$(\mathbf{S}) \;=_{def}$ ***traces***(s_0) – множество конечных трасс $\mathbf{S}$.

· s *терминально* $\;=_{def}$ ***init***$(s)=\varnothing$.

· s *стабильно* $\;=_{def}\; s\text{—}\tau\gamma\not\to \;=_{def}\; s\text{—}\tau\not\to$ & $s\text{—}\gamma\not\to$.

· Конвергентность: $s\!\downarrow \;=_{def}\; \forall S\in$ ***runs***$^{\infty}(s)$ ***trace***$(S)\neq\epsilon$

– в состоянии s не начинается бесконечный маршрут τ-переходов;

$U\!\downarrow \;=_{def}\; \forall s\in U\ s\!\downarrow$.

· Дивергентность: $s\!\uparrow \;=_{def}\; \neg(s\!\downarrow)$;

$U\!\uparrow \;=_{def}\; \exists s\in U\ s\!\uparrow$.

· Строгая конвергентность: $\mathbf{S}\!\Downarrow \;=_{def}$ ***der***$(\mathbf{S})\!\downarrow$.

· s *детерминировано* $\;=_{def}\; \forall\sigma\in C^*\ |s$ ***after*** $\sigma|\leq 1$.

---

[41] Для выразительности записи там, где это не приведёт к недоразумениям, мы, допуская вольность речи, будем писать $s\text{—}z\to s'$ вместо (s,z,s').

[42] Очевидно, ***traces***$(s)\in$ ***Trees***(C), ***traces***$^{\omega}(s)\in$ ***Trees***$^{\omega}(C)$, а ***traces***$^{\infty}(s)$ не дерево.

· **S** *детерминирована* $=_{def}$ s_0 детерминировано.
· **S** *конечна* $=_{def}$ $|\{e \in E_S | \boldsymbol{pre}(e) \in \boldsymbol{der}(S)\}| \neq \infty$
– конечно множество переходов, определенных в достижимых состояниях.
· **S** *является LTS-деревом* $=_{def}$ $\forall s \in \boldsymbol{der}(S)\ |\boldsymbol{runs}(S) \cap \omega^{-1}(s)| = 1$
– в каждом достижимом состоянии заканчивается только один маршрут.
· **S** *конечно-ветвящаяся* $=_{def}$ $\forall s \in \boldsymbol{der}(S)\ |\{e \in E_S | \boldsymbol{pre}(e) = s\}| \neq \infty$
– в каждом достижимом состоянии определено конечное число переходов (это эквивалентно конечно-ветвимости дерева маршрутов LTS).
· **S** *локально-конечно-ветвящаяся*
$=_{def}$ $\forall s \in \boldsymbol{der}(S)\ \forall t \in C_\tau\ |\{s` | s \text{—} t \rightarrow s`\}| \neq \infty$
– в каждом достижимом состоянии определено конечное число переходов по каждому символу (включая τ).

□

Определение 41: Две LTS в одном алфавите $A, B \in \boldsymbol{LTS}(C)$ называют *изоморфными*, если существует такая биекция множеств достижимых состояний $\boldsymbol{f}: \boldsymbol{der}(A) \rightarrow \boldsymbol{der}(B)$, что

$(\forall a, a` \in V_A\ \forall t \in C_\tau\ a \text{—} t \rightarrow a` \Leftrightarrow \boldsymbol{f}(a) \text{—} t \rightarrow \boldsymbol{f}(a`))\ \&\ \boldsymbol{f}(a_0) = b_0.$

Биекцию $\boldsymbol{f}$ называют в этом случае *изоморфизмом*.

□

2.3.2. LTS-модель

Структура раздела:
· Определение LTS-модели, стационарного состояния и βδ-отказов.
· LTS-модели без блокировок и/или разрушения.
· Преобразование LTS-модели во множество βγδ-трасс.
· Распространение LTS-обозначений на βγδ-трассы.
· βγδ-трассы маршрутов.
· Распространение трассовых обозначений на LTS.
· Символы, разрушающие и безопасные в состоянии.
· Объединение LTS-моделей.

Определение LTS-модели, стационарного состояния и βδ-отказов.

LTS в алфавите $C \subseteq Z$ иногда называют *Input-Output Labelled Transition System* (IOLTS) [JJTV99], *Input-Output Transition System* (IOTS) [Tret96] или *Input-Output Automaton* (IOA) [LT87]. Различия между этими понятиями маргинальны[43]. Допуская возможность разрушения, мы будем моделировать спецификацию и реализацию LTS в алфавите C_γ, где C – базовый алфавит.

[43] IOTS и IOA обычно считаются всюду определёнными по стимулам (input-enabled).

Определение 42: Будем предполагать, что $\gamma\neq\tau$. LTS-моделью в алфавите $C\subseteq Z$ будем называть LTS в алфавите C_γ. Множество LTS-моделей в алфавите $C\subseteq Z$ обозначим $\boldsymbol{LTS}_{\beta\gamma\delta}(C) =_{def} \boldsymbol{LTS}(C_\gamma)$, а множество всех LTS-моделей $\boldsymbol{LTS}_{\beta\gamma\delta}=\cup\{\boldsymbol{LTS}_{\beta\gamma\delta}(C)\,|\,C\subseteq Z\}$.

□

Определение 43: Для $C\subseteq Z$, $\mathbf{S}\in\boldsymbol{LTS}_{\beta\gamma\delta}(C)$ и состояния $s\in V_{\mathbf{S}}$

- s *стационарно* (quiescent), если в нём определены только переходы по стимулам:
  $\delta(s) =_{def} \boldsymbol{init}(s)\subseteq ?C$;
- Для состояния s через $\beta\delta(s)$ будем обозначать множество $\beta\delta$-отказов, порождаемых состоянием:
  $\beta\delta(s) =_{def} \beta\delta(\{\ z\in C \mid s\text{—}\tau\gamma\!\!\rightarrow \ \&\ s\text{—}z\!\!\rightarrow\ \})$.

□

$\beta\delta(s)$ пусто в нестабильном состоянии. В стабильном состоянии $\beta\delta(s)=\beta\delta(C\setminus\boldsymbol{init}(s))$, то есть состоит из блокировок стимулов, переходы по которым не определены в состоянии, и стационарности, если в состоянии не определены переходы по реакциям. Тем самым, если в стабильном состоянии определены переходы по каждому стимулу и хотя бы одной реакции, множество $\beta\delta(s)$ пусто.

LTS-модели без блокировок и/или разрушения.

Определение 44: Пусть задан алфавит $C\subseteq Z$. Обозначим:

1. модели без блокировок:
   $\boldsymbol{LTS}_{\gamma\delta}(C) =_{def} \{\mathbf{S}\in\boldsymbol{LTS}(C_\gamma)\,|\,\forall s\in\boldsymbol{der}(\mathbf{S})\ \beta\delta(s)\cap\beta(C)=\varnothing\}$,
   $\boldsymbol{LTS}_{\gamma\delta} =_{def} \cup\{\boldsymbol{LTS}_{\gamma\delta}(C)\,|\,C\subseteq Z\}$;
2. модели без разрушения (включая дивергенцию):
   $\boldsymbol{LTS}_{\beta\delta}(C) =_{def} \{\mathbf{S}\in\boldsymbol{LTS}(C_\gamma)\,|\,\forall s\in\boldsymbol{der}(\mathbf{S})\ s\text{—}\gamma\!\!\rightarrow\ \&\ s\!\downarrow\}$,
   $\boldsymbol{LTS}_{\beta\delta} =_{def} \cup\{\boldsymbol{LTS}_{\beta\delta}(C)\,|\,C\subseteq Z\}$;
3. модели без блокировок и разрушения (включая дивергенцию):
   $\boldsymbol{LTS}_{\delta}(C) =_{def} \boldsymbol{LTS}_{\gamma\delta}(C)\cap\boldsymbol{LTS}_{\beta\delta}(C)$,
   $\boldsymbol{LTS}_{\delta} =_{def} \cup\{\boldsymbol{LTS}_{\delta}(C)\,|\,C\subseteq Z\} = \boldsymbol{LTS}_{\gamma\delta}\cap\boldsymbol{LTS}_{\beta\delta}$.

□

Преобразование LTS-модели во множество $\beta\gamma\delta$-трасс.

Определим преобразование $\boldsymbol{LTS}_{\beta\gamma\delta}(C)\rightarrow\boldsymbol{LTS}(C_{\beta\gamma\delta})$ и возьмём множество трасс полученной LTS (Рис.47). В стационарном состоянии добавляется переход-петля по стационарности δ; в стабильном состоянии, где не определён переход по стимулу $?x$, добавляется переход-петля по

блокировке стимула {?x}; в дивергентном состоянии определяется γ-переход в специальное терминальное состояние γ; это же состояние становится постсостоянием для всех γ-переходов, уже имевшихся в исходной LTS[44]. Заметим, что в трассе преобразованной LTS разрушение может быть только конечным символом.

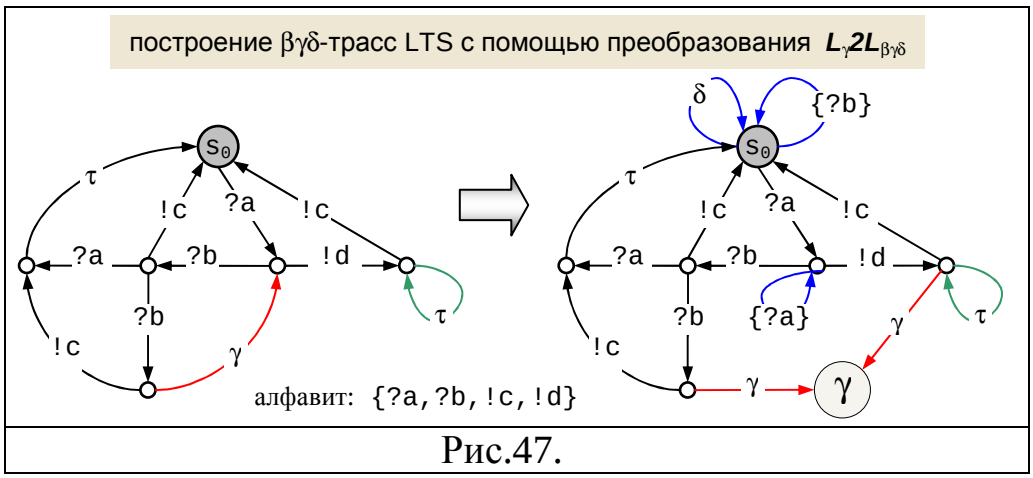


Рис.47.

Определение 45: Для $C \subseteq Z$ определим преобразование $L_\gamma 2L_{\beta\gamma\delta} : LTS_{\beta\gamma\delta}(C) \to LTS(C_{\beta\gamma\delta})$. Для $S=LTS(V_S, C_\gamma, E_S, s_0)$ $L_\gamma 2L_{\beta\gamma\delta}(S)=M=LTS(V_S \cup \{\gamma\}, C_{\beta\gamma\delta}, E_M, s_0)$, где $\gamma \notin V_S$, а E_M – наименьшее множество, порождаемое следующими правилами вывода: $\forall s, s` \in V_S$

$z \in C_\tau \;\&\; s \text{—}z\to s` \;\vdash\; s \text{—}z\to s`,$

$o \in \beta\delta(s) \;\vdash\; s \text{—}o\to s,$

$s \text{—}\gamma\to \;\vee\; s\uparrow \;\vdash\; s \text{—}\gamma\to\gamma.$

□

Определение 46: Пусть $C \subseteq Z$ и $S \in LTS_{\beta\gamma\delta}(C)$.

- βγδ-маршрутом в LTS S будем называть маршрут в LTS $M=L_\gamma 2L_{\beta\gamma\delta}(S)$: для состояния $s \in V_S$:
  - $runs_{\beta\gamma\delta}(s)$ $=_{def}$ $runs(s)$ в $L_\gamma 2L_{\beta\gamma\delta}(S)$ – конечные βγδ-маршруты с началом в s,
  - $runs^{\infty}_{\beta\gamma\delta}(s)$ $=_{def}$ $runs^{\infty}(s)$ в $L_\gamma 2L_{\beta\gamma\delta}(S)$ – бесконечные βγδ-маршруты с началом в s,
  - $runs^{\omega}_{\beta\gamma\delta}(s)$ $=_{def}$ $runs^{\omega}(s)$ в $L_\gamma 2L_{\beta\gamma\delta}(S)$ – все βγδ-маршруты с началом в s;
  - $runs_{\beta\gamma\delta}(S)$ $=_{def}$ $runs_{\beta\gamma\delta}(s_0)$ – конечные βγδ-маршруты S,

[44] Если LTS уже содержит состояние, которое называется γ, то перед преобразованием переименуем его.

- $\boldsymbol{runs}^{\infty}_{\beta\gamma\delta}(\mathbf{S}) =_{def} \boldsymbol{runs}^{\infty}_{\beta\gamma\delta}(s_0)$ – бесконечные βγδ-маршруты $\mathbf{S}$,
- $\boldsymbol{runs}^{\omega}_{\beta\gamma\delta}(\mathbf{S}) =_{def} \boldsymbol{runs}^{\omega}_{\beta\gamma\delta}(s_0)$ – все βγδ-маршруты $\mathbf{S}$.

· βγδ-трассой в LTS $\mathbf{S}$ будем называть трассу в LTS $\mathbf{M}=\boldsymbol{L}_{\gamma}\boldsymbol{2}\boldsymbol{L}_{\beta\gamma\delta}(\mathbf{S})$:
для состояния $s\in V_{\mathbf{S}}$:

- $\boldsymbol{traces}_{\beta\gamma\delta}(s) =_{def} \boldsymbol{traces}(s)$ в $\boldsymbol{L}_{\gamma}\boldsymbol{2}\boldsymbol{L}_{\beta\gamma\delta}(\mathbf{S})$
  – конечные βγδ-трассы с началом в s,
- $\boldsymbol{traces}^{\infty}_{\beta\gamma\delta}(s) =_{def} \boldsymbol{traces}^{\infty}(s)$ в $\boldsymbol{L}_{\gamma}\boldsymbol{2}\boldsymbol{L}_{\beta\gamma\delta}(\mathbf{S})$
  – бесконечные βγδ-трассы с началом в s,
- $\boldsymbol{traces}^{\infty}_{\beta\gamma\delta}(s) =_{def} \boldsymbol{traces}^{\infty}(s)$ в $\boldsymbol{L}_{\gamma}\boldsymbol{2}\boldsymbol{L}_{\beta\gamma\delta}(\mathbf{S})$
  – все βγδ-трассы с началом в s;
- $\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S}) =_{def} \boldsymbol{traces}_{\beta\gamma\delta}(s_0)$ – конечные βγδ-трассы $\mathbf{S}$,
- $\boldsymbol{traces}^{\infty}_{\beta\gamma\delta}(\mathbf{S}) =_{def} \boldsymbol{traces}^{\infty}_{\beta\gamma\delta}(s_0)$ – бесконечные βγδ-трассы $\mathbf{S}$,
- $\boldsymbol{traces}^{\infty}_{\beta\gamma\delta}(\mathbf{S}) =_{def} \boldsymbol{traces}^{\infty}_{\beta\gamma\delta}(s_0)$ – все βγδ-трассы $\mathbf{S}$.

· Две LTS-модели будем считать βγδ-*эквивалентными*, если они имеют одно и то же множество *конечных* βγδ-трасс.

□

Распространение LTS-обозначений на βγδ-трассы.

Введя βγδ-маршруты и βγδ-трассы, мы можем распространить LTS-обозначения, использующие последовательности $\sigma\in C_{\tau}^{\omega}$ и трассы $\sigma\in C^{\omega}$ на βγδ-последовательности $\sigma\in C_{\beta\gamma\delta\tau}^{\omega}$ и βγδ-трассы $\sigma\in C_{\beta\gamma\delta}^{\omega}$. Это относится к стрелкам $\mathrel{—}\sigma\!\!\rightarrow$, $\mathrel{—}\sigma\!\!\not\rightarrow$, $=\sigma\!\!\Rightarrow$, $=\sigma\!\!\not\Rightarrow$ и оператору ***after*** σ.

Стрелки имеют один и тот же смысл для LTS $\mathbf{S}$ и LTS $\boldsymbol{L}_{\gamma}\boldsymbol{2}\boldsymbol{L}_{\beta\gamma\delta}(\mathbf{S})$, если последовательность σ (трасса для двойной стрелки) не содержит отказов и не заканчивается символом разрушения γ. Если σ содержит отказы, стрелки, очевидно, применяются к LTS $\boldsymbol{L}_{\gamma}\boldsymbol{2}\boldsymbol{L}_{\beta\gamma\delta}(\mathbf{S})$, где есть переходы-петли по отказам. Двусмысленность возможна лишь в том случае, когда σ не содержит отказов и заканчивается символом γ, что происходит из-за того, что дивергенцию мы моделируем разрушением: в LTS $\boldsymbol{L}_{\gamma}\boldsymbol{2}\boldsymbol{L}_{\beta\gamma\delta}(\mathbf{S})$ есть дополнительные γ-переходы. По умолчанию мы будем считать, что для такой стрелки подразумевается LTS $\mathbf{S}$, а если имеется в виду LTS $\boldsymbol{L}_{\gamma}\boldsymbol{2}\boldsymbol{L}_{\beta\gamma\delta}(\mathbf{S})$, то это будем указывать явно.

Тем самым, вообще говоря, $\sigma\in\boldsymbol{traces}_{\beta\gamma\delta}(s) \neq s=\sigma\!\!\Rightarrow$,

хотя $\sigma\in\boldsymbol{traces}_{\beta\gamma\delta}(s) = s=\sigma\!\!\Rightarrow$ в $\boldsymbol{L}_{\gamma}\boldsymbol{2}\boldsymbol{L}_{\beta\gamma\delta}(\mathbf{S})$.

Определение 47: Пусть $C \subseteq Z$ и $\mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$. LTS-операторы-стрелки $\text{—}\sigma\rightarrow$, $\text{—}\sigma\not\rightarrow$, $=\sigma\Rightarrow$ и $=\sigma\not\Rightarrow$ распространим на $\beta\delta$-отказы, $\beta\gamma\delta\tau$-последовательности и $\beta\gamma\delta$-трассы: для $t \in C_{\beta\delta\tau}$, $z \in C_{\beta\delta}$, $\mu \in C_{\beta\gamma\delta\tau}{}^{\omega}$, $\sigma \in C_{\beta\gamma\delta}{}^{\omega}$ таких, что $\mu \in C_{\gamma\tau}{}^{\omega} \Rightarrow \gamma \notin \boldsymbol{Im}(\mu)$ и $\sigma \in C_{\gamma}{}^{\omega} \Rightarrow \gamma \notin \boldsymbol{Im}(\sigma)$:

$\ldots\text{—}\mu\ldots =_{def} \ldots\text{—}\mu\ldots$ в $\boldsymbol{L}_{\gamma}\boldsymbol{2}\boldsymbol{L}_{\beta\gamma\delta}(\mathbf{S})$;

$\ldots\text{—}t\ldots =_{def} \ldots\text{—}t\ldots$ в $\boldsymbol{L}_{\gamma}\boldsymbol{2}\boldsymbol{L}_{\beta\gamma\delta}(\mathbf{S}) = \ldots\text{—}\langle t\rangle\ldots$;

$\ldots=\sigma\ldots =_{def} \ldots=\sigma\ldots$ в $\boldsymbol{L}_{\gamma}\boldsymbol{2}\boldsymbol{L}_{\beta\gamma\delta}(\mathbf{S})$;

$\ldots=z\ldots =_{def} \ldots=z\ldots$ в $\boldsymbol{L}_{\gamma}\boldsymbol{2}\boldsymbol{L}_{\beta\gamma\delta}(\mathbf{S}) = \ldots=\langle z\rangle\ldots$.

□

Оператор ***after*** для $\beta\gamma\delta$-трассы, содержащей отказы, очевидно, относится к преобразованной LTS $\boldsymbol{L}_{\gamma}\boldsymbol{2}\boldsymbol{L}_{\beta\gamma\delta}(\mathbf{S})$. Если $\beta\gamma\delta$-трасса σ не содержит отказы (является трассой), то s ***after*** σ в $\mathbf{S} = s$ ***after*** σ в $\boldsymbol{L}_{\gamma}\boldsymbol{2}\boldsymbol{L}_{\beta\gamma\delta}(\mathbf{S})$ только в том случае, когда трасса σ не заканчивается разрушением. Если $\beta\gamma\delta$-трасса заканчивается разрушением, то в LTS $\boldsymbol{L}_{\gamma}\boldsymbol{2}\boldsymbol{L}_{\beta\gamma\delta}(\mathbf{S})$ она заканчивается в единственном состоянии γ. Тем самым, мы можем корректно распространить оператор ***after*** в LTS $\mathbf{S}$ на $\beta\gamma\delta$-трассы, не заканчивающиеся разрушением.

Определение 48: Пусть $C \subseteq Z$ и $\mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$. LTS-оператор ***after*** распространим на $\beta\gamma\delta$-трассы, не заканчивающиеся разрушением: для $s \in V_{\mathbf{s}}$, $\sigma \in \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$ такой, что $\gamma \notin \boldsymbol{Im}(\sigma)$:

s ***after*** $\sigma =_{def} s$ ***after*** σ в $\boldsymbol{L}_{\gamma}\boldsymbol{2}\boldsymbol{L}_{\beta\gamma\delta}(\mathbf{S})$;

$\mathbf{S}$ ***after*** $\sigma =_{def} s_0$ ***after*** $\sigma = \boldsymbol{L}_{\gamma}\boldsymbol{2}\boldsymbol{L}_{\beta\gamma\delta}(\mathbf{S})$ ***after*** σ;

□

$\beta\gamma\delta$-трассы маршрутов.

Каждому (конечному или бесконечному) маршруту S LTS $\mathbf{S}$ поставим в соответствие маршруты LTS $\boldsymbol{L}_{\gamma}\boldsymbol{2}\boldsymbol{L}_{\beta\gamma\delta}(\mathbf{S})$, то есть $\beta\gamma\delta$-маршруты LTS $\mathbf{S}$, которые отличаются от S:

a) *возможным* наличием петель $\beta\delta$-отказов

плюс b) *обязательной* заменой первого γ-перехода на γ-переход в γ-состояние и удалением переходов после γ-перехода, если этот маршрут содержит γ-переход,

или c) *возможным* дополнительным последним γ-переходом, моделирующим дивергентное конечное состояние конечного маршрута S, если этот маршрут не содержит γ-переходов.

Поясним случай b). Продолжения маршрутов после первых γ-переходов нас не интересуют, поскольку они не влияют на βγδ-трассы LTS.

Поясним случай c). Мы сопоставляем с маршрутом не только те βγδ-маршруты, которые заканчиваются моделирующим дивергенцию γ-переходом, но и их префиксы до этого γ-перехода. Это делается для того, чтобы в LTS объединение множества βγδ-маршрутов (βγδ-трасс) её маршрутов было деревом и совпало с множеством βγδ-маршрутов (βγδ-трасс) LTS согласно Определение 46:.

Определение 49: Пусть $C\subseteq Z$ и $\mathbf{S}\in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$. Для маршрута $S\in \boldsymbol{runs}^{\omega}(\mathbf{S})$ определим множество соответствующих ему βγδ-маршрутов (маршрутов LTS $\boldsymbol{L}_{\gamma}\boldsymbol{2}\boldsymbol{L}_{\beta\gamma\delta}(\mathbf{S})$):

a) $\gamma\notin \boldsymbol{Im}_{\circ}\boldsymbol{symbol}(S)$ & (S конечен $\Rightarrow \omega(S)\downarrow$)

: $\boldsymbol{runs}_{\beta\gamma\delta}(S) =_{def} \boldsymbol{runs}_{\beta\delta}(S)$,

: $\boldsymbol{runs}^{\omega}_{\beta\gamma\delta}(S) =_{def} \boldsymbol{runs}^{\omega}_{\beta\delta}(S)$,

a+b) $S=P\cdot\langle s,\gamma,s`\rangle\cdot R$ & $\gamma\notin \boldsymbol{Im}_{\circ}\boldsymbol{symbol}(P)$

: $\boldsymbol{runs}^{\omega}_{\beta\gamma\delta}(S) =_{def} \boldsymbol{runs}_{\beta\gamma\delta}(S) =_{def} \boldsymbol{runs}_{\beta\delta}(P)\cdot\{\langle\omega(P)\text{—}\gamma\rightarrow\gamma\rangle\}$,

a+c) $\gamma\notin \boldsymbol{Im}_{\circ}\boldsymbol{symbol}(S)$ & S конечен & $\omega(S)\uparrow$

: $\boldsymbol{runs}^{\omega}_{\beta\gamma\delta}(S) =_{def} \boldsymbol{runs}_{\beta\gamma\delta}(S) =_{def} \boldsymbol{runs}_{\beta\delta}(S)\cdot\{\epsilon,\langle\omega(S)\text{—}\gamma\rightarrow\gamma\rangle\}$,

где для любого маршрута R

$\boldsymbol{runs}_{\beta\delta}(R) =_{def} \{M\in \boldsymbol{runs}_{\beta\gamma\delta}(\mathbf{S}) \mid M\uparrow(V_{\mathbf{S}}\times\beta\delta(C)\times V_{\mathbf{S}})=R\}$,

$\boldsymbol{runs}^{\omega}_{\beta\delta}(R) =_{def} \{M\in \boldsymbol{runs}^{\omega}_{\beta\gamma\delta}(\mathbf{S}) \mid M\uparrow(V_{\mathbf{S}}\times\beta\delta(C)\times V_{\mathbf{S}})=R\}$.

□

Множество трасс соответствующих βγδ-маршрутов будем называть множеством βγδ-трасс маршрута S.

Определение 50: Пусть $C\subseteq Z$ и $\mathbf{S}\in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$. Для маршрута $S\in \boldsymbol{runs}(\mathbf{S})$ определим множество его βγδ-трасс как множество трасс соответствующих ему βγδ-маршрутов:

$\boldsymbol{traces}_{\beta\gamma\delta}(S) =_{def} \boldsymbol{trace}_{\circ}\boldsymbol{runs}_{\beta\gamma\delta}(S)$.

□

Из определения множества βγδ-маршрутов и βγδ-трасс маршрута вытекают очевидные следствия для множества βγδ-маршрутов и βγδ-трасс состояния s (или LTS, то есть её начального состояния s_0):

$\boldsymbol{runs}_{\beta\gamma\delta}(s) = \cup_{\circ}\boldsymbol{runs}_{\beta\gamma\delta\circ}\boldsymbol{runs}(s)$,

$\boldsymbol{runs}_{\beta\gamma\delta}(\mathbf{S}) = \cup_{\circ}\boldsymbol{runs}_{\beta\gamma\delta\circ}\boldsymbol{runs}(\mathbf{S})$;

$\boldsymbol{traces}_{\beta\gamma\delta}(s)=\boldsymbol{trace}_{\circ}\boldsymbol{runs}_{\beta\gamma\delta}(s)=\boldsymbol{trace}_{\circ}\cup_{\circ}\boldsymbol{runs}_{\beta\gamma\delta\circ}\boldsymbol{runs}(s)=\cup_{\circ}\boldsymbol{traces}_{\beta\gamma\delta\circ}\boldsymbol{runs}(s)$,

$\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})=\boldsymbol{trace}_{\circ}\boldsymbol{runs}_{\beta\gamma\delta}(\mathbf{S})=\boldsymbol{trace}_{\circ}\cup_{\circ}\boldsymbol{runs}_{\beta\gamma\delta\circ}\boldsymbol{runs}(\mathbf{S})=\cup_{\circ}\boldsymbol{traces}_{\beta\gamma\delta\circ}\boldsymbol{runs}(\mathbf{S})$.

В дальнейшем, по умолчанию, мы будем рассматривать конечные βγδ-маршруты и βγδ-трассы. Только в Глава 4.2 нам потребуются бесконечные βγδ-трассы.

Распространение трассовых обозначений на LTS.

Очевидно, множество βγδ-трасс состояния (или LTS) является βγδ-деревом. Обозначения, введённые для βγδ-деревьев, распространим на LTS. Для различения мы будем использовать нижний индекс "$_{\beta\gamma\delta}$".

Определение 51: Пусть $C \subseteq Z$, $\mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$, $s \in V_{\mathbf{S}}$, $U \subseteq V_{\mathbf{S}}$, $\sigma \in \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$.

- Множество начальных символов βγδ-трасс, начинающихся в s:
  $\boldsymbol{head}_{\beta\gamma\delta}(s) =_{def} \boldsymbol{head} \circ \boldsymbol{traces}_{\beta\gamma\delta}(s)$.
- Состояние безопасно, если из него по τ-переходам недостижимо разрушение:
  $s\ \boldsymbol{safe} =_{def} \langle\gamma\rangle \notin \boldsymbol{traces}_{\beta\gamma\delta}(s)$.
- Множество состояний безопасно, если все его состояния безопасны:
  $U\ \boldsymbol{safe} =_{def} \forall s \in U\ s\ \boldsymbol{safe}$.
- LTS безопасна, если её начальное состояние безопасно:
  $\mathbf{S}\ \boldsymbol{safe} =_{def} s_0\ \boldsymbol{safe}$.
- Множества βγδ-трасс:
  - безопасные βγδ-трассы, начинающиеся в s:
    $\boldsymbol{safe}_{\beta\gamma\delta}(s) =_{def} \boldsymbol{safe} \circ \boldsymbol{traces}_{\beta\gamma\delta}(s)$.
  - безопасные трассы, начинающиеся в s:
    $\boldsymbol{safe}(s) =_{def} \boldsymbol{safe}_{\beta\gamma\delta}(s) \cap \boldsymbol{traces}(s)$.
  - безопасные βγδ-трассы LTS:
    $\boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S}) =_{def} \boldsymbol{safe} \circ \boldsymbol{traces}_{\beta\gamma\delta}(s_0) = \boldsymbol{safe}_{\beta\gamma\delta}(s_0)$.
  - безопасные трассы LTS:
    $\boldsymbol{safe}(\mathbf{S}) =_{def} \boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S}) \cap \boldsymbol{traces}(\mathbf{S}) = \boldsymbol{safe}(s_0)$.
  - тестовые βγδ-трассы, начинающиеся в состоянии s:
    $\boldsymbol{tt}_{\beta\gamma\delta}(s) =_{def} \boldsymbol{tt} \circ \boldsymbol{traces}_{\beta\gamma\delta}(s)$.
  - тестовые βγδ-трассы LTS:
    $\boldsymbol{tt}_{\beta\gamma\delta}(\mathbf{S}) =_{def} \boldsymbol{tt} \circ \boldsymbol{traces}_{\beta\gamma\delta}(s_0) = \boldsymbol{tt}_{\beta\gamma\delta}(s_0)$.

□

Символы, разрушающие и безопасные в состоянии.

Определение символа, *безопасного в состоянии*, можно дать на основе определения стимула или реакции, *разрушающего в состоянии*. Стимул и его блокировка безопасны в состоянии, если в этом состоянии стимул не

разрушающий. Все реакции и стационарность безопасны в состоянии, если в этом состоянии каждая реакция не разрушающая. Множество символов, безопасных в состоянии s, обозначим $\boldsymbol{safesymbols}_{\beta\gamma\delta}(s)$.

Можно дать два различных определения стимула или реакции z, разрушающего в состоянии s: 1) в состоянии s есть βγδ-трасса ⟨z,γ⟩, 2) в состоянии s есть переход по z в опасное состояние (состояние, в котором есть βγδ-трасса ⟨γ⟩). См. Рис.48.

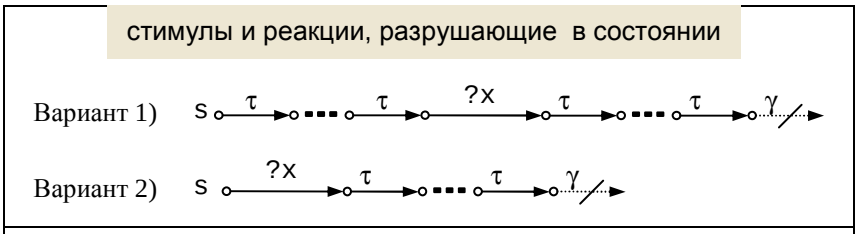


Рис.48.

В варианте 1) есть βγδ-маршрут, начинающийся в состоянии s и имеющий βγδ-трассу ⟨z,γ⟩. Такой βγδ-маршрут содержит один переход по z, до которого и после которого могут быть τ-переходы, и последний переход βγδ-маршрута – это γ-переход[45]. Множество символов, безопасных в состоянии, определяется так: $\boldsymbol{safesymbols}_{\beta\gamma\delta}(s) =_{def} \boldsymbol{safesymbols}\circ\boldsymbol{traces}_{\beta\gamma\delta}(s)$. В варианте 2) учитываются только такие βγδ-маршруты, которые начинаются сразу с z-перехода.

На самом деле нам никогда не понадобится определять безопасность символа в одном отдельно взятом состоянии, но только его безопасность после безопасной βγδ-трассы. Последнее для LTS означает безопасность символа u в каждом состоянии после безопасной βγδ-трассы σ: $u \in \cap\circ \boldsymbol{safesymbols}_{\beta\gamma\delta}(\mathbf{S}\ \boldsymbol{after}\ \sigma)$. А в этом случае оба определения оказываются эквивалентными. Для дальнейшей алгоритмизации вариант 2) предпочтительнее, поскольку требует меньше проверок.

Определение 52: Для $C \subseteq Z$, $\mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$, $s \in V_{\mathbf{S}}$, $z \in C$:

- z разрушающий в s $=_{def}$ $\exists s`\ s \text{—} z \rightarrow s`\ \&\ \neg\ s`$ ***safe***;
- z неразрушающий в s $=_{def}$ $\neg$ z разрушающий в s;
- $\boldsymbol{safesymbols}_{\beta\gamma\delta}(s) =_{def} \{\ ?x \in C \mid$?x неразрушающий в s$\}$

---

[45] Маршрут, соответствующий этому βγδ-маршруту, может не заканчиваться γ-переходом, а заканчиваться в дивергентном состоянии.

$\cup\{\{?x\}\in C \mid \quad ?x$ неразрушающий в $s\}$
$\cup\{\ !y\in C \mid \forall !t\in C\ !t$ неразрушающий в $s\}$
$\cup\{\quad \delta \quad \mid \forall !t\in C\ !t$ неразрушающий в $s\}$.

□

Объединение LTS-моделей.

Определение 53: Пусть $\mathbf{S}_i=LTS(V_i,C_i,E_i,s_{i0})$ LTS в алфавите $C_i\subseteq Z$, где индекс $i\in I$. Объединением множества LTS (см. Рис.49) назовём LTS $\mathbf{S}=\cup(\{\mathbf{S}_i|i\in I\})=LTS(V,C,E,s_0)$, где
$s_0\notin\cup(\{\{i\}\times V_i|i\in I\})$, $V=\{s_0\}\cup(\cup(\{\{i\}\times V_i|i\in I\}))$,
$C = \cup(\{C_i|i\in I\})$,
$E = \{s_0 \text{—}\tau\text{→}(i,s_{i0}) \mid i\in I\}$
$\cup(\cup(\{\{(i,s)\text{—}z\text{→}(i,s`) \mid s\text{—}z\text{→}s`\in E_i\} \mid i\in I\}))$.

□

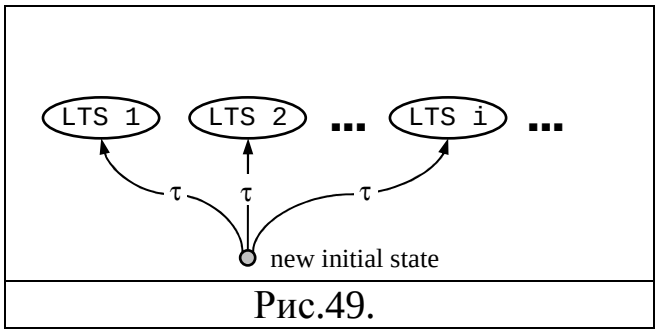


Рис.49.

Утверждение 34: Пусть $\mathbf{S}_i$ LTS-модель в алфавите $C_i\subseteq Z$, где $i\in I$. Объединение $\mathbf{S}=\cup(\{\mathbf{S}_i|i\in I\})$ LTS-моделей является LTS-моделью в объединённом алфавите $C=\cup(\{C_i|i\in I\})$.

□389

Заметим, что множество βγδ-трасс объединения LTS, вообще говоря, не равно объединению βγδ-моделей, являющихся множествами βγδ-трасс LTS-компонентов. Для того, чтобы это было так, нужно рассматривать расширенную βγδ-модель каждого компонента, то есть выполнить её замыкание по ***DRTIns***-операциям в объединённом алфавите (Объединение и пересечение βγδ-моделей. стр. 117). Если все LTS в одном алфавите, то множество βγδ-трасс объединения LTS равно объединению βγδ-трасс LTS-компонентов.

2.3.3. LTS-модель как βγδ-машина

Теперь мы покажем, что любая LTS-модель может рассматриваться как βγδ-машина (Рис.50). Для этого нам нужно определить правила работы LTS, помещённой внутрь «чёрного ящика» машины. Мы дадим такое описание в терминах LTS $\boldsymbol{L_\gamma 2L_{\beta\gamma\delta}(S)}$, которая однозначно строится из $\mathbf{S}$ (Определение 45:).

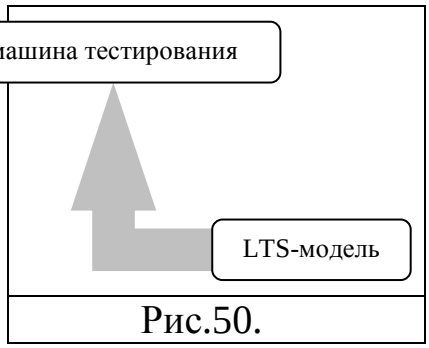


Рис.50.

Определение 54: Пусть $C \subseteq Z$, $\mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$. Правила работы LTS $\mathbf{M}=\boldsymbol{L_\gamma 2L_{\beta\gamma\delta}(S)}$ внутри «чёрного ящика» следующие:

LTS 1) В каждый момент времени $\mathbf{M}$ находится в некотором состоянии $m \in \boldsymbol{der}(\mathbf{M})$; в начальный момент времени $\mathbf{M}$ находится в начальном состоянии m_0.

LTS 2) Если $\mathbf{M}$ находится в нестабильном состоянии m и не нажата кнопка, $\mathbf{M}$ должна за конечное ограниченное время пройти любой (выбираемый недетерминированным образом) *конечный* маршрут $M \in \boldsymbol{runs}(m)$ и перейти в его конечное состояние $\omega(M)$, которое либо стабильно, либо является состоянием γ.[46]

LTS 3) Если $\mathbf{M}$ находится в состоянии m и нажата кнопка “?x” или “!”, $\mathbf{M}$ должна за конечное ограниченное время пройти любой (выбираемый недетерминированным образом) *конечный* маршрут $M \in \boldsymbol{runs}(m)$ и перейти в его конечное состояние $\omega(M)$, причём должно быть выполнено одно из трёх условий:

a) $\boldsymbol{trace}(M)=\langle ?x \rangle$ или, соответственно,
$\boldsymbol{trace}(M)=\langle !y \rangle$, где $!y \in C$;

b) $\boldsymbol{trace}(M)=\langle \gamma \rangle$;

c) $\boldsymbol{trace}(M)=\langle \{?x\} \rangle$ или, соответственно,
$\boldsymbol{trace}(M)=\langle \delta \rangle$.[46]

LTS 4) Когда $\mathbf{M}$ проходит переход $m \text{—}u\rightarrow m`$ для $u \in C_{\beta\gamma\delta}$ (то есть $u \neq \tau$), она выдаёт на печать символ u и отжимает нажатую кнопку (если такая была) при $u \neq \gamma$ (если $u=\gamma$, кнопка может как отжиматься, так и оставаться нажатой).

□

[46] Очевидно, что такой маршрут M всегда существует.

Вместо требования прохождения *конечного* маршрута в правилах LTS 2 и 3 можно было бы использовать модифицированное преобразование $L_\gamma 2L_{\beta\gamma\delta}$ без изменения множества трасс. Для этого достаточно в LTS $M=L_\gamma 2L_{\beta\gamma\delta}(S)$ выполнить следующие дополнительные преобразования:

1) каждый переход по стимулу или реакции, ведущий в дивергентное состояние m, превратить в «веер» переходов, ведущих во все дивергентные состояния, достижимые по τ-маршрутам из m;

2) удалить все τ-переходы, ведущие в дивергентные состояния.

После этого в LTS вообще не будет бесконечных τ-маршрутов и, тем самым, бесконечных маршрутов с конечной (в частности, одноэлементной) трассой.

Утверждение 35: Пусть $C \subseteq Z$, $S \in LTS_{\beta\gamma\delta}(C)$. Тогда машина, внутрь «чёрного ящика» которой помещена LTS $M=L_\gamma 2L_{\beta\gamma\delta}(S)$ с правилами работы, согласно Определение 54:, является βγδ-машиной.

□389

2.3.4. Эквивалентность βγδ-моделей и LTS-моделей

Структура раздела:

- LTS → βγδ-модель.
- βγδ-модель → LTS.
- βγδ-модель → компактная LTS.
- Утверждение об эквивалентности.

Для завершения «круга» эквивалентности (Рис.14) βγδ-машины, трассовой и LTS-моделей нам осталось показать, что для каждой трассовой модели существует LTS, имеющая эту модель в качестве множества своих βγδ-трасс (Рис.51). Тем не менее, нам кажется полезным доказать напрямую также и обратное утверждение: для каждой LTS-модели множество её βγδ-трасс, определяемое преобразованием $L_\gamma 2L_{\beta\gamma\delta}$, является трассовой моделью (Рис.52).

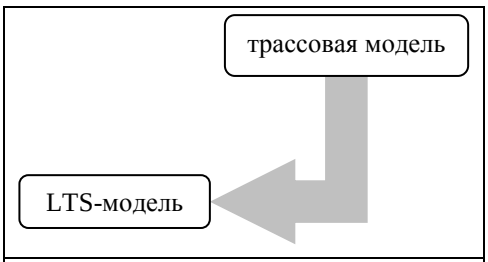


Рис.51.

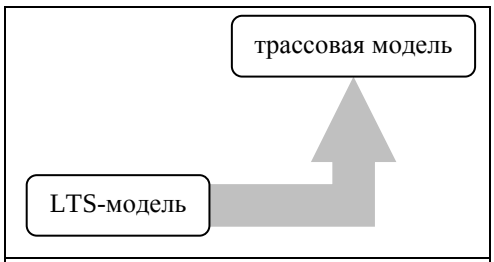


Рис.52.

LTS → βγδ-модель.

Утверждение 36: Пусть $C \subseteq Z$, $S \in LTS_{\beta\gamma\delta}(C)$ и $\Sigma = traces_{\beta\gamma\delta}(S)$.
Тогда $\Sigma \in MODEL_{\beta\gamma\delta}(C)$.

□391

βγδ-модель → LTS.

Теперь по βγδ-модели построим LTS, имеющую эту βγδ-модель в качестве множества своих βγδ-трасс. Состояниями такой LTS будут βγδ-трассы заданной βγδ-модели (Рис.53). τ-переход ведёт из βγδ-трассы μ, не заканчивающейся на отказ, в βγδ-трассу $\mu\cdot o$, заканчивающуюся на непустую трассу отказов o. Переход по стимулу из реакции z ведёт из любой βγδ-трассы μ в βγδ-трассу $\mu\cdot\langle z\rangle$. LTS строго-конвергентна и является LTS-деревом.

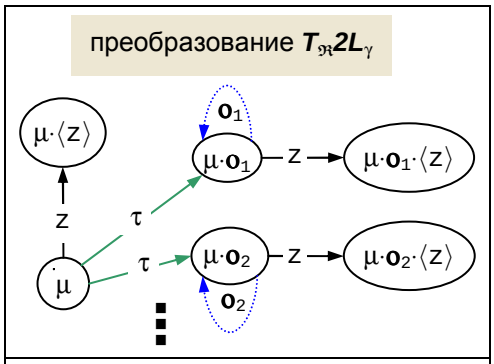


Рис.53.

Можно заметить, что это дерево конечно только тогда, когда βγδ-модель (как множество βγδ-трасс) конечна. А это может быть только в том случае, когда 1) алфавит стимулов конечен (по βγδ-конвергентности, βγδ-трасса продолжается каждым стимулом или его блокировкой), 2) βγδ-трассы модели не содержат βδ-отказов (по βγδ-замкнутости, они могут повторяться любое число раз), 3) нет бесконечно возрастающей цепочки трасс (βγδ-трасс без отказов)[47]. Дерево ограничено, если выполняются последние два условия.

Определение 55: Для $C \subseteq Z$ определим преобразование $T_{\beta\gamma\delta}2L_\gamma : MODEL_{\beta\gamma\delta}(C) \to LTS_{\beta\gamma\delta}(C)$.
Для $\Sigma \in MODEL_{\beta\gamma\delta}(C)$ $T_{\beta\gamma\delta}2L_\gamma(\Sigma) = S = LTS(\Sigma, C_\gamma, E_S, \epsilon)$, где E_S – наименьшее множество, порождаемое следующими правилами вывода: $\forall \mu \in \Sigma \ \forall z \in C_\gamma \ \forall o \in \beta\delta(C)^*$

(1) $\mu\cdot\langle z\rangle \in \Sigma \vdash \mu \text{—}z\to \mu\cdot\langle z\rangle$;

(2) $RefT(\mu) = \epsilon \ \&\ o \neq \epsilon \ \&\ \mu\cdot o \in \Sigma \vdash \mu \text{—}\tau\to \mu\cdot o$.

□

Сначала докажем два вспомогательных утверждения о LTS $T_{\beta\gamma\delta}2L_\gamma(\Sigma)$.

---

[47] Остановить возрастание цепочки βγδ-трасс может только разрушение.

Утверждение 37: О стабильных состояниях. Пусть $C \subseteq Z$, $\Sigma \in MODEL_{\beta\gamma\delta}(C)$ и $S = T_{\beta\gamma\delta}2L_{\gamma}(\Sigma)$. В LTS S состояние стабильно и порождает хотя бы один βδ-отказ тогда и только тогда, когда это βγδ-трасса, заканчивающаяся на разрушение, $\mu \cdot \langle\gamma\rangle \in \Sigma$ (терминальное состояние) или βγδ-трасса, заканчивающаяся на βδ-отказ, $\mu \cdot o \in \Sigma$, где $o \neq \epsilon$, $o \in \beta\delta(C)^*$. В последнем случае $init(\mu \cdot o) = head(\Sigma \ after \ \mu \cdot o) \cap C$.

□392

Утверждение 38: Пусть $C \subseteq Z$, $\Sigma \in MODEL_{\beta\gamma\delta}(C)$ и $S = T_{\beta\gamma\delta}2L_{\gamma}(\Sigma)$. Тогда для каждого маршрута $S \in runs(S)$ его βγδ-трассы $traces_{\beta\gamma\delta}(S) = DRTIns(\omega(S))$.

□392

Теперь докажем основное утверждение о LTS $T_{\beta\gamma\delta}2L_{\gamma}(\Sigma)$.

Утверждение 39: Пусть $C \subseteq Z$, $\Sigma \in MODEL_{\beta\gamma\delta}(C)$ и $S = T_{\beta\gamma\delta}2L_{\gamma}(\Sigma)$. Тогда $S\Downarrow$ и $\Sigma = traces_{\beta\gamma\delta}(S)$.

□394

βγδ-модель → компактная LTS.

Преобразование $T_{\beta\gamma\delta}2L_{\gamma}$ создаёт очень «некомпактную» LTS. Вместо этого можно делать преобразование, используя разложение βγδ-модели на стабильные и контрстабильные поддеревья.

В Утверждение 7: показано, что каждая βγδ-модель Σ разлагается в объединение наибольшего контрстабильного поддерева T_C и множества T_S стабильных поддеревьев. Мы будем строить LTS, в которой начальное состояние соответствует всей βγδ-модели Σ (Рис.54). Если $T_S \neq \varnothing$, начальное состояние нестабильно: мы проводим из него τ-переходы в стабильные состояния, соответствующие всем стабильным поддеревьям из T_S. Если $T_S = \varnothing$, начальное состояние будет стабильно. В любом случае, если хотя бы одна βγδ-трасса из Σ начинается с некоторого стимула или реакции $z \in C$, проводим из начального состояния переход по z в состояние, соответствующее βγδ-модели $\Sigma \ after \ \langle z \rangle$. Если есть трасса $\langle\gamma\rangle$ также делаем переход в $\Sigma \ after \ \langle\gamma\rangle = \{\epsilon\}$.[48] Совокупность таких переходов и продолжающих их βγδ-моделей соответствует наибольшему контрстабильному поддереву

[48] Напомним, что множество $\{\epsilon\}$ не является βγδ-моделью (нет βγδ-неконвергентности).

$$\mathbf{T}_c = \{\ \{\langle z\rangle\}\cdot(\mathbf{T}_c\ \textit{after}\ \langle z\rangle)\ |\ z\in \textit{head}(\mathbf{T}_c)\ \ \}$$
$$= \{\ \{\langle z\rangle\}\cdot(\mathbf{\Sigma}\ \textit{after}\ \langle z\rangle)\ |\ z\in \textit{head}(\mathbf{\Sigma})\cap C_\gamma\}.$$

Разлагая модель на каждом шаге, мы получаем искомую LTS.

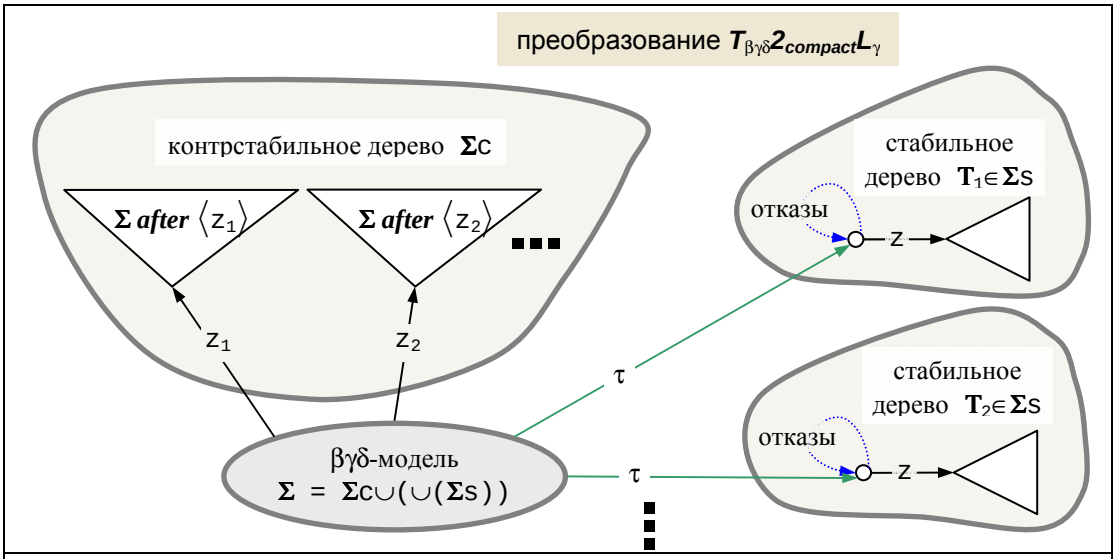


Рис.54.

Заметим, что такое преобразование создаёт максимально «компактную» LTS. В частности, если для данной βγδ-модели существует конечная LTS с таким же множеством βγδ-трасс, то таким преобразованием будет получена конечная LTS. Формально состояниями такой LTS будут все возможные βγδ-модели в базовом алфавите. Однако, множество достижимых состояний «минимально», в частности, конечно.

Определение 56: Для $C\subseteq Z$ определим преобразование $\boldsymbol{T}_{\beta\gamma\delta}\boldsymbol{2}_{\textit{compact}}\boldsymbol{L}_\gamma : \textit{MODEL}_{\beta\gamma\delta}(C)\to \textit{LTS}_{\beta\gamma\delta}(C)$. Для $\mathbf{\Sigma}\in \textit{MODEL}_{\beta\gamma\delta}(C)$ определим $\boldsymbol{T}_{\beta\gamma\delta}\boldsymbol{2}_{\textit{compact}}\boldsymbol{L}_\gamma(\mathbf{\Sigma})=\mathbf{S}=\mathrm{LTS}(V_S, C_\gamma, E_S, \mathbf{\Sigma})$, где $V_S=\{\epsilon\}\cup \textit{MODEL}_{\beta\gamma\delta}(C)$, а E_S – наименьшее множество, порождаемое следующими правилами вывода: $\forall \mathbf{T}\in \textit{MODEL}_{\beta\gamma\delta}(C)$

(1) $z\in \textit{head}(\mathbf{T})\cap C_\gamma \quad \vdash\ \mathbf{T}\text{—}z\to\mathbf{T}\ \textit{after}\ \langle z\rangle$,

(2) $\mathbf{T}\ \textit{unstable}\ \&\ \mathbf{T}=\mathbf{T}_c\cup(\cup(\mathbf{T}_s))\ \&\ \mathbf{T}`\in\mathbf{T}_s \quad \vdash\ \mathbf{T}\text{—}\tau\to\mathbf{T}`$,

где $\mathbf{T}=\mathbf{T}_c\cup(\cup(\mathbf{T}_s))$ – разложение βγδ-модели по Утверждение 7:.

□

Сначала докажем вспомогательное утверждение о стабильных состояниях LTS $\boldsymbol{T}_{\beta\gamma\delta}\boldsymbol{2}_{\textit{compact}}\boldsymbol{L}_\gamma(\mathbf{\Sigma})$.

Утверждение 40: О стабильных состояниях. Пусть $C\subseteq Z$, $\mathbf{\Sigma}\in \textit{MODEL}_{\beta\gamma\delta}(C)$ и $\mathbf{S}=\boldsymbol{T}_{\beta\gamma\delta}\boldsymbol{2}_{\textit{compact}}\boldsymbol{L}_\gamma(\mathbf{\Sigma})$. В LTS $\mathbf{S}$ состояние стабильно

тогда и только тогда, когда это терминальное состояние $\{\epsilon\}$ или стабильное дерево $\mathbf{T}$. В последнем случае $\boldsymbol{init}(\mathbf{T})=\boldsymbol{head}(\mathbf{T})\cap C$.

□395

Теперь докажем основное утверждение о LTS $\boldsymbol{T}_{\beta\gamma\delta}\boldsymbol{2}_{\boldsymbol{compact}}\boldsymbol{L}_{\gamma}(\boldsymbol{\Sigma})$.

Утверждение 41: Пусть $C\subseteq Z$, $\boldsymbol{\Sigma}\in\boldsymbol{MODEL}_{\beta\gamma\delta}(C)$ и $\mathbf{S}=\boldsymbol{T}_{\beta\gamma\delta}\boldsymbol{2}_{\boldsymbol{compact}}\boldsymbol{L}_{\gamma}(\boldsymbol{\Sigma})$. Тогда $\mathbf{S}\Downarrow$ и $\boldsymbol{\Sigma}=\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$.

□396

Утверждение об эквивалентности.

Утверждение 36: вместе с Утверждение 39: или Утверждение 41: дают следующее утверждение об эквивалентности моделей.

Утверждение 42: Для каждого $C\subseteq Z$ βγδ- и LTS-модели в алфавите C эквивалентны: $\boldsymbol{traces}_{\beta\gamma\delta}\circ\boldsymbol{LTS}_{\beta\gamma\delta}(C) = \boldsymbol{MODEL}_{\beta\gamma\delta}(C)$. Более того, βγδ-модели эквивалентны строго-конвергентным LTS-моделям: $\boldsymbol{traces}_{\beta\gamma\delta}(\{\mathbf{S}\in\boldsymbol{LTS}_{\beta\gamma\delta}(C)\mid\mathbf{S}\Downarrow\}) = \boldsymbol{MODEL}_{\beta\gamma\delta}(C)$.

□399

Замечание: Одной βγδ-модели соответствует, вообще говоря, множество βγδ-эквивалентных (имеющих то же множество βγδ-трасс) LTS-моделей, среди которых всегда есть строго-конвергентная (Рис.55).

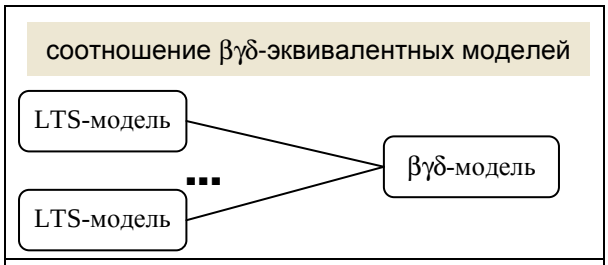


Рис.55.

2.3.5. Гипотеза о безопасности и соответствие $\boldsymbol{ioco}_{\beta\gamma\delta}$

В силу эквивалентности βγδ- и LTS-моделей мы можем распространить отношения ***safe for*** и $\boldsymbol{ioco}_{\beta\gamma\delta}$, определённые для βγδ-модели, на LTS-модель.

Определение 57: **Гипотеза о безопасности.** Для $C\subseteq Z$, $\mathbf{I},\mathbf{S}\in\boldsymbol{LTS}_{\beta\gamma\delta}(C)$

- реализация $\mathbf{I}$ *S*-тестируемая для спецификации $\mathbf{S}$:
  $\mathbf{I}$ ***safe for*** $\mathbf{S}$ $=_{\text{def}}$ $\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{I})$ ***safe for*** $\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$;
- множество *S*-тестируемых реализаций для спецификации $\mathbf{S}$:
  $\boldsymbol{safeIMPL}(\mathbf{S})$ $=_{\text{def}}$ $\{\mathbf{I}\in\boldsymbol{LTS}_{\beta\gamma\delta}(C)\mid\mathbf{I}$ ***safe for*** $\mathbf{S}\}$.

□

Условие безопасности можно записать также в виде:

$\mathbf{I}$ ***safe for*** $\mathbf{S}$ $= tt_{\circ}traces_{\beta\gamma\delta}(\mathbf{S}) \cap traces_{\beta\gamma\delta}(\mathbf{I}) \subseteq tt_{\circ}traces_{\beta\gamma\delta}(\mathbf{I})$
$= tt_{\beta\gamma\delta}(\mathbf{S}) \cap traces_{\beta\gamma\delta}(\mathbf{I}) \subseteq tt_{\beta\gamma\delta}(\mathbf{I})$.

Определение 58: **Соответствие** $ioco_{\beta\gamma\delta}$. Для $C \subseteq Z$, $\mathbf{I}, \mathbf{S} \in LTS_{\beta\gamma\delta}(C)$
- $\mathbf{I}\ ioco_{\beta\gamma\delta}\ \mathbf{S} =_{def} traces_{\beta\gamma\delta}(\mathbf{I})\ ioco_{\beta\gamma\delta}\ traces_{\beta\gamma\delta}(\mathbf{S})$.
- $confIMPL(\mathbf{S}) =_{def} \{\mathbf{I} \in LTS_{\beta\gamma\delta}(C) \mid \mathbf{I}\ ioco_{\beta\gamma\delta}\ \mathbf{S}\}$.

□

В определение $ioco_{\beta\gamma\delta}$ для βγδ-моделей, кроме условия ***safe for***, входит условие, которое может быть записано в трёх эквивалентных формах:
- $\forall \sigma \in safe(\mathbf{\Sigma})\ \forall u \in safesymbols(\mathbf{\Sigma}\ after\ \sigma)\ \sigma \cdot \langle u \rangle \in \mathbf{I} \Rightarrow \sigma \cdot \langle u \rangle \in \mathbf{\Sigma}$,
- $\forall \sigma \in safe(\mathbf{\Sigma})$
  $head(\mathbf{I}\ after\ \sigma) \cap safesymbols(\mathbf{\Sigma}\ after\ \sigma) \subseteq head(\mathbf{\Sigma}\ after\ \sigma)$,
- $tt(\mathbf{\Sigma}) \cap \mathbf{I} \subseteq \mathbf{\Sigma}$.

Соответствующее условие для LTS-моделей также запишем в аналогичных трёх формах:
- $\forall \sigma \in safe_{\beta\gamma\delta}(\mathbf{S})\ \forall u \in \cap_{\circ} safesymbols_{\beta\gamma\delta}(\mathbf{S}\ after\ \sigma)$
  $u \in \cup_{\circ} head_{\beta\gamma\delta}(\mathbf{I}\ after\ \sigma) \Rightarrow u \in \cup_{\circ} head_{\beta\gamma\delta}(\mathbf{S}\ after\ \sigma)$,
- $\forall \sigma \in safe_{\beta\gamma\delta}(\mathbf{S})$
  $(\cup_{\circ} head_{\beta\gamma\delta}(\mathbf{I}\ after\ \sigma)) \cap (\cap_{\circ} safesymbols_{\beta\gamma\delta}(\mathbf{S}\ after\ \sigma))$
  $\subseteq \cup_{\circ} head_{\beta\gamma\delta}(\mathbf{S}\ after\ \sigma)$,
- $tt_{\beta\gamma\delta}(\mathbf{S}) \cap traces_{\beta\gamma\delta}(\mathbf{I}) \subseteq traces_{\beta\gamma\delta}(\mathbf{S})$.

Полное условие конформности для βγδ-моделей было записано в виде:
$tt(\mathbf{\Sigma}) \cap \mathbf{I} \subseteq \mathbf{\Sigma} \cap tt(\mathbf{I})$.

Соответственно, для LTS-моделей полное условие конформности имеет вид:
$tt_{\beta\gamma\delta}(\mathbf{S}) \cap traces_{\beta\gamma\delta}(\mathbf{I}) \subseteq traces_{\beta\gamma\delta}(\mathbf{S}) \cap tt_{\beta\gamma\delta}(\mathbf{I})$.

Определение 59: LTS-модели $\mathbf{S}$ и $\mathbf{S}`$ $ioco_{\beta\gamma\delta}$-эквивалентны, если $ioco_{\beta\gamma\delta}$-эквивалентны множества их βγδ-трасс:
$\mathbf{S} \sim_{ioco_{\beta\gamma\delta}} \mathbf{S}` =_{def} traces_{\beta\gamma\delta}(\mathbf{S}) \sim_{ioco_{\beta\gamma\delta}} traces_{\beta\gamma\delta}(\mathbf{S}`)$.

□

2.3.6. Композиция LTS и тесты

Структура раздела:
- Параллельная композиция LTS.
- LTS-тесты.

Для LTS-модели мы определили *S*-тестируемые (удовлетворяющие гипотезе о безопасности) реализации и отношение $\boldsymbol{ioco}_{\beta\gamma\delta}$, используя эквивалентность βγδ- и LTS-моделей и соответствующие определения βγδ-теории. Тестирование для LTS традиционно определяется через оператор композиции LTS-реализации и LTS-теста, который строит новую, композиционную LTS, состояния которой – это пара состояний теста и реализации. В тестовой LTS есть только два терминальных состояния ***pass*** и ***fail*** (или одно из них). LTS-реализация *проходит* LTS-тест, если недостижимо композиционное состояние $(\mathtt{i},\boldsymbol{fail})$, где $\mathtt{i}$ соответствующее состояние реализации.

Используя LTS-отношение ***passes***, можно было бы определить значимые, исчерпывающие и полные наборы LTS-тестов аналогично тому, как это делается в βγδ-теории, используя отношение ***passes*** для βγδ-моделей. Вместо этого мы построим LTS-тест $\boldsymbol{LTest}(\mathbf{T})$ из тестового βγδ-дерева $\mathbf{T}\in\boldsymbol{TEST}\circ\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$. С одной стороны, мы имеем βγδ-отношение ***passes*** между βγδ-реализацией $\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{I})$ и тестовым деревом $\mathbf{T}$. С другой стороны, мы имеем LTS-отношение ***passes*** между LTS-реализацией $\mathbf{I}$ и LTS-тестом $\boldsymbol{LTest}(\mathbf{T})$. Мы покажем, что эти отношения эквивалентны (Рис.56). Тем самым, будет показано, что множество $\boldsymbol{LTest}(\mathbf{TT})$ LTS-тестов $\boldsymbol{LTest}(\mathbf{T})$, где $\mathbf{T}$ пробегает полный тестовый βγδ-набор $\mathbf{TT}\subseteq\boldsymbol{TEST}\circ\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$ для βγδ-спецификации $\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$, является полным тестовым LTS-набором для LTS-спецификации $\mathbf{S}$.

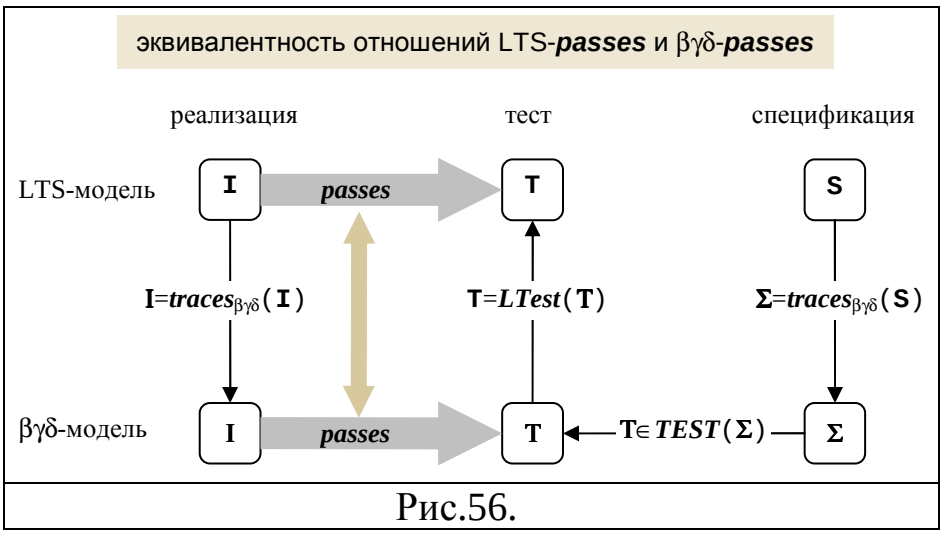


Рис.56.

Параллельная композиция LTS.

Определение 60: Определим биекцию, ставящую в соответствие каждому внешнему действию *противоположное* ему действие (стимулу – реакцию, реакции – стимул):

$\underline{\ .\ }: Z \to Z \ : \ \forall w \in W \ \underline{?w} = !w \ \& \ \underline{!w} = ?w.$

Мы распространим эту биекцию на отказы и разрушение:

$\forall u \in Z_{\beta\gamma\delta} \setminus Z \ \underline{u} = u.$

□

Параллельное выполнение двух LTS в алфавитах A и B понимается так, что переходы по противоположным действиям $?w$ и $!w$, где $?w \in A$ и $!w \in B$ или, наоборот, $!w \in A$ и $?w \in B$, выполняются синхронно, то есть, в обеих LTS одновременно, причём в композиции это становится τ-переходом. Остальные переходы выполняются асинхронно, то есть, в одной из LTS при сохранении состояния другой LTS. Это соответствует определению параллельной композиции в CCS – Calculus of Communicating Systems [Miln89] (также [JJTV99]).

Заметим, что при параллельной композиции в CSP – Communicating Sequential Processes [Hoare85] требуется дополнительное применение оператора ***Hide*** [BRT03], который превращает композиционные переходы, соответствующие синхронной паре переходов в компонентах, в τ-переходы.

Кроме этого, мы вводим новый переход по разрушению γ, который при параллельной композиции происходит асинхронно – так же как и τ-переход.

При тестировании отказ реализуется в тестере с помощью тайм-аута. Блокировка стимула $\{?x\}$ – это истечение тайм-аута при попытке передачи стимула из тестера в реализацию, которая находится в стабильном состоянии, и в этом состоянии отсутствует переход $?x$ по этому стимулу. Стационарность δ – это истечение тайм-аута при попытке принять в тестер любую реакцию из реализации, которая находится в стабильном состоянии, и в этом состоянии реализации отсутствуют переходы $!y$ по всем реакциям. Таким образом, отказу соответствует переход по тайм-ауту в LTS-тестере. Такой переход называют θ-переходом и маркируют символом $\theta \notin Z_\gamma$ (см., например, [Tret96]). В общем, θ-переход предназначен для разрешения тупиков и происходит тогда, когда никакие другие переходы невозможны.

Мы не будем вводить двух обозначений для оператора параллельной композиции LTS без θ-переходов и с θ-переходами (как это делает Treetmans в [Tret96], используя обозначения $\|$ и $\urcorner\|$), считая, что мы просто расширяем оператор параллельной композиции для LTS с θ-переходами.

Определение 61: Для алфавитов-подмножеств Z определим композицию алфавитов $.\urcorner\lfloor.: \wp(Z) \times \wp(Z) \to \wp(Z)$ следующим образом: $\forall A, B \subseteq Z \ A \urcorner\lfloor B =_{def} (A \setminus \underline{B}) \cup (B \setminus \underline{A})$.

Пересечения $A\cap\underline{B}$ и $B\cap\underline{A}$ будем называть синхронными подалфавитами алфавита A и алфавита B, соответственно. Элементы синхронного подалфавита будем называть синхронными символами.
Аналогично, разности $A\setminus\underline{B}$ и $B\setminus\underline{A}$ будем называть асинхронными подалфавитами алфавита A и алфавита B, соответственно. Элементы асинхронного подалфавита будем называть асинхронными символами.

□

Очевидно, композиция алфавитов является коммутативной операцией.

Определение 62: Пусть заданы алфавиты $A,B\subseteq Z$.
Определим композицию $.\!\upharpoonright\!\downharpoonright.:\boldsymbol{LTS}(A_{\gamma\theta})\times\boldsymbol{LTS}(B_{\gamma\theta})\to\boldsymbol{LTS}(C_{\gamma\theta})$, где $C=A\!\upharpoonright\!\downharpoonright B$, следующим образом: $\forall\mathbf{I}\in\boldsymbol{LTS}(A_{\gamma\theta})\ \forall\mathbf{T}\in\boldsymbol{LTS}(B_{\gamma\theta})$ $\mathbf{I}\!\upharpoonright\!\downharpoonright\mathbf{T}=_{\text{def}}\mathrm{LTS}(V_{\mathbf{I}}\times V_{\mathbf{T}},C_{\gamma\theta},E,i_0t_0)$, где E – наименьшее множество, порождаемое следующими правилами вывода: $\forall i\in V_{\mathbf{I}}\ \forall t\in V_{\mathbf{T}}$

$(\!\upharpoonright\!\downharpoonright 1)\quad z\in A_{\gamma\tau}\setminus\underline{B}\ \&\ i\text{—}z\to i` \quad\vdash it\text{—}z\to i`t;$

$(\!\upharpoonright\!\downharpoonright 2)\quad z\in B_{\gamma\tau}\setminus\underline{A}\ \&\ t\text{—}z\to t` \quad\vdash it\text{—}z\to it`;$

$(\!\upharpoonright\!\downharpoonright 3)\quad z\in A\cap\underline{B}\ \&\ i\text{—}z\to i`\ \&\ t\text{—}\underline{z}\to t` \quad\vdash it\text{—}\tau\to i`t`;$

$(\!\upharpoonright\!\downharpoonright 4)\quad i\text{—}\theta\to i`\ \&\ i\text{—}\tau\gamma\not\to\ \&\ t\text{—}\tau\gamma\not\to$
$\&\ (\forall z\in A\cap\underline{B}\ \ i\text{—}z\not\to\ \vee\ t\text{—}\underline{z}\not\to) \quad\vdash it\text{—}\theta\to i`t;$

$(\!\upharpoonright\!\downharpoonright 5)\quad t\text{—}\theta\to t`\ \&\ i\text{—}\tau\gamma\not\to\ \&\ t\text{—}\tau\gamma\not\to$
$\&\ (\forall z\in A\cap\underline{B}\ \ i\text{—}z\not\to\ \vee\ t\text{—}\underline{z}\not\to) \quad\vdash it\text{—}\theta\to it`.$

□

Очевидно, композиция LTS является коммутативной операцией (с точностью до изоморфизма, т.е. именования состояний it или ti).

LTS-тесты.

Для спецификации и реализации как LTS-моделей, то есть LTS в алфавите $C_\gamma\subseteq Z_\gamma$ тестер будет моделироваться LTS в противоположном алфавите $\underline{C}_\theta$. Поскольку мы рассматриваем реализации без приоритетов, но с разрушением, в LTS-реализации нет θ-переходов, но могут быть γ-переходы. В LTS-тесте, наоборот, могут быть θ-переходы, но нет γ-переходов (тест не должен разрушаться). Формально, для композиции в Определение 62: реализация рассматривается в надалфавите $C_{\gamma\theta}$, но по-прежнему не имеет θ-переходов, а тест рассматривается в надалфавите $\underline{C}_{\gamma\theta}$, но по-прежнему не имеет γ-переходов.

В LTS-модели синхронное взаимодействие тестера и реализации означает прохождение некоторого маршрута с τθ-трассой композиции LTS-реализации и LTS-теста. При этом тест проходит маршрут с $\underline{C}_\theta$ трассой, а реализация – маршрут с C_γ трассой. При безопасном тестировании трасса реализации не должна содержать (заканчиваться) разрушения, то есть должна быть трассой в алфавите C. В терминах βγδ-маршрутов реализации мы можем считать, при прохождении тестом θ-перехода реализация проходит βδ-отказ-петлю. Тем самым, можно считать, что реализация проходит βγδ-маршрут с βγδ-трассой σ, то есть трассу в алфавите $C_{\beta\gamma\delta}$ ($C_{\beta\delta}$ при безопасном тестировании).

Базовый алфавит композиции реализации и теста пуст: $C \rfloor\!\lfloor \underline{C} = \varnothing$. Поэтому возможны лишь τ-, γ- и θ-переходы. Если реализация удовлетворяет гипотезе о безопасности, то в композиции γ-переходы недостижимы. Выполнение такой композиции – это прохождение любой её τθ-трассы. При этом τ-переход выполняется за время, ограниченное тайм-аутом, и соответствует синхронному переходу или асинхронному τ-переходу реализации, а θ-переход выполняется за время тайм-аута и соответствует θ-переходу теста.

θ-переход в стабильном[49] состоянии теста t интерпретируется как отказ A, где $A=\underline{\textit{init}(t)\backslash\{\theta\}}$. В данной работе мы не рассматриваем отказы общего вида, ограничиваясь только βδ-отказами: блокировками стимулов и стационарностью. Поэтому в состоянии t, кроме θ-перехода, может быть определён a) только переход по выдаче одного реализационного стимула $?x$ ($\textit{init}(t)\backslash\{\theta\}=\{!x\}$) или b) только переходы по приёму всех реализационных реакций ($\textit{init}(t)\backslash\{\theta\}=\underline{!C}$). Это соответствует управляемости (***controllable***) трассовых тестов. θ-переход интерпретируется как наблюдение a) блокировки $\{?x\}$ или b) стационарности δ. Тогда τθ-трасса композиции LTS-реализации и LTS-теста соответствует некоторой βγδ-трассе σ реализации и противоположной βγδ-трассе $\underline{\sigma}$ теста, преобразованного в соответствии с этой интерпретацией (замена θ на соответствующий βδ-отказ).

Для LTS-модели мы определим тестовый пример и тестовый набор, используя преобразование LTS $\mathbf{S}$ в дерево её βγδ-трасс $\mathbf{\Sigma}$ и соответствующие определения βγδ-теории ($\textit{TEST}(\mathbf{\Sigma})$). По трассовому тесту $\mathbf{T}$ мы будем строить LTS-тест $\textit{LTest}(\mathbf{T})$. Мы будем использовать

[49] В нестабильном состоянии θ-переход никогда не выполняется.

преобразование теста $L_\theta 2L_{\beta\delta}$, совмещающее переопределение θ-переходов и последующую замену символов на противоположные. Преобразованная тестовая LTS «проходит» ту же самую βγδ-трассу σ, что реализация и трассовый тест: $traces \circ L_\theta 2L_{\beta\delta} \circ LTest(T) = T$.

Дополнительно мы потребуем детерминированности тестовой LTS. В такой LTS нет τ-переходов и каждой трассе σ соответствует только один маршрут T_σ такой, что $trace(T_\sigma)=\sigma$. Поскольку трассовый тест – это нётерово дерево трасс без разрушения, в детерминированном LTS-тесте все маршруты конечны[50], что соответствует конечности времени выполнения теста. Такой LTS-тест – это ациклический граф. Трасса σ максимальна тогда и только тогда, когда маршрут заканчивается в терминальном состоянии $\omega(T_\sigma)\in\{pass, fail\}$. В тестовой LTS вердикт определяется терминальным состоянием, что для детерминированной LTS эквивалентно вердикту на максимальных трассах. LTS-реализация *проходит* LTS-тест, если в любом достижимом композиционном состоянии it с терминальным состоянием t теста это состояние t=*pass*.

Композиция такого LTS-теста с LTS-реализацией, удовлетворяющей гипотезе о безопасности, – это ациклический граф с конечными маршрутами, все достижимые переходы которого – это τ- и θ-переходы, а все достижимые терминальные состояния имеют вид i*fail* или i*pass*. Выполнение такой композиционной LTS завершается за конечное время.

Определение 63: Для $C\subseteq Z$, $S\in LTS_{\beta\gamma\delta}(C)$, $T\in LTS(\underline{C}_\theta)$:

· Определим преобразование $L_\theta 2L_{\beta\delta}: LTS(\underline{C}_\theta)\to LTS(C_{\beta\delta})$.
Для $T=LTS(V_T, \underline{C}_\theta, E_T, t_0)$ $L_\theta 2L_{\beta\delta}(T)=T1=LTS(V_T, C_{\beta\delta}, E_{T1}, t_0)$, где E_{T1} – наименьшее множество, порождаемое следующими правилами вывода: $\forall t,t`\in der(T)$

$\underline{z}\in\underline{C}_\tau \quad \& \; t\text{—}\underline{z}\to t` \quad \vdash t\text{—}z\to t`;$

$\underline{!C}\subseteq init(t) \quad \& \; t\text{—}\theta\to t` \quad \vdash t\text{—}\delta\to t`;$

$\underline{?x}\in init(t) \quad \& \; t\text{—}\theta\to t` \quad \vdash t\text{—}\{?x\}\to t`.$

□

Определение 64: Для $C\subseteq Z$, $S\in LTS_{\beta\gamma\delta}(C)$, $T\in LTS(\underline{C}_\theta)$:

· *Тестовым примером (test case)* будем называть детерминированную LTS **T**, в которой не более двух терминальных состояний ***pass*** и ***fail***,

---

[50] Бесконечный маршрут имел бы либо бесконечную трассу, что противоречит нётеровости, либо бесконечный постфикс τ-переходов.

traces$\circ L_\theta 2L_{\beta\delta}$(**T**)$\in$***TEST***$\circ$***traces***$_{\beta\gamma\delta}$(**S**), и для любого маршрута $T_\sigma \in$ ***runs***$\circ L_\theta 2L_{\beta\delta}$(**T**), заканчивающегося в терминальном состоянии и имеющего $\beta\gamma\delta$-трассу σ=***trace***(T_σ), ***verdict***$(\sigma)=\omega(T_\sigma)$.

- Множество всех тестовых примеров обозначим ***TEST***(**S**).
- *Тестовый набор (test suite)* – это множество тестовых примеров: **TT**$\subseteq$***TEST***(**S**).
- Для **I**$\in$***LTS***$_{\beta\gamma\delta}$(C), **T**$\in$***TEST***(**S**)

  I ***passes*** **T** $=_{\text{def}}$ в LTS **I**$\wr$**T** $\forall it \in$ ***der***$(i_0 t_0)$ $t \neq$ ***fail***.

□

Определим преобразование ***LTest*** $\beta\gamma\delta$-теста в LTS-тест.

Определение 65: Для C$\subseteq$Z, спецификации **S**$\in$***LTS***$_{\beta\gamma\delta}$(C) и $\beta\gamma\delta$-теста **T**$\in$***TEST***$\circ$***traces***$_{\beta\gamma\delta}$(**S**) определим тестовую LTS ***LTest***(**T**)=**T**=LTS$(V_T, \underline{C}_\theta, E_T, t_0) \in$ ***TEST***(**S**), в которой состояния – это немаксимальные трассы $\beta\gamma\delta$-теста и значения вердикта V_T=(**T*****max**(**T**))$\cup\{$***pass***,***fail***$\}$, $t_0=\epsilon$, а E_T – наименьшее множество, порождаемое следующими правилами вывода: $\forall\sigma\in$**T*****max**(**T**) $\forall ?x \in C$ $\forall !y \in C$

σ ***send in*** **T** & $\sigma\cdot\langle ?x\rangle \in$ **T*****max**(**T**) $\vdash \sigma$—!x→$\sigma\cdot\langle ?x\rangle$;

σ ***send in*** **T** & $\sigma\cdot\langle ?x\rangle \in$ ***max***(**T**) $\vdash \sigma$—!x→***verdict***$(\sigma\cdot\langle ?x\rangle)$;

σ ***send in*** **T** & $\sigma\cdot\langle \{?x\}\rangle \in$ **T*****max**(**T**) $\vdash \sigma$—θ→$\sigma\cdot\langle \{?x\}\rangle$;

σ ***send in*** **T** & $\sigma\cdot\langle \{?x\}\rangle \in$ ***max***(**T**) $\vdash \sigma$—θ→***verdict***$(\sigma\cdot\langle \{?x\}\rangle)$;

σ ***wait in*** **T** & $\sigma\cdot\langle !y\rangle \notin$ ***max***(**T**) $\vdash \sigma$—?y→$\sigma\cdot\langle !y\rangle$;

σ ***wait in*** **T** & $\sigma\cdot\langle !y\rangle \in$ ***max***(**T**) $\vdash \sigma$—?y→***verdict***$(\sigma\cdot\langle !y\rangle)$;

σ ***wait in*** **T** & $\sigma\cdot\langle \delta\rangle \notin$ ***max***(**T**) $\vdash \sigma$—θ→$\sigma\cdot\langle \delta\rangle$;

σ ***wait in*** **T** & $\sigma\cdot\langle \delta\rangle \in$ ***max***(**T**) $\vdash \sigma$—θ→***verdict***$(\sigma\cdot\langle \delta\rangle)$.

□

Утверждение 43: Пусть C$\subseteq$Z.

1) $\forall$**S**$\in$***LTS***$_{\beta\gamma\delta}$(C) $\forall$**T**$\in$***TEST***$\circ$***traces***$_{\beta\gamma\delta}$(**S**)
   traces$\circ L_\theta 2L_{\beta\delta}\circ$***LTest***(**T**) = **T**.
2) ***TEST***$\circ$***traces***$_{\beta\gamma\delta}$(**S**) = ***traces***$\circ L_\theta 2L_{\beta\delta}\circ$***TEST***(**S**).

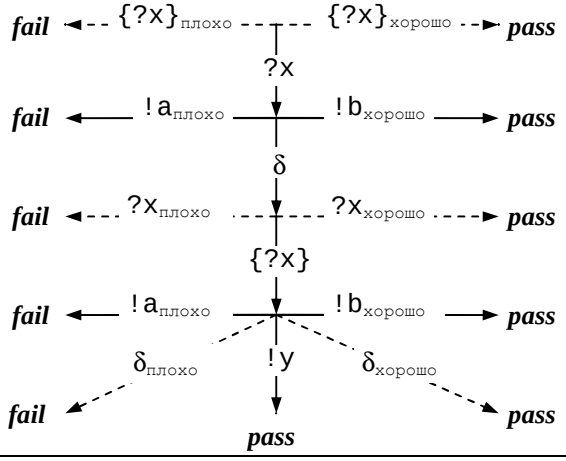
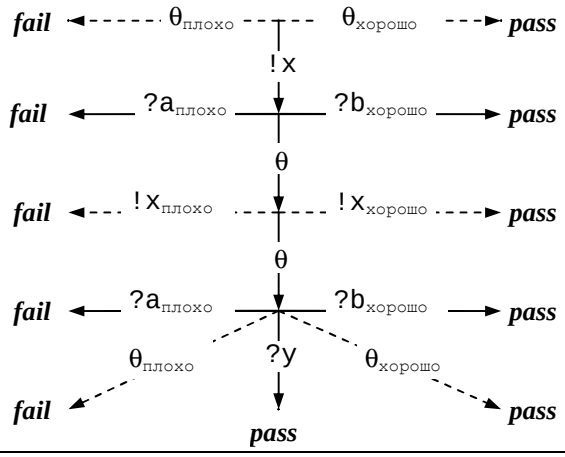
□399

Утверждение 44: Пусть C$\subseteq$Z, LTS-спецификация **S**$\in$***LTS***$_{\beta\gamma\delta}$(C), реализация **I**$\in$***safeIMPL***(**S**) и $\beta\gamma\delta$-тест **T**$\in$***TEST***$\circ$***traces***$_{\beta\gamma\delta}$(**S**). Тогда **I** ***passes*** ***LTest***(**T**) $\Leftrightarrow$ ***traces***$_{\beta\gamma\delta}$(**I**) ***passes*** **T**.

□399

В LTS-тесте, построенном по трассовому тесту-«ёршику», все маршруты имеют ограниченную длину, что соответствует ограниченности времени выполнения теста. На Рис.57 показан пример генерации LTS-теста по βγδ-тесту-«ёршику», который строится по одной безопасной βγδ-трассе спецификации.

генерация LTS-«ёршика» по βγδ-«ёршику» ***Test***(σ) для безопасной трассы $\sigma=\langle ?x, \delta, \{?x\}, !y \rangle$

порождающий граф βγδ-«ёршика»	LTS-«ёршик»
	

Символ снабжён индексом «$_{\text{хорошо}}$», если предшествующая трасса продолжается этим символом в спецификации; иначе – индексом «$_{\text{плохо}}$».

Пунктиром показаны взаимоисключающие продолжения, ведущие из одной вершины порождающего графа βγδ-теста, и взаимоисключающие переходы из одного состояния LTS-теста.

Рис.57.

2.3.7. Алгоритмизация

Структура раздела:

- Работа βγδ- и LTS-тестов.
- Конечное время выполнения теста.
- *S*-ветвящиеся LTS.
- Два способа задания LTS-модели (способы L1 и L2).
- Переходы только по безопасным стимулам и реакциям (способ L1).
- Переходы по всем стимулам и реакциям, и по разрушению (способ L2).
- Сравнение способов задания LTS.

Работа βγδ- и LTS-тестов.

Напомним, как работает βγδ-тест, построенный на основе полууправляемого поддерева безопасных трасс.

Пусть $C \subseteq Z$, $\Sigma \in MODEL_{\beta\gamma\delta}(C)$, а A нётерово полууправляемое поддерево безопасных трасс. Тест $T = Test(A)$ работает следующим образом. При получении символа u после безопасной трассы $\sigma \in A$ проверяется $\sigma \cdot \langle u \rangle \in safe(\Sigma)$. Если это не так, выдаётся вердикт ***fail***. В противном случае выдаётся вердикт ***pass***, если $\sigma \cdot \langle u \rangle \in max(T)$, или нажимается кнопка машины тестирования, определяемая немаксимальной трассой $\sigma \cdot \langle u \rangle$.

Соответственно, для LTS-спецификации $S \in LTS_{\beta\gamma\delta}(C)$ и $\Sigma = traces_{\beta\gamma\delta}(S)$ LTS-тест $T = LTest(T)$ работает следующим образом. После получения символа u в состоянии теста σ проверяется $\sigma \cdot \langle u \rangle \in safe_{\beta\gamma\delta}(S)$. Если это не так, выдаётся вердикт ***fail*** (переход в ***fail***-состояние). В противном случае выдаётся вердикт ***pass*** (переход в ***pass***-состояние), если состояние теста терминально, или тест переходит в состояние $\sigma \cdot \langle u \rangle$ (нажимается кнопка машины тестирования, определяемая состоянием теста $\sigma \cdot \langle u \rangle$).

Конечное время выполнения теста.

Если βγδ-тест T имеет конечное число немаксимальных трасс, то LTS-тест $T = LTest(T)$ имеет конечное число состояний и конечное число переходов в нетерминальные состояния. В терминальные состояния ***pass*** и ***fail*** может вести только конечное число θ-переходов и переходов по выдаче стимулов, но, быть может, бесконечное число переходов по приёму реакций. Для вынесения вердикта достаточно алгоритма, определяющего правильность реакции после трассы.

Определение 66: Пусть $C \subseteq Z$ и спецификация $S \in LTS_{\beta\gamma\delta}(C)$. Будем говорить, что выполнено **условие конечности времени выполнения LTS-теста**, если для βγδ-спецификации $\Sigma = traces_{\beta\gamma\delta}(S)$ выполнено условия конечности времени выполнения βγδ-теста (Определение 32:).

□

***S*-ветвящиеся LTS.**

Опираясь на эквивалентность βγδ- и LTS-моделей, мы можем использовать алгоритмы, которые мы описали выше для βγδ-моделей. Для этого мы, во-первых, соответствующим образом определим ограничения и алгоритмы, задающие LTS-спецификацию $S \in LTS_{\beta\gamma\delta}(C)$, где $C \subseteq Z$, и, во-вторых, дадим способ построения алгоритмов, задающих βγδ-спецификацию $\Sigma = traces_{\beta\gamma\delta}(S)$.

Нам нужно переформулировать требования к спецификации в терминах не трасс, а состояний и переходов. Мы будем считать, что LTS-спецификация задаётся *локальными алгоритмами*, то есть алгоритмами, которые перечисляют переходы из заданного состояния.

Требуется построить алгоритмы, задающие трассовую спецификацию и имеющие в качестве параметра безопасную βγδ-трассу, по алгоритмам, определённым для LTS-спецификации и имеющим в качестве параметра состояние, достижимое по безопасной βγδ-трассе (такое состояние будем называть *S-достижимым*). В частности, нужно уметь за конечное время определять, продолжается ли безопасная βγδ-трасса заданным стимулом или реакцией безопасным образом: оракулы $\mathbf{Extend1}_{\Sigma}$ и $\mathbf{Gamma1}_{\Sigma}$ или оракулы $\mathbf{Extend2}_{\Sigma}$ и $\mathbf{Gamma2}_{\Sigma}$. Очевидно, для этого нужно, чтобы безопасная βγδ-трасса заканчивалась в конечном числе состояний. Для этого, в частности, в LTS-спецификации в каждом *S*-достижимом состоянии должно быть определено конечное число переходов по символу τ и по каждому символу (стимулу или реакции), безопасному после некоторой βγδ-трассы, заканчивающейся в этом состоянии.

Определение 67: Пусть $C \subseteq Z$ и $\mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$. Состояние LTS будем называть *S-достижимым*, если оно достижимо по некоторой безопасной βγδ-трассе. Состояние LTS будем называть *S-ветвящимся*, если в нём определено конечное число переходов по символу τ и по каждому стимулу или реакции, безопасному после некоторой безопасной βγδ-трассы, заканчивающейся в этом состоянии:

$$(\forall z \in C \ \forall \sigma \cdot \langle z \rangle \in \boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S}) \ s \in (\mathbf{S} \ \boldsymbol{after} \ \sigma) \Rightarrow |\{s` \mid s \text{—} z \rightarrow s`\}| \neq \infty)$$

$$\& \ |\{s` \mid s \text{—} \tau \rightarrow s`\}| \neq \infty.$$

Будем говорить, что LTS *S-ветвящаяся*[51], если каждое её *S*-достижимое состояние *S*-ветвящееся.

Множество *S*-ветвящихся LTS-моделей в заданном алфавите $C \subseteq Z$ обозначим $\boldsymbol{SBLTS}_{\beta\gamma\delta}(C)$.

Множество всех *S*-ветвящихся LTS-моделей обозначим $\boldsymbol{SBLTS}_{\beta\gamma\delta} = \cup\{\boldsymbol{SBLTS}_{\beta\gamma\delta}(C) \mid C \subseteq Z\}$.

□

Утверждение 45: Пусть $C \subseteq Z$ и $\mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$. *S*-ветвимость LTS $\mathbf{S}$ эквивалентна следующему свойству: для каждой безопасной βγδ-трассы σ множество маршрутов, имеющих βγδ-трассу σ, конечно: $\forall \sigma \in \boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S}) \ |\mathbf{R}_{\sigma}| \neq \infty$, где $\mathbf{R}_{\sigma} = \{R \in \boldsymbol{runs}(\mathbf{S}) \mid \sigma \in \boldsymbol{traces}_{\beta\gamma\delta}(R)\}$.

---

[51] *S-* от safe - безопасная.

Также как для βγδ-модели, для LTS-модели мы будем предполагать, что все символы – стимулы, реакции, блокировки и стационарность – задаются конечными представлениями (последовательностями конечной длины в конечном алфавите). Также будем считать, что мы можем алгоритмически определять, является ли символ стимулом (например, префикс “?”), реакцией (например, префикс “!”), блокировкой (например, префикс “{?” и постфикс “}”) или стационарностью (например, “δ”), и по стимулу `?x` мы можем вычислить его блокировку `{?x}`, и наоборот. Наконец, будем предполагать, что состояния LTS также задаются конечными представлениями.

Два способа задания LTS-модели (способы `L1` и `L2`).

Так же как для βγδ-спецификации, мы рассмотрим два способа задания LTS-спецификации, отличающиеся тем, что указывается *для состояния*:

1) неразрушающие стимулы и реакции и переходы по тем из них, которые безопасны после какой-нибудь безопасной βγδ-трассы, заканчивающейся в состоянии, а также τ-переходы или
2) все переходы, определённые в состоянии, включая τ-переходы и γ-переходы.

Важно отметить, что при первом способе задания LTS мы определяем переходы из состояния `s` *не по всем* стимулам и реакциям, которые безопасны в состоянии `s`, а только по таким, которые *безопасны после некоторой безопасной βγδ-трассы*, заканчивающейся в `s`. Однако это не мешает проверять безопасность стимула или реакции `z` после такой βγδ-трассы. Для этого нужно убедиться, что `z` *безопасен в каждом состоянии* после этой βγδ-трассы. Для для проверки безопасности *стимула в состоянии* нужно убедиться, что *этот* стимул неразрушающий в этом состоянии, а для проверки безопасности *реакции в состоянии* нужно убедиться, что *все* реакции неразрушающие в этом состоянии.

Заметим, что стимул или реакция, опасные в состоянии, опасны после каждой безопасной βγδ-трассы, заканчивающейся в этом состоянии. Однако, если они безопасны в состоянии, то они могут быть безопасны после каждой безопасной βγδ-трассы, заканчивающейся в этом состоянии, или опасны после некоторой (или даже каждой) безопасной βγδ-трассы, заканчивающейся в этом состоянии.

Во втором случае задаются все переходы из состояния по стимулам и реакциям. Для проверки безопасности стимула или реакции `z` нужно найти

постсостояния всех переходов по z (из *заданного* состояния или из *каждого* состояния после безопасной βγδ-трассы), а также все состояния, достижимые из них по τ-переходам, и проверить, что все они конвергентны и ни в одном из этих состояний не определён γ-переход.

Дивергенция в LTS может быть двух видов (Рис.58):

a) на конечном числе состояний, то есть образованная только циклами τ-переходов, и
b) на бесконечном числе различных состояний, через которые проходят бесконечные маршруты τ-переходов.

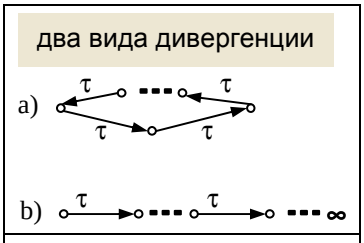


Рис.58.

При задании LTS способом 1) безопасность подразумевает отсутствие дивергенции обоих видов.

При задании LTS способом 2) мы можем проверять дивергенцию вида a), но должны предполагать, что дивергенции вида b) либо нет, либо она сочетается с наличием γ-перехода или дивергенцией вида a).

Заметим, что βγδ-трасса может продолжаться как стимулом ?x, так и его блокировкой {?x}, но не может не продолжаться ни тем, ни другим, если она конвергентна. В состоянии стимул не может и приниматься и блокироваться, но может не приниматься и не блокироваться, если состояние нестабильно. Аналогично, βγδ-трасса может продолжаться как реакцией !y, так и стационарностью δ, но не может не продолжаться ни стационарностью, ни какой-либо реакцией, если она конвергентна. Состояние не может быть стационарно и выдавать реакции, но состояние может быть нестационарно и не выдавать ни одной реакции, если оно нестабильно.

Также, как для трассовой модели, при задании LTS-спецификации в виде пред- и пост-условий рассматриваемые два способа по-разному распределяют проверки между предикатами пред- и постусловия:

1) Предусловие запрещает стимулы и реакции, разрушающие в состоянии, а постусловие определяет те переходы из состояния по стимулам и реакциям, которые могут принадлежать маршрутам безопасных βγδ-трасс.
2) Предусловие запрещает стимулы и реакции, по которых из состояния нет переходов, а постусловие определяет все переходы из состояния: по стимулам, реакциям, τ- и γ-переходы.

Замечание о разрушающих и опасных стимулах и реакциях. Это замечание аналогично соответствующему замечанию для трассовой модели (см. раздел 2.2.9). Можно было бы рассматривать модифицированный способ 1a), когда

вместо оракула разрушаемости используется оракул опасности стимулов и реакций в состоянии. Более того, можно было бы использовать понятие опасности в состоянии s стимула или реакции z в трассовом смысле: $s=\langle z,\gamma\rangle\Rightarrow$, а не так, как мы это делаем сейчас: $s\text{—}z\rightarrow s`=\gamma\Rightarrow$. Соответственно, предусловие запрещает те и только те стимулы, которые при безопасном тестировании нельзя посылать, если LTS оказалась в состоянии s. Также запрещаются все реакции тогда и только тогда, когда их нельзя принимать при безопасном тестировании, если LTS оказалась в состоянии s. Для стимулов здесь большой разницы нет, поскольку $s\text{—}z\rightarrow s`=\gamma\Rightarrow$ влечёт $s=\langle z,\gamma\rangle\Rightarrow$. Однако для реакций разница более существенная: реакция безопасна, если *все* реакции неразрушающие. Способ 1a) задаёт LTS-модель с меньшей точностью, однако, достаточной для генерации тестов для соответствия $\boldsymbol{ioco}_{\beta\gamma\delta}$. В то же время способ 1) предпочтительнее для задания спецификации в пред- и постусловиях безотносительно к выбору того или иного соответствия, поскольку в этом случае мы не закладываемся на те тестовые возможности, которые разрешают наблюдение лишь одного вида отказов, связанных с реакциями, а именно стационарности как отсутствия всех реакций. Если у нас есть возможность наблюдать, например, отсутствие указанной реакции (отдельная кнопка машины тестирования для приёма одной указанной реакции), то способ 1a) не годится: может оказаться, что указанная реакция неразрушающая, хотя другая реакция разрушающая.

Аналогично алгоритмизации трассовой модели мы потребуем конечности алфавита реакций.[52]

Переходы только по безопасным стимулам и реакциям (способ L1).

Определение 68: Пусть $C\subseteq Z$. Будем говорить, что LTS-модели $\mathbf{A},\mathbf{B}\in\boldsymbol{LTS}_{\beta\gamma\delta}(C)$ *F-изоморфны*, если

a) эти LTS либо обе безопасны, либо обе опасны: **A** ***safe*** $\Leftrightarrow$ **B** ***safe***;

a) изоморфны их под-LTS, порождаемые безопасными βγδ-трассами;

b) в соответствующих состояниях, достижимых по безопасным βγδ-трассам исходных LTS[53], одно и то же множество стимулов и реакций, разрушающих в этих состояниях:

---

[52] Вместо конечности алфавита реакций можно было бы потребовать конечности множества переходов по реакциям из каждого состояния, достижимого по безопасной βγδ-трассе. Соответственно меняется алгоритмическое задание LTS-модели: вместо итератора реакций и итератора переходов по каждой реакции нужен итератор продолжающих переходов по всем реакциям (для первого способа ещё нужно определять разрушающая реакция или нет).

$\exists \boldsymbol{f}: \mathbf{A}_{safe} \to \mathbf{B}_{safe}$ изоморфизм такой, что
$\forall a \in \cup(\mathbf{A}\ \textit{after}\ \textit{safe}_{\beta\gamma\delta}(\mathbf{A}))\ \forall z \in C$
z разрушающий в a в LTS $\mathbf{A} \Leftrightarrow z$ разрушающий в $\boldsymbol{f}(a)$ в LTS $\mathbf{B}$, где $\mathbf{S}_{safe}=LTS(V_S, C, E, s_0)$ и
$E=\cup(\{E_S \cap \textit{Im}(S) \mid S \in \textit{runs}_{\beta\gamma\delta}(\mathbf{S})\ \&\ \textit{trace}(S) \in \textit{safe}_{\beta\gamma\delta}(\mathbf{S})\})$.
Изоморфизм $\boldsymbol{f}$ будем в этом случае называть *F-изоморфизмом*.

□

Среди множества *F*-изоморфных LTS-моделей можно выделить одну (с точностью до изоморфизма) LTS-модель, которая получается с помощью следующих операций:

a) добавляем специальное γ-состояние, в котором определяется единственный переход – γ-петля: $\gamma \text{—} \gamma \to \gamma$;

b) если LTS опасна, то начальным состоянием вместо s_0 объявляем γ-состояние;

c) если состояние s *S*-достижимо, перенаправляем каждый переход $s \text{—} z \to s'$ по стимулу или реакции z, разрушающему в s ($s' = \gamma \Rightarrow$), в γ-состояние: $s \text{—} z \to \gamma$;

d) если состояние s *S*-достижимо, удаляем все переходы из s по стимулам и реакциям, которые не разрушающие в s, но опасны после каждой безопасной βγδ-трассы, заканчивающейся в s.

Такую LTS-модель будем называть *F*-канонической.

Определение 69: Пусть $C \subseteq Z$. Будем говорить, что LTS-модель $\mathbf{S} \in \textit{LTS}_{\beta\gamma\delta}(C)$ *F-каноническая*, если:

a) $\gamma \in V_S\ \&\ \gamma \text{—} \gamma \to \gamma\ \&\ |\{e \in E_S \mid \textit{pre}(e)=\gamma\}|=1$

b) $\&\ (\neg\ \mathbf{S}\ \textit{safe} \Rightarrow s_0=\gamma)$

$\&\ \forall s \in \cup(\mathbf{S}\ \textit{after}\ \textit{safe}_{\beta\gamma\delta}(\mathbf{S}))\ \forall z \in C\ \forall s, s'$

c) $(s \text{—} z \to s'\ \&\ \neg\ s'\ \textit{safe} \Rightarrow s'=\gamma)$

d) $\&\ (s \text{—} z \to s'\ \&\ s' \neq \gamma \Rightarrow \exists \sigma\ \sigma \cdot \langle z \rangle \in \textit{safe}_{\beta\gamma\delta}(\mathbf{S})\ \&\ s \in (\mathbf{S}\ \textit{after}\ \sigma))$.

□

Замечание о неалгоритмизуемости построения *F*-канонической модели из заданной LTS. Для того, чтобы убедиться, что нужно удалить переход по

---

[53] Достижимые состояния под-LTS совпадают с состояниями, достижимыми по безопасным βγδ-трассам, с точностью до начального состояния. Начальное состояние в опасной LTS недостижимо по безопасной βγδ-трассе, но, разумеется, принадлежит под-LTS.

символу z из состояния s, нужно проверить, что после всех безопасных βγδ-трасс, заканчивающихся в состоянии s, символ z опасен. Такую проверку, вообще говоря, нельзя выполнить за конечное время, если LTS задана локальными алгоритмами, поскольку в состоянии s может заканчиваться бесконечное число безопасных βγδ-трасс, проходящих (суммарно) через бесконечное число состояний, то есть нужно исследовать бесконечное число маршрутов. В то же время для конечной LTS эта задача алгоритмизуема.

Первым способом LTS задаётся с точностью до *F*-изоморфизма. Можно также сказать, что первым способом однозначно (с точностью до изоморфизма) задаётся *F*-каноническая LTS (Рис.59). Очевидно, что *F*-изоморфные LTS имеют одно и то же множество безопасных βγδ-трасс и, следовательно, ***ioco***$_{\beta\gamma\delta}$-эквивалентны.

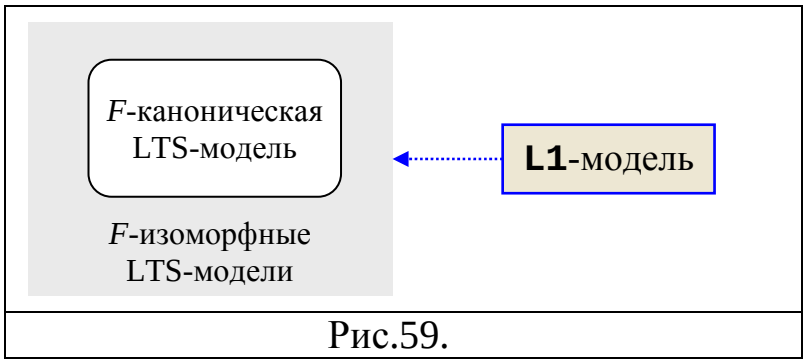


Рис.59.

Определение 70: Пусть $C \subseteq Z$. Будем говорить, что *S*-ветвящаяся LTS-модель $S \in SBLTS_{\beta\gamma\delta}(C)$ алгоритмически задана способом **L1** (**L1**-модель), если

1) задан оракул безопасности $\mathbf{Safe}_S()$, который возвращает ***true***, если модель безопасна: $\mathbf{Safe}_S() = S$ ***safe***;
2) *множество стимулов перечислимо*, и задан итератор (алгоритм перечисления) $\mathbf{X}_C$, перечисляющий множество ?C;
3) *множество реакций конечно*, и задан итератор (алгоритм перечисления) $\mathbf{Y}_C$, перечисляющий множество !C;

далее для каждого *S*-достижимого состояния s:

4) *множество разрушающих в состоянии стимулов и реакций разрешимо относительно множества всех стимулов и реакций*, и задан оракул (алгоритм разрешения) $\mathbf{Gamma1}_S(s,z)$, который для каждого стимула или реакции $z \in C$ возвращает ***true***, если z разрушающий в s;

5) [из *S*-ветвимости следует, что *множество переходов из состояния по каждому стимулу или реакции, безопасному после некоторой безопасной βγδ-трассы, заканчивающейся в этом состоянии, конечно;*] задан итератор (алгоритм перечисления) **Extend1**$_{\mathtt{s}}$`(s,z)`, который для каждого стимула или реакции `z∈C` такого, что $\exists\sigma\ \sigma\cdot\langle z\rangle\in$ ***safe***$_{\beta\gamma\delta}$`(S)` & `s∈(S` ***after*** σ`)`, перечисляет множество постсостояний `{s`|s—z→s`}`;

6) [из *S*-ветвимости следует, что *множество τ-переходов из состояния конечно;*] задан итератор (алгоритм перечисления) **Tau1**$_{\mathtt{s}}$`(s)`, который перечисляет множество постсостояний τ-переходов `{s`|s—τ→s`}`.

Множество **L1**-моделей в алфавите `C` будем обозначать **L1**`(C)`, а множество всех **L1**-моделей будем обозначать **L1**`=∪{`**L1**`(C)|C⊆Z}`.

□

Если LTS задана способом **L1**, то мы знаем все её финальные βγδ-трассы и их ***DRT***-замыкание. Про другие βγδ-трассы LTS нам ничего не известно. Если это *F*-каноническая LTS, то других βγδ-трасс в ней нет.

Утверждение 46: Существует алгоритм преобразования **L12T1**`:`**L1**`→`**T1**, который по каждой **L1**-спецификации строит модель её финальных трасс как **T1**-спецификацию. Как следствие, для **L1**-спецификации выполнено условие конечности времени выполнения LTS-теста, множество всех строго-значимых LTS-тестов с конечным числом нетерминальных состояний перечислимо, и можно построить алгоритм его перечисления.

□400

Переходы по всем стимулам и реакциям, и по разрушению (способ L2).

Теперь будем считать, что нам не задан алгоритм, проверяющий разрушаемость стимула или реакции в состоянии, но мы можем узнать, определён в состоянии переход по разрушению или нет. В этом случае проверять разрушаемость стимула или реакции нам придётся самим. Стимул или реакция разрушающие, если в состоянии есть переход по этому стимулу или реакции, постсостояние которого дивергентно или из этого постсостояния по τ-маршрутам достижим γ-переход.

Для того, чтобы за конечное время можно было проверить разрушение после стимула или реакции, требуется перечислимость переходов по стимулам и реакциям. Заметим, что *S*-ветвимость требует только конечности (и, следовательно, перечислимости) множества переходов по стимулам и реакциям, безопасным после трассы, но не налагает никаких ограничений на

мощность и перечислимость множества переходов по опасным стимулам или реакциям.

Перечислимость переходов по стимулам и реакциям необходимое, но недостаточное условие для того, чтобы можно было алгоритмически проверять разрушение после стимула или реакции в *S*-достижимом состоянии. Нам нужно ещё уметь строить τ-замыкание после каждого перехода по такому стимулу или реакции для того, чтобы в состояниях этого τ-замыкания проверять наличие перехода по разрушению или обнаруживать дивергенцию.

Чтобы можно было проверять дивергенцию, нужно, чтобы a) дивергенция порождалась τ-циклом, быть может, после цепочки τ-переходов или b) по τ-переходам был достижим γ-переход (Рис.60).

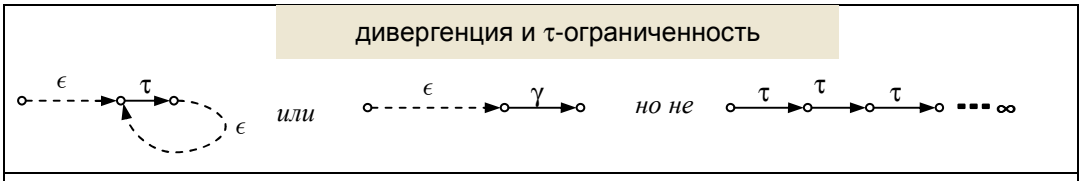


Рис.60.

При этом условии для того, чтобы можно было определить дивергентность состояния, нам было бы достаточно перечислимости дерева τ-маршрутов, начинающихся в состоянии. Однако мы хотим задавать LTS локальными алгоритмами, то есть алгоритмами, которые перечисляют переходы из состояния, а не маршруты, начинающиеся в состоянии. В таком случае нам требуется перечислимость τ-переходов, определённых в состоянии, то есть дерево τ-маршрутов, начинающихся в состоянии будет перечислимо-ветвящимся, а не просто перечислимым.

Заметим, что из перечислимости дерева τ-маршрутов, вообще говоря, не следует его перечислимо-ветвимость. Пример показан на Рис.61. Здесь в состояниях `b` и `c` определены τ-переходы в неперечислимые подмножества $\mathcal{B}$ и $\mathcal{N}\setminus\mathcal{B}$ множества натуральных чисел $\mathcal{N}$, которыми обозначены постсостояния τ-маршрутов длины `2`. Таким образом в состояниях `b` и `c` множества τ-переходов неперечислимы, однако перечислить постсостояния всех τ-маршрутов с началом в `a` легко: сначала перечисляем состояние `a` (τ-маршрут длины `0`), затем состояния `b` и `c` (τ-маршруты длины `1`), затем все натуральные числа (τ-маршруты длины `2`).

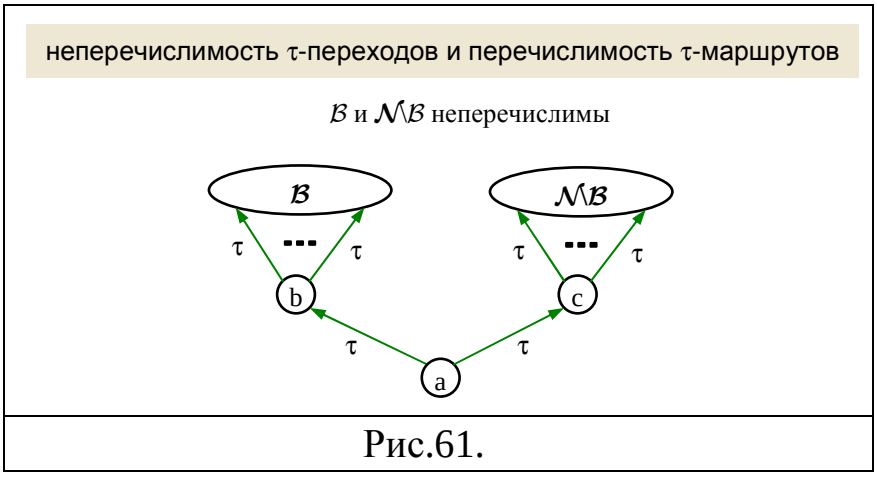


Рис.61.

С другой стороны, любое перечислимо-ветвящееся дерево конечных последовательностей перечислимо.

Для проверки безопасности стимула или реакции z в состоянии s необходимо перечисление множества постсостояний переходов $s\text{—}z\rightarrow s`$, которое должно быть конечным, если z безопасен, или должно быть перечислимым и содержать начальное состояние τ-маршрута с самопересечением или заканчивающегося в состоянии с γ-переходом.

Определение 71: Пусть $C\subseteq Z$ и $\mathbf{S}\in\boldsymbol{LTS}_{\beta\gamma\delta}(C)$. Будем говорить, что множество состояний $U\subseteq V_{\mathbf{S}}$ *τ-ограничено*, если оно перечислимо и дерево τ-маршрутов, начинающихся в состояниях из U и не проходящих через состояния с γ-переходом (кроме, быть может, конечного состояния), либо a) конечно, либо b) перечислимо-ветвящееся и содержит маршрут с самопересечением или заканчивающийся в состоянии с γ-переходом:

U перечислимо & ($|\mathbf{R}_U|\neq\infty$ ∨ $\mathbf{R}_U$ перечислимо-ветвящееся

& $\exists R\in\mathbf{R}_U$ $(\omega(R)\text{—}\gamma\rightarrow$ ∨ $\omega(R)\in\boldsymbol{Im}_\circ\boldsymbol{pre}_\circ R)$), где

$\mathbf{R}_U=\{R\in\boldsymbol{runs}(s)\,|\,s\in U$ & $\boldsymbol{trace}(R)=\epsilon$ & $\forall s`\in\boldsymbol{Im}_\circ\boldsymbol{pre}_\circ R$ $s`\text{—}\gamma\not\rightarrow\}$.

Состояние будем называть τ-ограниченным, если τ-ограничено множество из одного этого состояния.

□

Определение 72: *τ-ограниченной* LTS будем называть LTS, в которой все достижимые состояния τ-ограничены.

□

Определение 73: Пусть $C\subseteq Z$ и $\mathbf{S}\in\boldsymbol{LTS}_{\beta\gamma\delta}(C)$. Будем говорить, что LTS $\mathbf{S}$ *S-τ-ограничена*, если τ-ограничено начальное состояние и множество постсостояний $s`$ переходов $s\text{—}z\rightarrow s`$ для каждого *S*-достижимого состояния s и каждого стимула или реакции z:

s_0 τ-ограничено & $\forall\sigma\in$ ***safe***$_{\beta\gamma\delta}$**(S)** $\forall s\in$(**S** ***after*** σ) $\forall z\in C$ $\{s` \mid s\text{—}z\rightarrow s`\}$ τ-ограничено.

□

Заметим, что конечная LTS, очевидно, *S*-τ-ограничена, хотя может иметь дивергенцию. Однако *S*-ветвящаяся LTS, вообще говоря, не *S*-τ-ограничена, поскольку не регламентируется число переходов из *S*-достижимых состояний по опасным символам и τ-маршруты после таких переходов.

Утверждение 47: *S*-τ-ограниченная LTS *S*-ветвящаяся.

□402

Определение 74: Пусть $C\subseteq Z$, **S**$\in$***LTS***$_{\beta\gamma\delta}(C)$ и $s\in V_{\mathbf{s}}$. Определим трансфинальные βγδ-трассы LTS:

- Трансфинальные βγδ-трассы, начинающиеся в s:
  tfinal$_{\beta\gamma\delta}(s) =_{def}$ ***tfinal***$\circ$***traces***$_{\beta\gamma\delta}(s)$.
- Трансфинальные βγδ-трассы LTS:
  tfinal$_{\beta\gamma\delta}$**(S)** $=_{def}$ ***tfinal***$\circ$***traces***$_{\beta\gamma\delta}(s_0)$ = ***tfinal***$_{\beta\gamma\delta}(s_0)$.

□

Определение 75: Пусть $C\subseteq Z$. Будем говорить, что LTS-модели **A, B**$\in$***LTS***$_{\beta\gamma\delta}(C)$ *TF-изоморфны*, если

a) изоморфны их под-LTS, порождаемые трансфинальными βγδ-трассами без разрушения, в которых удалены τ-переходы из состояний с γ-переходами, и

b) в соответствующих достижимых состояниях этих под-LTS либо в обоих определены переходы по разрушению, либо в обоих не определены:

$\exists$***f*** **:** $\mathbf{A}_{tfinal}\rightarrow\mathbf{B}_{tfinal}$ изоморфизм такой, что**:**

$\forall a\in$***der***$(\mathbf{A}_{tfinal})$ $a\text{—}\gamma\rightarrow$ в LTS **A** $\Leftrightarrow$ ***f***$(a)\text{—}\gamma\rightarrow$ в LTS **B**,

где $\mathbf{S}_{tfinal}=LTS(V_{\mathbf{s}}, C, E, s_0)$ и

$E = \cup(\{E_{\mathbf{s}}\cap$***Im***$(S) \mid S\in$***runs***$_{\beta\gamma\delta}$**(S)** & ***trace***$(S)\in C_{\beta\delta}^{*}\cap$***tfinal***$_{\beta\gamma\delta}$**(S)**$\}$
$\setminus \{(s,\tau,s`)\in E_{\mathbf{s}} \mid s\text{—}\gamma\rightarrow\}$.

Изоморфизм ***f*** будем в этом случае называть *TF-изоморфизмом*.

□

Среди множества *TF*-изоморфных LTS-моделей можно выделить одну (с точностью до изоморфизма) LTS-модель, которая получается удалением всех переходов по стимулам и реакциям из состояний, которые не *S*-достижимы, и заменой каждого γ-перехода на γ-петлю[54]. Такую LTS-модель будем называть *TF*-канонической.

---

[54] Поскольку LTS определяется с точностью до изоморфизма, достаточно удалять переходы по стимулам и реакциям и заменять γ-переходы на γ-петли только в тех

Определение 76: Пусть $C \subseteq Z$. Будем говорить, что LTS-модель $\mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$ *TF-каноническая*, если в состоянии, достижимом по трансфинальной βγδ-трассе, но не *S*-достижимом, могут быть определены либо только τ-переходы, либо только γ-петля:

$$\forall s \in \cup(\mathbf{S}\ \textit{after}\ \textit{tfinal}_{\beta\gamma\delta}(\mathbf{S})) \setminus \cup(\mathbf{S}\ \textit{after}\ \textit{safe}_{\beta\gamma\delta}(\mathbf{S}))$$

$$\forall z \in C\ \ s \text{—} z \not\rightarrow\ \&\ (\forall s`\ \ s \text{—} \gamma \rightarrow s` \Rightarrow s \text{—} \tau \not\rightarrow\ \&\ s = s`).$$

□

Замечание о неалгоритмизуемости построения *TF*-канонической модели из заданной LTS. Для того, чтобы убедиться, что нужно удалить все переходы по стимулам и реакциям из одного состояния s, нужно проверить, что это состояние не *S*-достижимо. Такую проверку, вообще говоря, нельзя выполнить за конечное время, если LTS задаётся локальными алгоритмами, поскольку может оказаться бесконечным число безопасных βγδ-трасс, проходящих (суммарно) через бесконечное число состояний, то есть нужно исследовать бесконечное число маршрутов. В то же время для конечной LTS эта задача алгоритмизуема.

По каждой *TF*-канонической LTS можно однозначно построить *F*-изоморфную ей *F*-каноническую LTS так же, как это делается для любой LTS с той же моделью финальных трасс. Однако, по замечанию о неалгоритмизуемости построения *F*-канонической модели, такое построение в общем случае неалгоритмизуемо.

Вторым способом LTS задаётся с точностью до *TF*-изоморфизма. Можно также сказать, что этим способом однозначно задаётся *TF*-каноническая LTS (Рис.62). Очевидно, что *TF*-изоморфные LTS имеют одно и то же множество трансфинальных βγδ-трасс и, следовательно, $\boldsymbol{ioco}_{\beta\gamma\delta}$-эквивалентны.

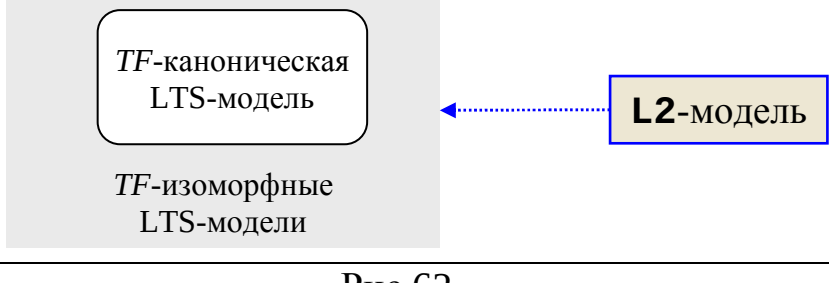


Рис.62.

состояниях, которые достижимы по пустой трассе, если LTS опасна, или только по безопасным βγδ-трассам, продолженным опасным стимулом или опасной реакцией.

Состояния можно разделить на четыре группы (цвета) в зависимости от того, какие переходы в этом состоянии определяются при алгоритмическом задании LTS:

- *Зелёное состояние* – заданы все переходы.
- *Жёлтое состояние* – заданы только τ- и γ-переходы.
- *Красное состояние* – заданы только γ-переходы.
- *Чёрное состояние* – не заданы никакие переходы.

Переходы задаются соответствующими алгоритмами. Эти алгоритмы имеют предусловия, которые описывают требования на цвета состояний в терминах той или иной достижимости этих состояний.

Наличие γ-перехода в состоянии делает несущественным наличие или отсутствие в нём других переходов, поэтому в таком состоянии мы не будем задавать других переходов. Более того, нам важен лишь сам факт наличия в состоянии γ-перехода, но не важно сколько таких переходов и в какие постсостояния они ведут.

Решающее значение имеет, какие состояния объявляются зелёными. Цвет остальных состояний определяется однозначно: (Рис.63):

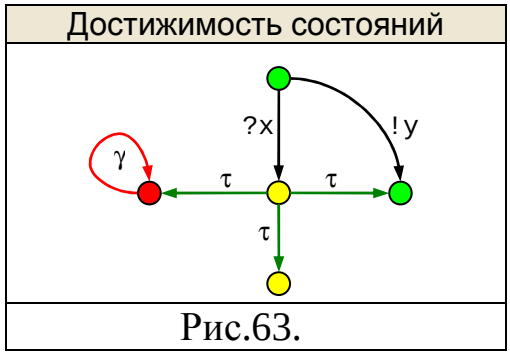


Рис.63.

Если начальное состояние или постсостояние перехода по стимулу или реакции из зелёного состояния не зелёное, то оно жёлтое или красное в зависимости от того, есть в этом постсостоянии γ-переход или нет.
Аналогично, если постсостояние τ-перехода из жёлтого состояния не зелёное, то оно жёлтое или красное в зависимости от того, есть в этом постсостоянии γ-переход или нет.
Остальные состояния чёрные.

Это индуктивное определение эквивалентно следующему неиндуктивному определению:

- Состояние жёлтое, если из него нет γ-перехода, оно не зелёное и либо начальное, либо достижимо из зелёного состояния по одному переходу по стимулу или реакции и далее по τ-маршруту, в каждом состоянии которого нет γ-перехода.
- Состояние красное, если из него есть γ-переход, оно не зелёное и либо начальное, либо достижимо из зелёного состояния по одному переходу по стимулу или реакции и далее по τ-маршруту, в каждом неконечном состоянии которого нет γ-перехода.

· Остальные состояния чёрные.

Заметим, что красное и жёлтое состояния определяются одинаково, за исключением наличия (в красном) или отсутствия (в жёлтом) состоянии γ-перехода.

Для генерации тестов, и вообще верификации конформности, нам будет необходимо и достаточно отождествить зелёные состояния спецификации с *S*-достижимыми состояниями, то есть состояниями, достижимыми по безопасным βγδ-трассам. При таком определении зелёных состояний жёлтые и красные состояния будем называть *S+-достижимыми* и *S+γ-достижимыми*, соответственно.

Определение 77: Пусть $C \subseteq Z$ и $\mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$. Определим *S+-достижимые* и *S+γ-достижимые* состояния следующим индуктивным способом: $\forall s \in V_S \ \forall z \in C \ \forall s` \in V_S$ : $s`$ не *S*-достижимо

$s`=s_0$ & $s` \not\xrightarrow{\gamma}$ $\vdash$ $s`$ *S+*-достижимо,

$s`=s_0$ & $s` \xrightarrow{\gamma}$ $\vdash$ $s`$ *S+*γ-достижимо,

s *S*-достижимо & $s \xrightarrow{z} s`$ & $s` \not\xrightarrow{\gamma}$ $\vdash$ $s`$ *S+*-достижимо,

s *S*-достижимо & $s \xrightarrow{z} s`$ & $s` \xrightarrow{\gamma}$ $\vdash$ $s`$ *S+*γ-достижимо,

s *S+*-достижимо & $s \xrightarrow{\tau} s`$ & $s` \not\xrightarrow{\gamma}$ $\vdash$ $s`$ *S+*-достижимо,

s *S+*-достижимо & $s \xrightarrow{\tau} s`$ & $s` \xrightarrow{\gamma}$ $\vdash$ $s`$ *S+*γ-достижимо.

□

Определение 78: Пусть $C \subseteq Z$. Будем говорить, что *S*-τ-ограниченная LTS-модель **S** в алфавите C алгоритмически задана способом **L2** (**L2**-модель), если

1) *множество стимулов перечислимо*, и задан итератор (алгоритм перечисления) $\mathbf{X}_C$, перечисляющий множество $?C$;
2) *множество реакций конечно*, и задан итератор (алгоритм перечисления) $\mathbf{Y}_C$, перечисляющий множество $!C$;
3) для каждого *S*-достижимого состояния s [из *S*-τ-ограниченности следует, что *множество переходов из* s *по каждому стимулу или реакции перечислимо*, а из *S*-ветвимости следует, что *множество переходов из* s *по каждому стимулу или реакции, безопасному после некоторой безопасной βγδ-трассы, заканчивающейся в* s, *конечно;*] задан итератор (алгоритм перечисления) $\mathbf{Extend2}_S(s,z)$, который

для каждого стимула или реакции $z \in C$ перечисляет множество постсостояний $\{s` \mid s \text{—} z \rightarrow s`\}$;

4) для каждого *S*- и *S*+-достижимого состояния s [из *S*-τ-ограниченности следует, что *множество τ-переходов из* s *перечислимо,* а из *S*-ветвимости следует, что, *если* s *S-достижимо, то множество τ-переходов из* s *конечно;*] задан итератор (алгоритм перечисления) $\mathbf{Tau2}_s(s)$, который перечисляет множество постсостояний τ-переходов $\{s` \mid s \text{—} \tau \rightarrow s`\}$;

5) для каждого *S*- , *S*+- и *S*+γ-достижимого состояния s задан оракул $\mathbf{Gamma2}_s(s)$, который возвращает ***true***, если в состоянии определён γ-переход $s \text{—} \gamma \rightarrow$, то есть оно *S*+γ-достижимо.

Множество **L2**-моделей в алфавите C будем обозначать $\mathbf{L2}(C)$, а множество всех **L2**-моделей будем обозначать $\mathbf{L2} = \cup\{\mathbf{L2}(C) \mid C \subseteq Z\}$.

□

Если LTS задана способом **L1**, то мы знаем все её трансфинальные βγδ-трассы и, очевидно, их ***DRT***-замыкание. Про другие βγδ-трассы LTS нам ничего не известно. Если это *TF*-каноническая LTS, то других βγδ-трасс в ней нет.

Утверждение 48: Существует алгоритм преобразования $\mathbf{L2}2\mathbf{L1} : \mathbf{L2} \rightarrow \mathbf{L1}$, который по каждой **L2**-спецификации строит *F*-изоморфную ей **L1**-спецификацию. Как следствие, для **L2**-спецификации выполнено условие конечности времени выполнения LTS-теста, множество всех строго-значимых LTS-тестов с конечным числом нетерминальных состояний перечислимо, и можно построить алгоритм его перечисления.

□403

Сравнение способов задания LTS.

В отличие от трассовых **T1**- и **T2**-спецификаций, по **L1**-спецификации, вообще говоря, нельзя алгоритмически построить **L2**-спецификацию. Дело в том, что возможна ситуация, когда стимул или реакция z безопасен в состоянии s, но опасен после каждой безопасной βγδ-трассы, заканчивающейся в состоянии s. В этой ситуации **L1**-спецификация не итерирует переходы $s \text{—} z \rightarrow s`$, а **L2**-спецификация должна итерировать такие переходы. Можно было бы итерировать пустое множество переходов, выделяя одну такую **L2**-спецификацию, однако, по замечанию о неалгоритмизуемости построения *F*-канонической модели, мы не можем за конечное время определить, имеет место указанная ситуация или нет.

Разумеется, отсюда не следует, что по **L1**-спецификации нельзя алгоритмически построить **L2**-спецификацию в каких-то частных случаях, когда у нас есть какая-то дополнительная информация о задаваемой LTS (см., например, ниже Утверждение 83: в разделе 3.6.2).

Для **L2**-спецификации **S** можно построить алгоритмы, задающие модель её трансфинальных трасс как **T2**-спецификацию. Доказательство аналогично доказательству Утверждение 46:.

Ниже на Рис.64 приведена таблица сравнения различных способов задания спецификации для LTS-моделей. Стрелками показана выводимость алгоритмов, задающих спецификацию на конце стрелки, из алгоритмов, задающих спецификацию в начале стрелки.

L1	**S** *S*-ветвящаяся
	задан оракул безопасности $\mathbf{Safe_S}()$, который возвращает ***true***, если модель безопасна: $\mathbf{Safe_S}() = \mathbf{S}$ ***safe***
	множество стимулов перечислимо и задан итератор стимулов $\mathbf{X_C}$: ?C
	множество реакций конечно и задан итератор реакций $\mathbf{Y_C}$: !C
	для каждого *S*-достижимого состояния s :
T1 ◄	*множество разрушающих в состоянии стимулов и реакций разрешимо относительно множества всех стимулов и реакций* и задан оракул $\mathbf{Gamma1_S}(s,z)=(z$ разрушающий в $s)$ для $z\in C$
	[*множество переходов из состояния по каждому стимулу или реакции, безопасному после некоторой безопасной βγδ-трассы, заканчивающейся в этом состоянии, конечно*] и задан итератор $\mathbf{Extend1_S}(s,z)$: $\{s` \mid s—z→s`\}$ где $z\in C$ такое, что $\exists\sigma\ \sigma\cdot\langle z\rangle\in \mathbf{safe}_{\beta\gamma\delta}(\mathbf{S})\ s\in(\mathbf{S}\ \mathbf{after}\ \sigma)$
	[*множество τ-переходов из состояния конечно*] и задан итератор $\mathbf{Tau1_S}(s)$: $\{s` \mid s—\tau→s`\}$

▲

L2	**S** *S*-τ-ограниченная
	множество стимулов перечислимо и задан итератор стимулов $\mathbf{X_C}$: ?C
	множество реакций конечно и задан итератор реакций $\mathbf{Y_C}$: !C
	для каждого *S*-достижимого состояния s [*множество переходов из* s *по каждому стимулу или реакции перечислимо, а по каждому стимулу или реакции, безопасному после некоторой безопасной βγδ-трассы, заканчивающейся в* s, *конечно*] и задан
T2 ◄	итератор $\mathbf{Extend2_S}(s,z)$: $\{s` \mid s—z→s`\}$ для $z\in C$
	для каждого *S*- и *S*+-достижимого состояния s [*множество τ-переходов из* s *перечислимо, а если* s *S-достижимо, то конечно*] и задан итератор $\mathbf{Tau2_S}(s)$: $\{s` \mid s—\tau→s`\}$
	для каждого *S*- , *S*+- и *S*+γ-достижимого состояния s задан *оракул наличия перехода по разрушению из состояния* $\mathbf{Gamma2_S}(s)= s—\gamma→$

Рис.64.

Глава 2.4. Сравнение моделей

Структура главы:
1. Сравнение трассовых моделей и LTS-модели
2. Сравнение моделей безопасных и финальных трасс с βγδ-моделью
3. Дивергенция вместо разрушения. Модель строго-конвергентных трасс
4. Различение разрушения и дивергенции

Выше мы рассмотрели четыре типа моделей: модель безопасных трасс, модель финальных трасс, βγδ-модель и LTS-модель. По степени «детальности» и, соответственно, избыточности, по отношению к соответствию $\boldsymbol{ioco}_{\beta\gamma\delta}$, они соотносятся так, как показано на Рис.65.

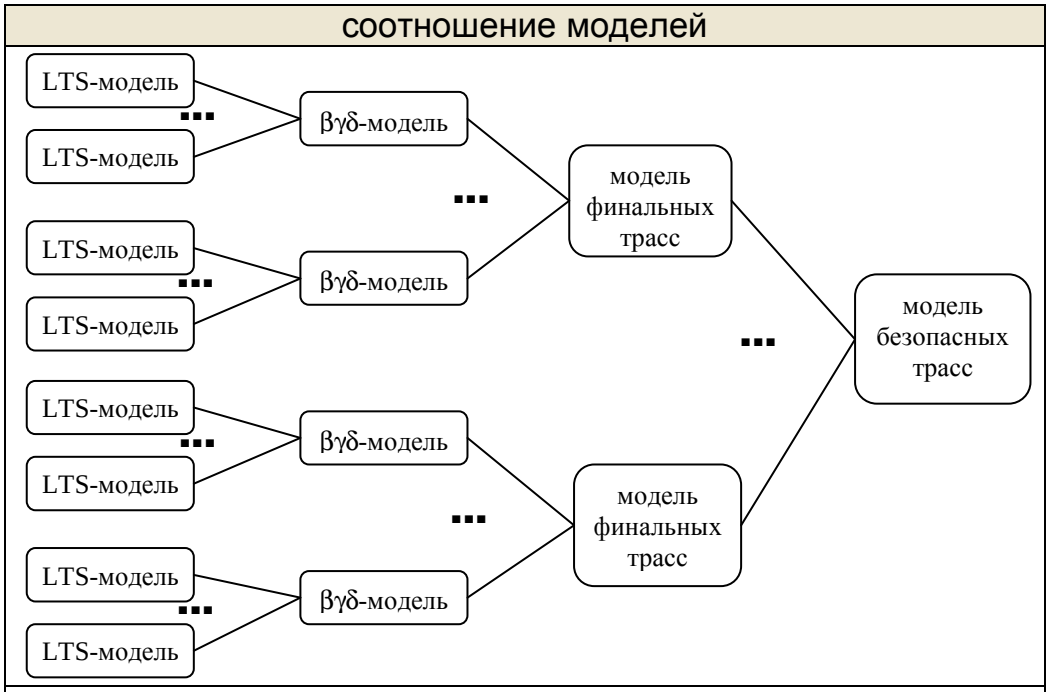


Рис.65.

2.4.1. Сравнение трассовых моделей и LTS-модели

Самой «избыточной» моделью является LTS-модель. Однако эта модель наиболее наглядна и удобна для человеческого восприятия. Многие вещи можно увидеть в ней, так сказать, непосредственным созерцанием (когда пишут, что это «очевидно»). Другие модели не обладают такой наглядностью.

В частности, определение βγδ-трасс, безопасных и финальных βγδ-трасс на LTS достаточно просто и понятно. В то же время эксплицитные определения трассовых моделей достаточно сложны и далеко не очевидны.

Но самым главным преимуществом LTS-модели является возможность определения композиции моделей. Вообще говоря, в трассовой теории также можно определить композицию моделей, имеющую тот же смысл. Однако для этого недостаточно βγδ-трасс, как трасс наблюдений. Причина в том, что для композиции мы должны иметь больше знаний об устройстве системы, чем те знания, которые можем получить из трасс наблюдений. В Часть 3 мы введём, так называемые, ϕ-трассы и определим композицию ϕ-трасс таким образом, что ϕ-трассы композиции LTS-моделей совпадают с композицией ϕ-трасс этих моделей. По ϕ-трассам LTS-модели можно получить все её βγδ-трассы, однако обратное неверно.

Хотя мы не приводим в этой работе эксплицитного определения ϕ-модели, такое определение можно дать, но оно также будет достаточно сложным.

Поэтому LTS-модель остаётся базовой моделью, по которой сверяются трассовые модели.

2.4.2. Сравнение моделей безопасных и финальных трасс с βγδ-моделью

Самой «неизбыточной» моделью является модель безопасных трасс, поскольку, по Утверждение 21:, только те модели безопасных трасс ***ioco***$_{βγδ}$-эквивалентны, которые совпадают. Всю трассовую теорию, включая генерацию тестов, можно было бы построить, опираясь только на модель безопасных трасс. В модели финальных трасс более явно прописаны причины, по которым некоторые безопасные трассы неконвергентны: наличие разрушающих трасс. В остальном модели безопасных и финальных трасс очень похожи.

Существуют веские причины для выбора в качестве трассовой модели именно βγδ-модели . Таких причин две.

Первая причина: βγδ-модель лучше, чем модель безопасных или финальных трасс, отвечает семантике машины тестирования.

Хотя при безопасном тестировании могут наблюдаться только безопасные трассы тестируемой модели, тем не менее мы определяли разрушение как условно-наблюдаемое действие, и поэтому другие, небезопасные, трассы модели также можно считать условно-наблюдаемыми.

Конечно, наблюдение самого разрушения есть некоторая абстракция, поскольку под разрушением мы понимаем любое нежелательное поведение, в том числе и такое, которое вовсе «не кажется» разрушением: приём

стимулов, выдача реакции и т.п. Следуя этому пониманию, было бы естественно запретить наблюдение разрушающих трасс, то есть трасс, заканчивающихся разрушением.

Однако, наблюдаемая трасса может не быть безопасной даже в том случае, когда не содержит разрушение: например, трасса $\mu\cdot\langle\{?x\},!y\rangle$ или даже $\mu\cdot\langle?x,!y\rangle$ при наличии трассы $\mu\cdot\langle?x,\gamma\rangle$. При посылке стимула $?x$ наблюдаемая трасса выбирается недетерминированно и вполне может быть выбрана одна из первых двух, неразрушающих, трасс и не выбрана третья, разрушающая, трасса. Запрет на наблюдение всех таких трасс объясняется возможностью выбора третьей, разрушающей, трассы, однако это не отменяет существование других трасс. Ограничиваясь только безопасными или финальными трассами, мы, фактически, утверждаем, что после u-дивергентной безопасной трассы проверка символа u *всегда* вызывает разрушение, что вовсе не обязательно.

Вторая причина: βγδ-модель лучше, чем модель безопасных или финальных трасс, соответствует LTS-модели.

Для представления βγδ-модели в виде LTS достаточно понимать отказ как виртуальную петлю в стабильном состоянии: состоянии, в котором нет τ- и γ-переходов, и нет переходов по действиям, принадлежащим отказу. Таково обычное понимание трасс с отказами в LTS – это просто трассы LTS с добавленными виртуальными петлями.

Безопасные или финальные трассы, вообще говоря, нельзя представить в виде LTS аналогичным образом, поскольку эти модели могут не являться βγδ-моделями.

2.4.3. Дивергенция вместо разрушения. Модель строго-конвергентных трасс

Мы рассмотрели модели в которых дивергенция моделируется разрушением. С формальной, математической, точки зрения имеет право на существование и обратный подход: моделирование разрушения дивергенцией. Для этого достаточно в LTS-модели $\mathbf{S}$ заменить каждый γ-переход на τ-петлю с помощью преобразования $\boldsymbol{L_\gamma 2L}(\mathbf{S})$:

$s,s`\in V_{\mathbf{S}}$ & $z\in C_\tau$ & $s\text{—}z\rightarrow s`$ $\vdash$ $s\text{—}z\rightarrow s`$,

$s\text{—}\gamma\rightarrow$ $\vdash$ $s\text{—}\tau\rightarrow s$.

Теперь мы можем рассматривать LTS в алфавите $C \subseteq Z$, как это обычно и делают, а не в алфавите C_γ, как мы делали выше.

Определим преобразование $\boldsymbol{L2L}_{\beta\delta}$ для добавления виртуальных петель отказов:

$$s, s` \in V_S \;\&\; z \in C_\tau \;\&\; s \text{—}z\rightarrow s` \;\vdash\; s \text{—}z\rightarrow s`,$$

$$s \in V_S \;\&\; o \in \beta\delta(s) \;\vdash\; s \text{—}o\rightarrow s.$$

βδ-трассы LTS $\mathbf{S}$ определим как трассы LTS $\boldsymbol{L2L}_{\beta\delta}(\mathbf{S})$:

$$\boldsymbol{traces}_{\beta\delta}(\mathbf{S}) =_{def} \boldsymbol{traces} \circ \boldsymbol{L2L}_{\beta\delta}(\mathbf{S}).$$

Назовём *строго-конвергентной* такую βδ-трассу σ, которая проходит только через конвергентные состояния, и будем обозначать:

$$\sigma\Downarrow =_{def} \forall \mu \le \sigma \; \forall s \in (\mathbf{S} \; \boldsymbol{after} \; \mu) \; s\downarrow.$$

Множество строго-конвергентных βδ-трасс LTS $\mathbf{S}$ обозначим:

$$\Downarrow(\mathbf{S}) =_{def} \{\sigma \in \mathbf{S} \mid \sigma\Downarrow\}.$$

Определим множество *тестовых трасс* через ***t***-оператор взятия тестовых трасс от заданной трассы (Определение 21:):

$$\boldsymbol{tt}_{\beta\delta}(\mathbf{S}) =_{def} \cup \circ \boldsymbol{t} \circ \Downarrow(\mathbf{S}).$$

Полное условие конформности выглядит аналогично тому, что мы сформулировали в 2.3.5:

$$\boldsymbol{tt}_{\beta\delta}(\mathbf{S}) \cap \boldsymbol{traces}_{\beta\delta}(\mathbf{I}) \subseteq \boldsymbol{traces}_{\beta\delta}(\mathbf{S}) \cap \boldsymbol{tt}_{\beta\delta}(\mathbf{I}).$$

Вся остальная часть теории, включая генерацию тестов, строится аналогично.

Легко видеть, что трасса LTS $\mathbf{S}$ в алфавите C_γ безопасна тогда и только тогда, когда она строго-конвергентна в LTS $\boldsymbol{L}_\gamma\boldsymbol{2L}(\mathbf{S})$:

$$\forall \mathbf{S} \in \boldsymbol{LTS}(C_\gamma) \;\; \boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S}) = \Downarrow \circ \boldsymbol{L}_\gamma\boldsymbol{2L}(\mathbf{S}).$$

Поэтому модель строго-конвергентных трасс, на самом деле, совпадает с моделью безопасных трасс. Различие между безопасными и строго-конвергентными трассами в том, каким способом они выделены среди всех βδ-трасс LTS: с помощью разрушения, моделирующего дивергентность, или с помощью дивергенции, моделирующей разрушение. Заметим также, что в композиции LTS дивергенция ведёт себя аналогично разрушению (асинхронно): если хотя бы одно из состояний LTS-компонентов дивергентно, то композиционное состояние также дивергентно; если хотя бы в одном из состояний LTS-компонентов определён γ-переход, то в композиционном состоянии также будет определён γ-переход. Более того, новая дивергенция, не производная от дивергенции состояний-компонентов, соответствует новому разрушению, которое только моделирует эту новую дивергенцию.

Таким образом, с формальной, математической, точки зрения оба подхода: с разрушением или с дивергенцией, – эквивалентны.

2.4.4. Различение разрушения и дивергенции

Тем не менее, подход с разрушением нам кажется предпочтительнее по следующим причинам.

Во-первых, неформальная семантика разрушения шире семантики дивергенции; она включает вообще любое поведение, которое мы признаём нежелательным, а не только дивергенцию, в том числе и реальное разрушение тестируемой системы.

Во-вторых, модель с разрушением имеет лучшие перспективы в будущем, когда мы захотим тестировать саму дивергенцию. Иными словами, когда дивергенция уже не обязательно признаётся настолько нежелательным поведением, что его нужно избегать при тестировании. Мы уже отмечали в 1.3.1, что дивергенция вовсе не обязательно означает ошибку в тестируемой системе. Иногда дивергенция может быть частью функциональности системы, и требоваться спецификацией. Проблема дивергенции не в том, что это всегда ошибка, а в том, что бывает невозможно отличить дивергенцию от отказа.

Обозначим через λ специальное наблюдение «дивергенция или отказ» [Glab90]. В некоторых случаях можно отличить дивергенцию от отказа. Например, если после посылки стимула $?x$ наблюдается λ, а затем после повторной посылки стимула $?x$ наблюдается сам стимул $?x$, значит λ была дивергенцией. Если в этой ситуации спецификация запрещает дивергенцию, мы фиксируем неконформность.

Другой подход основан на введении приоритетов. Например, мы можем требовать, чтобы приём стимулов (или некоторых стимулов) имел больший приоритет, чем τ-переход, и рассматривать только такие модели реализации и спецификации. Наблюдение λ после посылки такого стимула будет всегда означать отказ. Если после приёма реакций, мы наблюдаем λ, а далее после посылки стимула снова наблюдаем λ, значит первая λ также была отказом. И т.д.

Здесь мы не будем больше развивать идею тестирования дивергенции. Для нас сейчас важно лишь то, что при таком возможном подходе потребуется различать дивергенцию и разрушение: дивергенцию мы будем тестировать, а разрушение, по-прежнему, избегать.

Глава 2.5. Пополнение спецификации

Структура главы:
1. Пополнение состояний
2. Трассовое пополнение
3. Классическая семантика *ioco*
4. Алгоритмизация

В LTS-модели часто недопустимость стимула (отсутствие перехода по стимулу) в том или ином состоянии трактуется как «неспецифицированное» поведение по этому стимулу. При этом предполагается, что для неспецифицированного поведения существует некоторая трактовка, определяющая «умалчиваемое» поведение по этому стимулу (допущение полноты – completeness assumption). Такая трактовка называется пополнением спецификации. Таких пополнений предложено довольно много. Обзор различных видов пополнений можно найти в [BP94],[LY96]. Пополнением по умолчанию можно считать «отсутствие пополнения»: если стимул не допустим, он не принимается. В нестабильном состоянии тупик не возникает, поскольку может быть выполнен τ-переход, а в стабильном состоянии стимул блокируется.

Заметим, что то или иное допущение полноты делается ещё до того, как встаёт вопрос об отношении конформности. Везде, где в предыдущих главах речь шла о LTS- или βγδ-спецификации **S**, теперь мы должны иметь в виду пополненную спецификацию.

С другой стороны, эквивалентные пополнения спецификаций, то есть пополнения, сохраняющие множество конформных реализаций, используются для решения других задач. Ниже мы будем искать решение двух таких задач для отношения ***ioco***: обеспечение рефлексивности и монотонности отношения.

Сначала рассмотрим пополнение состояний LTS, а затем трассовое пополнение.

2.5.1. Пополнение состояний

Определение 79: Для $C \subseteq Z$ пополнением состояний называется преобразование $\boldsymbol{Scomp}: \boldsymbol{LTS}_{\beta\gamma\delta}(C) \to \boldsymbol{LTS}_{\beta\gamma\delta}(C)$, которое удовлетворяет условию: старые состояния и переходы, а также начальное состояние, сохраняются, и любой добавленный переход из старого состояния – это переход по стимулу, по которому раньше не было переходов из этого состояния. Формально:

$\forall \mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$ $\boldsymbol{Scomp}(\mathbf{S}) = \mathbf{M} = LTS(V_M, C_\gamma, E_M, m_0)$, где
$V_S \subseteq V_M$ & $E_S \subseteq E_M$ & $m_0 = s_0$
& $\forall s \in V_S$ $\forall (s,z,m) \in E_M \setminus E_S$ $z \in ?C$ & $\forall s'$ $(s,z,s') \notin E_S$).

□

Теперь мы рассмотрим несколько видов пополнений состояний. Каждый из них представим в двух вариантах в зависимости от того, определяется ли переход по стимулу в любом состоянии, в котором не было такого перехода, или только в стабильном состоянии, то есть в состоянии, где отсутствие перехода по стимулу означает его блокировку. Условие стабильности для второго варианта заключено ниже в квадратных скобках "[]".

Блокировка: $\boldsymbol{Scomp}_\beta(\mathbf{S})$.
Пополнение по умолчанию: стимулы, не специфицированные в старых состояниях, не принимаются: никаких переходов по стимулам из старых состояний не добавляется. LTS (точнее, достижимая её часть) не изменяется, но теперь считается полностью специфицированной. Можно считать, что при таком пополнении LTS не изменяется (если изменения с точностью до изоморфизма не учитывать).

Определение 80: Для $C \subseteq Z$, $\mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$ $\boldsymbol{Scomp}_\beta(\mathbf{S}) = \mathbf{S}$.

□

Теперь рассмотрим три вида пополнений состояний, при которых для неспецифицированного стимула определяется переход по этому стимулу с тем или иным дальнейшим продолжением, которое полностью определено по стимулам, но не приводит к разрушению.

Ангельское пополнение (*angelic completion* в [BRT03]): $\boldsymbol{Scomp}_a(\mathbf{S})$.
Для неспецифицированного стимула определяется переход-петля по этому стимулу: стимул принимается, но игнорируется – не меняет состояние (Рис.66).

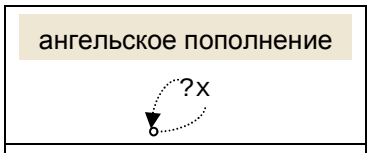


Рис.66.

Определение 81: Для $C \subseteq Z$, $\mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$ $\boldsymbol{Scomp}_a(\mathbf{S}) = \mathbf{M} \in \boldsymbol{LTS}_{\gamma\delta}(C)$, где $V_M = V_S$, $m_0 = s_0$, $E_M = E_S \cup E$, где E – наименьшее множество, порождаемое следующими правилами вывода: $\forall s \in V_S$ $\forall ?x \in C$
$[s \text{—}\tau\!\!\not\rightarrow] \;\&\; s \text{—}?x\!\!\not\rightarrow \;\vdash\; s \text{—}?x\!\rightarrow\! s$.

□

Демоническое пополнение (*demonic completion* [BRT03]): $\boldsymbol{Scomp}_{\mathrm{d}}(\mathbf{S})$.
Для неспецифицированного стимула определяется переход по этому стимулу с произвольным дальнейшим поведением без блокировок и разрушения (Рис.67).

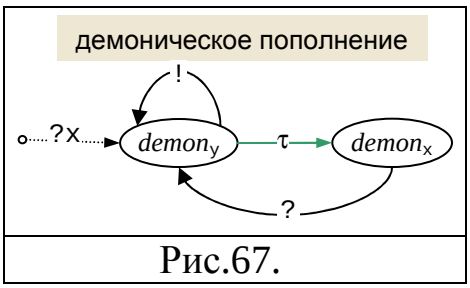


Рис.67.

Определение 82: Для $\mathtt{C}\subseteq\mathtt{Z}$, $\mathbf{S}\in\boldsymbol{LTS}_{\beta\gamma\delta}(\mathtt{C})$ $\boldsymbol{Scomp}_{\mathrm{d}}(\mathbf{S})=\mathtt{M}\in\boldsymbol{LTS}_{\gamma\delta}(\mathtt{C})$, где $\mathtt{V_M}=\mathtt{V_S}\cup\{demon_{\mathtt{Y}},demon_{\mathtt{X}}\}$, где $demon_{\mathtt{Y}},demon_{\mathtt{X}}\notin\mathtt{V_S}$, $\mathtt{m}_0=\mathtt{s}_0$, $\mathtt{E_M}=\mathtt{E_S}\cup\mathtt{E}$, где $\mathtt{E}$ – наименьшее множество, порождаемое следующими правилами вывода:
$\forall \mathtt{s}\in\mathtt{V_S}\ \ \forall ?\mathtt{x},?\mathtt{x}`\in\mathtt{C}\ \ \forall !\mathtt{y}\in\mathtt{C}$

$$[\mathtt{s}\text{—}\tau\not\rightarrow]\ \&\ \mathtt{s}\text{—}?\mathtt{x}\not\rightarrow$$

$$\vdash \mathtt{s}\text{—}?\mathtt{x}\rightarrow demon_{\mathtt{Y}}\text{—}!\mathtt{y}\rightarrow demon_{\mathtt{Y}}\text{—}\tau\rightarrow demon_{\mathtt{X}}\text{—}?\mathtt{x}`\rightarrow demon_{\mathtt{Y}}.$$

□

Реакция об ошибке: $\boldsymbol{Scomp}_{\mathrm{e}}(\mathbf{S})$.
Для неспецифицированного стимула определяется переход в специальное состояние с последующей выдачей реакции об ошибке (Рис.68).

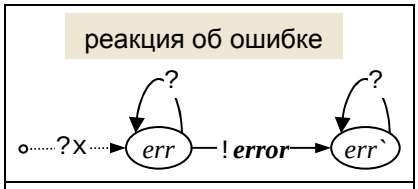


Рис.68.

Определение 83: Для $\mathtt{C}\subseteq\mathtt{Z}$, $\mathbf{S}\in\boldsymbol{LTS}_{\beta\gamma\delta}(\mathtt{C})$ $\boldsymbol{Scomp}_{\mathrm{e}}(\mathbf{S})=\mathtt{M}\in\boldsymbol{LTS}_{\gamma\delta}(\mathtt{C})$, где $\mathtt{V_M}=\mathtt{V_S}\cup\{err,err`\}$, где $err,err`\notin\mathtt{V_S}$, $\mathtt{m}_0=\mathtt{s}_0$, $\mathtt{E_M}=\mathtt{E_S}\cup\mathtt{E}$, где $\mathtt{E}$ – наименьшее множество, порождаемое следующими правилами вывода:
$\forall \mathtt{s}\in\mathtt{V_S}\ \ \forall ?\mathtt{x},?\mathtt{x}`\in\mathtt{C}$

$$[\mathtt{s}\text{—}\tau\not\rightarrow]\ \&\ \mathtt{s}\text{—}?\mathtt{x}\not\rightarrow$$

$$\vdash \mathtt{s}\text{—}?\mathtt{x}\rightarrow err`\text{—}?\mathtt{x}`\rightarrow err`\text{—}!\boldsymbol{error}\rightarrow err`\text{—}?\mathtt{x}`\rightarrow err`.$$

□

Это пополнение можно трактовать двояко:

1) Реакция об ошибке $!\boldsymbol{error}$ добавляется в алфавит реакций реализации. Реализация должна после приёма стимула $?\mathtt{x}$ выдавать такую реакцию и не должна выдавать другую реакцию или переходить в стационарное состояние.
2) Реакция об ошибке $!\boldsymbol{error}$ *не* добавляется в алфавит реакций реализации. Любая реакция или стационарность реализации после приёма стимула $?\mathtt{x}$ считается ошибочной. Это означает, что конформная реализация не должна проходить при тестировании трассу, после которой в спецификации (до пополнения) стимул $?\mathtt{x}$ не специфицирован (ни в каком состоянии).

Гамма-пополнение: $\boldsymbol{Scomp}_{\gamma}(\mathbf{S})$.
Для неспецифицированного стимула определяется переход по этому стимулу, ведущий к разрушению (Рис.69).

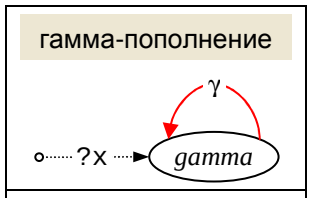


Рис.69.

Определение 84: Для $C \subseteq Z$, $\mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$ $\boldsymbol{Scomp}_{\gamma}(\mathbf{S}) = \mathbf{M} \in \boldsymbol{LTS}_{\gamma\delta}(C)$, где $V_M = V_S \cup \{gamma\}$, где $gamma \notin V_S$, $m_0 = s_0$, $E_M = E_S \cup E$, где E – наименьшее множество, порождаемое следующими правилами вывода: $\forall s \in V_S \ \forall ?x \in C$

$$[s \text{—}\tau\!\!\rightarrow] \ \&\ s \text{—}?x\!\!\rightarrow \ \vdash s \text{—}?x\!\rightarrow gamma \text{—}\gamma\!\rightarrow gamma.$$

□

Возможны комбинации этих вариантов в зависимости от состояния LTS.

Комбинированный вариант 1: рецептивные и нерецептивные состояния. В [HP04] предлагается различать состояния, в которых есть специфицированные стимулы, и состояния, в которых таких стимулов нет (Рис.70). Состояния первого вида можно было бы назвать *рецептивными*: это состояния, в которых ведётся приём стимулов. Допущение полноты требует, чтобы в рецептивном состоянии принимались все стимулы: если какой-то стимул не специфицирован, то считается, что такой стимул принимается с последующим произвольным поведением. В нерецептивных состояниях приём стимулов вообще не ведётся. Если нерецептивное состояние стабильно, то все стимулы блокируются.

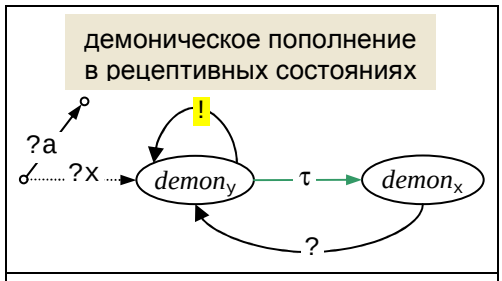


Рис.70.

Формально, для $demon_Y, demon_X \notin V_S$: $V_M = V_S \cup \{demon_Y, demon_X\}$, $m_0 = s_0$, $E_M = E_S \cup E$, где E – наименьшее множество, порождаемое следующими правилами вывода: $\forall s \in V_S \ \forall ?x, ?x` \in C \ \forall !y \in C$

$$\boldsymbol{init}(s) \cap ?C \neq \varnothing \ \&\ s \text{—}?x\!\!\rightarrow$$

$$\vdash s \text{—}?x\!\rightarrow demon_Y \text{—}!y\!\rightarrow demon_Y \text{—}\tau\!\rightarrow demon_X \text{—}?x`\!\rightarrow demon_Y.$$

Комбинированный вариант 2: стационарные и нестационарные состояния. В работах ИСП РАН [KKPPB03] различаются стационарные (нет внутренней активности и выдачи реакций) и нестационарные (есть внутренняя активность или выдача реакций) состояния (Рис.71). В стационарном состоянии неспецифицированный

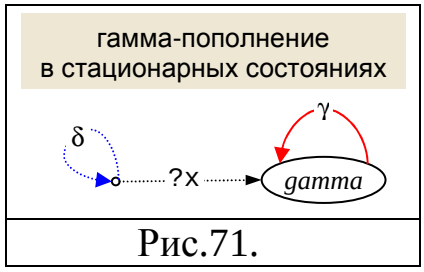


Рис.71.

стимул принимается и ведёт к разрушению, а в нестационарном – не принимается. Если нестационарное состояние стабильно, неспецифицированный стимул блокируется.

Формально, для $gamma \notin V_S$: $V_M = V_S \cup \{gamma\}$, $m_0 = s_0$, $E_M = E_S \cup E$, где E – наименьшее множество, порождаемое следующими правилами вывода: $\forall s \in V_S \ \forall ?x \in C$

$$\delta(s) \ \& \ s \text{—}?x\!\!\rightarrow \ \vdash \ s \text{—}?x \rightarrow gamma \text{—}\gamma \rightarrow gamma.$$

Основанием для такого пополнения является предположение, что система может продолжать свою работу, пока её инициатива не исчерпана: имеются τ-переходы и переходы по реакциям. Иными словами, если системе есть, что делать (внутренние действия или выдача реакций), она не принимает стимул, для которого у неё не определено поведение. Но, если системе нечего делать, кроме как принимать стимулы, она обязана принимать любой стимул, и, если для принятого стимула у неё не определено поведение, это ведёт (более строго, может вести) к разрушению системы. Поскольку мы должны избегать разрушения, тестер должен выдавать только такой стимул, который допустим во всех стационарных состояниях в конце пройденной βγδ-трассы, то есть, (в терминах самой реализации) во всех стационарных состояниях, достижимых из текущего состояния по τ-переходам.

Заметим, что при асинхронном тестировании, когда реализация погружается в тестовый контекст из неограниченных входной и выходной очередей, переходы реализации по реакциям становятся τ-переходами, и любой стимул принимается из тестера (во входную очередь). Теперь тестер должен выдавать только такой стимул, который допустим во всех стационарных состояниях, достижимых из состояний в конце пройденной βγδ-трассы по τ-переходам и переходам по реакциям, то есть, (в терминах самой реализации) во всех стационарных состояниях, достижимых из текущего состояния по τ-переходам и переходам по реакциям.

Более грубым вариантом асинхронного тестирования является, так называемое, стационарное тестирование [BKK03]: стимулы посылаются в реализацию только в стационарных состояниях. В этом случае, фактически, считается, что в нестационарных состояниях стимулы не специфицированы, а в стационарных – специфицированы: безопасно принимаются, принимаются с возможным разрушением или блокируются.

2.5.2. Трассовое пополнение

Определение 85: Пусть $C \subseteq Z$.

- Преобразование $\boldsymbol{Comp}:\boldsymbol{MODEL}_{\beta\gamma\delta}(C)\to\boldsymbol{MODEL}_{\beta\gamma\delta}(C)$ назовём *трассовым пополнением*, если существует такое пополнение состояний ***Scomp***, что: $\forall\boldsymbol{\Sigma}\in\boldsymbol{MODEL}_{\beta\gamma\delta}(C)\ \exists\mathbf{S}\in\boldsymbol{LTS}_{\beta\gamma\delta}(C)$ $\boldsymbol{\Sigma}=\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})\ \&\ \boldsymbol{Comp}(\boldsymbol{\Sigma})=\boldsymbol{traces}_{\beta\gamma\delta}\circ\boldsymbol{Scomp}(\mathbf{S})$.
- Тождественное трассовое пополнение будем называть *пополнением по умолчанию*.
- Соответственно, преобразования
  a) $\boldsymbol{TT}:\boldsymbol{MODEL}_{\beta\gamma\delta}\to\boldsymbol{MODEL}_{\beta\gamma\delta}$,
  b) $\boldsymbol{TL}:\boldsymbol{MODEL}_{\beta\gamma\delta}\to\boldsymbol{LTS}_{\beta\gamma\delta}$,
  c) $\boldsymbol{LT}:\boldsymbol{LTS}_{\beta\gamma\delta}\to\boldsymbol{MODEL}_{\beta\gamma\delta}$,
  d) $\boldsymbol{LL}:\boldsymbol{LTS}_{\beta\gamma\delta}\to\boldsymbol{LTS}_{\beta\gamma\delta}$
  будем также называть трассовыми пополнениями, если множество βγδ-трасс результата является трассовым пополнением множества βγδ-трасс аргумента: существует такое трассовое пополнение ***Comp*** βγδ-моделей, что
  a) $\boldsymbol{TT}(\boldsymbol{\Sigma})=\boldsymbol{Comp}(\boldsymbol{\Sigma})$,
  b) $\boldsymbol{traces}_{\beta\gamma\delta}\circ\boldsymbol{TL}(\boldsymbol{\Sigma})=\boldsymbol{Comp}(\boldsymbol{\Sigma})$,
  c) $\boldsymbol{LT}(\mathbf{S})=\boldsymbol{Comp}\circ\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$,
  d) $\boldsymbol{traces}_{\beta\gamma\delta}\circ\boldsymbol{LL}(\mathbf{S})=\boldsymbol{Comp}\circ\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$.

□

Здесь мы отметим лишь некоторые свойства трассового пополнения: какие βγδ-трассы могут быть добавлены, а какие удалены.

Любой переход $s—z\to s`$ по стимулу или реакции $z$ в LTS может быть заменён на два перехода, введением промежуточного состояния $(s,z,s`)$: $s—z\to(s,z,s`)—\tau\to s`$. Очевидно, множество βγδ-трасс при этом не изменяется. Заметим, что в добавленном состоянии не определён переход по любому стимулу $?x$, и, следовательно, при пополнении он может добавиться. Поэтому для любой βγδ-модели $\boldsymbol{\Sigma}$, любой её βγδ-трассы μ, не заканчивающейся на отказ, и любого стимула $?x$ существует LTS $\mathbf{S}$, имеющая то же множество βγδ-трасс, $\boldsymbol{\Sigma}=\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$, при пополнении которой βγδ-трасса μ может продолжиться стимулом $?x$.

С другой стороны, если βγδ-трасса μ заканчивается на отказ, то все состояния после трассы стабильны. Чтобы продолжить такую трассу стимулом $?x$, нужно, чтобы хотя бы в одном из этих стабильных состояний не было перехода по стимулу $?x$, то есть была блокировка $\{?x\}$. Следовательно, трассу, заканчивающуюся на отказ, может продолжить стимулом только в том случае, когда до пополнения она продолжалась блокировкой $\{?x\}$.

Наконец, при любом пополнении состояний никакие переходы не удаляются, а добавляется только такой переход из старого состояния, который является переходом по стимулу, по которому до пополнения из этого состояния не было перехода. Если состояние стабильно, то при таком добавлении в нём исчезает блокировка стимула. Тем самым, при пополнении может быть удалена только такая βγδ-трасса исходной LTS, которая имеет вид $\mu\cdot\langle\{?x\}\rangle\cdot\lambda$, где βγδ-трасса μ сохраняется и появляется (или была раньше) βγδ-трасса $\mu\cdot\langle ?x\rangle$.

Эти свойства необходимы, что доказывается следующим утверждением.

Утверждение 49: Для $C\subseteq Z$ любое пополнение состояний $\boldsymbol{Scomp}:\boldsymbol{LTS}_{\beta\gamma\delta}(C)\to\boldsymbol{LTS}_{\beta\gamma\delta}(C)$ удовлетворяет следующим свойствам: для $S\in\boldsymbol{LTS}_{\beta\gamma\delta}(C)$, $\Sigma=\boldsymbol{traces}_{\beta\gamma\delta}(S)$ и $T=\boldsymbol{traces}_{\beta\gamma\delta}\circ\boldsymbol{Scomp}(S)$

$$T\backslash\Sigma \subseteq \{\ \mu\cdot\langle ?x\rangle\cdot\lambda \mid \mu\in\Sigma\ \&\ \boldsymbol{RefT}(\mu)\neq\epsilon \Rightarrow \mu\cdot\langle\{?x\}\rangle\in\Sigma\ \}$$

$$\&\ \Sigma\backslash T \subseteq \{\ \mu\cdot\langle\{?x\}\rangle\cdot\lambda \mid \mu\cdot\langle ?x\rangle\in T\ \}.$$

□404

Вместе с тем, рассматриваемые свойства не являются достаточными: контрпример на Рис.72.

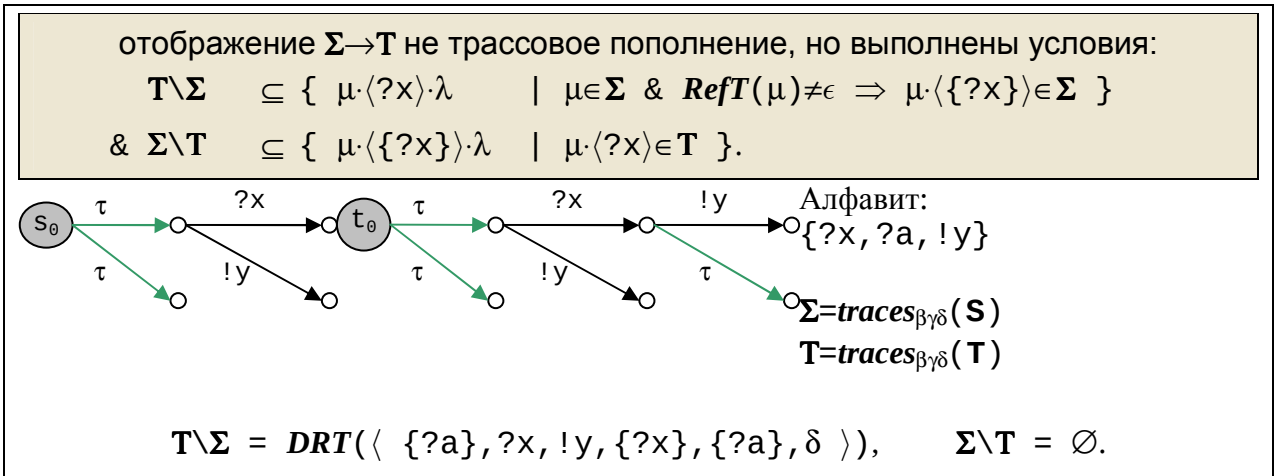


Рис.72.

Остаётся нерешённой проблема существования для заданной βγδ-модели Σ «универсальной» LTS S с множеством βγδ-трасс $\boldsymbol{traces}_{\beta\gamma\delta}(S)=\Sigma$ такой, что любое трассовое пополнение $\boldsymbol{Comp}$ βγδ-модели Σ является некоторым пополнением состояний $\boldsymbol{Scomp}$ этой «универсальной» LTS: $\boldsymbol{Comp}(\Sigma)=\boldsymbol{traces}_{\beta\gamma\delta}\circ\boldsymbol{Scomp}(S)$.

В следующем разделе мы рассмотрим один вид трассового пополнения, а именно, пополнения, удаляющего все блокировки и сохраняющего классическую семантику отношения ***ioco***: пополненная спецификация ***ioco***-эквивалентна исходной и не содержит блокировок стимулов.

2.5.3. Классическая семантика *ioco*

Структура раздела:
- Определение отношения *ioco*.
- Постановка задачи трассового пополнения модели.
- Несохранение *ioco* при пополнении состояний.
- Демоническое и гамма-пополнения состояний.
- Решение проблемы ослабления соответствия: базовое пополнение.
- Удаление блокировок после базового пополнения.
- Трассовые пополнения, сохраняющие *ioco*.
- Сравнение демонического и гамма-пополнений.
- Расширения семантики *ioco*.

Определение отношения *ioco*.

Классическое отношение ***ioco*** определяется для трасс в алфавите C_δ, которые называются трассами с задержками (suspension traces). Мы будем называть эти трассы δ-трассами.

В реализации и спецификации нет разрушения, в том числе и дивергенции (строго-конвергентная модель).

Реализация, кроме того, не должна блокировать стимулы, то есть, должна быть полностью (во всех достижимых стабильных состояниях) определена по стимулам (input-enabled).

Спецификация, напротив, может быть определена частично.

Однако, если в спецификации δ-трасса μ продолжается как стимулом `?x`, так и его блокировкой `{?x}`, то блокировка игнорируется (в конформной реализации δ-трасса μ продолжается только стимулом `?x`).

Если же в спецификации δ-трасса μ не продолжается стимулом `?x`, то есть продолжается только его блокировкой `{?x}`, то используется, так называемое, *слабое отношение* [BP94] и, соответственно, *слабое тестирование* [LY96]: не проверяется поведение реализации по стимулу

?x, не специфицированному после δ-трассы μ, то есть, тестер не посылает в реализацию такой стимул после δ-трассы μ.

Правила отношения ***ioco*** имеют вид "*Реализация может … только тогда, когда для спецификации это возможно в такой же ситуации, то есть, после той же самой δ-трассы*", где многоточие означает одно из следующих:

Правило реакции: "выдать данную реакцию".

Правило стационарности: "не выдавать никаких реакций".

Формально отношение ***ioco*** определяется для трассовой и LTS-моделей следующим образом:

Определение 86: Пусть $C \subseteq Z$.

1. Для трассовой спецификации без разрушения $\mathbf{\Sigma} \in \boldsymbol{MODEL}_{\beta\delta}(C)$ и полностью определённой по стимулам реализации без разрушения $\mathbf{I} \in \boldsymbol{MODEL}_{\delta}(C)$

   $\mathbf{I}$ ***ioco*** $\mathbf{\Sigma} =_{def}$

   $\forall \mu \in \mathbf{\Sigma} \cap \mathbf{I}\ \ \mu \cdot \langle \delta \rangle \in \mathbf{I} \Rightarrow \mu \cdot \langle \delta \rangle \in \mathbf{\Sigma}\ \ \&\ \ \forall !y \in C\ \ \mu \cdot \langle !y \rangle \in \mathbf{I} \Rightarrow \mu \cdot \langle !y \rangle \in \mathbf{\Sigma}.$
2. Для LTS-спецификации без разрушения[55] $\mathbf{S} \in \boldsymbol{LTS}_{\beta\delta}(C)$ и полностью определённой по стимулам реализации без разрушения $\mathbf{I} \in \boldsymbol{LTS}_{\delta}(C)$

   $\mathbf{I}$ ***ioco*** $\mathbf{S}$ $=_{def}$ $\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{I})$ ***ioco*** $\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$

□

Для спецификаций без разрушения и блокировок (полностью определённых по стимулам), очевидно, отношения ***ioco*** и $\boldsymbol{ioco}_{\beta\gamma\delta}$ совпадают. По Утверждение 24:, реализации без разрушения *S*-тестируемы при любой спецификации, а если спецификации без разрушения, то отношение $\boldsymbol{ioco}_{\beta\gamma\delta}$ эквивалентно вложенности βδ-трасс. Тем самым, для спецификаций без разрушения и блокировок отношения ***ioco*** и $\boldsymbol{ioco}_{\beta\gamma\delta}$ эквивалентны вложенности δ-трасс.

Постановка задачи трассового пополнения модели.

Поскольку отношение ***ioco*** использует трассы без блокировок и разрушения, естественно возникает задача преобразования (трассового пополнения) спецификации без разрушения в ***ioco***-эквивалентную ей спецификацию без блокировок и разрушения. На самом деле, кроме "естественности", есть две более важные причины решать эту задачу:

---

[55] Поскольку дивергенция является частным случаем разрушения, отсутствие разрушения в LTS-модели означает, в том числе, её строгую конвергентность.

1) Нерефлексивность отношения ***ioco***: если спецификация имеет блокировки стимулов, то она не конформна сама себе. С практической точки зрения это очень неудобно: казалось бы, если реализация написана как "калька" со спецификации, то всё должно быть правильно. В то же время, полностью определённая по стимулам (input-enabled) спецификация конформна сама себе.
2) Немонотонность отношения ***ioco***: композиция LTS-реализаций, конформных LTS-спецификациям с блокировками стимулов, может быть неконформна композиции этих спецификаций. В то же время, для полностью определённых по стимулам спецификаций отношение ***ioco*** оказывается монотонным. Это будет показано ниже в Часть 3. Верификация композиции для отношения $\boldsymbol{ioco}_{\beta\gamma\delta}$, совпадающего для таких реализаций и спецификаций с отношением ***ioco***.

Мы сформулируем эту задачу для трассового пополнения типа $\boldsymbol{TL}:\boldsymbol{MODEL}_{\beta\delta}\rightarrow\boldsymbol{LTS}_{\delta}$, а затем рассмотрим её для остальных типов трассовых пополнений ***TT***, ***LL*** и ***LT***.

Постановка задачи:

для трассовой спецификации без разрушения $\mathbf{\Sigma}\in\boldsymbol{MODEL}_{\beta\delta}(\mathtt{C})$ найти такую полностью определённую по стимулам LTS-спецификацию без разрушения $\mathtt{S`}\in\boldsymbol{LTS}_{\delta}(\mathtt{C`})$[56], чтобы $\mathbf{I}\ \boldsymbol{ioco}\ \mathbf{\Sigma}$ было эквивалентно $\mathbf{I}\ \boldsymbol{ioco}\ \boldsymbol{traces}_{\beta\gamma\delta}(\mathtt{S`})$.

Тогда $\mathbf{I}\ \boldsymbol{ioco}\ \mathbf{\Sigma} \Leftrightarrow \mathbf{I}\ \boldsymbol{ioco}\ \boldsymbol{traces}_{\beta\gamma\delta}(\mathtt{S`}) \Leftrightarrow \mathbf{I}\ \boldsymbol{ioco}_{\beta\gamma\delta}\ \boldsymbol{traces}_{\beta\gamma\delta}(\mathtt{S`}) \Leftrightarrow \mathbf{I} \subseteq \boldsymbol{traces}_{\beta\gamma\delta}(\mathtt{S`})$.

Мы покажем, что такая LTS-спецификация $\mathtt{S`}$ существует и является ***TL***-пополнением трассовой спецификации $\mathbf{\Sigma}$. Теперь задача формулируется так: найти такое ***TL***-пополнение ***TLDemonic***, что

(1) для $\mathbf{\Sigma}\in\boldsymbol{MODEL}_{\beta\delta}(\mathtt{C})$ $\boldsymbol{TLDemonic}(\mathbf{\Sigma})\in\boldsymbol{LTS}_{\delta}(\mathtt{C`})$
и для $\mathbf{I}\in\boldsymbol{MODEL}_{\delta}(\mathtt{C})$
$\mathbf{I}\ \boldsymbol{ioco}\ \mathbf{\Sigma} \Leftrightarrow \mathbf{I}\ \boldsymbol{ioco}\ \boldsymbol{traces}_{\beta\gamma\delta\circ\boldsymbol{TLDemonic}}(\mathbf{\Sigma}) \Leftrightarrow \mathbf{I}\ \boldsymbol{ioco}_{\beta\gamma\delta}\ \boldsymbol{traces}_{\beta\gamma\delta\circ\boldsymbol{TLDemonic}}(\mathbf{\Sigma}) \Leftrightarrow \mathbf{I} \subseteq \boldsymbol{traces}_{\beta\gamma\delta\circ\boldsymbol{TLDemonic}}(\mathbf{\Sigma})$.

Вместе с тем, отношение $\boldsymbol{ioco}_{\beta\gamma\delta}$ может применяться и к тем реализациям, в которых есть блокировки и разрушение (при условии выполнения гипотезы

---

[56] Мы допускаем расширение алфавита при пополнении $\mathtt{C}\subset\mathtt{C`}$. Необходимость этого будет показана ниже.

о безопасности). Поэтому естественно рассматривать такое ***TL***-пополнение, которое может добавлять разрушение в спецификацию:

(2) для $\mathbf{\Sigma} \in \boldsymbol{MODEL}_{\beta\delta}(C)$ $\boldsymbol{TLGamma}(\mathbf{\Sigma}) \in \boldsymbol{LTS}_{\gamma\delta}(C`)$
и для $\mathbf{I} \in \boldsymbol{MODEL}_{\delta}(C)$
$\mathbf{I}$ ***ioco*** $\mathbf{\Sigma}$ $\Leftrightarrow$ $\mathbf{I}$ $\boldsymbol{ioco}_{\beta\gamma\delta}$ $\boldsymbol{traces}_{\beta\gamma\delta\circ}\boldsymbol{TLGamma}(\mathbf{\Sigma})$.

Несохранение *ioco* при пополнении состояний.

Прежде всего отметим, что все рассмотренные выше пополнения состояний $\boldsymbol{Scomp}_{\ldots}$, удаляющие все блокировки стимулов, не сохраняют семантику ***ioco***: множество реализаций, ***ioco***-соответствующих $\mathbf{S}$, не совпадает с множеством реализаций, ***ioco***-соответствующих $\boldsymbol{Scomp}_{\ldots}(\mathbf{S})$.

Здесь выделяются два случая интерпретации блокировки {?x} после δ-трассы μ – в зависимости от того, A) продолжается δ-трасса самим стимулом ?x или B) нет.

A) Сначала рассмотрим интерпретацию блокировки {?x} после трассы μ, которая продолжается как стимулом ?x, так и его блокировкой {?x}.

Все рассмотренные выше пополнения состояний, кроме одного случая, ослабляют соответствие.

Демоническое пополнение допускает любое поведение реализации после трассы $\mu\cdot\langle ?x\rangle$, хотя в исходной спецификации могло допускаться не любое такое поведение.

После ангельского пополнения дополнительно допускается после трассы $\mu\cdot\langle ?x\rangle$ то поведение, которое в исходной спецификации было после трассы $\mu\cdot\langle \{?x\}\rangle$, то есть, после трассы μ в состоянии, где не был определён переход по стимулу ?x, хотя такого поведения могло не быть в исходной спецификации после трассы $\mu\cdot\langle ?x\rangle$.

Пополнение «реакция об ошибке» имеет две трактовки.

1) Реакция об ошибке $!\boldsymbol{error}$ добавляется в алфавит стимулов и реакций реализации. В этом случае дополнительно допускается трасса $\mu\cdot\langle ?x\rangle\cdot\langle !\boldsymbol{error}\rangle$, хотя такого поведения могло не быть в исходной спецификации. Это ослабление соответствия.

2) Реакция об ошибке $!\boldsymbol{error}$ *не* добавляется в алфавит стимулов и реакций реализации. В этом случае реализация не может выдать $!\boldsymbol{error}$, и *нет ослабления* соответствия.

После γ-пополнения не проверяется (и не тестируется) поведение реализации после трассы $\mu\cdot\langle ?x\rangle$, хотя в исходной спецификации могло допускаться не любое такое поведение.

В) Теперь рассмотрим интерпретацию блокировки $\{?x\}$ после трассы μ, которая не продолжается стимулом $?x$.

Все рассмотренные выше пополнения состояний, кроме двух, усиливают соответствие.

Демоническое пополнение для полностью определённых по стимулам (input enabled) реализаций *не усиливает* соответствие, поскольку допускается любое поведение реализации после трассы $\mu\cdot\langle ?x\rangle$, а до пополнения вообще не проверялось, какое поведение имеет реализация после этой трассы (стимул не посылался в реализацию при тестировании).

Ангельское пополнение усиливает соответствие: после трассы $\mu\cdot\langle ?x\rangle$ допускается только то поведение, которое в исходной спецификации было после трассы μ.

Пополнение «реакция об ошибке» усиливает соответствие:
1) Если реакция об ошибке $!\boldsymbol{error}$ добавляется в алфавит стимулов и реакций реализации, то теперь после трассы $\mu\cdot\langle ?x\rangle$ допускается единственное поведение – реакция $!\boldsymbol{error}$.
2) Если реакция об ошибке $!\boldsymbol{error}$ *не* добавляется в алфавит стимулов и реакций реализации, то теперь любое поведение после трассы $\mu\cdot\langle ?x\rangle$ не допускается.

γ-пополнение *не усиливает* соответствие: не проверяется стимул $?x$ после трассы μ, поскольку он разрушающий после этой трассы и его запрещено посылать в реализацию при тестировании.

Демоническое и гамма-пополнения состояний.

Таким образом, единственные пополнения состояний, которые *не усиливают* соответствие, – это демоническое пополнение $\boldsymbol{Scomp}_{\mathrm{d}}$ и гамма-пополнение $\boldsymbol{Scomp}_{\gamma}$.

Мы будем рассматривать пополнения состояний, которые определяют переход по стимулу только в стабильном состоянии, то есть в состоянии, где отсутствие перехода по стимулу означает его блокировку.

Сравним эти два вида пополнения состояний: демоническое и гамма-пополнение.

Демоническое пополнение состояний эквивалентно использованию для соответствия ***ioco*** не всех δ-трасс, а только их подмножества ***Utraces*** [BRT03]. Это такие δ-трассы, в которых каждый префикс, продолжаемый в трассе стимулом, не имеет в спецификации продолжения блокировкой:
для $\mathbf{S}\in \boldsymbol{LTS}_{\beta\delta}(\mathtt{C})$ $\boldsymbol{Utraces}(\mathbf{S}) =_{\mathrm{def}}$
$\{\sigma\in \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S}) \mid \forall ?x\in \mathtt{C}\ \ \forall \mu\ \ \mu\cdot\langle ?x\rangle\le\sigma \ \Rightarrow\ \mu\cdot\langle\{?x\}\rangle\notin \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})\}$.

Очевидно, это эквивалентно использованию для соответствия ***ioco*** безопасных δ-трасс после гамма-пополнения состояний спецификации, поскольку $\boldsymbol{Utraces}(\mathbf{S})=\boldsymbol{safe}_{\beta\gamma\delta}\circ\boldsymbol{Scomp}_{\gamma}(\mathbf{S})$ для $\mathbf{S}\in \boldsymbol{LTS}_{\beta\delta}(\mathtt{C})$. Гамма-пополнение и использование затем для тестирования его безопасных трасс удобно ещё и тем, что мы сохраняем информацию о тех стимулах, которыми не продолжались δ-трассы: теперь эти стимулы принимаются с дальнейшим разрушением, в то время как при демоническом пополнении или при использовании ***Utraces*** эта информация теряется [BRT03]. Поясним это на примерах.

A) Пусть трасса μ продолжалась в $\mathbf{S}$ как стимулом $?x$, так и его блокировкой $\{?x\}$.
   - A1) После демонического пополнения трасса μ продолжается стимулом $?x$ и далее произвольным поведением: $\mu\cdot\langle ?x\rangle\cdot\lambda$, где λ любая согласованная δ-трасса.
   - A2) В $\boldsymbol{Utraces}(\mathbf{S})$ трасса μ не продолжается стимулом $?x$, а блокировок вообще нет в таких трассах.
   - A3) После гамма-пополнения трасса μ продолжается стимулом $?x$ и далее тем поведением, которого после него было до пополнения, а также разрушением:
     $\mu\cdot\langle ?x\rangle\cdot\lambda$, где $\lambda\in(\ \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})\ \boldsymbol{after}\ \mu\cdot\langle ?x\rangle\ )$, и $\mu\cdot\langle ?x,\gamma\rangle$.

Во всех случаях A1÷3 после $\mu\cdot\langle ?x\rangle$ в реализации допускается любое поведение, что как раз и означает ослабление соответствия для случая A.

B) Пусть трасса μ не продолжалась в **S** стимулом ?x, то есть продолжалась только его блокировкой {?x}.
   - B1) После демонического пополнения трасса μ продолжается стимулом ?x и далее произвольным поведением: $\mu\cdot\langle ?x\rangle\cdot\lambda$, где λ любая согласованная δ-трасса.
   - B2) В ***Utraces*(S)** трасса μ не продолжается стимулом ?x, а блокировок вообще нет в таких трассах.
   - B3) После гамма-пополнения трасса μ продолжается стимулом ?x и далее разрушением: $\mu\cdot\langle ?x,\gamma\rangle$.

   Во всех случаях B1÷3 после $\mu\cdot\langle ?x\rangle$ в реализации допускается любое поведение, что точно соответствует семантике ***ioco*** для исходной спецификации.

Мы видим, что случаи A1 и B1 неразличимы, также неразличимы случаи A2 и B2. Однако, случаи A3 и B3 различаются: если трасса $\mu\cdot\langle ?x\rangle$ после пополнения продолжается только разрушением, то это случай B3) – до пополнения трасса μ не продолжалась стимулом ?x, а если трасса $\mu\cdot\langle ?x\rangle$ после пополнения продолжается не только разрушением, то это случай A3 – до пополнения трасса μ продолжалась стимулом ?x.

Если нам удастся избежать ослабления соответствия (в случаях A1÷3), то в случаях B1÷3 при демоническом пополнении и при использовании ***Utraces*(S)** мы всё равно будем вынуждены проводить избыточное тестирование. После пополнения мы не различаем то поведение, которое было до пополнения и то, которое добавлено при пополнении. В отличие от этого, при гамма-пополнении мы эти поведения различаем: было поведение без разрушения, а добавлено разрушение. Мы будем тестировать только поведение без разрушения.

Решение проблемы ослабления соответствия: базовое пополнение.

Итак, демоническое и гамма-пополнения ослабляют, но не усиливают соответствие. Причина ослабления – в использовании пополнения состояний, а не трассового пополнения. Теперь нам нужно построить соответствующие трассовые пополнения спецификации.

Каждое из трассовых пополнений ***TLDemonic*** и ***TLGamma*** мы представим в виде композиции преобразований $\boldsymbol{TLDemonic=Scomp_d \circ Comp_0}$ и $\boldsymbol{TLGamma=Scomp_\gamma \circ Comp_0}$

Преобразование $\boldsymbol{Comp_0}(\Sigma): \boldsymbol{MODEL}_{\beta\delta}(C) \to \boldsymbol{LTS}_{\beta\delta}(C`)$ выполняет «обязательную» часть работы: после этого преобразования никакая δ-трасса не продолжается и стимулом и его блокировкой. Тем самым устраняется случай A, при котором возникало ослабление соответствия.

Блокировки стимулов, неспецифицированных после δ-трасс, удаляются дальнейшем пополнением состояний: демоническим пополнением $\boldsymbol{Scomp_d}$ или гамма-пополнением $\boldsymbol{Scomp_\gamma}$.

Заметим, что преобразование $\boldsymbol{Comp_0}$, вообще говоря, не является трассовым пополнением, но композиция ***Demonic*** и ***Gamma*** будут трассовыми пополнениями.

Идея преобразования $\boldsymbol{Comp_0}$ заключается в следующем.

A. Пусть δ-трасса μ продолжается в **Σ** как стимулом ?x, так и его блокировкой {?x} (Рис.73).

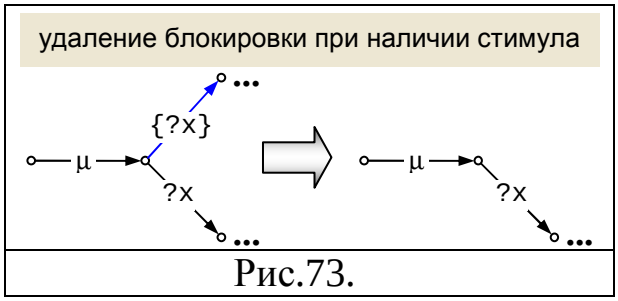


Рис.73.

Отношение ***ioco*** для такой δ-трассы игнорирует её продолжение блокировкой.

Поэтому преобразование $\boldsymbol{Comp_0}$ удаляет эту блокировку, то есть удаляет все βγδ-трассы вида $\mu\cdot\langle\{?x\}\rangle\cdot\lambda$.

B. Теперь пусть как δ-трасса μ, так и её подтрасса базовых символов, получаемая удалением стационарности, $\mu_0=\mu\uparrow\{\delta\}$ не продолжаются в **Σ** стимулом ?x (Рис.74).

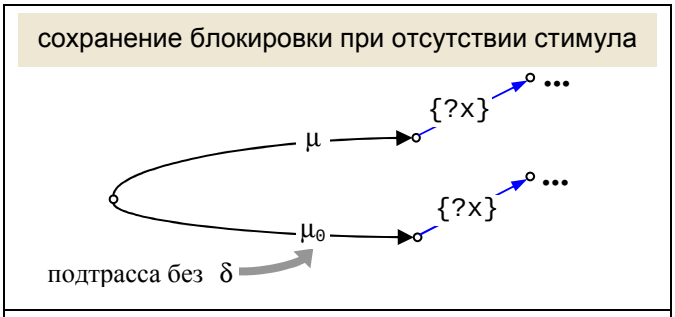


Рис.74.

В этом случае преобразование $\boldsymbol{Comp_0}$ сохраняет блокировку {?x} после δ-трассы μ.

C. Наконец, пусть δ-трасса μ не продолжается в $\mathbf{\Sigma}$ стимулом $?x$, но её подтрасса базовых символов, получаемая удалением стационарности, $\mu_0=\mu\uparrow\{\delta\}$ продолжается в $\mathbf{\Sigma}$ стимулом $?x$ (Рис.75).

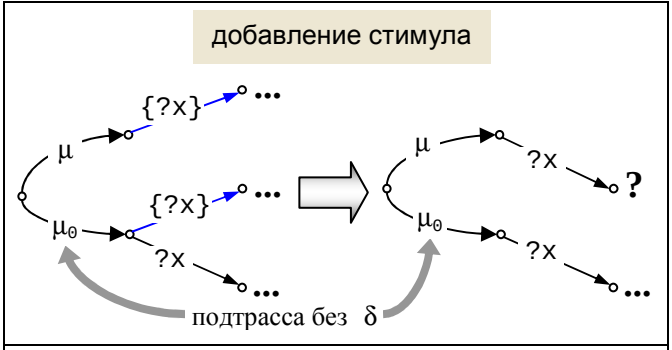


Рис.75.

Рассмотрим случай C подробно.

В этом случае, поскольку δ-трасса μ продолжается в $\mathbf{\Sigma}$ блокировкой $\{?x\}$, трасса μ_0 продолжается в $\mathbf{\Sigma}$ как стимулом $?x$, так и его блокировкой $\{?x\}$.

Мы должны определить продолжения δ-трассы μ стимулом $?x$, то есть добавить βγδ-трассы вида $\mu\cdot\langle ?x\rangle\cdot\lambda$, удалив блокировку $\{?x\}$ после этой δ-трассы. При этом не должно происходить усиление или ослабление соответствия.

Рассмотрим подтрассы $\mu_0,\mu_1,\mu_2,\ldots$, которые получаются из μ удалением некоторых стационарностей и которые продолжаются в $\mathbf{\Sigma}$ стимулом $?x$.

Если бы для некоторого i δ-трасса $\lambda\notin(\mathbf{\Sigma}\ \textbf{\textit{after}}\ \mu_i\cdot\langle ?x\rangle)$, то после добавления трассы $\mu\cdot\langle ?x\rangle\cdot\lambda$ стало бы $\lambda\in(\textbf{\textit{traces}}_{\beta\gamma\delta^\circ}\textbf{\textit{Comp}}_0(\mathbf{\Sigma})\ \textbf{\textit{after}}\ \mu_i\cdot\langle ?x\rangle)$. Тем самым, произошло бы *ослабление* соответствия, поскольку после преобразования мы разрешили бы для реализации после δ-трассы $\mu_i\cdot\langle ?x\rangle$ новое поведение λ, которое не разрешалось до преобразования.

Наоборот, если бы некоторая δ-трасса $\lambda`\in\cap(\{\mathbf{\Sigma}\ \textbf{\textit{after}}\ \mu_i\cdot\langle ?x\rangle\,|\,i=0,1,2,\ldots\}$, продолжавшая до преобразования все δ-трассы $\mu_i\cdot\langle ?x\rangle$, не была бы при преобразовании добавлена после δ-трассы $\mu\cdot\langle ?x\rangle$, то есть $\lambda`\notin(\textbf{\textit{traces}}_{\beta\gamma\delta^\circ}\textbf{\textit{Comp}}_0(\mathbf{\Sigma})\ \textbf{\textit{after}}\ \mu\cdot\langle ?x\rangle)$, то произошло бы *усиление* соответствия, поскольку конформная реализация может иметь δ-трассу $\mu\cdot\langle ?x\rangle\cdot\lambda`$ (до преобразования стимул $?x$ не проверялся после δ-трассы

μ), а после преобразования мы будем ловить ложную ошибку при прохождении такой δ-трассы.

Таким образом, мы должны продолжить δ-трассу μ стимулом ?x так, чтобы продолжения после стимула были теми и только теми δ-трассами, которые до преобразования продолжали все δ-трассы $\mu_i\cdot\langle ?x\rangle$ (Рис.76):

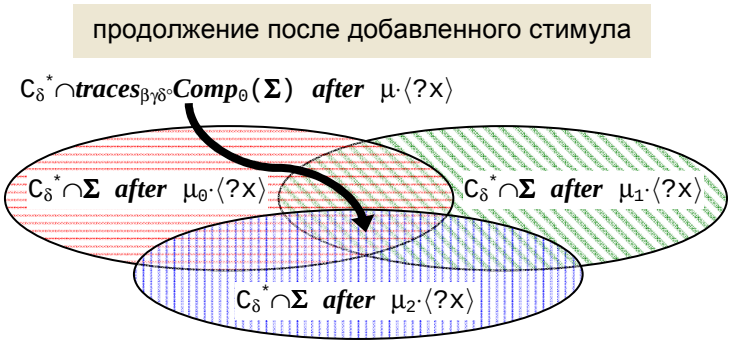


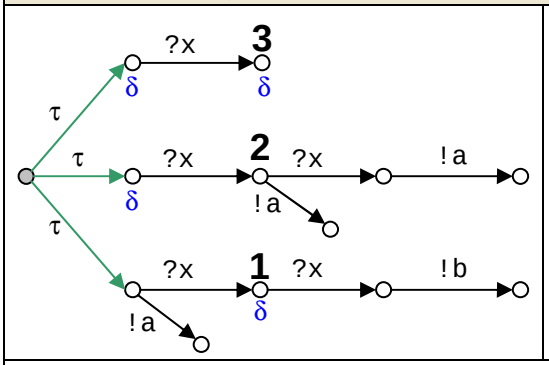
Рис.76.

$$C_\delta^*\cap traces_{\beta\gamma\delta^\circ}Comp_0(\Sigma)\ \ after\ \ \mu\cdot\langle ?x\rangle$$
$$= \cap\{C_\delta^*\cap\Sigma\ \ after\ \ \mu_i\cdot\langle ?x\rangle \mid i=0,1,2,...\}.$$

Если после δ-трассы из этого множества некоторый стимул не специфицирован, то блокировка такого стимула будет в дальнейшем удаляться с помощью одного из пополнений состояний $Scomp_d$ или $Scomp_\gamma$. Поэтому, если каждая δ-трасса этого множества конвергентна по реакциям, то есть продолжается хотя бы одной реакцией или стационарностью, то всё хорошо.

Однако, это не всегда так. Пример приведён на Рис.77.

пересечение продолжений подтрасс, получающихся удалением из трассы μ некоторых δ, может быть неконвергентно по реакциям



Алфавит: {?x,!a,!b} LTS **S**
μ = ⟨δ,?x,δ,?x⟩

1: $traces_{\gamma\delta}$**(S)** ***after*** ⟨?x,δ,?x⟩ = ⟨!b⟩·{δ}*
2: $traces_{\gamma\delta}$**(S)** ***after*** ⟨δ,?x,?x⟩ = ⟨!a⟩·{δ}*

1∩2: = ∅

Рис.77.

В этом примере δ-трасса ⟨δ,?x,δ⟩ заканчивается в состоянии 3, где не определён стимул ?x, а её подтрассы μ_2=⟨δ,?x⟩ и μ_1=⟨?x,δ⟩ заканчиваются, кроме состояния 3, в состояниях 2 и 1, соответственно, где стимул ?x определён. Однако, в состоянии 2 после стимула ?x

определена реакция $!a$, а в состоянии 1 – реакция $!b$. Тем самым, никакой общей реакции или стационарности нет, и в пересечении $\mathbf{\Sigma}\ \boldsymbol{after}\ \mu_2$ и $\mathbf{\Sigma}\ \boldsymbol{after}\ \mu_1$ пустая трасса не конвергентна по реакциям. Эта неконвергентность сохраняется и при дальнейшем пересечении с $\mathbf{\Sigma}\ \boldsymbol{after}\ \mu_0$, где подтрасса $\mu_0=\langle ?x\rangle$.

Что означает эта неконвергентность?

Она означает, что ни одна конформная реализация не может иметь δ-трассу $\langle \delta, ?x, \delta\rangle$, хотя такая δ-трасса есть в спецификации. По сути, эта δ-трасса просто не согласована с другими δ-трассами спецификации.

Что нужно делать?

Мы могли бы вообще удалить из спецификацию «лишнюю» δ-трассу $\langle \delta, ?x, \delta\rangle$. Однако, при итеративном построении преобразования заранее неизвестно, является эта δ-трасса лишней или нет. В нашем примере мы могли бы после состояний 1 и 2 между стимулом $?x$ и реакциями $!a$ и $!b$, соответственно, расположить сколь угодно длинную δ-трассу – одну и ту же в обоих случаях. Только после прохождения этой δ-трассы (а не пустой трассы, как на Рис.77) в процессе построения преобразования мы обнаружим неконвергентность по реакциям. Поэтому, если мы хотим строить преобразование алгоритмически, это решение не годится.

Предлагается другое решение:

1) Определяем стимул после «лишней» δ-трассы и строим пересечение $\cap\{C_\delta{}^*\cap\mathbf{\Sigma}\ \boldsymbol{after}\ \mu_i\cdot\langle ?x\rangle \mid i=0,1,2,...\}$ до тех пор, пока не будет обнаружена неконвергентность по реакциям: некоторая построенная δ-трасса не может быть продолжена ни какой-либо реакцией, ни стационарностью.
2) Далее продолжаем неконвергентную по реакциям δ-трассу специальной ошибочной реакцией $!\boldsymbol{error}\notin C$ – аналогично тому, как это делается в пополнении состояний $\boldsymbol{Scomp}_e$. Эта реакция добавляется к алфавиту спецификации: $C`=C\cup\{!\boldsymbol{error}\}$, а алфавит реализации остаётся прежним.

Любая конформная реализация не проходит «лишнюю» δ-трассу, а неконформная реализация после прохождения «лишней» δ-трассы выдаёт реакцию из алфавита C или стационарность, что заведомо не совпадает с единственной разрешённой в спецификации после такой δ-трассы реакцией $!\boldsymbol{error}$.

Формально можно считать, что такая спецификация допускает конформные реализации и в расширенном алфавите $C`$, в частности, такой реализацией является сама спецификация. Но такие реализации должны после прохождения «лишней» трассы выдавать именно реакцию $!\boldsymbol{error}$.

Дадим формальное определение преобразования $\boldsymbol{Comp}_0:\boldsymbol{MODEL}_{\beta\delta}(C)\to\boldsymbol{LTS}_{\beta\delta}(C`)$. Для аргумента – βδ-модели $\boldsymbol{\Sigma}$ достижимыми состояниями результирующей LTS будут δ-трассы, a) в которых нет двух идущих подряд символов δ, и b) которые либо сами принадлежат βδ-модели $\boldsymbol{\Sigma}$, либо из них могут быть получены δ-трассы βδ-модели $\boldsymbol{\Sigma}$ удалением некоторых стационарностей. Начальное состояние – пустая трасса.

Определение 87: Для $C\subseteq Z$ определим преобразование $\boldsymbol{Comp}_0:\boldsymbol{MODEL}_{\beta\delta}(C)\to\boldsymbol{LTS}_{\beta\delta}(C`)$, где $C`=C\cup\{!\boldsymbol{error}\}$ и $!\boldsymbol{error}\notin C$. Для $\boldsymbol{\Sigma}\in\boldsymbol{MODEL}_{\beta\delta}(C)$ $\boldsymbol{Comp}_0(\boldsymbol{\Sigma})=\boldsymbol{S}=LTS(C_\delta^*,C`,E_S,\epsilon)$, где E_S – наименьшее множество, порождаемое следующими правилами вывода: $\forall\mu\in C_\delta^*$ $\forall ?x\in C$ $\forall !y\in C$

(1) $\exists\mu`\in\boldsymbol{D}(\mu)\cap\boldsymbol{\Sigma}$ $\quad\mu`\cdot\langle ?x\rangle\in\boldsymbol{\Sigma}\quad\vdash\ \mu\text{—}?x\to\mu\cdot\langle ?x\rangle$,

(2) $\forall\mu`\in\boldsymbol{D}(\mu)\cap\boldsymbol{\Sigma}$ $\quad\mu`\cdot\langle !y\rangle\in\boldsymbol{\Sigma}\quad\vdash\ \mu\text{—}!y\to\mu\cdot\langle !y\rangle$,

(3) $\forall\mu`\in\boldsymbol{D}(\mu)\cap\boldsymbol{\Sigma}\ \boldsymbol{RefT}(\mu)=\epsilon\ \&\ \mu`\cdot\langle\delta\rangle\in\boldsymbol{\Sigma}\quad\vdash\ \mu\text{—}\tau\to\mu\cdot\langle\delta\rangle$,

(4) $\forall z\in !C_\delta\ \exists\mu`\in\boldsymbol{D}(\mu)\cap\boldsymbol{\Sigma}$ $\quad\mu`\cdot\langle z\rangle\notin\boldsymbol{\Sigma}\quad\vdash\ \mu\text{—}!\boldsymbol{error}\to\mu\cdot\langle !\boldsymbol{error}\rangle$.

□

Удаление блокировок после базового пополнения.

Конформная реализация для оставшихся блокировок может иметь следующее поведение в зависимости от класса реализаций:

(1) реализации без блокировок и разрушения – принимать стимул с любыми продолжениями без блокировок и разрушения;

(2) реализации общего вида (блокировки и разрушение допускаются) – блокировать стимул или принимать стимул с любыми продолжениями (блокировки и разрушение допускаются в этих продолжениях).

Эти оставшиеся блокировки удаляются из LTS $\boldsymbol{Comp}_0(\boldsymbol{\Sigma})$ последующим пополнением состояний этой LTS:

(1) Пополнение $\boldsymbol{Scomp}_d$ выполняет демоническое пополнение: заменяет блокировку на приём стимула со всеми возможными продолжениями без блокировок и разрушения.

(2) Пополнение $\boldsymbol{Scomp_\gamma}$ выполняет гамма-пополнение: заменяет блокировку на приём стимула с единственным продолжением разрушением. Это запрещает при тестировании проверять такой стимул после «лишней» трассы, то есть разрешает конформной реализации как блокировать стимул, так и принимать его с любыми продолжениями со всеми возможными блокировками и разрушением.

Определение 88: $\boldsymbol{TLDemonic} =_{def} \boldsymbol{Scomp_d \circ Comp_0}$,
$\boldsymbol{TLGamma} =_{def} \boldsymbol{Scomp_\gamma \circ Comp_0}$

□

Утверждение 50: Преобразования ***TLDemonic*** и ***TLGamma*** являются трассовыми ***TL***-пополнениями, не оставляющими блокировок в спецификации и сохраняющими семантику отношения ***ioco***.

□405

Трассовые пополнения, сохраняющие *ioco*.

Имея трассовое ***TL***-пополнение, сохраняющее семантику ***ioco***, легко получить остальные трассовые пополнения с помощью преобразования LTS-модели в трассовую модель (Рис.78).

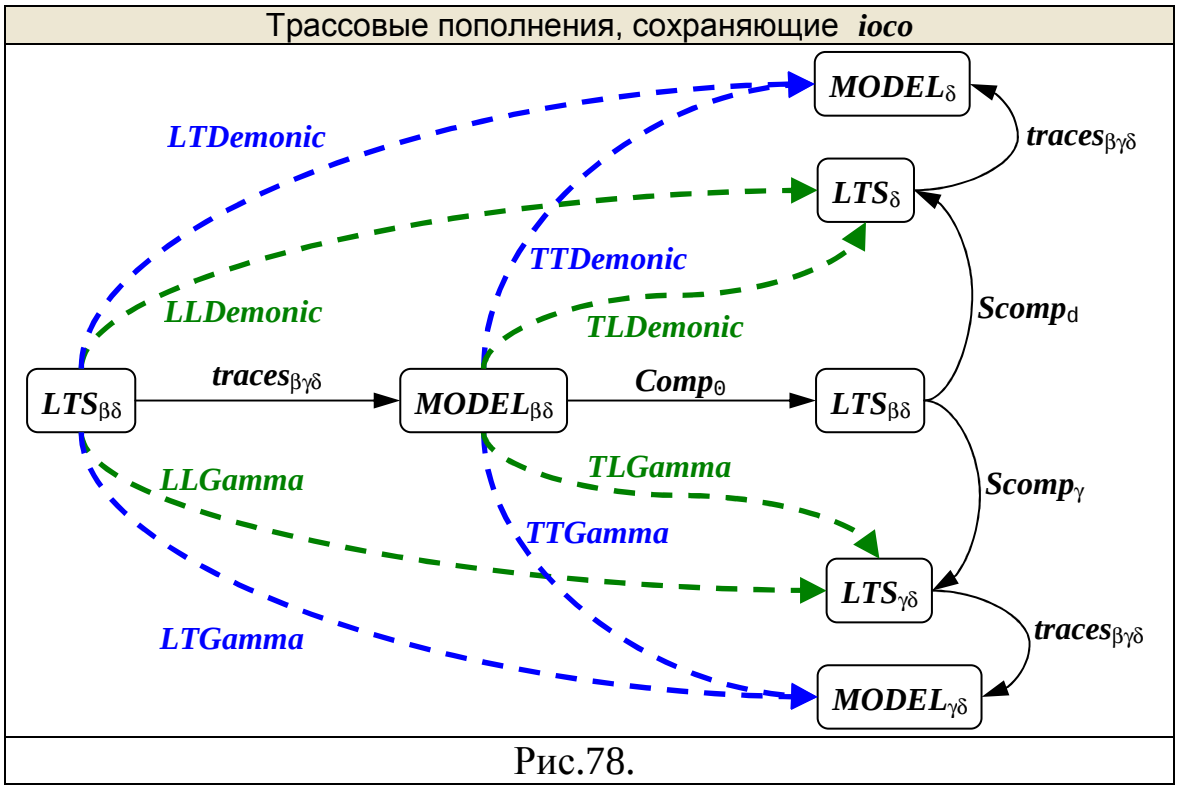


Рис.78.

Определение 89: Трассовые пополнения, сохраняющие ***ioco***:

$\boldsymbol{TTDemonic} =_{def} \boldsymbol{traces_{\beta\gamma\delta} \circ TLDemonic}$,
$\boldsymbol{LLDemonic} =_{def} \boldsymbol{TLDemonic \circ traces_{\beta\gamma\delta}}$,
$\boldsymbol{LTDemonic} =_{def} \boldsymbol{traces_{\beta\gamma\delta} \circ TLDemonic \circ traces_{\beta\gamma\delta}}$,

$$\textbf{\textit{TTGamma}} =_{def} \textbf{\textit{traces}}_{\beta\gamma\delta} \circ \textbf{\textit{TLGamma}},$$
$$\textbf{\textit{LLGamma}} =_{def} \textbf{\textit{TLGamma}} \circ \textbf{\textit{traces}}_{\beta\gamma\delta},$$
$$\textbf{\textit{LTGamma}} =_{def} \textbf{\textit{traces}}_{\beta\gamma\delta} \circ \textbf{\textit{TLGamma}} \circ \textbf{\textit{traces}}_{\beta\gamma\delta}.$$

□

Сравнение демонического и гамма-пополнений.

Проведём сравнение демонического и гамма-пополнений, для определённости имея в виду ***TT***-пополнения.

Трассовое демоническое пополнение ***Demonic*** и трассовое гамма-пополнение ***Gamma*** сохраняют классическую семантику ***ioco***. Различие между ними проявляется, прежде всего, для реализаций, в которых могут быть блокировки. При определении соответствия ***ioco*** предполагается, что в каждом стабильном состоянии реализации определён переход по каждому стимулу, то есть в реализации нет блокировок. Однако, если в спецификации **S** трасса μ не продолжается стимулом ?x, то на самом деле не проверяется, может ли реализация после трассы μ принимать или блокировать стимул ?x. Иными словами, классическая семантика ***ioco*** на самом деле требует только, чтобы в конформной реализации после общей трассы μ были определены переходы по всем тем стимулам, которыми трасса μ может продолжаться в спецификации.

Резюмируя, можно провести следующее сравнение отношений ***ioco*** и $\textbf{\textit{ioco}}_{\beta\gamma\delta}$ и трассовых демонического и гамма-пополнений:

1) Домен $\textbf{\textit{ioco}}_{\beta\gamma\delta}$ шире домена ***ioco***:
   ioco-спецификации = $\textbf{\textit{ioco}}_{\beta\gamma\delta}$-спецификации без разрушения,
   ioco-реализации = $\textbf{\textit{ioco}}_{\beta\gamma\delta}$-реализации без блокировок и разрушения.
2) На домене отношения ***ioco*** (спецификации без разрушения, реализации без блокировок и разрушения):
   $$\mathbf{I}\ \textbf{\textit{ioco}}\ \mathbf{\Sigma} \Leftrightarrow \mathbf{I}\ \textbf{\textit{ioco}}\ \textbf{\textit{Demonic}}(\mathbf{\Sigma}) \Leftrightarrow \mathbf{I}\ \textbf{\textit{ioco}}_{\beta\gamma\delta}\ \textbf{\textit{Demonic}}(\mathbf{\Sigma}) \Leftrightarrow \mathbf{I}\ \textbf{\textit{ioco}}_{\beta\gamma\delta}\ \textbf{\textit{Gamma}}(\mathbf{\Sigma}).$$
3) Для отношения ***ioco*** спецификация с блокировками не самоприменима, то есть не является конформной реализацией, поскольку конформные реализации не могут иметь блокировок. В то же время для $\textbf{\textit{ioco}}_{\beta\gamma\delta}$ любая спецификация конформна сама себе. Можно также отметить, что спецификация $\textbf{\textit{Demonic}}(\mathbf{\Sigma})$ самоприменима, поскольку в ней нет блокировок. Тем самым, решается первая из указанных целей: достигается рефлексивность отношения конформности.

Предлагаемое нами гамма-пополнение имеет ряд преимуществ по сравнению с демоническим пополнением.

1) Тестирование по гамма-пополненной спецификации не требует (запрещает) посылать в реализацию стимул ?x, неспецифицированный в исходной спецификации после трассы μ, и выполнять дальнейшие проверки, а тестирование по демонически пополненной спецификации вынуждает делать эти лишние действия. Причина этого в том, что при гамма-пополнении мы сохраняем информацию об отсутствующем после трассы стимуле в виде разрушения γ после этого стимула, а при демоническом пополнении эта информация теряется.

2) Гамма-пополнение расширяет классическую семантику ***ioco*** на не полностью определённые по стимулам реализации, тогда как демоническое пополнение усиливает соответствие для таких реализаций, поскольку запрещает блокировку стимула, которым до пополнения не продолжалась трасса.

3) Семантика разрушения шире семантики произвольного поведения, поскольку включает «нежелательное» поведение, которого следует избегать при тестировании. Мотивы «нежелательности» могут быть различными.

   Кроме угрозы реального разрушения реализации, могут учитываться соображения не полной реализации (при отладке), безопасности (конфиденциальности) и т.п. Для обнаружения отказов (стационарности для ***ioco***) при тестировании таким нежелательным поведением является дивергенция.

   Более того, при необходимости любое поведение можно объявить нежелательным в данном тестовом эксперименте. Для этого достаточно каждый переход в спецификации по стимулу или реакции s—z→s` заменить двумя переходами s—z→s``—*→s`. При генерации тестов для некоторого тестового эксперимента определяем *=τ, если переход желателен, и *=γ, если переход нежелателен в этом тестовом эксперименте.

Последние два пункта, фактически, указывают на то, что, если рассматривать реализации на классе всех LTS, разрешая в них блокировки и разрушение, то демоническое пополнение не сохраняет соответствие ***ioco***.

Расширения семантики *ioco*.

Если после базового пополнения спецификации блокировок не остаётся, для такой спецификации отношения ***ioco***$_{\beta\gamma\delta}$ и ***ioco*** совпадают: множества реализацией, конформных такой спецификации по обоим этим отношениям одинаковы. В этом случае множество трасс, которые используются при тестировании ***ioco***, является деревом δ-трасс и βγδ-моделью, причём все δ-трассы спецификации конвергентны: продолжаются каждым стимулом.

Такие спецификации после пополнения не имеют блокировок и разрушения, их трассы – это δ-трассы, то есть трассы в алфавите C_δ.

Однако в общем случае после базового пополнения могут остаться блокировки. В этом случае множество трасс, которые используются при тестировании ***ioco***, также является деревом δ-трасс, но уже не является βγδ-моделью, поскольку некоторые δ-трассы окажутся неконвергентными: не продолжаются некоторым стимулом и не продолжаются блокировкой этого стимула. При тестировании такие стимулы не посылаются в реализацию.

Можно предложить два расширения отношения ***ioco***: расширение домена реализаций и расширение домена спецификаций.

Расширение домена реализаций.

Если при тестировании некоторые стимулы не проверяются после некоторых δ-трасс, то, с нашей точки зрения, это эквивалентно продолжению таких δ-трасс стимулом и далее разрушением: такие разрушающие после трассы стимулы нельзя посылать в реализацию при безопасном тестировании. Тем самым, можно естественно расширить отношение ***ioco*** на реализации с разрушением и блокировками. Если стимул не проверяется, то в реализации он может блокироваться или приниматься. После приёма стимула возможно любое поведение, включая разрушение и блокировки любых стимулов.

Расширение домена спецификаций.

Итак, мы видим, что при гамма-пополнении спецификации в ней может появиться разрушение. Поэтому следующее естественное расширение отношения ***ioco*** – разрешить иметь в спецификации разрушение с самого начала. Такое разрушение можно разрешить не только после приёма стимула, но также после выдачи реакции или с самого начала (саморазрушающаяся спецификация, которой конформна любая реализация). При пополнении могут добавиться новые разрушения, а блокировки, по-прежнему, удаляются.

Такие расширения – это сужение отношения $\boldsymbol{ioco}_{\beta\gamma\delta}$ на поддомен спецификаций без блокировок. Если спецификация не полностью определена по стимулам, то она предварительно пополняется преобразованием ***Gamma***. Трассы таких пополненных спецификаций – это трассы в алфавите $\mathtt{C}_{\gamma\delta}$.

По Утверждение 24:, для $\boldsymbol{\Sigma}\in\boldsymbol{MODEL}_{\beta\gamma\delta}(\mathtt{C})$ (или $\boldsymbol{\Sigma}\in\boldsymbol{MODEL}_{\beta\delta}(\mathtt{C})$ при расширении только домена реализаций): $\boldsymbol{MODEL}_{\delta}(\mathtt{C})\subseteq\boldsymbol{safeIMPL}\circ\boldsymbol{Gamma}(\boldsymbol{\Sigma})$. Поэтому реализации, которые обычно рассматриваются для отношения ***ioco***, то есть полностью определённые по стимулам и не содержащие разрушения, *S*-тестируемы. Однако, для расширенного отношения ***ioco*** допускаются и другие реализации $\boldsymbol{safeIMPL}\circ\boldsymbol{Gamma}(\boldsymbol{\Sigma})\setminus\boldsymbol{MODEL}_{\delta}(\mathtt{C})$.

2.5.4. Алгоритмизация

Структура раздела:
- Пополнения состояний.
- Трассовые пополнения *Demonic* и *Gamma*, сохраняющие *ioco*.

Пополнения состояний.

Легко проверить, что все (за одним исключением) описанные выше (2.5.1) пополнения состояний $\boldsymbol{Scomp}_{\ldots}$ алгоритмизуемы. Для LTS, заданной первым способом **L1** (и, тем самым, для сводимых к ним LTS, заданным вторым способом **L2**), алгоритмы, задающие пополненную LTS, строятся из алгоритмов, задающих исходную LTS.

Исключением является комбинированный вариант 1, где используется предикат, проверяющий отсутствие в состоянии переходов по стимулам $\boldsymbol{init}(\mathtt{s})\cap?\mathtt{C}\neq\varnothing$. Нужен специальный оракул, реализующий этот предикат. Если число стимулов конечно, такой оракул можно построить по алгоритмам, задающим **L1**-спецификацию.

Трассовые пополнения *Demonic* и *Gamma*, сохраняющие *ioco*.

Поскольку демоническое пополнение состояний $\boldsymbol{Scomp}_{\mathtt{d}}$ и гамма-пополнение состояний $\boldsymbol{Scomp}_{\gamma}$ алгоритмизуемы, нам достаточно показать алгоритмизуемость преобразования $\boldsymbol{Comp}_{\theta}$.

βδ-модели, заданные вторым способом **T2**, и LTS-модели, заданные способом **L1** или **L2**, сводятся к βδ-модели, заданной первым способом

T1, а **L2**-модель сводится к моделям остальных типов. Поэтому нам достаточно иметь алгоритм преобразования $\boldsymbol{Comp_0}:\boldsymbol{MODEL}_{\beta\delta}(C)\to\boldsymbol{LTS}_{\beta\delta}(C`)$ типа **T1→L2**.

Утверждение 51: Существует алгоритм $\boldsymbol{AComp_0}$, определённый на классе **T1**-моделей без разрушения, который выполняет преобразование $\boldsymbol{Comp_0}$: для каждой LTS **S** из его домена возвращает **L2**-LTS $\boldsymbol{Comp_0}(\mathbf{S})$.

□407

Таким образом ***TL***-пополнения $\boldsymbol{TLDemonic}=\boldsymbol{Scomp_d}\circ\boldsymbol{Comp_0}$ и $\boldsymbol{TLGamma}=\boldsymbol{Scomp_\gamma}\circ\boldsymbol{Comp_0}$ алгоритмизуемы. Остальные трассовые пополнения на базе демонического и гамма-пополнений состояний также алгоритмизуемы, поскольку алгоритмизуемо преобразование LTS-модели в трассовую модель. Для моделей без разрушения первый и второй способы задания модели, фактически[57], совпадают: **T1=T2** и **L1=L2**, и мы будем обозначать их просто **T** и **L**. Соответственно преобразование $\boldsymbol{traces}_{\beta\gamma\delta}$ выполняется алгоритмом **L2T=L12T1=L22T2** (Рис.79).

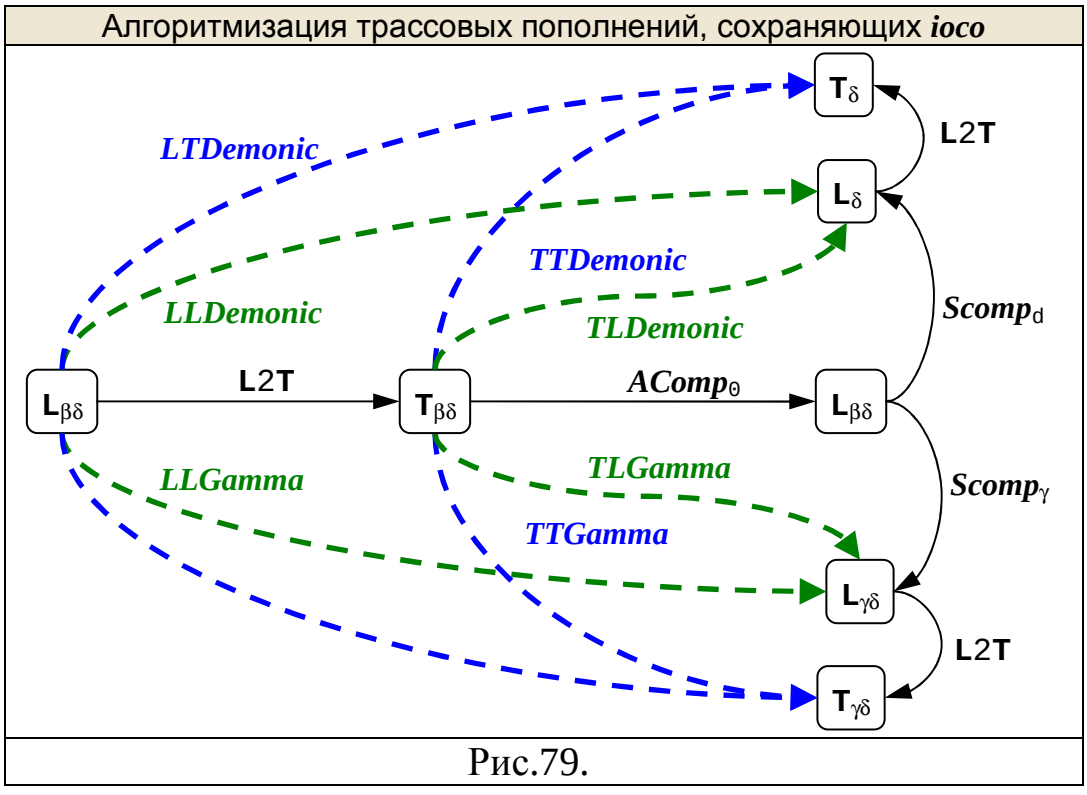
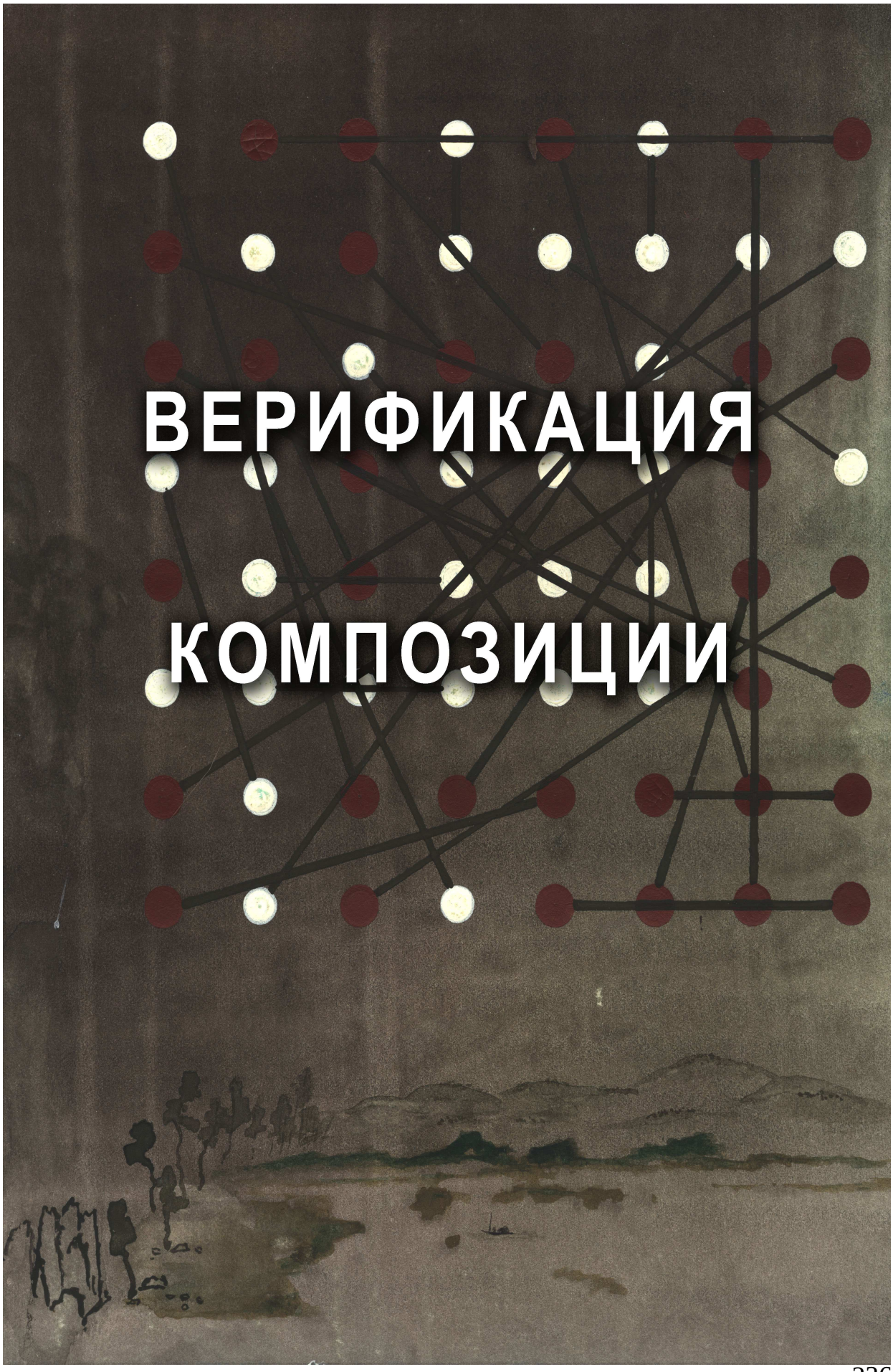


Рис.79.

[57] С точностью до константных оракулов: безопасности $\mathbf{Safe}_\Sigma\equiv\mathbf{Safe}_S\equiv\boldsymbol{true}$ и разрушения $\mathbf{Gamma1}_\Sigma\equiv\mathbf{Gamma1}_S\equiv\mathbf{Gamma2}_\Sigma\equiv\mathbf{Gamma2}_S\equiv\boldsymbol{false}$.


ВЕРИФИКАЦИЯ
КОМПОЗИЦИИ

Часть 3. Верификация композиции

Структура части:

1. Общая теория монотонности

   В этой главе излагается формальная теория монотонности в самых общих терминах модели и соответствия. Определяются понятия корректной спецификации композиции, косой композиции и монотонности. Формулируются восемь достаточных условий монотонности и излагается план доказательства этих условий.

2. Мажорирование βγδ-трасс и отношение $ioco_{\beta\gamma\delta}$

   Определяется отношение мажорирования βγδ-трасс и показывается эквивалентность мажорирования (для множеств βγδ-трасс LTS) отношению $\boldsymbol{ioco}_{\beta\gamma\delta}$.

3. ϕ-трассы

   Вводится новый вид трасс – ϕ-трассы, которые, вообще говоря, не являются трассами наблюдений. Теория ϕ-трасс – это вспомогательный инструмент для определения преобразования спецификаций и доказательства его монотонности. В отличие от βγδ-трасс аппарат ϕ-трасс позволяет выразить композицию LTS на трассовом уровне с точностью до конформности.

4. Общий случай: Мажорирование ϕ-трасс

   В общем случае произвольных LTS-спецификаций определяется отношение мажорирования ϕ-трасс и доказывается выполнение его свойств, требуемых достаточными условиями монотонности.

5. Общий случай: Преобразование $\mathcal{T}_{\beta\gamma\delta}$

   В общем случае произвольных *S*-ветвящихся LTS-спецификаций определяется преобразование спецификаций и доказывается его монотонность.

6. Общий случай: Алгоритмизация

   В этой главе рассматриваются проблемы алгоритмизации преобразования произвольных *S*-ветвящихся LTS-спецификаций и композиции преобразованных спецификаций (косой композиции).

Композиционная система – это составная система, собранная из компонентов с помощью применения определенных правил композиции. В этой работе компоненты моделируются LTS, а правила композиции – оператором $\between$ параллельной композиции LTS (Определение 62:). Мы допускаем θ-переходы только в тестере, но не в реализации. Поэтому компоненты системы моделируются LTS без θ-переходов, а их композиция – оператором $\between$ без правил вывода $(\between 4,5)$.

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

При тестировании соответствия правильность реализации отдельного компонента определяется как её соответствие спецификации компонента, а правильность системы – как её соответствие спецификации системы. Правильность реализации компонента проверяется синхронным тестированием компонента. Очевидно, одной из причин неправильной работы системы может оказаться неправильная работа её компонентов, которые при автономном тестировании не были полностью проверены. С учетом бесконечности полного набора тестов это вполне возможно и действительно часто встречается на практике.

Однако проблемы композиционных систем этим не исчерпываются. Реализации всех компонентов могут соответствовать своим спецификациям, а собранная из этих компонентов система не соответствует спецификации системы в целом. В чём причина и что делать в такой ситуации?

Проблема здесь в том, что само соотношение спецификаций компонентов и спецификации системы неправильно. Это уже ошибка декомпозиции спецификации системы на спецификации её компонентов. Такие ошибки значительно хуже ошибок в отдельных компонентах, поскольку их труднее обнаруживать, и они имеют более печальные последствия. Поэтому системное или комплексное тестирование не просто продолжает незаконченное тестирование компонентов, но предназначено также для обнаружения ошибок архитектурного уровня, которые не обнаруживаются автономными тестами. Такие ошибки могут привести к достаточно радикальным изменениям в спецификациях, что потребует модификации или даже повторной реализации всех или части компонентов.

Правильное соотношение спецификаций компонентов и спецификации системы должно удовлетворять *условию монотонности*: композиция конформных своим спецификациям компонентов конформна спецификации системы. Спецификация системы *корректна*, если она удовлетворяет условию монотонности при заданных спецификациях компонентов. Самая сильная (то есть, предъявляющая максимальные функциональные требования) корректная спецификация системы определяется самим условием монотонности. Однако это определение неявно, и не ясно, существует ли такая самая сильная корректная спецификация и можно ли её построить алгоритмически.

Примеры показывают, что самая сильная корректная спецификация системы, если она существует, оказывается слабее композиции спецификаций, проводимой по тем же правилам, что и композиция реализаций

компонентов. Если такую композицию называть *прямой*, то нужна другая – *косая* – композиция спецификаций, совпадающая с самой сильной корректной спецификацией системы, если она существует.

Это вызвано разными уровнями абстракции, используемыми в определениях соответствия и прямой композиции. Соответствия основаны на наблюдаемых действиях реализационной модели, то есть трассах, а композиция, кроме того, на ненаблюдаемых напрямую состояниях. Для отношения $\boldsymbol{ioco}_{\beta\gamma\delta}$ наблюдаемые действия — это прием или блокировка стимула, выдача реакции или отсутствие выдачи какой-либо реакции (стационарность). Композиция дополнительно учитывает состояния, ненаблюдаемые действия и соотношение состояний и действий.

Таким образом, необходимо определить косую композицию для выбранного отношения конформности.

Пополнение спецификаций – это наиболее простой подход к решению этой проблемы. Он основан на том, что блокировки в спецификации являются причиной несохранения соответствия для отношений, их не учитывающих, в частности для отношения $\boldsymbol{ioco}$. Поэтому можно избавиться от блокировок, пополняя спецификацию, но сохраняя композицию спецификаций по тем же правилам, что и композицию реализаций.

В данной части дано определение косой композиции для отношения $\boldsymbol{ioco}_{\beta\gamma\delta}$, когда блокировки допускаются. Будет показано, что при определённых условиях косая композиция совпадает с прямой композицией *преобразованных* спецификаций. Если такое преобразование существует, то соответствие будем называть монотонным относительно этого преобразования, а преобразование – монотонным для этого соответствия.

Одно такое преобразование спецификаций описывается в этой части. Следует отметить, что это преобразование не единственное, удовлетворяющее условию монотонности. В частности, мы рассмотрим оптимизацию этого преобразования, которая строит несколько другую LTS-модель.

В некоторых частных случаях такое преобразование может быть более простым. Например, для отношения $\boldsymbol{ioco}$ (без блокировок и разрушения), достаточно пополнения не полностью определенных спецификаций, удаляющего все блокировки. Этот и другие частные случаи рассматриваются в следующей части.

Асинхронное тестирование – это тестирование системы из двух компонентов, один из которых – реализация, а другой – фиксированная и известная среда передачи. Здесь монотонное преобразование нужно применять только к спецификации реализации, а среда остаётся неизменной. Мы будем говорить о левомонотонном преобразовании, имея в виду сохранение среды, задаваемой как правый операнд композиции. Название условно в связи с коммутативностью композиции.

В дальнейшем под спецификацией будем понимать спецификацию, для которой уже выполнено необходимое пополнение.

Глава 3.1. Общая теория монотонности

Структура главы:
1. Косая композиция и монотонное соответствие
2. Восемь достаточных условий монотонности соответствия
3. План доказательства достаточных условий

В этой главе излагается формальная теория монотонности в самых общих терминах модели и соответствия, являющегося отношением предпорядка на множестве моделей.

В разделе 3.1.1 формально определяются понятия корректной спецификации композиции, косой композиции и монотонности.

В разделе 3.1.2 формулируются восемь достаточных условий монотонности в предположении, что на моделях заданы βγδ- и ϕ-трассы и операции над ними. Под трассами здесь будут пониматься произвольные элементы, обладающие ровно теми свойствами, которые необходимы для теории монотонности. Нас не будет интересовать семантика этих трасс и операций над ними.

В разделе 3.1.3 излагается план доказательства этих условий при интерпретации соответствия как $\boldsymbol{ioco}_{\beta\gamma\delta}$, моделей как LTS и βγδ-трасс как βγδ-трасс LTS. В дальнейшем будут определены ϕ-трассы LTS.

3.1.1. Косая композиция и монотонное соответствие

Определим формально понятия корректной спецификации композиции, косой композиции и монотонного соответствия.

Определение 90: Пусть заданы:
- множество $\mathfrak{M}$ объектов, которые мы будем называть моделями, а также реализациями и спецификациями;
- коммутативная *прямая* композиция моделей $\mathbf{.}\!\uparrow\!\downarrow\!\mathbf{.} : \mathfrak{M}\times\mathfrak{M}\to\mathfrak{M}$;
- отношение предпорядка (рефлексивное и транзитивное бинарное отношение) $\boldsymbol{rel} \subseteq \mathfrak{M}\times\mathfrak{M}$; как правило, левый операнд будем называть реализацией, а правый – спецификацией.

□

Отношение предпорядка индуцирует *эквивалентность*. На фактор-множестве по этой эквивалентности предпорядок становится частичным порядком [Ско90]. Поэтому мы можем использовать понятия верхнего/нижнего конуса и точной верхней/нижней грани подмножества.

Определение 91: Пусть $\mathbf{S},\mathbf{S`}\in\mathfrak{M}$ и $\mathfrak{N}\subseteq\mathfrak{M}$. Обозначим:

- эквивалентность моделей:

  $\mathbf{S} \sim_{rel} \mathbf{S`} =_{def} \mathbf{S}\ rel\ \mathbf{S`}\ \&\ \mathbf{S`}\ rel\ \mathbf{S}$;

- верхний (нижний) конус подмножества – множество элементов больше или равных (меньше или равных) каждому элементу подмножества:

  $\mathfrak{N}^{\triangle} =_{def} \{\mathbf{S_1}\in\mathfrak{M} \mid \forall\mathbf{S}\in\mathfrak{N}\ \mathbf{S}\ rel\ \mathbf{S_1}\}$,

  $\mathfrak{N}^{\triangledown} =_{def} \{\mathbf{S_1}\in\mathfrak{M} \mid \forall\mathbf{S}\in\mathfrak{N}\ \mathbf{S_1}\ rel\ \mathbf{S}\}$;

- точная верхняя (нижняя) грань подмножества – множество наименьших (наибольших) элементов верхнего (нижнего) конуса подмножества:

  $sup(\mathfrak{N}) =_{def} \{\mathbf{S_2}\in\mathfrak{N}^{\triangle} \mid \forall\mathbf{S_1}\in\mathfrak{N}^{\triangle}\ \mathbf{S_2}\ rel\ \mathbf{S_1}\}$,

  $inf(\mathfrak{N}) =_{def} \{\mathbf{S_2}\in\mathfrak{N}^{\triangledown} \mid \forall\mathbf{S_1}\in\mathfrak{N}^{\triangledown}\ \mathbf{S_1}\ rel\ \mathbf{S_2}\}$.

□

Определение 92: **Корректная спецификация и косая композиция.**

- Обозначим множество всех реализаций, конформных (по отношению ***rel***) данной спецификации $\mathbf{S}\in\mathfrak{M}$:

  $\mathfrak{I}(\mathbf{S}) =_{def} \{\mathbf{S}\}^{\triangledown} = \{\mathbf{S_1}\in\mathfrak{M} \mid \mathbf{S_1}\ rel\ \mathbf{S}\}$.

- Для спецификации $\mathbf{S_1}\in\mathfrak{M}$ и реализации $\mathbf{I_2}\in\mathfrak{M}$ *левокорректной* спецификацией их композиции назовём спецификацию $\mathbf{S}$, которой конформна композиция любой реализации, конформной спецификации $\mathbf{S_1}$, с реализацией $\mathbf{I_2}$. Множество левокорректных спецификаций обозначим:

  $\mathfrak{S}(\mathbf{S_1})(\mathbf{I_2}) =_{def} (\mathfrak{I}(\mathbf{S_1})\wr\mathbf{I_2})^{\triangle}$

  $= \{\mathbf{S}\in\mathfrak{M} \mid \forall\mathbf{I_1}\in\mathfrak{I}(\mathbf{S_1})\ \mathbf{I_1}\wr\mathbf{I_2}\ rel\ \mathbf{S}\}$.

- Для двух спецификаций $\mathbf{S_1},\mathbf{S_2}\in\mathfrak{M}$ *корректной* спецификацией их композиции назовём спецификацию $\mathbf{S}$, которой конформна композиция любых реализацией, конформных спецификациям $\mathbf{S_1}$ и $\mathbf{S_2}$. Множество корректных спецификаций обозначим:

  $\mathfrak{S}(\mathbf{S_1},\mathbf{S_2}) =_{def} (\mathfrak{I}(\mathbf{S_1})\wr\mathfrak{I}(\mathbf{S_2}))^{\triangle}$

  $= \{\ \mathbf{S}\in\mathfrak{M} \mid \forall\mathbf{I_1}\in\mathfrak{I}(\mathbf{S_1})\ \forall\mathbf{I_2}\in\mathfrak{I}(\mathbf{S_2})\ \mathbf{I_1}\wr\mathbf{I_2}\ rel\ \mathbf{S}\ \}$.

- *Левокосой композицией* спецификации $\mathbf{S_1}\in\mathfrak{M}$ и реализации $\mathbf{I_2}\in\mathfrak{M}$ будем называть самую сильную корректную спецификацию их композиции. Множество левокосых композиций обозначим:

  $\mathbf{S_1}\not\wr\mathbf{I_2} =_{def} sup(\mathfrak{I}(\mathbf{S_1})\wr\mathbf{I_2})$

$$= \{\ S \in \mathfrak{S}(S_1)(I_2)\ \mid\ \forall S` \in \mathfrak{S}(S_1)(I_2)\ \ S\ \ rel\ \ S`\ \}.$$

- *Косой композицией* двух спецификаций компонентов $S_1, S_2 \in \mathfrak{M}$ будем называть самую сильную корректную спецификацию их композиции. Множество косых композиций обозначим:

$$S_1 \not/ S_2 =_{def} sup(\mathfrak{I}(S_1) \upharpoonright\!\downarrow \mathfrak{I}(S_2))$$

$$= \{S \in \mathfrak{S}(S_1, S_2)\ \mid\ \forall S` \in \mathfrak{S}(S_1, S_2)\ \ S\ \ rel\ \ S`\}.$$

□

Заметим, что мы могли бы аналогично определить правокорректную спецификацию и правокосую композицию. Однако, в силу коммутативности оператора прямой композиции $\upharpoonright\!\downarrow$, это излишне. То же самое относится к понятию монотонности, которое мы определяем ниже.

Определение 93: **Монотонность соответствия.** Пусть выделен класс спецификаций $\mathfrak{C} \subseteq \mathfrak{M}$ и задано преобразование $\mathcal{T}: \mathfrak{C} \to \mathfrak{C}$.

- Будем говорить, что преобразование $\mathcal{T}$ *инвариантно* (на классе $\mathfrak{C}$), если оно сохраняет эквивалентность $\sim_{rel}$: $\forall P \in \mathfrak{C}\ \ \mathcal{T}(P)\ \sim_{rel}\ P$.[58]
- Будем говорить, что отношение ***rel*** *$\mathcal{T}$-левомонотонно* (на классе $\mathfrak{C}$), если преобразование $\mathcal{T}$ инвариантно и прямая композиция преобразованной спецификации из $\mathfrak{C}$ с любой реализацией из $\mathfrak{C}$ является левокосой композицией: $\forall S_1, I_2 \in \mathfrak{C}\ \ \mathcal{T}(S_1) \upharpoonright\!\downarrow I_2 \in S_1 \not\backslash I_2$. Преобразование $\mathcal{T}$ будем называть ***rel***-левомонотонным преобразованием (на классе $\mathfrak{C}$), или просто левомонотонным преобразованием (на классе $\mathfrak{C}$), если отношение ***rel*** подразумевается.
- Будем говорить, что отношение ***rel*** *$\mathcal{T}$-монотонно* (на классе $\mathfrak{C}$), если преобразование $\mathcal{T}$ инвариантно и прямая композиция преобразованных спецификаций из класса $\mathfrak{C}$ является косой композицией: $\forall S_1, S_2 \in \mathfrak{C}\ \ \mathcal{T}(S_1) \upharpoonright\!\downarrow \mathcal{T}(S_2) \in S_1 \not/ S_2$. Преобразование $\mathcal{T}$ будем называть ***rel***-монотонным преобразованием (на классе $\mathfrak{C}$), или просто монотонным преобразованием (на классе $\mathfrak{C}$), если отношение ***rel*** подразумевается.

---

[58] Формально, сохранение эквивалентности означает: $a \sim_{rel} b \Rightarrow \mathcal{T}(a) \sim_{rel} b$. Это выполняется, поскольку, если $\mathcal{T}(a) \sim_{rel} a$, то, по транзитивности эквивалентности, $\mathcal{T}(a) \sim_{rel} a\ \&\ a \sim_{rel} b \Rightarrow \mathcal{T}(a) \sim_{rel} b$.

- Будем говорить, что отношение $\boldsymbol{rel}$ *левомонотонно* или *монотонно* (на классе $\mathfrak{C}$), если оно, соответственно, $\mathcal{T}$*-левомонотонно* или $\mathcal{T}$*-монотонно* (на классе $\mathfrak{C}$) для тождественного преобразования $\mathcal{T}$.

□

3.1.2. Восемь достаточных условий монотонности соответствия

Определение 94: Пусть заданы:

- класс спецификаций $\mathfrak{C}\subseteq\mathfrak{M}$;
- множество $\mathbf{G}$, элементы которого мы будем называть βγδ-трассами;
- множество $\mathbf{F}$, элементы которого мы будем называть ϕ-трассами, и которые предполагаются отличными от моделей $\mathfrak{M}\cap\mathbf{F}=\varnothing$;[59]
- отображение $g:\mathfrak{M}\to\wp(\mathbf{G})$, задающее множество βγδ-трасс модели;
- отображение $f:\mathfrak{M}\to\wp(\mathbf{F})$, задающее множество ϕ-трасс модели;
- оператор $\xi:\mathbf{F}\to\wp(\mathbf{G})$, строящий по ϕ-трассе множество *порождаемых* (*генерируемых*) ею βγδ-трасс;
- отображение $.\!\upharpoonright\!.:\mathbf{F}\times\mathbf{F}\to\wp(\mathbf{F})$, называемое композицией ϕ-трасс;
- отношение «один к многим» $\preccurlyeq\subseteq\mathbf{G}\times\wp(\mathbf{G})$, называемое мажорированием βγδ-трасс; если $a\preccurlyeq B$, то будем говорить, что B *мажорирует* a (a *мажорируется* B);
- отношение «один к многим» $\preccurlyeq\subseteq\mathbf{F}\times\wp(\mathbf{F})$, называемое мажорированием ϕ-трасс, которое совпадает с мажорированием βγδ-трасс на пересечении доменов $(\mathbf{F}\cap\mathbf{G})\times(\wp(\mathbf{F})\cap\wp(\mathbf{G}))$; если $a\preccurlyeq B$, то будем говорить, что B *мажорирует* a (a *мажорируется* B).

□

Мы будем считать также, что отображения и отношения естественным образом распространены на случай, когда операндом является не элемент, а множество элементов.

Определение 95: Будем говорить, что спецификация *мажорантна*, если множество её ϕ-трасс мажорирует множество ϕ-трасс каждой конформной реализации. Преобразование $\mathcal{T}:\mathfrak{C}\to\mathfrak{C}$ будем называть *мажорантным* (на

---

[59] Мы требуем различения ϕ-трасс и моделей по техническим причинам: мы перегружаем оператор параллельной композиции, используя одно и то же обозначение $\upharpoonright$ как для ϕ-трасс, так и для моделей.

классе $\mathfrak{C}$), если для каждой спецификации $\mathbf{S}\in\mathfrak{C}$ преобразованная спецификация $\mathcal{T}(\mathbf{S})$ мажорантна.

□

Определение 96: **Условия монотонности**.

Монотонность 1: Эквивалентность соответствия мажорированию βγδ-трасс:

$\forall \mathbf{I},\mathbf{S}\in\mathfrak{M}\ \ \mathbf{I}\ \ \boldsymbol{rel}\ \ \mathbf{S} \Leftrightarrow \boldsymbol{g}(\mathbf{I}) \preccurlyeq \boldsymbol{g}(\mathbf{S})$.

Монотонность 2: Генеративность ϕ-трасс:

$\forall \mathbf{S}\in\mathfrak{M}\ \cup_{\circ}\xi_{\circ}\boldsymbol{f}(\mathbf{S}) = \boldsymbol{g}(\mathbf{S})$.

Монотонность 3: Аддитивность ϕ-трасс относительно композиции:

$\forall \mathbf{S}_1,\mathbf{S}_2\in\mathfrak{M}\ \ \boldsymbol{f}(\mathbf{S}_1\between\mathbf{S}_2)=\cup(\boldsymbol{f}(\mathbf{S}_1)\between\boldsymbol{f}(\mathbf{S}_2))$.

Монотонность 4: Генеративность мажорирования ϕ-трасс:

$\forall \mathbf{I},\mathbf{S}\in\mathfrak{M}\ \ \boldsymbol{f}(\mathbf{I})\preccurlyeq\boldsymbol{f}(\mathbf{S}) \Rightarrow \cup_{\circ}\xi_{\circ}\boldsymbol{f}(\mathbf{I})\preccurlyeq\cup_{\circ}\xi_{\circ}\boldsymbol{f}(\mathbf{S})$.

Монотонность 5: Композиционность мажорирования ϕ-трасс на классе $\mathfrak{C}$:

$\forall \mathbf{I}_1,\mathbf{I}_2\in\mathfrak{M}\ \ \forall \mathbf{S}_1,\mathbf{S}_2\in\mathfrak{C}$

$\boldsymbol{f}(\mathbf{I}_1)\preccurlyeq\boldsymbol{f}(\mathbf{S}_1)\ \&\ \boldsymbol{f}(\mathbf{I}_2)\preccurlyeq\boldsymbol{f}(\mathbf{S}_2) \Rightarrow \cup(\boldsymbol{f}(\mathbf{I}_1)\between\boldsymbol{f}(\mathbf{I}_2))\preccurlyeq\cup(\boldsymbol{f}(\mathbf{S}_1)\between\boldsymbol{f}(\mathbf{S}_2))$.

Монотонность 6: Конформность преобразования $\mathcal{T}:\mathfrak{C}\to\mathfrak{C}$:

$\forall \mathbf{S}\in\mathfrak{C}\ \ \mathcal{T}(\mathbf{S})\ \ \boldsymbol{rel}\ \ \mathbf{S}$.

Монотонность 7: Мажорантность преобразования $\mathcal{T}:\mathfrak{C}\to\mathfrak{C}$:

$\forall \mathbf{S}\in\mathfrak{C}\ \cup_{\circ}\boldsymbol{f}_{\circ}\mathcal{I}(\mathbf{S}) \preccurlyeq \boldsymbol{f}_{\circ}\mathcal{T}(\mathbf{S})$.

Монотонность 8: Рефлексивность мажорирования ϕ-трасс на классе $\mathfrak{C}$:

$\forall \mathbf{S}\in\mathfrak{C}\ \ \boldsymbol{f}(\mathbf{S}) \preccurlyeq \boldsymbol{f}(\mathbf{S})$.

□

Утверждение 52: Для того, чтобы отношение ***rel*** было $\mathcal{T}$-монотонным на классе $\mathfrak{C}$ достаточно выполнения условий монотонности 1÷7. Для того, чтобы отношение ***rel*** было $\mathcal{T}$-левомонотонным на классе $\mathfrak{C}$ достаточно выполнения условий монотонности 1÷8.

□409

Замечание: Если класс $\mathfrak{C}$ замкнут по композиции моделей $\between$, то результат композиции может использоваться как операнд в дальнейших композициях с сохранением монотонности соответствия. Это важно, если композиционная система состоит из многих компонентов (больше двух). Тем не менее, мы не требуем замкнутости по композиции в общем случае, поскольку, например, для асинхронного тестирования используется ровно одна композиция двух моделей: реализации и среды передачи.

3.1.3. План доказательства достаточных условий

В дальнейшем мы под моделями будем понимать LTS-модели (Определение 42:), под βγδ-трассами – βγδ-трассы (Определение 5:) LTS-моделей (Определение 46:) и под соответствием – отношение $\boldsymbol{ioco}_{\beta\gamma\delta}$ (Определение 23: и Определение 58:): $\mathfrak{M}=\boldsymbol{LTS}_{\beta\gamma\delta}$, $\mathbf{G}=Z_{\beta\gamma\delta}^{*}$ и $\boldsymbol{g}=\boldsymbol{traces}_{\beta\gamma\delta}$, $\boldsymbol{rel}=\boldsymbol{ioco}_{\beta\gamma\delta}$. Композицию моделей $\between$ будем понимать как композицию LTS согласно Определение 62:.

В Глава 3.2 мы определим мажорирование $\preccurlyeq$ βγδ-трасс и покажем Эквивалентность соответствия мажорированию βγδ-трасс: (Монотонность 1:).

В Глава 3.3 определяются ϕ-трассы $\mathbf{F}=\mathbf{\Phi}^{\omega}$, их связь с LTS $\boldsymbol{f}=\boldsymbol{traces}_{\phi}^{\omega}$, ξ-оператор и композиция $\between$ ϕ-трасс. Доказываются Генеративность ϕ-трасс: (Монотонность 2:) и Аддитивность ϕ-трасс относительно композиции: (Монотонность 3:).

В качестве класса $\mathfrak{C}$ выбирается класс всех *S*-ветвящихся LTS-спецификаций.

В Глава 3.4 определяется мажорирование ϕ-трасс $\preccurlyeq$ и доказывается его генеративность (Монотонность 4:). Также доказываются рефлексивность (Монотонность 8:), и композиционность (Монотонность 5:) мажорирования ϕ-трасс, причём на классе всех LTS-спецификаций, а не только *S*-ветвящихся.

В Глава 3.5 определяется преобразование $\mathcal{T}=\mathcal{T}_{\beta\gamma\delta}$ и доказываются его конформность (Монотонность 6:), причём на классе всех LTS-спецификаций, а не только *S*-ветвящихся. Также доказывается мажорантность преобразования (Монотонность 7:) на классе *S*-ветвящихся LTS-спецификаций $\mathfrak{C}=\boldsymbol{SBLTS}_{\beta\gamma\delta}$.

Таким образом, из четырёх последних условий монотонности, в которых спецификация выбирается из класса $\mathfrak{C}$, только для мажорантности преобразования (Монотонность 7:) требуется рассматривать не класс всех LTS-спецификаций, а только его подкласс *S*-ветвящихся LTS-спецификаций.

Для частных случаев LTS без разрушения $\mathfrak{C}=SBLTS_{\beta\delta}$, LTS без блокировок $\mathfrak{C}=SBLTS_{\gamma\delta}$ и LTS без блокировок и разрушения $\mathfrak{C}=SBLTS_{\delta}$ мы определим своё мажорирование ϕ-трасс, сохраняя обозначение $\preccurlyeq$, и своё преобразование, которое будем обозначать, соответственно, $\mathcal{T}_{\beta\delta}$, $\mathcal{T}_{\gamma\delta}$, $\mathcal{T}_{\delta}$. Этим частным случаям посвящена следующая Часть 4.

Для удобства, перепишем условия монотонности в более конкретных обозначениях (оставляя обозначение класса $\mathfrak{C}$ и преобразования $\mathcal{T}$, поскольку они различны в различных случаях):

Монотонность 1: Эквивалентность соответствия мажорированию βγδ-трасс:

$\forall \mathbf{I},\mathbf{S}\in LTS_{\beta\gamma\delta}\ \mathbf{I}\ ioco_{\beta\gamma\delta}\ \mathbf{S} \Leftrightarrow traces_{\beta\gamma\delta}(\mathbf{I}) \preccurlyeq traces_{\beta\gamma\delta}(\mathbf{S})$.

Монотонность 2: Генеративность ϕ-трасс:

$\forall \mathbf{S}\in LTS_{\beta\gamma\delta}\ \cup\circ\xi\circ traces_{\phi}^{\omega}(\mathbf{S}) = traces_{\beta\gamma\delta}(\mathbf{S})$.

Монотонность 3: Аддитивность ϕ-трасс относительно композиции:

$\forall \mathbf{S_1},\mathbf{S_2}\in LTS_{\beta\gamma\delta}\ traces_{\phi}^{\omega}(\mathbf{S_1}\!\upharpoonright\!\downharpoonright\!\mathbf{S_2}) = \cup(traces_{\phi}^{\omega}(\mathbf{S_1})\!\upharpoonright\!\downharpoonright\! traces_{\phi}^{\omega}(\mathbf{S_2}))$.

Монотонность 4: Генеративность мажорирования ϕ-трасс:

$\forall \mathbf{I},\mathbf{S}\in LTS_{\beta\gamma\delta}$

$traces_{\phi}^{\omega}(\mathbf{I}) \preccurlyeq traces_{\phi}^{\omega}(\mathbf{S}) \Rightarrow \cup\circ\xi\circ traces_{\phi}^{\omega}(\mathbf{I}) \preccurlyeq \cup\circ\xi\circ traces_{\phi}^{\omega}(\mathbf{S})$.

Монотонность 5: Композиционность мажорирования ϕ-трасс на классе $\mathfrak{C}$:

$\forall \mathbf{I_1},\mathbf{I_2}\in LTS_{\beta\gamma\delta}\ \forall \mathbf{S_1},\mathbf{S_2}\in \mathfrak{C}$

$traces_{\phi}^{\omega}(\mathbf{I_1}) \preccurlyeq traces_{\phi}^{\omega}(\mathbf{S_1})\ \&\ traces_{\phi}^{\omega}(\mathbf{I_2}) \preccurlyeq traces_{\phi}^{\omega}(\mathbf{S_2})$

$\Rightarrow \cup(traces_{\phi}^{\omega}(\mathbf{I_1})\!\upharpoonright\!\downharpoonright\! traces_{\phi}^{\omega}(\mathbf{I_2})) \preccurlyeq \cup(traces_{\phi}^{\omega}(\mathbf{S_1})\!\upharpoonright\!\downharpoonright\! traces_{\phi}^{\omega}(\mathbf{S_2}))$.

Монотонность 6: Конформность преобразования $\mathcal{T}\colon \mathfrak{C}\to\mathfrak{C}$:

$\forall \mathbf{S}\in \mathfrak{C}\ \mathcal{T}(\mathbf{S})\ ioco_{\beta\gamma\delta}\ \mathbf{S}$.

Монотонность 7: Мажорантность преобразования $\mathcal{T}\colon \mathfrak{C}\to\mathfrak{C}$:

$\forall \mathbf{S}\in \mathfrak{C}\ \cup\circ traces_{\phi}^{\omega}\circ\mathfrak{I}(\mathbf{S}) \preccurlyeq traces_{\phi}^{\omega}\circ\mathcal{T}(\mathbf{S})$.

Монотонность 8: Рефлексивность мажорирования ϕ-трасс на классе $\mathfrak{C}$:

$\forall \mathbf{S}\in \mathfrak{C}\ traces_{\phi}^{\omega}(\mathbf{S}) \preccurlyeq traces_{\phi}^{\omega}(\mathbf{S})$.

Глава 3.2. Мажорирование βγδ-трасс и отношение $ioco_{βγδ}$

Структура главы:
1. Мажорирование βγδ-трасс.
2. Эквивалентность $ioco_{βγδ}$ мажорированию βγδ-трасс (Монотонность 1:)

$\mathfrak{M}=LTS_{βγδ}$	Под моделями общей теории монотонности мы будем понимать LTS-модели (Определение 42:).
$\between$ для моделей	Под прямой композицией моделей общей теории монотонности будем понимать композицию LTS $.\between.:LTS_{βγδ}\times LTS_{βγδ}\to LTS_{βγδ}$ (Определение 62:).
$G=Z_{βγδ}^*$	Под множеством G общей теории монотонности будем понимать множество всех βγδ-трасс $Z_{βγδ}^*$, где алфавит Z – это множество всех возможных стимулов и реакций (Определение 5:).
$g=traces_{βγδ}$	Под отображением g общей теории монотонности будем отображение $traces_{βγδ}$, которое даёт множество βγδ-трасс LTS-операнда (Определение 46:).
$rel=ioco_{βγδ}$	Под отношением rel общей теории монотонности будем понимать отношение $ioco_{βγδ}$ (Определение 23: и Определение 58:), которое, по Утверждение 20:, является предпорядком.
$\preccurlyeq$ для βγδ-трасс	Введём отношение мажорирования βγδ-трасс и докажем выполнение условия Монотонность 1:.

3.2.1. Мажорирование βγδ-трасс.

Для общей теории монотонности нам нужно мажорирование βγδ-трасс как отношение «один к многим». Мы определим мажорирование βγδ-трасс как отношение «один к одному» и распространим его на «один к многим» (и «многие к многим»), то есть на множества βγδ-трасс.

Напомним, что при фиксированном алфавите $C\subseteq Z$ мы обозначаем $δ=!C$ и для произвольного множества A:
$?A=A\cap(\{?\}\times W)$, $!A=A\cap(\{!\}\times W)$ и
$βδ(A)=\{\{?x\}\,|\,?x\in ?A\}\cup\{δ\,|\,!A=δ\}$.
Очевидно, $βδ(A)=\{βδ(z)\,|\,z\in ?A\cup !A\}$, где $βδ(z)$ означает βδ-отказ, которому принадлежит z.

Определение 97: Пусть заданы алфавит $C\subseteq Z$ и две βγδ-трассы $σ,μ\in C_{βγδ}^*$.

- Будем говорить, что σ *мажорирует* μ (μ *мажорируется* σ), и обозначать $\mu \preccurlyeq \sigma$, если A) $\beta\gamma\delta$-трассы σ и μ совпадают, или B) имеет место γ-мажорирование: $\beta\gamma\delta$-трасса σ заканчивается разрушением и либо B1) $\sigma=\langle\gamma\rangle$, либо B2) её префикс π до последнего стимула или реакции a не включительно совпадает с префиксом $\beta\gamma\delta$-трассы μ, продолжаемый в μ либо B21) $\beta\delta$-отказом $\beta\delta(a)$, либо B22) стимулом или реакцией $b \in \beta\delta(a)$:[60]

  $\mu \preccurlyeq \sigma \;=_{def}\; \mu=\sigma \vee \mu \preccurlyeq^{\gamma} \sigma,$

  $\mu \preccurlyeq^{\gamma} \sigma \;=_{def}\; \sigma=\langle\gamma\rangle \;\vee\; \exists \pi \;\exists a \in C$

  $$\sigma=\pi\cdot\langle a,\gamma\rangle \;\&\; (\pi\cdot\langle\beta\delta(a)\rangle \leq \mu \;\vee\; \exists b \in \beta\delta(a)\;\; \pi\cdot\langle b\rangle \leq \mu).$$

- Распространим мажорирование $\beta\gamma\delta$-трасс на множества $\beta\gamma\delta$-трасс:

  $\forall a \in C_{\beta\gamma\delta}^{*} \;\forall B \subseteq C_{\beta\gamma\delta}^{*} \;\; a \preccurlyeq B \;=_{def}\; \exists b \in B \;\; a \preccurlyeq b.$

  $\forall A, B \subseteq C_{\beta\gamma\delta}^{*} \;\; A \preccurlyeq B \;=_{def}\; \forall a \in A \;\exists b \in B \;\; a \preccurlyeq b.$

- Две LTS-модели **A** и **B** будем называть эквивалентными по мажорированию $\beta\gamma\delta$-трасс, если множества $\beta\gamma\delta$-трасс этих LTS мажорируют друг друга:

  $\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{A}) \preccurlyeq \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{B}) \;\&\; \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{B}) \preccurlyeq \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{A}).$

□

Смысл мажорирования $\beta\gamma\delta$-трасс проясняется следующим утверждением.

Утверждение 53: Безопасная $\beta\gamma\delta$-трасса мажорируется только сама собой, а опасная $\beta\gamma\delta$-трасса имеет γ-мажоранту: $\forall C \subseteq Z \;\; \forall \mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$

1. $\forall \mu \in \boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S}) \;\; \forall \sigma \in \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S}) \;\; \mu \preccurlyeq \sigma \;\Rightarrow\; \mu=\sigma.$
2. $\forall \mu \in \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S}) \setminus \boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S}) \;\; \exists \sigma \in \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S}) \;\; \mu \preccurlyeq^{\gamma} \sigma.$

□412

3.2.2. Эквивалентность $ioco_{\beta\gamma\delta}$ мажорированию $\beta\gamma\delta$-трасс (Монотонность 1:)

Утверждение 54: **Условие Монотонность 1: выполнено:** реализация конформна спецификации тогда и только тогда, когда каждая $\beta\gamma\delta$-трасса реализации мажорируется некоторой $\beta\gamma\delta$-трассой спецификации:

---

[60] Если a стимул, то $\beta\delta(a)=\{a\}$, и в случае B22) $b=a$. Если a реакция, то $\beta\delta(a)=\delta$, и в случае B22) b любая реакция, не необязательно a.

$\forall \mathbf{I,S} \in \mathit{LTS}_{\beta\gamma\delta}\ \mathbf{I}\ \mathit{ioco}_{\beta\gamma\delta}\ \mathbf{S} \Leftrightarrow \mathit{traces}_{\beta\gamma\delta}(\mathbf{I}) \preccurlyeq \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$.

□412

Глава 3.3. ϕ-трассы

Структура главы:
1. ϕ-символы, ϕ-трассы и ϕ-маршруты LTS
2. ξ-оператор. Генеративность ϕ-трасс (Монотонность 2:)
3. Композиция ϕ-трасс. Аддитивность (Монотонность 3:)

Как мы уже отмечали выше (2.4.1), преимуществом LTS-модели по сравнению с моделями трасс наблюдений является возможность определения композиции моделей. Однако для этого недостаточно трасс наблюдений. Причина этого в том, что для композиции нам недостаточно тех знаний, которые мы можем вывести из наблюдения за поведением системы. Иными словами, одной модели трасс наблюдений соответствует множество различных LTS, в том числе LTS, ведущих себя по разному при композиции.

Тем не менее, в трассовой теории также можно определить композицию моделей, имеющую тот же смысл, если использовать другие трассы. Здесь мы вводим трассы, которые будем называть ϕ-трассами, и определяем композицию ϕ-трасс таким образом, что ϕ-трассы композиции LTS-моделей совпадают с композицией ϕ-трасс этих моделей. Такие трассы не являются, вообще говоря, трассами наблюдений, по крайней мере, для βγδ-машины тестирования. Вместе с тем, по ϕ-трассам LTS-модели можно получить все её βγδ-трассы, однако обратное неверно.

ϕ-трассы являются разновидностью трасс с отказами (failure traces). В отличие от общего вида таких трасс в каждом стабильном состоянии s в трассу помещается максимальный отказ (refusal set) - множество всех стимулов и реакций, по которым нет перехода из s: $C \backslash \mathit{init}(s)$. Кроме того, мы добавляем к этому отказу множество стимулов и реакций, по которым из s есть переходы в постсостояния с γ-переходами: $s \text{—} z \rightarrow s` \text{—} \gamma \rightarrow$.

3.3.1. ϕ-символы, ϕ-трассы и ϕ-маршруты LTS

Структура раздела:
- Определение ϕ-символов и ϕ-трасс.
- Преобразование LTS во множество ϕ-трасс.
- Распространение LTS-обозначений на ϕ-трассы.
- ϕ-трассы маршрутов.
- Замкнутость множества ϕ-трасс LTS по операциям над ϕ-трассами.

$\mathbf{F}=\mathbf{\Phi}^{\omega}$	Под множеством **F** общей теории монотонности будем понимать множество всех ϕ-трасс $\mathbf{\Phi}^{\omega}$, определяемое ниже.
$f=traces_{\phi}^{\omega}$	Под отображением ***f*** общей теории монотонности будем отображение $traces_{\phi}^{\omega}$, определяемое ниже.

Определение ϕ-символов и ϕ-трасс.

Определение 98: Пусть задан базовый алфавит символов $C \subseteq Z$.

- ϕ-*символом* будем называть пару из двух непересекающихся подмножеств базового алфавита, которые будем называть ***ref****-множеством* и ***gamma****-множеством*. Для ϕ-символа $o=(a,b)$ будем обозначать его ***ref***-множество и ***gamma***-множество через $o_r=a$ и $o_g=b$.
- ϕ-*алфавитом* будем называть множество всех ϕ-символов, и обозначать для данного базового алфавита
  $\Phi(C) =_{def} \{(a,b)|a\subseteq C \ \& \ b\subseteq C \ \& \ a\cap b=\varnothing\}$.
  Множество всех ϕ-символов обозначим $\Phi=\cup(\{\Phi(C)|C\subseteq Z\})$.
- Объединение алфавита и множества всех ϕ-символов обозначим
  $C_{\phi} =_{def} C\cup\Phi(C)$.
- *ϕ-трассой* в алфавите C будем называть (конечную или бесконечную) последовательность μ в алфавите базовых и ϕ-символов $C_{\phi\gamma}$, которая удовлетворяет следующим требованиям:
  - не продолжается после разрушения:
    $\forall i\in[1..|\mu|-1] \ \mu(i)\neq\gamma$;
  - за ϕ-символом следует либо тот же самый ϕ-символ, либо стимул или реакция, не принадлежащие ***ref***-множеству этого ϕ-символа:
    $\forall i\in[2..|\mu|]$
    $\mu(i-1)\in\Phi(C) \Rightarrow \mu(i)=\mu(i-1) \vee \mu(i)\in C \ \& \ \mu(i)\notin\mu(i-1)_r$.
- Для данного базового алфавита C множество конечных, бесконечных и всех ϕ-трасс в этом алфавите обозначим $\mathbf{\Phi}(C)$, $\mathbf{\Phi}^{\infty}(C)$, $\mathbf{\Phi}^{\omega}(C)$, соответственно.
- Объединение множеств конечных, бесконечных и всех ϕ-трасс во всех алфавитах обозначим $\mathbf{\Phi}^{\infty}=\cup(\{\mathbf{\Phi}^{\infty}(C)|C\subseteq Z\})$, $\mathbf{\Phi}=\cup(\{\mathbf{\Phi}(C)|C\subseteq Z\})$, $\mathbf{\Phi}^{\omega}=\cup(\{\mathbf{\Phi}^{\omega}(C)|C\subseteq Z\})$.

□

Мы будем обозначать ϕ-трассы строчными, а множества ϕ-трасс прописными жирными греческими буквами.

Вообще говоря, ϕ-трассы считаются разными, если они определены в разных алфавитах. Допуская вольность речи, мы будем под ϕ-трассой понимать последовательность в подразумеваемом алфавите.

Преобразование LTS во множество ϕ-трасс.

ϕ-символ состояния s – это пара множеств: ***ref***-множество = множество стимулов и реакций, по которым нет переходов из состояния s, то есть $C\backslash \mathit{init}(s)$, и ***gamma***-множество = множество стимулов и реакций, по которым из s есть переходы в постсостояния с γ-переходами: $s\text{—}z\rightarrow s`\text{—}\gamma\rightarrow$. Такие стимулы и реакции можно назвать *непосредственно разрушающими* в состоянии.

Определение 99: **ϕ-символ стабильного состояния.** Пусть $C\subseteq Z$, $s\in \mathit{LTS}_{\beta\gamma\delta}(C)$. Для стабильного состояния $s\in V_s$ определим:

- ***ref***-множество как множество отвергаемых стимулов и реакций $\phi(s)_r =_{def} C\backslash \mathit{init}(s)$,
- ***gamma***-множество как множество непосредственно разрушающих стимулов и реакций $\phi(s)_g =_{def} \{z\in C \mid \exists s` \; s\text{—}z\rightarrow s`\text{—}\gamma\rightarrow\}$.
- ϕ-символ состояния как пару $\phi(s) =_{def} (\phi(s)_r, \phi(s)_g)$.

Будем считать, что нестабильное состояние не имеет ***ref***-множества, ***gamma***-множества и ϕ-символа.

□

В целом мы различаем три вида разрушающих стимулов и реакций:

1) Символ z *непосредственно* разрушающий *в состоянии* s, если есть переход в состояние с γ-переходом: $s\text{—}z\rightarrow s`\text{—}\gamma\rightarrow$.
2) Символ z разрушающий *в состоянии* s, если есть переход в состояние с γ-трассой: $s\text{—}z\rightarrow s`=\gamma\Rightarrow$. В отличие от 1-го вида здесь γ-переход может начинаться не в самом постсостоянии $s`$, а в состоянии, достижимом из $s`$ по непустой цепочке τ-переходов. Возможно также, что γ-перехода нет, но постсостояние $s`$ дивергентно. На основе таких разрушающих в состоянии стимулов и реакций мы определяли множество символов, безопасных в состоянии $\mathit{safesymbols}_{\beta\gamma\delta}(s)$ (Определение 52:).
3) Символ z разрушающий *после βγδ-трассы* σ, если есть βγδ-трасса $\sigma\cdot\langle z,\gamma\rangle$. В отличие от 2-го вида здесь z-переход может начинаться не в самом состоянии s, а в состоянии, достижимом из s по непустой цепочке τ-переходов. На основе таких разрушающих после βγδ-трассы стимулов и реакций мы определяли множество символов, безопасных после βγδ-трассы $\mathit{safesymbols}(\mathit{traces}_{\beta\gamma\delta}(s)\ \mathit{after}\ \sigma)$ (Определение 11: и Определение 46:).

Очевидно, 1⇒2⇒3, но обратные импликации, вообще говоря, не верны.

Для ϕ-символа состояния мы выбираем трактовку 1) – непосредственно разрушающие символы, поскольку в этом случае ***gamma***-множество композиционного состояния ab однозначно определяется ***gamma***-множествами состояний-компонентов a и b. При других трактовках это не так. Трактовка 2) требует учёта τ-маршрутов, возникающих в результате синхронных переходов при композиции, которые бесконечны или ведут в состояния с γ-переходами. Трактовка 3) требует перечисления всех безопасных βγδ-трасс, заканчивающихся в состоянии, что, вообще говоря, невозможно сделать за конечное время, поскольку число таких βγδ-трасс может быть бесконечным (ср. Замечание о неалгоритмизуемости построения F-канонической модели в 2.3.7).

Выше мы вводили множество βγδ-трасс LTS через преобразование $\boldsymbol{LTS}_{\beta\gamma\delta}(C)\rightarrow\boldsymbol{LTS}(C_{\beta\gamma\delta})$ и взятие множества трасс полученной LTS. Аналогично множество ϕ-трасс LTS определим через преобразование $\boldsymbol{L_\gamma 2L_{\phi\gamma}}:\boldsymbol{LTS}_{\beta\gamma\delta}(C)\rightarrow\boldsymbol{LTS}(C_{\phi\gamma})$ и взятие множества трасс полученной LTS (Рис.47). Также, как βδ-отказ преобразуется в переход-петлю, в каждом стабильном состоянии s добавляется переход-петля по символу ϕ(s). Также, как в преобразовании $\boldsymbol{L_\gamma 2L_{\beta\delta}}$, для моделирования дивергенции разрушением в каждом дивергентном состоянии определяется γ-переход в специальное терминальное гамма-состояние; это же состояние становится постсостоянием для всех γ-переходов, уже имевшихся в исходной LTS[61]. Заметим, что в трассе преобразованной LTS разрушение может быть только конечным символом.

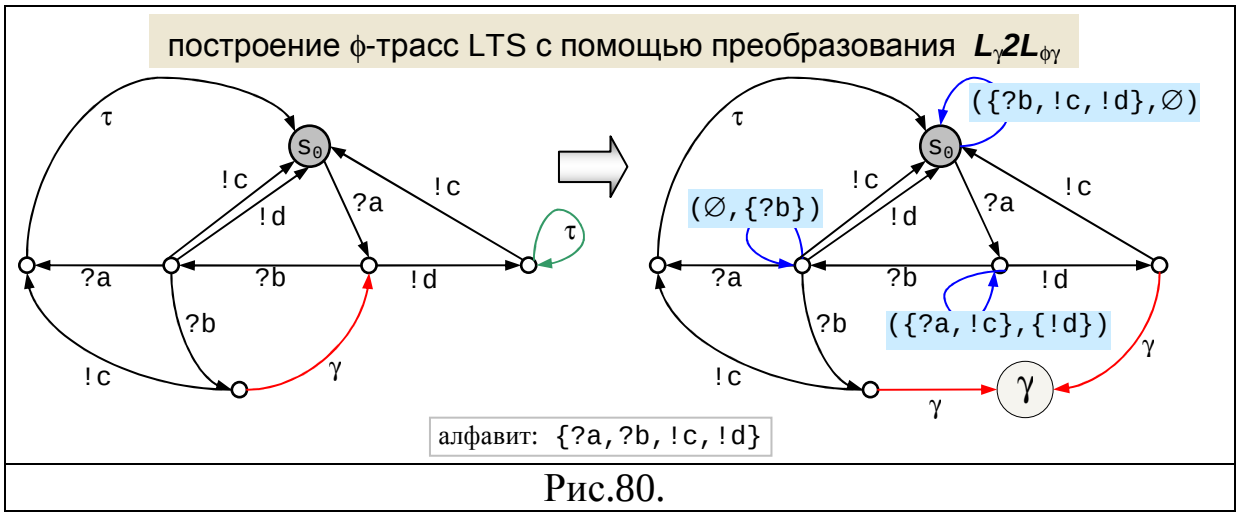


Рис.80.

[61] Если LTS уже содержит состояние, которое называется γ, то перед преобразованием переименуем его.

Определение 100: Для $C \subseteq Z$ определим преобразование $L_\gamma 2L_{\phi\gamma} : LTS_{\beta\gamma\delta}(C) \to LTS(C_{\phi\gamma})$. Для $S = LTS(V_S, C_\gamma, E_S, s_0)$ $L_\gamma 2L_{\phi\gamma}(S) = M = LTS(V_S \cup \{\gamma\}, C_{\phi\gamma}, E_M, s_0)$, где $\gamma \notin V_S$, а E_M – наименьшее множество, порождаемое следующими правилами вывода:

$\forall s, s' \in V_S \quad \forall z \in C_\tau$

$s \text{—} z \to s' \quad \vdash \quad s \text{—} z \to s';$

$s \text{—} \tau\gamma \nrightarrow \quad \vdash \quad s \text{—} \phi(s) \to s;$

$s \text{—} \gamma \to \ \vee \ s\uparrow \quad \vdash \quad s \text{—} \gamma \to \gamma.$

□

Трассы LTS $L_\gamma 2L_{\phi\gamma}(S)$ – это и есть ϕ-трассы LTS S. Очевидно, что они удовлетворяют всем ограничениям Определение 98:. В отличие от βγδ-трасс мы будем рассматривать не только конечные, но и бесконечные ϕ-трассы.

Определение 101: Пусть $C \subseteq Z$ и $S \in LTS_{\beta\gamma\delta}(C)$.

- ϕ-маршрутом в LTS S будем называть маршрут в LTS $L_\gamma 2L_{\phi\gamma}(S)$:
  для состояния $s \in V_S$:
  - $runs_\phi(s) =_{def} runs(s)$ в $L_\gamma 2L_\phi(S)$
    – конечные ϕ-маршруты с началом в состоянии s,
  - $runs_\phi^\infty(s) =_{def} runs^\infty(s)$ в $L_\gamma 2L_\phi(S)$
    – бесконечные ϕ-маршруты с началом в состоянии s,
  - $runs_\phi^\omega(s) =_{def} runs^\omega(s)$ в $L_\gamma 2L_\phi(S)$
    – все ϕ-маршруты с началом в состоянии s,
  - $runs_\phi(S) =_{def} runs_\phi(s_0)$ – конечные ϕ-маршруты S,
  - $runs_\phi^\infty(S) =_{def} runs_\phi^\infty(s_0)$ – бесконечные ϕ-маршруты S,
  - $runs_\phi^\omega(S) =_{def} runs_\phi^\omega(s_0)$ – все ϕ-маршруты S.
- ϕ-трассой LTS S будем называть трассу LTS $L_\gamma 2L_{\phi\gamma}(S)$:
  для состояния $s \in V_S$:
  - $traces_\phi(s) =_{def} traces(s)$ в $L_\gamma 2L_\phi$
    – конечные ϕ-трассы с началом в состоянии s,
  - $traces_\phi^\infty(s) =_{def} traces^\infty(s)$ в $L_\gamma 2L_\phi$
    – бесконечные ϕ-трассы с началом в s,
  - $traces_\phi^\omega(s) =_{def} traces^\omega(s)$ в $L_\gamma 2L_\phi$
    – все ϕ-трассы с началом в состоянии s,
  - $traces_\phi(S) =_{def} traces_\phi(s_0)$ – конечные ϕ-трассы S,
  - $traces_\phi^\infty(S) =_{def} traces_\phi^\infty(s_0)$ – бесконечные ϕ-трассы S,
  - $traces_\phi^\omega(S) =_{def} traces_\phi^\omega(s_0)$ – все ϕ-трассы S.

· ϕ-трасса $\sigma \in traces_\phi^\omega(s)$ *разрушающая*, если некоторый её префикс продолжается разрушением: $\exists \mu \le \sigma \;\; \mu \cdot \langle\gamma\rangle \in traces_\phi^\omega(s)$.

□

Распространение LTS-обозначений на ϕ-трассы.

Выше мы распространили LTS-обозначения (стрелки и оператор ***after***) на βγδ-последовательности и βγδ-трассы. Аналогично распространим эти LTS-обозначения на ϕ-последовательности $\sigma \in C_{\phi\tau}^\omega$ и ϕ-трассы $\sigma \in C_\phi^\omega$.

Стрелки имеют один и тот же смысл для LTS S и LTS $L_\gamma 2L_\phi(S)$, если последовательность σ (трасса для двойной стрелки) не содержит ϕ-символов и не заканчивается символом разрушения γ. Если σ содержит ϕ-символы, стрелки, очевидно, применяются к LTS $L_\gamma 2L_\phi(S)$, где есть переходы-петли по ϕ-символам. Двусмысленность возможна лишь в том случае, когда σ не содержит ϕ-символов и заканчивается символом γ, что происходит из-за того, что дивергенцию мы моделируем разрушением: в LTS $L_\gamma 2L_\phi(S)$ есть дополнительные γ-переходы. По умолчанию мы будем считать, что для такой стрелки подразумевается LTS S, а если имеется в виду LTS $L_\gamma 2L_\phi(S)$, то это будем указывать явно.

Тем самым, вообще говоря, $\sigma \in traces_\phi(s) \neq s{=}\sigma\Rightarrow$,

хотя $\sigma \in traces_\phi(s) = s{=}\sigma\Rightarrow$ в $L_\gamma 2L_\phi(S)$.

Определение 102: Пусть $C \subseteq Z$ и $S \in LTS_{\beta\gamma\delta}(C)$. LTS-операторы-стрелки $—\sigma\rightarrow$, $—\sigma\not\rightarrow$, $=\sigma\Rightarrow$ и $=\sigma\not\Rightarrow$ распространим на ϕ-символы, ϕτ-последовательности и ϕ-трассы: для $t \in C_{\phi\tau}$, $z \in C_\phi$, $\mu \in C_{\phi\tau}^\omega$, $\sigma \in C_\phi^\omega$ таких, что $\mu \in C_{\gamma\tau}^\omega \Rightarrow \gamma \notin Im(\mu)$ и $\sigma \in C_\gamma^\omega \Rightarrow \gamma \notin Im(\sigma)$:

$\ldots—\mu\ldots =_{def} \ldots—\mu\ldots$ в $L_\gamma 2L_\phi(S)$;

$\ldots—t\ldots =_{def} \ldots—t\ldots$ в $L_\gamma 2L_\phi(S) = \ldots—\langle t\rangle\ldots$;

$\ldots{=}\sigma\ldots =_{def} \ldots{=}\sigma\ldots$ в $L_\gamma 2L_\phi(S)$;

$\ldots{=}z\ldots =_{def} \ldots{=}z\ldots$ в $L_\gamma 2L_\phi(S) = \ldots{=}\langle z\rangle\ldots$.

□

Оператор ***after*** для ϕ-трассы, содержащей ϕ-символы, очевидно, относится к преобразованной LTS $L_\gamma 2L_\phi(S)$. Если ϕ-трасса σ не содержит ϕ-символы (является трассой), то s ***after*** σ в $S = s$ ***after*** σ в $L_\gamma 2L_\phi(S)$ только в том случае, когда трасса σ не заканчивается разрушением. Если ϕ-трасса заканчивается разрушением, то в LTS $L_\gamma 2L_\phi(S)$ она заканчивается в

единственном состоянии γ. Тем самым, мы можем корректно распространить оператор ***after*** в LTS **S** на ϕ-трассы, не заканчивающиеся разрушением.

Определение 103: Пусть $C \subseteq Z$ и $\mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$. LTS-оператор ***after*** распространим на конечные ϕ-трассы, не заканчивающиеся разрушением: для $s \in V_{\mathbf{S}}$, $\boldsymbol{\sigma} \in \boldsymbol{traces}_{\phi}(\mathbf{S})$ такой, что $\gamma \notin \boldsymbol{Im}(\boldsymbol{\sigma})$:

s ***after*** $\boldsymbol{\sigma}$ $=_{def}$ s ***after*** $\boldsymbol{\sigma}$ в $\boldsymbol{L_{\gamma}2L_{\phi}}(\mathbf{S})$;

S ***after*** $\boldsymbol{\sigma}$ $=_{def}$ s_0 ***after*** $\boldsymbol{\sigma}$ в $\boldsymbol{L_{\gamma}2L_{\phi}}(\mathbf{S})$ = $\boldsymbol{L_{\gamma}2L_{\phi}}(\mathbf{S})$ ***after*** $\boldsymbol{\sigma}$.

□

ϕ-трассы маршрутов.

Каждому маршруту S LTS **S** поставим в соответствие маршруты LTS $\boldsymbol{L_{\gamma}2L_{\phi}}(\mathbf{S})$, то есть ϕ-маршруты LTS **S**, которые могут отличаться от S:

a) *возможным* наличием петель ϕ-символов

плюс b) *обязательной* заменой первого γ-перехода на γ-переход в γ-состояние и удалением переходов после γ-перехода, если этот маршрут содержит γ-переход,

или c) *возможной* заменой бесконечного постфикса τ-переходов на моделирующий его γ-переход в γ-состояние, если такой постфикс есть в S и S не содержит γ-переходов.

Это сопоставление отличается от аналогичной процедуры для βγδ-маршрутов способом моделирования дивергенции γ-переходом, что, в свою очередь, объясняется введением в рассмотрение бесконечных маршрутов и ϕ-маршрутов. Моделирующий дивергенцию γ-переход βγδ-маршрута соответствует дивергентному конечному состоянию *конечного* маршрута S. В соответствующем ϕ-маршруте моделирующий дивергенцию γ-переход соответствует бесконечному постфиксу τ-переходов *бесконечного* маршрута S.

Поясним случай b). Продолжения маршрутов после первых γ-переходов нас не интересуют, поскольку они не влияют на ϕ-трассы LTS (аналогично тому, как они не влияют на βγδ-трассы LTS).

Поясним случай c). Мы сопоставляем с маршрутом все ϕ-маршруты, которые заканчиваются моделирующим дивергенцию γ-переходом. Этот γ-переход может идти из любого состояния в бесконечном постфиксе τ-переходов маршрута S. В дальнейшем нас будет интересовать лишь множество трасс этих ϕ-маршрутов. С этой точки зрения множество ϕ-маршрутов, соответствующих маршруту S, определено явно избыточно.

Достаточно было бы взять только те из них, в которых γ-переход начинается в каком-то одном произвольно выбранном состоянии s: для любого ϕ-маршрута, заканчивающегося моделирующим дивергенцию γ-переходом, найдётся ϕ-маршрут с той же трассой, в котором моделирующий дивергенцию γ-переход начинается в состоянии s. Более того, бесконечные ϕ-маршруты, соответствующие такому маршруту S, не добавляют новых ϕ-трасс.

Определение 104: Пусть $C \subseteq Z$ и $\mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$. Для маршрута $S \in \boldsymbol{runs}^{\omega}(\mathbf{S})$ определим множество соответствующих ему ϕ-маршрутов (маршрутов LTS $\boldsymbol{L}_{\gamma}\boldsymbol{2L}_{\phi}(\mathbf{S})$):

a) $\gamma \notin \boldsymbol{Im} \circ \boldsymbol{symbol}(S) \;\&\; (|S| < \infty \vee |\boldsymbol{trace}(S)| = \infty)$
$: \boldsymbol{runs}_{\phi}^{\omega}(S) =_{def} \boldsymbol{runs}_{\phi\text{-}\gamma}^{\omega}(S),$

a+b) $S = P \cdot \langle s, \gamma, s` \rangle \cdot R \;\&\; \gamma \notin \boldsymbol{Im} \circ \boldsymbol{symbol}(P)$
$: \boldsymbol{runs}_{\phi}(S) =_{def} \boldsymbol{runs}_{\phi\text{-}\gamma}(P) \cdot \{\langle \omega(P) — \gamma \rightarrow \gamma \rangle\},$

a+c) $\gamma \notin \boldsymbol{Im} \circ \boldsymbol{symbol}(S) \;\&\; |S| = \infty \;\&\; |\boldsymbol{trace}(R)| < \infty$
$: \boldsymbol{runs}_{\phi}(S) =_{def} \boldsymbol{runs}_{\phi\text{-}\gamma}^{\omega}(S)$
$\cup(\cup(\{\boldsymbol{runs}_{\phi\text{-}\gamma}(P) \cdot \{\langle \omega(P) — \gamma \rightarrow \gamma \rangle\} | S = P \cdot R \;\&\; R \text{ τ-маршрут}\})),$

где для любого маршрута R
$\boldsymbol{runs}_{\phi\text{-}\gamma}^{\omega}(R) =_{def} \{M \in \boldsymbol{runs}_{\phi}^{\omega}(\mathbf{S}) | M\uparrow(V_{\mathbf{S}} \times \Phi(C) \times V_{\mathbf{S}}) = R\}.$
$\boldsymbol{runs}_{\phi\text{-}\gamma}(R) =_{def} \boldsymbol{runs}_{\phi\text{-}\gamma}^{\omega}(R) \cap \boldsymbol{runs}_{\phi}(\mathbf{S}).$

□

Множество трасс соответствующих ϕ-маршрутов будем называть множеством ϕ-трасс маршрута S.

Определение 105: Пусть $C \subseteq Z$ и $\mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$. Для маршрута $S \in \boldsymbol{runs}^{\omega}(\mathbf{S})$ определим множество его ϕ-трасс как множество трасс соответствующих ϕ-маршрутов:

$\boldsymbol{traces}_{\phi}^{\omega}(S) =_{def} \boldsymbol{trace} \circ \boldsymbol{runs}_{\phi}^{\omega}(S),$
$\boldsymbol{traces}_{\phi}(S) =_{def} \boldsymbol{traces}_{\phi}^{\omega}(S) \cap \boldsymbol{traces}_{\phi}(\mathbf{S}),$
$\boldsymbol{traces}_{\phi}^{\infty}(S) =_{def} \boldsymbol{traces}_{\phi}^{\omega}(S) \cap \boldsymbol{traces}_{\phi}^{\infty}(\mathbf{S}).$

□

Из определения множества ϕ-маршрутов и ϕ-трасс маршрута вытекают очевидные следствия для множества ϕ-маршрутов и ϕ-трасс состояния s (или LTS, то есть её начального состояния s_0):

$\boldsymbol{runs}_{\phi}^{\omega}(s) = \cup \circ \boldsymbol{runs}_{\phi}^{\omega} \circ \boldsymbol{runs}^{\omega}(s),$
$\boldsymbol{runs}_{\phi}^{\omega}(\mathbf{S}) = \cup \circ \boldsymbol{runs}_{\phi}^{\omega} \circ \boldsymbol{runs}^{\omega}(\mathbf{S});$
$\boldsymbol{traces}_{\phi}^{\omega}(s) = \boldsymbol{trace} \circ \boldsymbol{runs}_{\phi}^{\omega}(s) = \boldsymbol{trace} \circ \cup \circ \boldsymbol{runs}_{\phi}^{\omega} \circ \boldsymbol{runs}^{\omega}(s) = \cup \circ \boldsymbol{traces}_{\phi}^{\omega} \circ \boldsymbol{runs}^{\omega}(s),$

$$traces_\phi^\omega(S)=trace_\circ runs_\phi^\omega(S)=trace_\circ\cup_\circ runs_\phi^\omega{}_\circ runs^\omega(S)=\cup_\circ traces_\phi^\omega{}_\circ runs^\omega(S).$$

Замкнутость множества ϕ-трасс LTS по операциям над ϕ-трассами.

Утверждение 55: Множество ϕ-трасс LTS замкнуто по следующим операциям над ϕ-трассами:

1) взятие префикса ϕ-трассы: $\mu\cdot\lambda\to\mu$,
2) удаление ϕ-символа o: $\mu\cdot\langle o\rangle\cdot\lambda\to\mu\cdot\lambda$,
3) продолжение ϕ-трассы, заканчивающейся на ϕ-символ o, неотвергаемым символом $z\notin o_r$: $\mu\cdot\langle o\rangle\to\mu\cdot\langle z\rangle$,
4) продолжение ϕ-трассы, заканчивающейся на ϕ-символ o, разрушающим символом $z\in o_g$ и разрушением: $\mu\cdot\langle o\rangle\to\mu\cdot\langle z,\gamma\rangle$.

□417

Можно было бы независимо от LTS рассматривать ϕ-модели и их свойства. Неявное определение, конечно, такое: множество ϕ-трасс является ϕ-моделью, если оно является множеством ϕ-трасс некоторой LTS. В этой работе мы не приводим эксплицитного определения ϕ-модели, как мы это сделали для других трассовых моделей: βγδ-модели, модели безопасных и модели финальных трасс. Тем не менее, такое определение можно дать и доказать его совпадение с неявным определением.

3.3.2. ξ-оператор. Генеративность ϕ-трасс (Монотонность 2:)

Структура раздела:

- ξ-оператор.
- Генеративность ϕ-трасс (Монотонность 2:).
- Безопасные ϕ-трассы и символы, безопасные после ϕ-трасс.

ξ	Определим ξ-оператор общей теории монотонности, строящий по ϕ-трассе множество *порождаемых* (генерируемых) ею βγδ-трасс и докажем выполнение условия Монотонность 2:.

ξ-оператор.

Этот оператор определяет порождаемую (генерируемую) βγδ-трассу заменой каждого ϕ-символа ϕ-трассы на конечную последовательность βδ-отказов, вложенных в ***ref***-множество этого ϕ-символа. Блокировка стимула может появиться в порождаемой βγδ-трассе, если стимул принадлежит ***ref***-множеству соответствующего ϕ-символа. Стационарность может появиться в порождаемой βγδ-трассе, если все реакции принадлежат ***ref***-множеству соответствующего ϕ-символа. Подтрассы базовых символов – стимулов,

реакций и разрушения – одинаковы в ϕ-трассе и порождаемых ею βγδ-трассах.

Заметим, что для генерации βγδ-трасс используются только ***ref***-множества ϕ-символов ϕ-трассы, а их ***gamma***-множества не используются.

При таком определении порождаемых βγδ-трасс для получения всех βγδ-трасс LTS нам достаточно только конечных ϕ-трасс. Мы формально распространим ξ-оператор на бесконечные ϕ-трассы как возвращающий пустое множество βγδ-трасс.

Определение 106: Определим ξ-оператор, строящий по ϕ-трассе множество *порождаемых* ею βγδ-трасс.

- Сначала определим ξ-оператор для конечных ϕ-трасс.
  Для $C \subseteq Z$, $o \in \Phi(C)$, $z \in C_\gamma$ и $\boldsymbol{\sigma} \in \boldsymbol{\Phi}(C)$
  - $\xi(o) =_{def} \beta\delta(o_r)^*$,
  - $\xi(z) =_{def} \{\langle z \rangle\}$,
  - $\xi(\boldsymbol{\sigma}) =_{def} \cdot(\xi_\circ\boldsymbol{\sigma}) = \xi(\boldsymbol{\sigma}(1))\cdot\ldots\cdot\xi(\boldsymbol{\sigma}(|\boldsymbol{\sigma}|))$.[62]
- Распространим ξ-оператор на бесконечные ϕ-трассы.
  Для $C \subseteq Z$ и $\boldsymbol{\sigma} \in \boldsymbol{\Phi}^\infty(C)$
  - $\xi(\boldsymbol{\sigma}) =_{def} \varnothing$.

□

Генеративность ϕ-трасс (Монотонность 2:).

Утверждение 56: **Условие Монотонность 2: выполнено:** ϕ-трассы генеративны: $\forall C \subseteq Z \ \forall \mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C) \ \cup_\circ\xi_\circ\boldsymbol{traces}_\phi(\mathbf{S}) = \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$.

□417

Безопасные ϕ-трассы и символы, безопасные после ϕ-трасс.

Определим безопасность ϕ-трассы через безопасность порождаемых ею βγδ-трасс. Аналогично определим символы, безопасные после ϕ-трассы.

Определение 107:

- ϕ-трассу LTS будем называть *безопасной*, если она генерирует хотя бы одну βγδ-трассу, безопасную в этой LTS. В противном случае ϕ-трассу будем называть *опасной*.
- Множество безопасных ϕ-трасс LTS **S** будем обозначать

---

[62] Конкатенация последовательности множеств последовательностей – это множество всех возможных конкатенаций последовательностей из этих множеств.

$safe_{\phi}(S) =_{def} \{ \sigma \in traces_{\phi}(S) \mid \xi(\sigma) \cap safe_{\beta\gamma\delta}(S) \neq \varnothing \}$.

- Символ (базовый символ или βδ-отказ) будем называть *безопасным* после ϕ-трассы, если он безопасен после хотя бы одной безопасной βγδ-трассы, генерируемой этой ϕ-трассой. В противном случае символ будем называть *опасным* после этой ϕ-трассы.
- Множество символов, безопасных после ϕ-трассы σ, обозначим[63] $safesymbols(\sigma) =_{def} \cup_{\circ} safesymbols(\Sigma \ after \ (\xi(\sigma) \cap safe(\Sigma)))$, где $\Sigma = traces_{\beta\gamma\delta}(S)$.

□

Заметим, что после опасной ϕ-трассы все символы опасны.

3.3.3. Композиция ϕ-трасс. Аддитивность (Монотонность 3:)

Структура раздела:
- Композиция ϕ-символов.
- Композиция ϕ-трасс.
- Композиция маршрутов.
- Аддитивность ϕ-трасс относительно композиции (Монотонность 3:).

⥮ для ϕ-трасс	Определим композицию ϕ-трасс общей теории монотонности и докажем выполнение условия Монотонность 3:.

Композиция ϕ-символов.

Определим композицию ϕ-символов так, чтобы ϕ-символ композиционного состояния был равен композиции ϕ-символов состояний-операндов.

Если композиционное состояние нестабильно, композицию ϕ-символов определим равной τ. Поскольку состояния-операнды стабильны, композиционное состояние нестабильно, если возможен синхронный переход: в одном состоянии-операнде есть переход по стимулу или реакции z, а в другом – по противоположному символу $\underline{z}$. В терминах ϕ-символов это означает, что существуют синхронные символы z и $\underline{z}$, не принадлежащие ***ref***-множествам соответствующих операндов.

Если композиционное состояние стабильно, то его ***ref***- и ***gamma***-множества состоят из тех асинхронных символов, которые принадлежат ***ref***- и ***gamma***-множествам состояний-операндов, соответственно.

---

[63] Множество символов, безопасных после ϕ-трассы, зависит не только от ϕ-трассы, но и от подразумеваемой LTS, из которой берётся эта ϕ-трасса.

Определение 108: Пусть $A,B\subseteq Z$. Определим композицию ϕ-символов $.\between.:\Phi(A)\times\Phi(B)\to\Phi(A\between B)_\tau$ следующим образом: $\forall a\in\Phi(A)$ $\forall b\in\Phi(B)$

$(A\backslash a_r)\cap(B\backslash b_r)\neq\varnothing \;:\; a\between b =_{def} \tau,$

$(A\backslash a_r)\cap(B\backslash b_r)=\varnothing \;:\; a\between b =_{def}$

$((A\backslash B)\cap a_r\cup(B\backslash A)\cap b_r,\ (A\backslash B)\cap a_g\cup(B\backslash A)\cap b_g).$

□

Утверждение 57: Пусть $A,B\subseteq Z$, $\mathbf{K}\in\boldsymbol{LTS}_{\beta\gamma\delta}(A)$, $\mathbf{L}\in\boldsymbol{LTS}_{\beta\gamma\delta}(B)$, и пусть состояния $k\in V_{\mathbf{K}}$ и $l\in V_{\mathbf{L}}$ стабильны. Тогда для состояния $kl\in V_{\mathbf{K}\between\mathbf{L}}$ имеет место:

a) $kl\text{—}\tau\gamma\!\!\rightarrow\!\!\!\!\!\rightarrow \Leftrightarrow \phi(k)\between\phi(l)\neq\tau$,

b) $kl\text{—}\tau\gamma\!\!\rightarrow\!\!\!\!\!\rightarrow \Rightarrow \phi(kl)=\phi(k)\between\phi(l)$.

□417

Композиция ϕ-трасс.

Определение 109: Пусть $A,B\subseteq Z$. Определим композицию ϕ-трасс $.\between.:\boldsymbol{\Phi}^\omega(A)\times\boldsymbol{\Phi}^\omega(B)\to\boldsymbol{\Phi}^\omega(A\between B)$ как наименьшее множество последовательностей, порождаемое следующими правилами вывода:

$\forall\boldsymbol{\kappa}\in\boldsymbol{\Phi}(A)\quad\forall\boldsymbol{\lambda}\in\boldsymbol{\Phi}(B)\quad\forall\boldsymbol{\mu}\in\boldsymbol{\Phi}(A\between B)$

$\forall\boldsymbol{\kappa}`\in\boldsymbol{\Phi}^\omega(A)\quad\forall\boldsymbol{\lambda}`\in\boldsymbol{\Phi}^\omega(B)\quad\forall\boldsymbol{\mu}`\in\boldsymbol{\Phi}^\omega(A\between B)$

(ϕ0) начальное правило: $\vdash \epsilon \in \epsilon \between \epsilon$,

(ϕ1) асинхронная композиция с левым базовым символом:
$\boldsymbol{\mu}\in\boldsymbol{\kappa}\between\boldsymbol{\lambda}\ \&\ z\in A_\gamma\backslash B\ \&\ \gamma\notin\boldsymbol{Im}(\boldsymbol{\mu}) \vdash \boldsymbol{\mu}\cdot\langle z\rangle \in (\boldsymbol{\kappa}\cdot\langle z\rangle)\between\boldsymbol{\lambda}$,

(ϕ2) асинхронная композиция с правым базовым символом:
$\boldsymbol{\mu}\in\boldsymbol{\kappa}\between\boldsymbol{\lambda}\ \&\ z\in B_\gamma\backslash A\ \&\ \gamma\notin\boldsymbol{Im}(\boldsymbol{\mu}) \vdash \boldsymbol{\mu}\cdot\langle z\rangle \in \boldsymbol{\kappa}\between(\boldsymbol{\lambda}\cdot\langle z\rangle)$,

(ϕ3) синхронная композиция базовых символов:
$\boldsymbol{\mu}\in\boldsymbol{\kappa}\between\boldsymbol{\lambda}\ \&\ z\in A\cap B \vdash \boldsymbol{\mu} \in (\boldsymbol{\kappa}\cdot\langle z\rangle)\between(\boldsymbol{\lambda}\cdot\langle z\rangle)$,

(ϕ4) синхронная композиция ϕ-символов:
$\boldsymbol{\mu}\in\boldsymbol{\kappa}\between\boldsymbol{\lambda}\ \&\ a\in\Phi(A)\ \&\ b\in\Phi(B) \vdash \boldsymbol{\mu}\cdot\langle a\between b\rangle\uparrow\{\tau\}\in(\boldsymbol{\kappa}\cdot\langle a\rangle)\between(\boldsymbol{\lambda}\cdot\langle b\rangle)$,

(ϕ5) моделирование дивергенции разрушением:
$\boldsymbol{\mu}\in\boldsymbol{\kappa}\between\boldsymbol{\lambda}\ \&\ \boldsymbol{\kappa}`=\boldsymbol{\lambda}`\in(A\cap B)^\infty \vdash \boldsymbol{\mu}\cdot\langle\gamma\rangle \in (\boldsymbol{\kappa}\cdot\boldsymbol{\kappa}`)\between(\boldsymbol{\lambda}\cdot\boldsymbol{\lambda}`)$,

(ϕ6) индуктивный предел: композиция ϕ-трасс, хотя бы одна из которых бесконечна, есть множество всех ϕ-трасс, каждая из которых определяется бесконечной последовательностью применения правил

$(\phi 0 \div 4)$ к ϕ-трассам-операндам, причём в этой последовательности используется каждый из символов каждой ϕ-трассы-операнда.

□

Правила $(\phi 1)$ и $(\phi 2)$ являются асинхронными. Поэтому, вообще говоря, операнды неоднозначно определяют результат. Иными словами, композиция двух ϕ-трасс – это множество ϕ-трасс. Например, для асинхронных базовых символов $a \in A \setminus \underline{B}$ и $b \in B \setminus \underline{A}$ композиция $\langle a \rangle \wr\lfloor \langle b \rangle = \{\langle a,b \rangle, \langle b,a \rangle\}$. Композиция (как множество) пуста, если операнды не компонуемы.

Аналогично один операнд $\boldsymbol{\kappa}$ и результирующая ϕ-трасса из композиционного множества $\boldsymbol{\mu} \in \boldsymbol{\kappa} \wr\lfloor \boldsymbol{\lambda}$, вообще говоря, неоднозначно определяют другой операнд $\boldsymbol{\lambda}$. Например, для синхронного базового символа $z \in A \cap \underline{B}$ и асинхронного базового символа $b \in A \setminus \underline{B}$ композиционная ϕ-трасса $\langle b \rangle \in \langle z \rangle \wr\lfloor \langle b, \underline{z} \rangle$ и $\langle b \rangle \in \langle z \rangle \wr\lfloor \langle \underline{z}, b \rangle$.

Префикс композиционной ϕ-трассы $\boldsymbol{\mu}` \leq \boldsymbol{\mu} \in \boldsymbol{\kappa} \wr\lfloor \boldsymbol{\lambda}$ принадлежит композиции некоторых префиксов операндов $\boldsymbol{\kappa}` \leq \boldsymbol{\kappa}$, $\boldsymbol{\lambda}` \leq \boldsymbol{\lambda}$, $\boldsymbol{\mu}` \in \boldsymbol{\kappa}` \wr\lfloor \boldsymbol{\lambda}`$. Обратно, для любого префикса одного операнда $\boldsymbol{\kappa}` \leq \boldsymbol{\kappa}$ найдётся (не обязательно единственный) префикс другого операнда $\boldsymbol{\lambda}` \leq \boldsymbol{\lambda}$, с которым он компонуется и каждая ϕ-трасса из композиции префиксов $\boldsymbol{\mu}` \in \boldsymbol{\kappa}` \wr\lfloor \boldsymbol{\lambda}`$ является префиксом некоторой ϕ-трассы из композиции операндов $\boldsymbol{\mu}` \leq \boldsymbol{\mu} \in \boldsymbol{\kappa} \wr\lfloor \boldsymbol{\lambda}$.

Если операнды $\boldsymbol{\kappa}$, $\boldsymbol{\lambda}$ и композиционная ϕ-трасса $\boldsymbol{\mu} \in \boldsymbol{\kappa} \wr\lfloor \boldsymbol{\lambda}$ фиксированы, то они однозначно определяют последовательность применяемых правил $(\phi 0 \div 4)$. Если хотя бы один из операндов бесконечен, то применяется правило индуктивного предела $(\phi 6)$ и, возможно, правило моделирования дивергенции разрушением $(\phi 5)$.

Если ϕ-трассы представимы в виде $\boldsymbol{\kappa} = \boldsymbol{\kappa}` \cdot \boldsymbol{\kappa}``$, $\boldsymbol{\lambda} = \boldsymbol{\lambda}` \cdot \boldsymbol{\lambda}``$, $\boldsymbol{\mu} = \boldsymbol{\mu}` \cdot \boldsymbol{\mu}``$ и префикс результата принадлежит композиции префиксов операндов $\boldsymbol{\mu}` \in \boldsymbol{\kappa}` \wr\lfloor \boldsymbol{\lambda}`$, то постфикс результата принадлежит композиции постфиксов операндов $\boldsymbol{\mu}`` \in \boldsymbol{\kappa}`` \wr\lfloor \boldsymbol{\lambda}``$.

Утверждение 58: Композиция ϕ-трасс коммутативна:
$\forall A,B \subseteq Z \ \forall \boldsymbol{\kappa} \in \boldsymbol{\Phi}^{\omega}(A) \ \forall \boldsymbol{\lambda} \in \boldsymbol{\Phi}^{\omega}(B) \ \boldsymbol{\kappa} \wr\lfloor \boldsymbol{\lambda} = \boldsymbol{\lambda} \wr\lfloor \boldsymbol{\kappa}$.

□418

Композиция маршрутов.

Напомним, что композиция LTS без θ-переходов определяется следующим образом: Пусть $A,B\subseteq Z$, $\mathbf{K}\in\boldsymbol{LTS}_{\beta\gamma\delta}(A)$, $\mathbf{L}\in\boldsymbol{LTS}_{\beta\gamma\delta}(B)$. Тогда $\mathbf{K}\|\mathbf{L}=LTS(V_{\mathbf{K}}\times V_{\mathbf{L}},(A\|B)_{\gamma},E,k_0l_0)$, где E – наименьшее множество, порождаемое следующими правилами вывода: $\forall k\in V_{\mathbf{K}}\ \forall l\in V_{\mathbf{L}}$

$(\|1)\quad z\in A_{\gamma\tau}\setminus\underline{B}\ \&\ k\text{—}z\to k`\ \vdash\ kl\text{—}z\to k`l;$

$(\|2)\quad z\in B_{\gamma\tau}\setminus\underline{A}\ \&\ l\text{—}z\to l`\ \vdash\ kl\text{—}z\to kl`;$

$(\|3)\quad z\in A\cap\underline{B}\ \&\ k\text{—}z\to k`\ \&\ l\text{—}\underline{z}\to l`\ \vdash\ kl\text{—}\tau\to k`l`.$

Определение 110: Композиция LTS индуцирует естественную композицию маршрутов:

- начальное правило: пара начал маршрутов-операндов k_0 и l_0 становится началом k_0l_0 композиции маршрутов;
- асинхронный переход: переход в одном из маршрутов-операндов по символу τ, символу γ или по асинхронному символу (из алфавита $A\setminus B$ или $B\setminus A$, соответственно), соответствует переходу по этому же символу композиции маршрутов, согласно правилам вывода $(\|1)$ и $(\|2)$;
- синхронный переход: пара переходов в маршрутах-операндах по противоположным синхронным символам (из алфавитов $A\cap\underline{B}$ и $B\cap\underline{A}$, соответственно), соответствует τ-переходу композиции маршрутов, согласно правилу вывода $(\|3)$;
- индуктивный предел: композиция маршрутов, хотя бы один из которых бесконечен, есть множество маршрутов (очевидно, бесконечных), каждый из которых определяется бесконечной последовательностью применения вышеописанных правил к маршрутам-операндам, причём в этой последовательности используется каждый из переходов каждого маршрута-операнда.

□

Заметим, что композиция маршрутов $K\|L$ определяется неоднозначно, то есть результатом композиции является множество маршрутов композиционной LTS. В то же время каждый маршрут $M\in K\|L$ однозначно определяется последовательностью применения правил $(\|1\div3)$ к маршрутам-операндам K и L. Если композиционный маршрут M бесконечен, то эта последовательность также бесконечная.

Обратно, при заданных маршрутах-операндах K и L и маршруте $M\in K\|L$ однозначно определяется последовательность применения правил $(\|1\div3)$

при композиции. Исключение составляет случай, когда каждый из маршрутов операндов K и L продолжается последовательностью τ-петель: тогда для формирования композиционного маршрута M годится любой порядок прохождения этих τ-петель. Это, однако, не влияет на справедливость утверждения: если префикс композиционного маршрута является композицией префиксов операндов, то его постфикс является композицией постфиксов операндов:

$M \in K \between L \;\&\; M=M_1 \cdot M_2 \;\&\; K=K_1 \cdot K_2 \;\&\; L=L_1 \cdot L_2 \;\&\; M_1 \in K_1 \between L_1 \Rightarrow M_2 \in K_2 \between L_2$.

Покажем как связаны композиция маршрутов и композиция ϕ-трасс этих маршрутов.

Утверждение 59: $\forall \mathbf{K}, \mathbf{L} \in \boldsymbol{LTS}_{\beta\gamma\delta} \;\forall K \in \boldsymbol{runs}^{\omega}(\mathbf{K}) \;\forall L \in \boldsymbol{runs}^{\omega}(\mathbf{L})$

1) $\forall M \in K \between L \;\forall \boldsymbol{\mu} \in \boldsymbol{traces}_{\phi}^{\omega}(M) \;\exists \boldsymbol{\kappa} \in \boldsymbol{traces}_{\phi}^{\omega}(K) \;\exists \boldsymbol{\lambda} \in \boldsymbol{traces}_{\phi}^{\omega}(L) \;\boldsymbol{\mu} \in \boldsymbol{\kappa} \between \boldsymbol{\lambda}$,

2) $\forall \boldsymbol{\kappa} \in \boldsymbol{traces}_{\phi}^{\omega}(K) \;\forall \boldsymbol{\lambda} \in \boldsymbol{traces}_{\phi}^{\omega}(L) \;\forall \boldsymbol{\mu} \in \boldsymbol{\kappa} \between \boldsymbol{\lambda} \;\exists M \in K \between L \;\boldsymbol{\mu} \in \boldsymbol{traces}_{\phi}^{\omega}(M)$.

□418

Аддитивность ϕ-трасс относительно композиции (Монотонность 3:).

Утверждение 60: **Условие Монотонность 3: выполнено:** ϕ-трассы аддитивны относительно композиции:

$\forall \mathbf{K}, \mathbf{L} \in \boldsymbol{LTS}_{\beta\gamma\delta} \;\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{K} \between \mathbf{L}) = \cup(\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{K}) \between \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{L}))$.

□424

Глава 3.4. Общий случай: Мажорирование ϕ-трасс

Структура главы:
1. Определение мажорирования ϕ-трасс
2. Мажорирование ϕ-трасс предпорядок (Монотонность 8:)
3. Генеративность мажорирования ϕ-трасс (Монотонность 4:)
4. Композиционность мажорирования ϕ-трасс (Монотонность 5:)

$\preccurlyeq$ для ϕ-трасс	Определим мажорирование ϕ-трасс общей теории монотонности и докажем выполнение условий Монотонность 8:,Монотонность 4: и Монотонность 5:.

3.4.1. Определение мажорирования ϕ-трасс

Теперь мы определим мажорирование ϕ-трасс как отношение «один к одному» и распространим его на «многие к многим».

Сначала определим мажорирование базовых символов и ϕ-символов. Для базовых символов a и b мажорирование $a \preccurlyeq b$ означает их равенство $a=b$. Для ϕ-символов в терминах стабильных состояний s и t, имеющих ϕ-символы a и b, соответственно, $a \preccurlyeq b$ означает: 1) если в s нет перехода по стимулу или реакции z, то либо в t также нет перехода по z, либо такой переход непосредственно разрушающий: $a_r \subseteq b_r \cup b_g$; 2) если в s есть непосредственно разрушающий переход по стимулу или реакции z, то в t также есть непосредственно разрушающий переход по z: $a_g \subseteq b_g$.

Отметим, что в мажорировании ϕ-символов речь идёт о непосредственно разрушающих стимулах и реакциях, а не просто о разрушающих. Если из состояния s есть переход $s \text{—} z \rightarrow s`$ по стимулу или реакция $z$, ведущий в опасное постсостояние $s`$, в котором, тем не менее, нет γ-перехода $s` \text{—} \gamma \nrightarrow$ (такой γ-переход достижим из $s`$ по непустому τ-маршруту $s` = \gamma \Rightarrow$ или $s`$ дивергентно), то символ z не попадает в ***gamma***-множество ϕ-символа состояния s.

Определение 111: **Мажорирование ϕ-символов.** Пусть задан алфавит $C \subseteq Z$. Определим отношение мажорирования базовых символов и ϕ-символов:

$\forall a,b \in C_\gamma \quad a \preccurlyeq b =_{def} a=b,$

$\forall a,b \in \Phi(C) \quad a \preccurlyeq b =_{def} a_r \subseteq b_r \cup b_g \ \& \ a_g \subseteq b_g.$

□

Мажорирование ϕ-трасс означает: 1) поэлементное мажорирование базовых символов и ϕ-символов для ϕ-трасс одной длины или 2) такое же поэлементное мажорирование префиксов ϕ-трасс и продолжение префикса мажорирующей ϕ-трассы разрушением.

Определение 112: **Мажорирование ϕ-трасс.** Пусть задан алфавит $C \subseteq Z$.

- Определим отношение мажорирования ϕ-трасс (в одном алфавите C):

  $\forall \mu, \sigma \in \Phi^{\omega}(C) \quad \mu \preccurlyeq \sigma =_{def} \mu \preccurlyeq^{\alpha} \sigma \vee \mu \preccurlyeq^{\gamma} \sigma$, где:

  A) α-мажорирование: ϕ-трассы имеют одну длину и каждый символ мажорируемой ϕ-трассы мажорируется соответствующим символом мажорирующей ϕ-трассы:

  $\mu \preccurlyeq^{\alpha} \sigma =_{def} |\mu|=|\sigma| \ \& \ \forall i \in [1..|\mu|] \ \mu(i) \preccurlyeq \sigma(i)$;

  B) γ-мажорирование: мажорирующая ϕ-трасса после префикса, α-мажорирующего префикс мажорируемой ϕ-трассы, продолжается разрушением:

  $\mu \preccurlyeq^{\gamma} \sigma =_{def} \exists \mu` \leq \mu \ \exists \sigma` \ \mu` \preccurlyeq^{\alpha} \sigma` \ \& \ \sigma = \sigma` \cdot \langle \gamma \rangle$.

- Распространим мажорирование ϕ-трасс на множества ϕ-трасс:

  $\forall I, \Sigma \subseteq \Phi^{\omega}(C) \ I \preccurlyeq \Sigma =_{def} \forall \mu \in I \ \exists \sigma \in \Sigma \ \mu \preccurlyeq \sigma$.

- Две LTS-модели A и B будем называть эквивалентными по мажорированию ϕ-трасс, если множества ϕ-трасс этих LTS мажорируют друг друга: $traces_{\phi}^{\omega}(A) \preccurlyeq traces_{\phi}^{\omega}(B) \ \& \ traces_{\phi}^{\omega}(B) \preccurlyeq traces_{\phi}^{\omega}(A)$.

□

Определение 113: *Нормальной* ϕ-трассой будем называть такую ϕ-трассу, в которой нет двух идущих подряд ϕ-символов.

□

3.4.2. Мажорирование ϕ-трасс предпорядок (Монотонность 8:)

Для общей теории монотонности требуется рефлексивность мажорирования множеств ϕ-трасс только на выбранном классе LTS-спецификаций $\mathfrak{C}$. Мы покажем рефлексивность и транзитивность мажорирования для любых отдельных ϕ-трасс и, тем самым, для множеств ϕ-трасс любых LTS-спецификаций.

Утверждение 61: Мажорирование ϕ-трасс является предпорядком (рефлексивно и транзитивно).

□424

Утверждение 62: **Условие Монотонность 8: выполнено:** мажорирование множеств ϕ-трасс LTS является предпорядком (рефлексивно и транзитивно) на классе всех LTS-спецификаций:

$\forall \mathbf{S} \in \mathit{LTS}_{\beta\gamma\delta}\ \ \mathit{traces}_{\phi}^{\omega}(\mathbf{S}) \preccurlyeq \mathit{traces}_{\phi}^{\omega}(\mathbf{S}),$

$\forall \mathbf{A},\mathbf{B},\mathbf{C} \in \mathit{LTS}_{\beta\gamma\delta} \quad \mathit{traces}_{\phi}^{\omega}(\mathbf{A}) \preccurlyeq \mathit{traces}_{\phi}^{\omega}(\mathbf{B}) \ \ \& \ \ \mathit{traces}_{\phi}^{\omega}(\mathbf{B}) \preccurlyeq \mathit{traces}_{\phi}^{\omega}(\mathbf{C})$

$\Rightarrow \quad \mathit{traces}_{\phi}^{\omega}(\mathbf{A}) \preccurlyeq \mathit{traces}_{\phi}^{\omega}(\mathbf{C}).$

□424

3.4.3. Генеративность мажорирования ϕ-трасс (Монотонность 4:)

Утверждение 63: **Условие Монотонность 4: выполнено:** мажорирование ϕ-трасс генеративно:

$\forall \mathbf{I},\mathbf{S} \in \mathit{LTS}_{\beta\gamma\delta}$

$\mathit{traces}_{\phi}^{\omega}(\mathbf{I}) \preccurlyeq \mathit{traces}_{\phi}^{\omega}(\mathbf{S}) \ \Rightarrow\ \cup\circ\xi\circ\mathit{traces}_{\phi}^{\omega}(\mathbf{I}) \preccurlyeq \cup\circ\xi\circ\mathit{traces}_{\phi}^{\omega}(\mathbf{S}).$

□424

3.4.4. Композиционность мажорирования ϕ-трасс (Монотонность 5:)

Композиционность мажорирования мы докажем на классе всех LTS-спецификаций, из чего будет следовать композиционность мажорирования на любом его подклассе. Более того, мы покажем композиционность мажорирования слева (при замене одного, для определённости, левого операнда на его мажоранту), а из неё уже выведем полную композиционность мажорирования (при замене обоих операндов на их мажоранты).

Удалено старое 64-ое утверждение

Утверждение 64: Мажорирование ϕ-трасс композиционно слева на классе всех LTS-спецификаций:

$\forall \mathbf{I}_1,\mathbf{I}_2,\mathbf{S}_1 \in \mathit{LTS}_{\beta\gamma\delta}\ \ \mathit{traces}_{\phi}^{\omega}(\mathbf{I}_1) \preccurlyeq \mathit{traces}_{\phi}^{\omega}(\mathbf{S}_1)$

$\Rightarrow \quad \cup(\mathit{traces}_{\phi}^{\omega}(\mathbf{I}_1) \between \mathit{traces}_{\phi}^{\omega}(\mathbf{I}_2)) \preccurlyeq \cup(\mathit{traces}_{\phi}^{\omega}(\mathbf{S}_1) \between \mathit{traces}_{\phi}^{\omega}(\mathbf{I}_2)).$

□427

Утверждение 65: **Условие Монотонность 5: выполнено:** мажорирование ϕ-трасс композиционно на классе всех LTS-спецификаций:

$\forall \mathbf{I}_1,\mathbf{I}_2,\mathbf{S}_1,\mathbf{S}_2 \in \mathit{LTS}_{\beta\gamma\delta}$

$\mathit{traces}_{\phi}^{\omega}(\mathbf{I}_1) \preccurlyeq \mathit{traces}_{\phi}^{\omega}(\mathbf{S}_1) \ \ \& \ \ \mathit{traces}_{\phi}^{\omega}(\mathbf{I}_2) \preccurlyeq \mathit{traces}_{\phi}^{\omega}(\mathbf{S}_2)$

$\Rightarrow \quad \cup(\mathit{traces}_{\phi}^{\omega}(\mathtt{I}_1) \between \mathit{traces}_{\phi}^{\omega}(\mathtt{I}_2)) \preccurlyeq \cup(\mathit{traces}_{\phi}^{\omega}(\mathtt{S}_1) \between \mathit{traces}_{\phi}^{\omega}(\mathtt{S}_2))$.

□432

Глава 3.5. Общий случай: Преобразование $\mathcal{T}_{\beta\gamma\delta}$

Структура главы:
1. Примеры и общая идея преобразования
2. Формальное определение преобразования
3. Конформность преобразования (Монотонность 6:)
4. *S*-ветвящиеся и локально-конечно-ветвящиеся LTS.
5. Мажорантность преобразования (Монотонность 7:)
6. Монотонность преобразования
7. Оптимизация преобразования

$\mathcal{T}=\mathcal{T}_{\beta\gamma\delta}$	Определим преобразование $\mathcal{T}: \mathfrak{C}\to\mathfrak{C}$ общей теории монотонности как преобразование $\mathcal{T}_{\beta\gamma\delta}$ на классе *S*-ветвящихся LTS-спецификаций $\mathfrak{C}=\boldsymbol{SBLTS}_{\beta\gamma\delta}$ и докажем в последующих разделах выполнение условий Монотонность 6: и Монотонность 7:.

В результате у нас окажутся доказанными все восемь достаточных условий монотонности для преобразования $\mathcal{T}_{\beta\gamma\delta}$ на классе *S*-ветвящихся LTS-спецификаций.

Далее рассмотрим простую оптимизацию преобразования, которая будет полезна, главным образом, для упрощения примеров.

3.5.1. Примеры и общая идея преобразования

Структура раздела:
- Сингуляризация состояний с безопасными реакциями.
- Особый случай: добавление полного состояния.
- Power-LTS.
- γ-однородные LTS.
- Состояния, неразличимые по безопасным βγδ-трассам.
- Разрушающие и непосредственно разрушающие символы; γ-состояние.
- Общая идея преобразования.

В этом разделе на нескольких примерах покажем идею преобразования $\mathcal{T}_{\beta\gamma\delta}$. Сначала рассмотрим преобразование спецификаций без разрушения, а потом дополнительные особенности преобразования, связанные с разрушением.

Сингуляризация состояний с безопасными реакциями.

В примере на Рис.81 в спецификации `P` в начальном состоянии `0` определены два перехода по разным безопасным реакциям `!y` и `!i`, а в

конформной реализации **K** в начальном состоянии определён переход только по одной реакции `!y`. LTS **L** конформна сама себе, и в ней в начальном состоянии определён переход по стимулу `?i`, противоположному реакции `!i`. В композиции **K**⫛**L** в начальном состоянии будет только один переход по асинхронной реакции `!y`, а в композиции **P**⫛**L**, кроме этого, ещё и τ-переход, соответствующий синхронной паре переходов LTS-операндов по реакции `!i` и стимулу `?i`. Из-за этого в композиции реализаций перед реакцией `!y` появляется блокировка стимула `{?x}`, а в композиции спецификаций этого нет: $\langle$`{?x},!y`$\rangle \in \boldsymbol{tt}(\boldsymbol{\Sigma}) \cap \mathbf{I}$, но $\langle$`{?x},!y`$\rangle \notin \boldsymbol{\Sigma}$, где $\mathbf{I}=\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{K}⫛\mathbf{L})$ и $\boldsymbol{\Sigma}=\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{P}⫛\mathbf{R})$. Тем самым композиция реализаций не конформна композиции спецификаций.

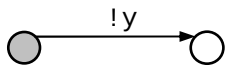
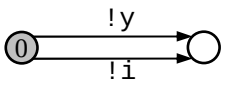
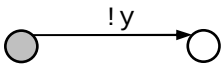
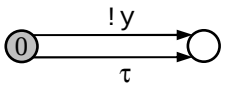
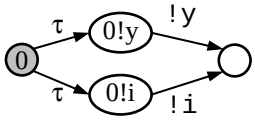
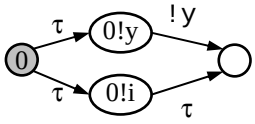
пример необходимости сингуляризации состояний с безопасными реакциями			
алфавиты:	реализации:	спецификации:	конформность:
`A={?x,!y,!i}`	**K**= 	**P**= 	**K** $\boldsymbol{ioco}_{\beta\gamma\delta}$ **P**
`B={?i}`	**L**= 		**L** $\boldsymbol{ioco}_{\beta\gamma\delta}$ **L**
`A⫛B={?x,!y}`	**K**⫛**L**= 	**P**⫛**L**= 	**K**⫛**L** $\neg\boldsymbol{ioco}_{\beta\gamma\delta}$ **P**⫛**L**
сингуляризация	**P`**= 	**P`**⫛**L**= 	**K**⫛**L** $\boldsymbol{ioco}_{\beta\gamma\delta}$ **P`**⫛**L** **P`** $\sim_{\boldsymbol{ioco}_{\beta\gamma\delta}}$ **P** $\boldsymbol{traces}_{\phi}(\mathbf{P`}) \neq \boldsymbol{traces}_{\phi}(\mathbf{P})$

Рис.81.

Если мы разделим в спецификации начальное состояние `0` на два состояния `0!y` и `0!i`, мы получим спецификацию **P`**, которая имеет то же множество βγδ-трасс, что исходная спецификация **P** и, следовательно, $\boldsymbol{ioco}_{\beta\gamma\delta}$-эквивалентна ей. Различие между спецификациями **P** и **P`** заключается в следующем: в реализации **K** есть ϕ-трасса $\langle$`({?x,!i},`$\varnothing)\rangle$ (в начальном состоянии), которая не мажорируется никакой ϕ-трассой спецификации **P**, но имееется в спецификации **P`** (в состоянии `0!y`) и, следовательно, мажорируется ϕ-трассами спецификации **P`**. В результате композиция реализаций **K**⫛**L** конформна композиции спецификаций **P`**⫛**L**.

Таким образом, для монотонности отношения $\boldsymbol{ioco}_{\beta\gamma\delta}$ требуется преобразование LTS-спецификации. Интуитивно ясно, и это видно в примере на Рис.81, что немонотонность может возникать из-за того, что после общей безопасной βγδ-трассы конформная реализация может заканчиваться в стабильном состоянии с ϕ-символом a, а спецификация – в стабильных состояниях с ϕ-символами $b_1,b_2,...$. Конформность требует[64], чтобы нашёлся такой ϕ-символ $b=b_i$, который порождает не меньшее множество βδ-отказов $\beta\delta(a_r)\subseteq\beta\delta(b_r)$. Однако, для мажорантности *при отсутствии разрушения*, как в примере на Рис.81, требуется вложенность самих ***ref***-множеств $a_r\subseteq b_r$.

Очевидно, что вложенность βδ-отказов влечёт вложенность стимулов $?(a_r)\subseteq?(b_r)$, однако это не так для реакций: может быть, что $!(a_r)$ не вложен в $!(b_r)$. В примере на Рис.81 пустая ϕ-трасса конформной реализации **K** заканчивается в начальном состоянии с ϕ-символом $a=(\{?x,!i\},\varnothing)$, а в спецификации **P** пустая ϕ-трасса заканчивается также в начальном состоянии 0 с ϕ-символом $b=(\{?x\},\varnothing)$: $?(a_r)=\{?x\}\subseteq\{?x\}=?(b_r)$, но $!(a_r)=\{!i\}\not\subseteq\varnothing=!(b_r)$.

Поэтому-то мы и преобразуем спецификацию так, чтобы «расщепить» стабильное состояние s, в котором определено несколько переходов по реакциям $!y_1,!y_2,...$, на несколько состояний $s!y_1,s!y_2,...$, оставляя в каждом состоянии $s!y_i$ только один переход по реакции $!y_i$. Тогда ϕ-символ $b=\phi(s)$ заменится на ϕ-символы $b_1=\phi(s!y_1),b_2=\phi(s!y_2),...$ такие, что для каждого i будет $?(b_{ir})=?(b_r)$ и $!(b_{ir})=!C\backslash\{!y_i\}\supseteq!(b_r)$. Тогда, очевидно, вложенность $\beta\delta(a_r)\subseteq\beta\delta(b_r)$ влечёт для каждого i вложенность $a_r\subseteq b_{ir}$. При этом, если после ϕ-символа ϕ-трасса конформной реализации продолжается реакцией $!y_i$, то в преобразованной спецификации после мажорирующего ϕ-символа b_i также будет продолжение реакцией $!y_i$.

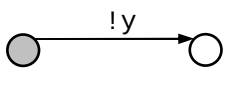
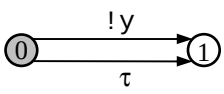
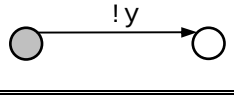
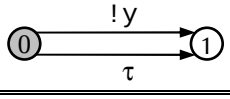
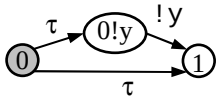
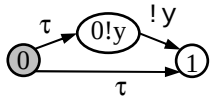
Такую операцию «расщепления» состояний мы будем называть *сингуляризацией*, состояние, в котором определено не более одного перехода по реакциям, – *сингулярным*, а LTS, в которой все стабильные состояния сингулярны, – *сингулярной*. Очевидно, что $\beta\delta(s)=\beta\delta(s!y_i)$ для каждого i, тем самым преобразование сохраняет βγδ-трассы спецификации.

---

[64] Для *S*-ветвящейся спецификации.

Особый случай: добавление полного состояния.

Особый случай возникает, когда после общей безопасной βγδ-трассы конформная реализация может заканчиваться в стабильном состоянии `k`, не порождающем βδ-отказы, то есть в этом состоянии определены переходы по каждому стимулу и хотя бы одной реакции. Такое состояние будем называть *полным*. Конформность не требует, чтобы эта βγδ-трасса в спецификации заканчивалась в полном состоянии, то есть какой-то стимул или реакция может продолжать эту βγδ-трассу в спецификации только в нестабильном состоянии `p`. Однако, при композиции реализационное состояние `k` может скомпоноваться с таким стабильным состоянием `l` другой реализации, что `kl` окажется стабильным состоянием, порождающим βδ-отказы (например, в `l` блокируется асинхронный стимул). В то же время композиция нестабильного спецификационного состояния `p` с любым состоянием `r` даёт нестабильное состояние `pr`. Тем самым соответствие может не сохраниться: βδ-отказу в композиции конформных реализаций не будет соответствовать βδ-отказ в композиции спецификаций.

<table>
<tr><th colspan="4">пример необходимости добавления стабильного состояния,
не порождающего βδ-отказы</th></tr>
<tr><td>алфавиты:</td><td>реализации:</td><td>спецификации:</td><td>конформность:</td></tr>
<tr><td>A={!y,!z}</td><td>K=  </td><td>P=  </td><td>K ioco_{βγδ} P</td></tr>
<tr><td>B={?x}</td><td colspan="2">L=  </td><td>L ioco_{βγδ} L</td></tr>
<tr><td>A⫴B={?x,!y,!z}</td><td>K⫴L=  </td><td>P⫴L=  </td><td>K⫴L ¬ioco_{βγδ} P⫴L</td></tr>
<tr><td>добавление полного состояния</td><td>P₁=  </td><td>P₁⫴L=  </td><td>K⫴L ioco_{βγδ} P₁⫴L
P₁ ~_{iocoβγδ} P
traces_ϕ(P₁)≠traces_ϕ(P)</td></tr>
<tr><td colspan="4">Рис.82.</td></tr>
</table>

В примере на Рис.82 этой ситуации отвечают начальные состояния спецификации **P** и конформной реализации **K**. Композиция реализаций **K⫴L** не конформна композиции спецификаций **P⫴L**: ⟨{?x},!y⟩∈***tt*(Σ)**∩**I**, но ⟨{?x},!y⟩∉**Σ**, где **I**=***traces***_{βγδ}(**K⫴L**) и **Σ**=***traces***_{βγδ}(**P⫴L**).

Поэтому мы должны так преобразовать спецификацию, чтобы добавить полное состояние после βγδ-трассы, если она продолжается каждым стимулом и хотя бы одной реакцией. Такую βγδ-трассу будем называть *полной*. В примере на Рис.82 добавляется состояние `0!y` (спецификация

сингулярна), а из старого состояния `0` проводится τ-переход в состояние `0!y`.

Power-LTS.

Заметим, что добавление полного мы должны делать после βγδ-трассы. Поэтому в общем случае нам требуется преобразовать исходную LTS **`S`** в power-LTS **`S`**`, состояния которой – это множества состояний LTS **`S`** после тех или иных βγδ-трасс, а переход по символу `z` ведёт из множества состояний `A` в множество состояний `B`, если в исходной LTS **`S`** хотя бы из одного состояния из `A` есть переход по `z`, и множество постсостояний всех таких переходов – это и есть множество `B`. Кроме того, если βγδ-трасса μ пуста или заканчивается на стимул или реакцию, из множества состояний **`S`** ***after*** μ проведём τ-переход в каждое подмножество состояний **`S`** ***after*** μ·o после продолжения βγδ-трассы μ последовательностью βδ-отказов o.

Замечание: Построение power-графа иногда называют «детерминизацией». Эта техника используется для построения детерминированного графа (или соответствующего автомата Мили), порождающего то же множество последовательностей, что исходный граф (автомат) [HMU02]. Если единственный вид наблюдений – это внешние действия (стимулы, реакции и разрушение), то есть в трассах наблюдений нет отказов, то power-LTS действительно получается детерминированной. В этом случае нет смысла определять ϕ-трассы. Однако при наличии отказов нужны ϕ-трассы, а тогда ϕ-трасса, не заканчивающаяся на ϕ-символ, может продолжаться несколькими разными ϕ-символами. Если ϕ-символы понимать в обычном для LTS-модели смысле как (виртуальные) петли в состояниях (в одном состоянии не более одного ϕ-символа), то это можно обеспечить только недетерминизмом: несколькими переходами по одному внешнему действию или τ-переходами. Поэтому, чтобы не вызывать ненужных ассоциаций, мы предпочитаем не использовать термин «детерминизация». Наша power-LTS будет «детерминированной» только в том смысле, что в каждом состоянии определено не более одного перехода по каждому внешнему действию, но зато есть τ-переходы, ведущие в стабильные состояния с разными ϕ-символами.

Необходимость в power-преобразовании безотносительно к добавлению полного состояния показана на Рис.83, Рис.84, Рис.85, Рис.86 и Рис.87.

Здесь пустая трасса спецификации **`P`** заканчивается в двух состояниях **`1`** и **`2`** (не считая начального, из которого ведут только τ-переходы в состояния

1 и **2**). Всего получается три непустых подмножества: **1**, **2** и **12**. Состояния **1** и **2** сингулярны, а множество **12** не сингулярно – из него выходят переходы по двум реакциям `!y` и `!z`. Поэтому мы разбиваем состояние **12** на два состояния: **12!y** и **12!z**. Отличие состояний **12!y** и **12!z** от исходных состояний **1** и **2** в том, что переходы по стимулу `?a` ведут из *каждого* состояния **12!y** и **12!z** во все возможные состояния, то есть в те, в которые вели переходы по стимулу `?a` из *хотя бы одного* состояния **1** или **2**. В результате получаем преобразованную спецификацию **P`** (Рис.83).

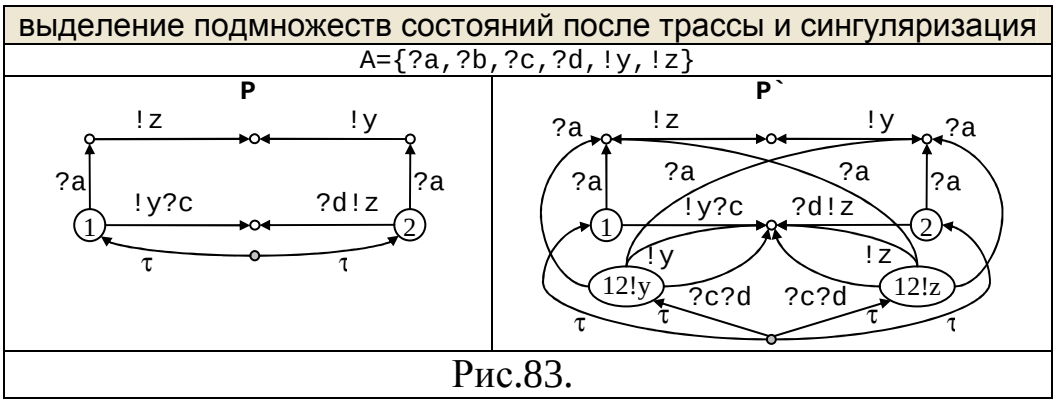


Рис.83.

Покажем необходимость каждого из четырёх состояний **1**, **2**, **12!y** и **12!z**.

Для состояния **12!z** выбирается конформная реализация $\mathbf{K}_1$, которая компонуется с LTS $\mathbf{L}_1$ (см. Рис.84). Прежде всего, заметим, что композиция реализаций $\mathbf{K}_1 \between \mathbf{L}_1$ не конформна композиции исходной спецификации $\mathbf{P} \between \mathbf{L}_1$, но конформна композиции преобразованной спецификации: $\mathbf{K}_1 \between \mathbf{L}_1$ ***ioco***$_{\beta\gamma\delta}$ $\mathbf{P}` \between \mathbf{L}_1$. Причина неконформности – в трассе ⟨`{?b},?a,!z`⟩, которая имеется в композиции реализаций $\mathbf{K}_1 \between \mathbf{L}_1$, но отсутствует в композиции исходной спецификации $\mathbf{P} \between \mathbf{L}_1$. Эта трасса, однако, имеется в композиции преобразованной спецификации $\mathbf{P}` \between \mathbf{L}_1$, где для получения этой трассы существенно используется состояние **12!z**.

Аналогично, для состояния **12!y** выбирается конформная реализация $\mathbf{K}_2$, которая компонуется с LTS $\mathbf{L}_2$ (Рис.85).

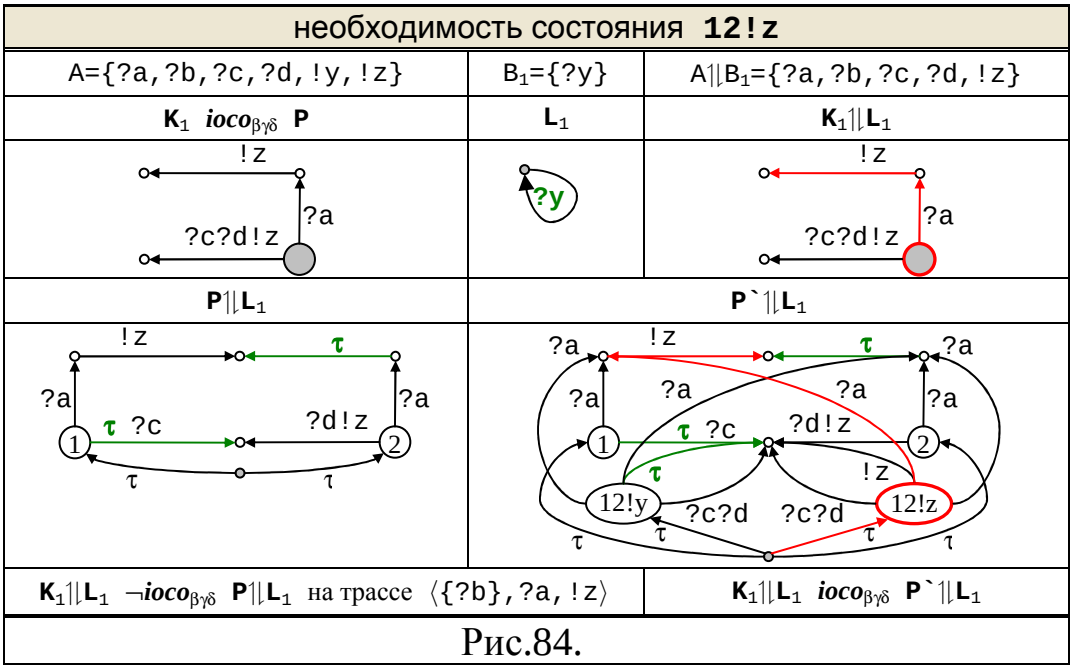


Рис.84.

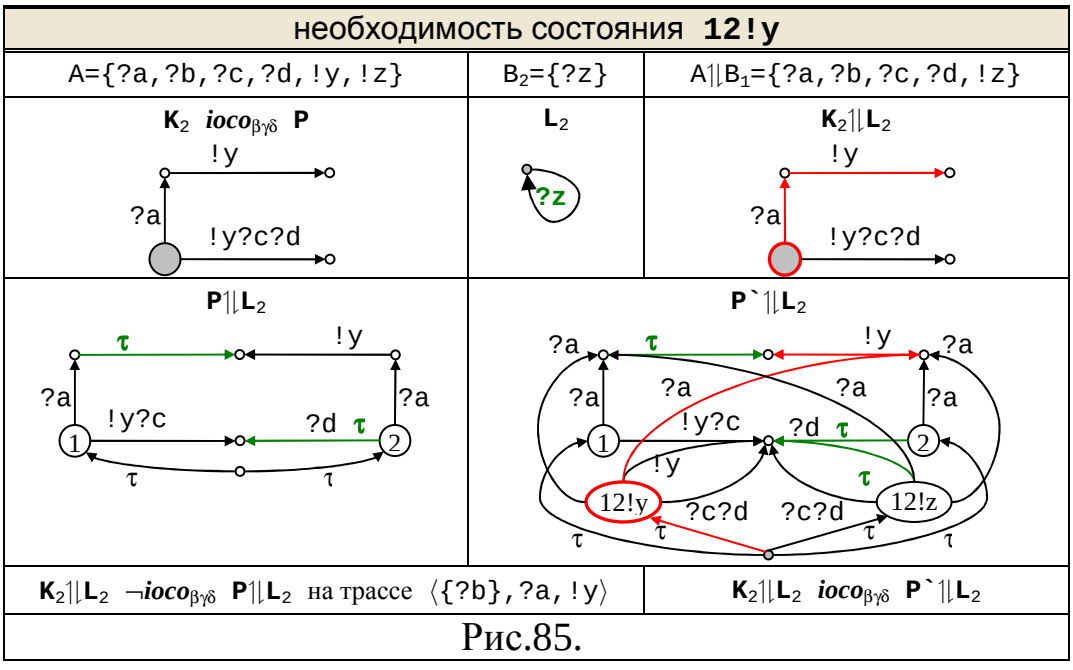


Рис.85.

Для состояний **1** и **2** в качестве конформной реализации выбирается сама спецификация **P**. При компоновке с LTS $\mathbf{L_3}$ получается трасса ⟨{?b},?a,!z⟩, для получения которой в композиции преобразованной спецификации $\mathbf{P`||L_3}$ требуется состояние **1** в **P`** (Рис.86). Аналогично, при компоновке с LTS $\mathbf{L_4}$ получается трасса ⟨{?b},?a,!y⟩, для получения которой в композиции преобразованной спецификации $\mathbf{P`||L_4}$ требуется состояние **2** в **P`** (Рис.87).

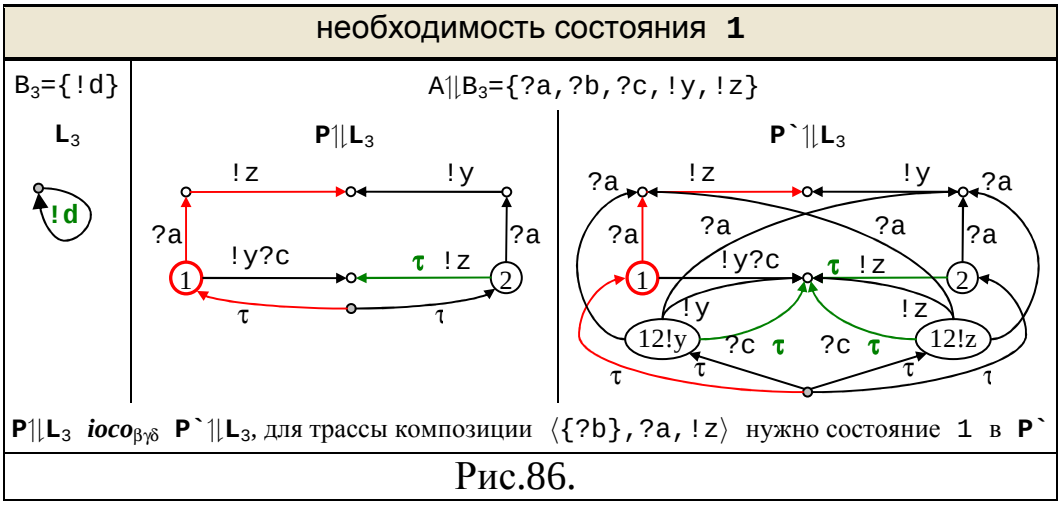


Рис.86.

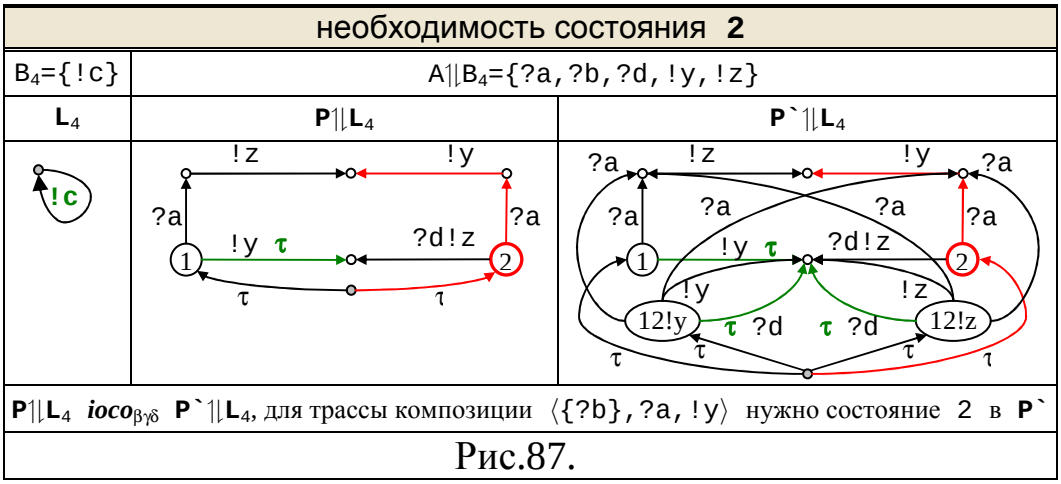


Рис.87.

γ-однородные LTS.

Теперь рассмотрим основную причину немонотонности, связанную с разрушением: нарушение γ-однородности.

В разделе 2.2.5 βγδ-модель определялась как γ-однородная, если в ней каждая βγδ-трасса, продолжаемая *некоторой* реакцией и далее разрушением, продолжалась *каждой* реакцией и далее разрушением. Определим соответствующим образом γ-однородную LTS.

Определение 114: Пусть для $C \subseteq Z$ задана LTS-модель $S \in LTS_{\beta\gamma\delta}(C)$. Будем говорить, что **S** *γ-однородна*, если γ-однородно множество её βγδ-трасс.

□

Покажем, что нарушение γ-однородности может служить причиной немонотонности.

В примере на Рис.88 спецификация **P** не γ-однородна: в начальном состоянии есть разрушающая реакция `!y`, но для реакции `!i` либо нет перехода, либо он не разрушающий (изображён пунктиром). В то же время в конформной реализации в начальном состоянии может быть определён переход по реакции `!i` с любым дальнейшим продолжением. На рисунке реализация **K** $\mathit{ioco}_{\beta\gamma\delta}$ **P** и в её начальном состоянии определён переход по реакции `!i` и далее переход по стимулу `?x` и по реакции `!y`. Здесь ϕ-трасса конформной реализации $\langle$`!i,?x,!y`$\rangle \in \mathit{traces}_{\phi}$(**K**) не мажорируется никакой ϕ-трассой спецификации **P**. Из-за этого в композиции βγδ-трасса $\langle$`?x,!y`$\rangle \in \mathit{tt}(\Sigma) \cap \mathbf{I}$, но $\langle$`?x,!y`$\rangle \notin \Sigma$, где $\mathbf{I}=\mathit{traces}_{\beta\gamma\delta}(\mathbf{K} \upharpoonright\!\downarrow \mathbf{L})$ и $\Sigma=\mathit{traces}_{\beta\gamma\delta}(\mathbf{P} \upharpoonright\!\downarrow \mathbf{L})$, то есть **I** $\neg\mathit{ioco}_{\beta\gamma\delta}$ **Σ**.

Для того, чтобы достичь монотонности, мы должны «раскопировать» разрушающий переход по реакции: если в состоянии есть разрушающий переход по одной реакции, то добавить разрушающие переходы по всем реакциям. Так получится спецификация **P`** и **K**$\upharpoonright\!\downarrow$**L** $\mathit{ioco}_{\beta\gamma\delta}$ **P`**$\upharpoonright\!\downarrow$**L**.

Заметим, что в этом примере мы специально выбрали спецификацию без блокировок.

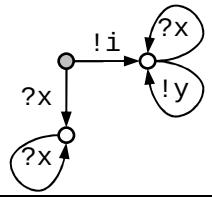
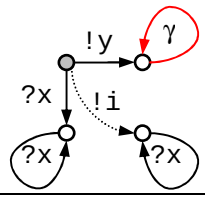
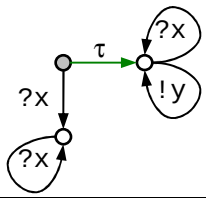
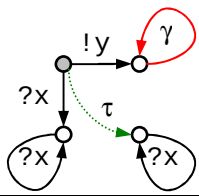
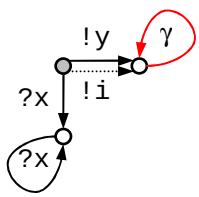
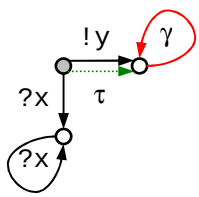
«раскопирование» разрушающего перехода для всех реакций			
алфавиты:	реализации:	спецификации (без блокировок):	конформность:
`A={?x,!y,!i}`	**K=** 	**P=** 	**K** $\mathit{ioco}_{\beta\gamma\delta}$ **P**
`B={?i}`	**L=** 		**L** $\mathit{ioco}_{\beta\gamma\delta}$ **L**
`A||B={?x,!y}`	**K**$\upharpoonright\!\downarrow$**L=** 	**P**$\upharpoonright\!\downarrow$**L=** 	**K**$\upharpoonright\!\downarrow$**L** $\neg\mathit{ioco}_{\beta\gamma\delta}$ **P**$\upharpoonright\!\downarrow$**L**
добавление разрушающего перехода по реакции	**P`=** 	**P`**$\upharpoonright\!\downarrow$**L=** 	**K**$\upharpoonright\!\downarrow$**L** $\mathit{ioco}_{\beta\gamma\delta}$ **P`**$\upharpoonright\!\downarrow$**L** **P`** $\sim_{\mathit{ioco}_{\beta\gamma\delta}}$ **P** traces_{ϕ}(**P`**)$\neq\mathit{traces}_{\phi}$(**P**)

Рис.88.

Состояния, неразличимые по безопасным βγδ-трассам.

Теперь рассмотрим ещё одну причину немонотонности, связанную с разрушением, и покажем, что она решается с помощью построения power-LTS, не требуя каких-то дополнительных преобразований.

Такой причиной немонотонности является наличие разрушающего перехода по реакции в одном состоянии при отсутствии разрушающего перехода по этой реакции *в другом* состоянии, если эти состояния *неразличимы* по безопасным βγδ-трассам, то есть достижимы по одним и тем же безопасным βγδ-трассам.

В примере на Рис.89 в спецификации **P** только одна реакция !y, которая в одном состоянии a разрушающая, а в другом состоянии b, достижимом *по тем же самым* безопасным βγδ-трассам, неразрушающая: переход b—!y→ (показан пунктиром) не приводит к разрушению или отсутствует. В то же время в конформной реализации **K** только в начальном состоянии есть переход по реакции !y, ведущий к разрушению.

В спецификации состояния a и b отличаются ведущими к ним ϕ-трассами: состояние a нестабильно и в нём нет ϕ-символа, а b стабильно и в нём есть ϕ-символ: a) (∅,∅), если b—!y→, или b) (δ,∅), если b—!y↛. Из-за этого при композиции спецификаций **P**⫛**L** композиционное состоянии be имеет ϕ-символ с βδ-отказом: a) ({?x},∅) или b) ({?x}∪δ,∅). В то же время при композиции реализаций **K**⫛**L** композиционное состояние ae не имеет ϕ-символа. В результате в композиции спецификаций **P**⫛**L** реакции безопасны после блокировки {?x}, а в композиции реализаций **K**⫛**L** это не так, что означает нарушение гипотезы о безопасности. Здесь ϕ-трасса конформной реализации $\langle(\varnothing,\varnothing),!y,\gamma\rangle\in \boldsymbol{traces}_{\phi}(\mathbf{K})$ не мажорируется никакой ϕ-трассой спецификации **P**. Из-за этого βγδ-трасса $\langle\{?x\},!y\rangle\in \boldsymbol{tt}(\mathbf{\Sigma})\cap\mathbf{I}$, но $\langle\{?x\},!y\rangle\notin \boldsymbol{tt}(\mathbf{I})$, где $\mathbf{I}=\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{K}⫛\mathbf{L})$ и $\mathbf{\Sigma}=\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{P}⫛\mathbf{L})$, то есть $\mathbf{I}\ \neg\boldsymbol{ioco}_{\beta\gamma\delta}\ \mathbf{\Sigma}$.

Рассмотрим в спецификации **P** ϕ-трассу **μ**, заканчивающуюся ϕ-символом состояния b: a) $\boldsymbol{\mu}=\langle(\varnothing,\varnothing)\rangle$ или b) $\boldsymbol{\mu}=\langle(\delta,\varnothing)\rangle$. Эта ϕ-трасса безопасна (генерирует безопасную пустую βγδ-трассу), но реакция !y опасна после

неё. Заметим, что в **P** ϕ-трасса **μ** не имеет мажоранты, которая продолжалась бы ⟨!y,γ⟩, а в **P`** такая мажоранта есть – это сама ϕ-трасса **μ**.

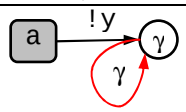
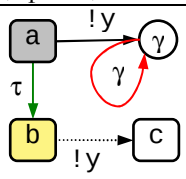
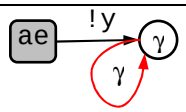
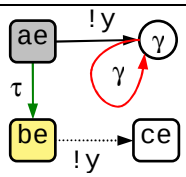
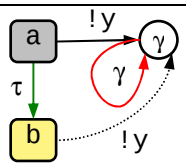
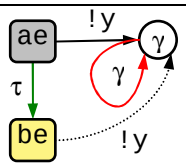
<table>
<tr><th colspan="4">добавление разрушающего перехода по опасной реакции</th></tr>
<tr><td>алфавиты:</td><td>реализации:</td><td>спецификации:</td><td>конформность:</td></tr>
<tr><td>A={!y}</td><td>K= 
</td><td>P= 
</td><td>K ioco_{βγδ} P</td></tr>
<tr><td>B={?x}</td><td colspan="2">L= 
</td><td>L ioco_{βγδ} L</td></tr>
<tr><td>A⅂⌊B={?x,!y}</td><td>K⅂⌊L= 
</td><td>P⅂⌊L= 
</td><td>K⅂⌊L ¬safe for P⅂⌊L</td></tr>
<tr><td>добавление разрушающего перехода по реакции</td><td>P`= 
</td><td>P`⅂⌊L= 
</td><td>K⅂⌊L ioco_{βγδ} P`⅂⌊L</b><br><b>P` ~_{iocoβγδ} P
traces_ϕ(P`)≠traces_ϕ(P)</td></tr>
</table>

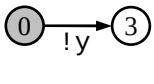
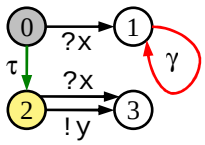
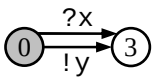
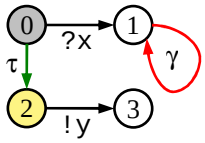
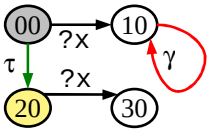
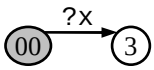
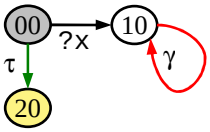
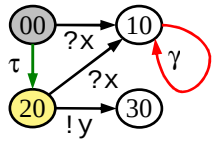
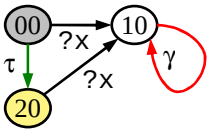
Рис.89.

Аналогичная ситуация показана в примере на Рис.90, но только для стимулов. В спецификации в состоянии 0 определён переход по стимулу ?x, ведущий в γ-состояние, а в состоянии 2, достижимого *по тем же самым* безопасным βγδ-трассам, такого перехода нет. Здесь рассмотрены два случая в зависимости от того, есть ли в состоянии 2 неразрушающий переход по стимулу ?x (случай 1) или в состоянии 2 вообще нет переходов по стимулу ?x (случай 2).

В обоих случаях композиция с LTS **L** демонстрирует нарушение монотонности. В случае 1 в композиции реализаций имеется «лишняя» блокировка стимула после стационарности: $\langle\delta,\{?x\}\rangle\in\boldsymbol{tt}(\Sigma_1)\cap\mathbf{I}_1$, но $\langle\delta,\{?x\}\rangle\notin\Sigma_1$, где $\mathbf{I}_1=\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{K}_1\text{⅂⌊}\mathbf{L})$ и $\Sigma_1=\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{P}_1\text{⅂⌊}\mathbf{L})$, то есть $\mathbf{I}_1$ $\neg\boldsymbol{ioco}_{\beta\gamma\delta}$ Σ_1. В случае 2, наоборот, имеется «лишний» приём стимула: $\langle\delta,?x\rangle\in\boldsymbol{tt}(\Sigma_2)\cap\mathbf{I}_2$, но $\langle\delta,?x\rangle\notin\Sigma_2$, где $\mathbf{I}_2=\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{K}_2\text{⅂⌊}\mathbf{L})$ и $\Sigma_2=\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{P}_2\text{⅂⌊}\mathbf{R})$, то есть $\mathbf{I}_2$ $\neg\boldsymbol{ioco}_{\beta\gamma\delta}$ Σ_2.

Здесь LTS **L** выбрана таким образом, чтобы удаление перехода по реакции создавало стационарность, после которой стимул ?x становится

безопасным в спецификации. Аналогично можно было бы выбрать LTS **L** так, чтобы стимул ?x становился безопасным после добавляемой блокировки некоторого нового стимула {?a}: для этого достаточно в LTS **L** вместо алфавита B={?y} использовать алфавит B={?a}.

<table>
<tr><th colspan="4">добавление разрушающего перехода по опасному стимулу</th></tr>
<tr><th>реализации:</th><th>спецификации:</th><th>алфавиты:</th><th>конформность:</th></tr>
<tr><td>K₁=  </td><td>P₁=  </td><td rowspan="2">A={!y,?x}</td><td>K₁ ioco_{βγδ} P₁</td></tr>
<tr><td>K₂=  </td><td>P₂=  </td><td>K₂ ioco_{βγδ} P₂</td></tr>
<tr><td colspan="2">L=  </td><td>B={?y}</td><td>L ioco_{βγδ} L</td></tr>
<tr><td>K₁||L=  </td><td>P₁||L=  </td><td rowspan="2">A||B={?x}</td><td>K₁||L ¬ioco_{βγδ} P₁||L</td></tr>
<tr><td>K₂||L=  </td><td>P₂||L=  </td><td>K₂||L ¬ioco_{βγδ} P₂||L</td></tr>
<tr><td>P`</b><sub>1</sub>= <b>P`₂=  </td><td>P`</b><sub>1</sub>||<b>L</b>= <b>P`₂||L=  </td><td colspan="2">K₁||L ioco_{βγδ} P`</b><sub>1</sub>||<b>L</b>=<b>P`₂||L ioco_{βγδ} K₂||L
P₁ ~_{iocoβγδ} P`</b><sub>1</sub>=<b>P`₂ ~_{iocoβγδ} P₂</td></tr>
<tr><td colspan="4">traces_ϕ(P₁)≠traces_ϕ(P`</b><sub>1</sub>)=<b><i>traces</i></b><sub>ϕ</sub>(<b>P`₂)≠traces_ϕ(P₂)</td></tr>
<tr><td colspan="4">Рис.90.</td></tr>
</table>

Для того, чтобы устранить эту причину немонотонности, нам не требуется делать какого-то специального преобразования. Достаточно построения power-LTS, поскольку, по построению, в power-LTS все её состояния (множества состояний исходной LTS) различаются по βγδ-трассам.

Разрушающие и непосредственно разрушающие символы; γ-состояние.

Стимул или реакция z разрушающий в состоянии s, если есть переход s—z→s`, ведущий в опасное постсостояние s`, то есть в s` есть γ-трасса: s`=γ⇒. Если γ-переход определён непосредственно в постсостоянии s`, то есть s`—γ→, то z непосредственно разрушающий в состоянии s. Разрушающий z-переход, вообще говоря, не является непосредственно разрушающим: γ-переход может быть достижим из s` по

τ-маршрутам $s—z→s`—\langle\tau...\tau\rangle→s``—\gamma→$, или состояние $s`$ может быть дивергентным, то есть в нём начинается бесконечный τ-маршрут $s—z→s`—\langle\tau...\rangle→$.

Можно показать, что, если в каждом состоянии s LTS $\mathbf{S}$ для каждого разрушающего z-перехода $s—z→s`=\gamma\Rightarrow$ добавить непосредственно разрушающий z-переход $s—z→s``—\gamma→$, то получится LTS $\mathbf{S}`$, ϕ-трассы которой мажорируют ϕ-трассы исходной LTS $\mathbf{S}$. При этом не важно, какие ещё переходы, кроме γ-перехода будут определены в постсостоянии $s``$ и куда будет вести γ-переход $s``—\gamma→$. Можно считать, что все добавляемые z-переходы ведут в единственное специальное γ-состояние, в котором определяется единственный переход γ-петля $\gamma—\gamma→\gamma$. Более того, «старые» разрушающие z-переходы можно удалить.

Для того, чтобы в этом убедиться, достаточно посмотреть, как меняются ϕ-трассы LTS. Пусть в исходной LTS $\mathbf{S}$ ϕ-трасса $\boldsymbol{\mu}$ не проходит через разрушающие переходы по стимулам и реакциям, но продолжается стимулом или реакцией z, разрушающим в состоянии s. Любая ϕ-трасса $\boldsymbol{\mu}\cdot\langle z\rangle\cdot\boldsymbol{\lambda}$ в LTS $\mathbf{S}$ γ-мажорируется ϕ-трассой $\boldsymbol{\mu}\cdot\langle z,\gamma\rangle$ в преобразованной LTS $\mathbf{S}`$. Если состояние $s$ стабильно и имеет ϕ-символ $\phi(s)=a$, то в LTS $\mathbf{S}$ есть также ϕ-трасса $\boldsymbol{\mu}\cdot\langle a,z\rangle\cdot\boldsymbol{\lambda}$. В преобразованной LTS $\mathbf{S}`$ $\phi(s)=b$ такое, что $a_r=b_r$ и $a_g\cup\{z\}\subseteq b_g$. Поэтому $a\preccurlyeq b$. Следовательно, ϕ-трасса $\boldsymbol{\mu}\cdot\langle a,z\rangle\cdot\boldsymbol{\lambda}$ γ-мажорируется ϕ-трассой $\boldsymbol{\mu}\cdot\langle b,z,\gamma\rangle$ в LTS $\mathbf{S}`$.

Легко также видеть, что такое преобразование не меняет множество безопасных βγδ-трасс, следовательно, оно инвариантно: $\mathbf{S}\sim_{\boldsymbol{ioco}_{\beta\gamma\delta}}\mathbf{S}`$.

Поскольку при композиции τ- и γ-переходы выполняются асинхронно, для любой LTS $\mathbf{T}$ композиции $\mathbf{S}\!\downarrow\!\!\uparrow\mathbf{T}$ и $\mathbf{S}`\!\downarrow\!\!\uparrow\mathbf{T}$ также $\boldsymbol{ioco}_{\beta\gamma\delta}$-эквивалентны.

Вместе с тем следует отметить, что замена разрушения на непосредственное разрушение не является необходимым условием монотонности. Например, рассмотрим в алфавите $\{!y\}$ спецификацию $\mathbf{S}$, состоящую только из переходов $s_0—!y→s_1—\tau→s_2—\gamma→s_3$. Любая безопасная LTS-реализация $\mathbf{I}$ в алфавите $\{!y\}$ конформна $\mathbf{S}$. Нетрудно видеть, что для любой LTS $\mathbf{T}$ композиция $\mathbf{I}\!\downarrow\!\!\uparrow\mathbf{T}$ $\boldsymbol{ioco}_{\beta\gamma\delta}$ $\mathbf{S}\!\downarrow\!\!\uparrow\mathbf{T}$, то есть отношение $\boldsymbol{ioco}_{\beta\gamma\delta}$ левомонотонно.

Общая идея преобразования.

Теперь опишем неформально преобразование $\mathcal{T}_{\beta\gamma\delta}$ в общем случае. Сначала мы определим его как преобразование трассовой модели в LTS, а потом как преобразование LTS→LTS.

Преобразование $\mathcal{T}_{\beta\gamma\delta}{}^{t}: \boldsymbol{MODEL}_{\beta\gamma\delta}(C) \rightarrow \boldsymbol{LTS}_{\beta\gamma\delta}(C)$ будет похоже на преобразование $\boldsymbol{T}_{\beta\gamma\delta}\boldsymbol{2}_{\boldsymbol{compact}}\boldsymbol{L}_{\gamma}: \boldsymbol{MODEL}_{\beta\gamma\delta}(C) \rightarrow \boldsymbol{LTS}_{\beta\gamma\delta}(C)$, которое строит по трассовой модели "компактную" LTS с тем же множеством βγδ-трасс (Определение 56:). Отличия определяются необходимостью сингуляризации, добавления полного состояния и проведением разрушающих переходов в специальное γ-состояние.

Пусть у нас есть βγδ-модель Σ в алфавите $C \subseteq Z$.
Для краткости будем обозначать подмодель после безопасной βγδ-трассы $[\mu] = \Sigma$ ***after*** μ.

Тау-трассой будем называть безопасную βγδ-трассу μ, которая не заканчивается βδ-отказом (пуста или заканчивается стимулом или реакцией). Для этой βγδ-трассы мы создадим нестабильное power-состояние $[\mu]$.

Определим переходы из такого нестабильного power-состояния по стимулам и реакциям, которые проведём также в нестабильные power-состояния.

Если βγδ-трасса μ продолжается безопасным стимулом $?x$ или безопасной реакцией $!y$, определим переход $[\mu] — ?x \rightarrow [\mu \cdot \langle ?x \rangle]$ или $[\mu] — !y \rightarrow [\mu \cdot \langle !y \rangle]$, соответственно. Если стимул $?x$ опасен после μ, определим переход по нему в специальное γ-состояние $[\mu] — ?x \rightarrow \gamma$. Если реакции опасны после μ, определим переход в γ-состояние по *каждой* реакции $[\mu] — !y \rightarrow \gamma$.

Теперь определим стабильные power-состояния, τ-переходы из нестабильного power-состояния в стабильные power-состояния и переходы по стимулам и реакциям из стабильных power-состояний.

Стабильной трассой будем называть безопасную βγδ-трассу, которая заканчивается βδ-отказом или полна (особый случай). *Гамма-трассой* будем называть безопасную βγδ-трассу, после которой реакции опасны, а *дельта-трассой* – безопасную βγδ-трассу, которая не продолжается реакциями.

Рассмотрим все стабильные βγδ-трассы $\mu\cdot\mathbf{o}$, где $\mathbf{o}$ трасса βδ-отказов ($\mathbf{o}=\epsilon$ только в особом случае, когда βγδ-трасса μ полна).

Если стабильная трасса $\mu\cdot\mathbf{o}$ безопасно продолжается реакцией $!y$, определим стабильное power-состояние как пару из множества состояний после $\mu\cdot\mathbf{o}$ и выделенной безопасной продолжающей реакции $\mathbf{[}\mu\cdot\mathbf{o]}!y$. Также определим τ-переход $\mathbf{[}\mu\mathbf{]}\text{—}\tau\rightarrow\mathbf{[}\mu\cdot\mathbf{o]}!y$ и переход по реакции $\mathbf{[}\mu\cdot\mathbf{o]}!y\text{—}!y\rightarrow\mathbf{[}\mu\cdot\mathbf{o}\cdot!y\rangle\mathbf{]}$.

Для стабильной гамма-трассы $\mu\cdot\mathbf{o}$ определим стабильное power-состояние $\mathbf{[}\mu\cdot\mathbf{o]}\gamma$. Также определим τ-переход $\mathbf{[}\mu\mathbf{]}\text{—}\tau\rightarrow\mathbf{[}\mu\cdot\mathbf{o]}\gamma$ и переходы по каждой реакции в γ-состояние $\mathbf{[}\mu\cdot\mathbf{o]}\gamma\text{—}!y\rightarrow\gamma$.

Для стабильной дельта-трассы $\mu\cdot\mathbf{o}$ определим стабильное power-состояние $\mathbf{[}\mu\cdot\mathbf{o]}\delta$. Также определим τ-переход $\mathbf{[}\mu\mathbf{]}\text{—}\tau\rightarrow\mathbf{[}\mu\cdot\mathbf{o]}\delta$ (переходов по реакциям не будет).

Аналогично переходам по стимулам из нестабильного power-состояния, определим переходы по продолжающим стимулам из стабильных power-состояний $\mathbf{[}\mu\cdot\mathbf{o]}t\text{—}?x\rightarrow\mathbf{[}\mu\cdot\mathbf{o}\cdot\langle?x\rangle\mathbf{]}$ или $\mathbf{[}\mu\cdot\mathbf{o]}t\text{—}?x\rightarrow\gamma$, где $t\in!C_{\gamma\delta}$.

В γ-состоянии определим единственный переход – γ-петлю.

Преобразование $\mathcal{T}_{\beta\gamma\delta}:\boldsymbol{LTS}_{\beta\gamma\delta}(C)\rightarrow\boldsymbol{LTS}_{\beta\gamma\delta}(C)$ отличается от преобразования $\mathcal{T}_{\beta\gamma\delta}^{t}:\boldsymbol{MODEL}_{\beta\gamma\delta}(C)\rightarrow\boldsymbol{LTS}_{\beta\gamma\delta}(C)$ только тем, что в качестве power-состояния мы будем использовать не трассовую подмодель после безопасной βγδ-трассы, а множество состояний исходной LTS после этой βγδ-трассы. Для LTS $\mathbf{S}$ мы будем обозначать для краткости $[\mu]=\mathbf{S}\ \boldsymbol{after}\ \mu$.

Можно показать, что преобразования $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ и $\mathcal{T}_{\beta\gamma\delta}^{t}\circ\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$ связаны между собой отношением строгой бисимуляции [Miln89], которую можно задать как отображение достижимых power-состояний первой LTS в достижимые power-состояния второй LTS:

$$\boldsymbol{f}:\boldsymbol{der}\circ\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})\rightarrow\boldsymbol{der}\circ\mathcal{T}_{\beta\gamma\delta}^{t}\circ\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$$
$$:\ [\mu]\rightarrow\mathbf{[}\mu\mathbf{]},\ [\mu]!y\rightarrow\mathbf{[}\mu\mathbf{]}!y,\ [\mu]\gamma\rightarrow\mathbf{[}\mu\mathbf{]}\gamma,\ [\mu]\delta\rightarrow\mathbf{[}\mu\mathbf{]}\delta,\ \gamma\rightarrow\gamma.$$
$$\forall u\in C_{\gamma\tau}\ \ \forall v,v`\in\boldsymbol{Dom}(\boldsymbol{f})\ \ \forall\mathbf{v},\mathbf{v}`\in\boldsymbol{Im}(\boldsymbol{f})$$
$$\boldsymbol{f}(v_0)=\mathbf{v_0}$$

& (v—u→v` ⇒ ***f***(v)—u→***f***(v`))

& (**v**—u→**v`** & ***f***(v)=**v** & ***f***(v`)=**v`** ⇒ v—u→v`).

3.5.2. Формальное определение преобразования

Структура раздела:
- Виды βγδ-трасс.
- Преобразование βγδ-модели.
- Преобразование LTS-модели.
- Вспомогательные утверждения.

Виды βγδ-трасс.

Определение 115: Пусть $C \subseteq Z$ и $\Sigma = \boldsymbol{MODEL}_{\beta\gamma\delta}(C)$.

- *Tau-трассой* будем называть безопасную βγδ-трассу, которая не заканчивается βδ-отказом:
  $\boldsymbol{T}(\Sigma) =_{def} \{\mu \in \boldsymbol{safe}(\Sigma) \mid \boldsymbol{RefT}(\mu) = \epsilon\}$.
- *Полной трассой* будем называть безопасную βγδ-трассу μ, которая продолжается каждым стимулом и хотя бы одной реакцией (не продолжается лишь некоторыми, но не всеми, реакциями):
  μ полна $=_{def}$ $C \setminus \boldsymbol{head}(\Sigma \ \boldsymbol{after}\ \mu) \subset !C$.
- *Стабильной трассой* будем называть безопасную βγδ-трассу, которая заканчивается βδ-отказом или полна:
  $\boldsymbol{S}(\Sigma) =_{def} \{\mu \in \boldsymbol{safe}(\Sigma) \mid \boldsymbol{RefT}(\mu) \neq \epsilon \vee \mu\text{полна}\}$.
- *Гамма-трассой* будем называть безопасную βγδ-трассу, после которой реакции опасны:
  $\boldsymbol{G}(\Sigma) =_{def} \{\mu \in \boldsymbol{safe}(\Sigma) \mid \exists !y \in C \ \mu \cdot \langle !y, \gamma \rangle \in \Sigma\}$.
- *Дельта-трассой* будем называть безопасную βγδ-трассу, которая не продолжается реакциями:
  $\boldsymbol{D}(\Sigma) =_{def} \{\mu \in \boldsymbol{safe}(\Sigma) \mid \forall !y \in C \ \mu \cdot \langle !y \rangle \notin \Sigma\}$.

□

Преобразование βγδ-модели.

Определение 116: Пусть $C \subseteq Z$ и $\Sigma = \boldsymbol{MODEL}_{\beta\gamma\delta}(C)$. Обозначим:

$\mu \in \Sigma : \boldsymbol{[\mu]} =_{def} \Sigma \ \boldsymbol{after}\ \mu$.

$M \subseteq \Sigma : \boldsymbol{[M]} =_{def} \Sigma \ \boldsymbol{after}\ M = \{\Sigma \ \boldsymbol{after}\ \mu \mid \mu \in M\} = \{\boldsymbol{[\mu]} \mid \mu \in M\}$.

□

Определение 117: Для $C \subseteq Z$ определим преобразование $\boldsymbol{\tau}_{\beta\gamma\delta}^{t} : \boldsymbol{MODEL}_{\beta\gamma\delta}(C) \to \boldsymbol{LTS}_{\beta\gamma\delta}$. Для $\boldsymbol{\Sigma} \in \boldsymbol{MODEL}_{\beta\gamma\delta}(C)$ определим

$\mathcal{T}_{\beta\gamma\delta}^{t}(\Sigma)=\mathtt{LTS}(\mathtt{V},\mathtt{C}_\gamma,\mathtt{E},\mathtt{t}_0)$, где $\mathtt{V}=[\boldsymbol{T}(\Sigma)]\cup([\boldsymbol{S}(\Sigma)]\times!\mathtt{C}_{\gamma\delta})\cup\{\gamma\}$, $\mathtt{t}_0=[\epsilon]=\boldsymbol{\Sigma}$, если $\boldsymbol{\Sigma}$ ***safe***, или $\mathtt{t}_0=\gamma$ в противном случае, а $\mathtt{E}$ – наименьшее множество, порождаемое следующими правилами вывода:

$\forall\mu\in\boldsymbol{T}(\Sigma)\ \ \forall!y\in\mathtt{C}\ \ \forall?x\in\mathtt{C}\ \ \forall\mathbf{o}\in\beta\delta(\mathtt{C})^*$

(γ)		$\vdash \gamma\text{—}\gamma\rightarrow\gamma$;
(T!T)	$\mu\cdot\langle!y\rangle\in\boldsymbol{T}(\Sigma)$	$\vdash [\mu]\text{—}!y\rightarrow[\mu\cdot\langle!y\rangle]$;
(T!γ)	$\mu\in\boldsymbol{G}(\Sigma)\cap\boldsymbol{T}(\Sigma)$	$\vdash [\mu]\text{—}!y\rightarrow\gamma$;
(T?T)	$\mu\cdot\langle?x\rangle\in\boldsymbol{T}(\Sigma)$	$\vdash [\mu]\text{—}?x\rightarrow[\mu\cdot\langle?x\rangle]$;
(T?γ)	$\mu\cdot\langle?x,\gamma\rangle\in\Sigma$	$\vdash [\mu]\text{—}?x\rightarrow\gamma$;
(S!T)	$\mu\cdot\mathbf{o}\in\boldsymbol{S}(\Sigma)$ & $\mu\cdot\mathbf{o}\cdot\langle!y\rangle\in\boldsymbol{T}(\Sigma)$	$\vdash [\mu]\text{—}\tau\rightarrow[\mu\cdot\mathbf{o}]!y$ $[\mu\cdot\mathbf{o}]!y\text{—}!y\rightarrow[\mu\cdot\mathbf{o}\cdot\langle!y\rangle]$;
(S?T)	—"— & —"— & $\mu\cdot\mathbf{o}\cdot\langle?x\rangle\in\boldsymbol{T}(\Sigma)$	$\vdash [\mu\cdot\mathbf{o}]!y\text{—}?x\rightarrow[\mu\cdot\mathbf{o}\cdot\langle?x\rangle]$;
(S?γ)	—"— & —"— & $\mu\cdot\mathbf{o}\cdot\langle?x,\gamma\rangle\in\Sigma$	$\vdash [\mu\cdot\mathbf{o}]!y\text{—}?x\rightarrow\gamma$;
(G!γ)	$\mu\cdot\mathbf{o}\in\boldsymbol{G}(\Sigma)\cap\boldsymbol{S}(\Sigma)$	$\vdash [\mu]\text{—}\tau\rightarrow[\mu\cdot\mathbf{o}]\gamma$ $[\mu\cdot\mathbf{o}]\gamma\text{—}!y\rightarrow\gamma$;
(G?T)	—"— & $\mu\cdot\mathbf{o}\cdot\langle?x\rangle\in\boldsymbol{T}(\Sigma)$	$\vdash [\mu\cdot\mathbf{o}]\gamma\text{—}?x\rightarrow[\mu\cdot\mathbf{o}\cdot\langle?x\rangle]$;
(G?γ)	—"— & $\mu\cdot\mathbf{o}\cdot\langle?x,\gamma\rangle\in\Sigma$	$\vdash [\mu\cdot\mathbf{o}]\gamma\text{—}?x\rightarrow\gamma$;
(D)	$\mu\cdot\mathbf{o}\in\boldsymbol{D}(\Sigma)\cap\boldsymbol{S}(\Sigma)$	$\vdash [\mu]\text{—}\tau\rightarrow[\mu\cdot\mathbf{o}]\delta$;
(D?T)	—"— & $\mu\cdot\mathbf{o}\cdot\langle?x\rangle\in\boldsymbol{T}(\Sigma)$	$\vdash [\mu\cdot\mathbf{o}]\delta\text{—}?x\rightarrow[\mu\cdot\mathbf{o}\cdot\langle?x\rangle]$.
(D?γ)	—"— & $\mu\cdot\mathbf{o}\cdot\langle?x,\gamma\rangle\in\Sigma$	$\vdash [\mu\cdot\mathbf{o}]\delta\text{—}?x\rightarrow\gamma$.

□

Преобразование LTS-модели.

Определение 118: Пусть $\mathtt{C}\subseteq\mathtt{Z}$ и $\mathbf{S}\in\boldsymbol{LTS}_{\beta\gamma\delta}(\mathtt{C})$. Обозначим:

$\mu\in\Sigma$: $[\mu]$ $=_{\text{def}}$ $\mathbf{S}$ ***after*** μ.

$\mathtt{M}\subseteq\Sigma$: $[\mathtt{M}]$ $=_{\text{def}}$ $\mathbf{S}$ ***after*** $\mathtt{M}=\{\mathbf{S}$ ***after*** $\mu\,|\,\mu\in\mathtt{M}\}=\{[\mu]\,|\,\mu\in\mathtt{M}\}$.

Множество состояний будем называть *полным*, если для каждого стимула в нём есть состояние, в котором определён переход по этому стимулу, а также есть состояние, в котором определён переход по какой-нибудь реакции.

□

Мы определим преобразование на классе всех LTS-спецификаций $\mathcal{T}:\boldsymbol{LTS}_{\beta\gamma\delta}\rightarrow\boldsymbol{LTS}_{\beta\gamma\delta}$, а потом покажем, что $\mathcal{T}(\mathfrak{C})=\mathfrak{C}$ для класса *S*-ветвящихся LTS-спецификаций $\mathfrak{C}=\boldsymbol{SBLTS}_{\beta\gamma\delta}$. Тем самым, при сужении преобразования

на домен $\mathfrak{C}$ будет выполнено требование общей теории монотонности $\mathcal{T}: \mathfrak{C} \to \mathfrak{C}$.

Определение 119: Определим преобразование $\mathcal{T}_{\beta\gamma\delta}: \boldsymbol{LTS}_{\beta\gamma\delta} \to \boldsymbol{LTS}_{\beta\gamma\delta}$.
Для $C \subseteq Z$, $\mathbf{S}=\mathtt{LTS}(V_S, C_\gamma, E_S, s_0)$ и $\Sigma = \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$ $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})=\mathtt{LTS}(V, C_\gamma, E, t_0)$, где $V = [\boldsymbol{T}(\Sigma)] \cup ([\boldsymbol{S}(\Sigma)] \times !C_{\gamma\delta}) \cup \{\gamma\}$, $t_0 = [\epsilon]$, если $\mathbf{S}$ ***safe***, или $t_0=\gamma$ в противном случае, а E – наименьшее множество, порождаемое следующими правилами вывода:

$\forall \mu \in \boldsymbol{T}(\Sigma)\ \forall !y \in C\ \forall ?x \in C\ \forall \mathbf{o} \in \beta\delta(C)^*$

(γ)		$\vdash \gamma \text{—} \gamma \to \gamma$;
(**T**!**T**)	$\mu \cdot \langle !y \rangle \in \boldsymbol{T}(\Sigma)$	$\vdash [\mu] \text{—} !y \to [\mu \cdot \langle !y \rangle]$;
(**T**!γ)	$\mu \in \boldsymbol{G}(\Sigma) \cap \boldsymbol{T}(\Sigma)$	$\vdash [\mu] \text{—} !y \to \gamma$;
(**T**?**T**)	$\mu \cdot \langle ?x \rangle \in \boldsymbol{T}(\Sigma)$	$\vdash [\mu] \text{—} ?x \to [\mu \cdot \langle ?x \rangle]$;
(**T**?γ)	$\mu \cdot \langle ?x, \gamma \rangle \in \Sigma$	$\vdash [\mu] \text{—} ?x \to \gamma$;
(**S**!**T**)	$\mu \cdot \mathbf{o} \in \boldsymbol{S}(\Sigma)$ & $\mu \cdot \mathbf{o} \cdot \langle !y \rangle \in \boldsymbol{T}(\Sigma)$	$\vdash [\mu] \text{—} \tau \to [\mu \cdot \mathbf{o}] !y$ $[\mu \cdot \mathbf{o}] !y \text{—} !y \to [\mu \cdot \mathbf{o} \cdot \langle !y \rangle]$;
(**S**?**T**)	—"— & —"— & $\mu \cdot \mathbf{o} \cdot \langle ?x \rangle \in \boldsymbol{T}(\Sigma)$	$\vdash [\mu \cdot \mathbf{o}] !y \text{—} ?x \to [\mu \cdot \mathbf{o} \cdot \langle ?x \rangle]$;
(**S**?γ)	—"— & —"— & $\mu \cdot \mathbf{o} \cdot \langle ?x, \gamma \rangle \in \Sigma$	$\vdash [\mu \cdot \mathbf{o}] !y \text{—} ?x \to \gamma$;
(**G**!γ)	$\mu \cdot \mathbf{o} \in \boldsymbol{G}(\Sigma) \cap \boldsymbol{S}(\Sigma)$	$\vdash [\mu] \text{—} \tau \to [\mu \cdot \mathbf{o}] \gamma$ $[\mu \cdot \mathbf{o}] \gamma \text{—} !y \to \gamma$;
(**G**?**T**)	—"— & $\mu \cdot \mathbf{o} \cdot \langle ?x \rangle \in \boldsymbol{T}(\Sigma)$	$\vdash [\mu \cdot \mathbf{o}] \gamma \text{—} ?x \to [\mu \cdot \mathbf{o} \cdot \langle ?x \rangle]$;
(**G**?γ)	—"— & $\mu \cdot \mathbf{o} \cdot \langle ?x, \gamma \rangle \in \Sigma$	$\vdash [\mu \cdot \mathbf{o}] \gamma \text{—} ?x \to \gamma$;
(**D**)	$\mu \cdot \mathbf{o} \in \boldsymbol{D}(\Sigma) \cap \boldsymbol{S}(\Sigma)$	$\vdash [\mu] \text{—} \tau \to [\mu \cdot \mathbf{o}] \delta$;
(**D**?**T**)	—"— & $\mu \cdot \mathbf{o} \cdot \langle ?x \rangle \in \boldsymbol{T}(\Sigma)$	$\vdash [\mu \cdot \mathbf{o}] \delta \text{—} ?x \to [\mu \cdot \mathbf{o} \cdot \langle ?x \rangle]$,
(**D**?γ)	—"— & $\mu \cdot \mathbf{o} \cdot \langle ?x, \gamma \rangle \in \Sigma$	$\vdash [\mu \cdot \mathbf{o}] \delta \text{—} ?x \to \gamma$.

□

Вспомогательные утверждения.

Утверждение 66: Пусть $C \subseteq Z$, $\mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$ и $\Sigma = \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$. Тогда в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$

1) для любой тау-трассы $\mu \in \boldsymbol{T}(\Sigma)$ множество состояний $[\mu]$ τ-замкнуто в LTS $\mathbf{S}$ (вместе с пресостоянием τ-перехода содержит и постсостояние этого перехода);
2) для любой тау-трассы $\mu \in \boldsymbol{T}(\Sigma)$ power-состояние $[\mu]$ нестабильно;

3) для любой стабильной трассы $\mu\in S(\Sigma)$ power-состояние вида $[\mu]t$, где $t\in !C_{\gamma\delta}$, стабильно.

□432

Утверждение 67: Каждое power-состояние LTS $\mathcal{T}_{\beta\gamma\delta}(S)$ вида $[\mu]$ или $[\mu]t$, где $t\in !C_{\gamma\delta}$, достижимо в LTS $\mathcal{T}_{\beta\gamma\delta}(S)$ по βγδ-трассе μ, и эта βγδ-трасса безопасна в S, то есть $\mu\in safe_{\beta\gamma\delta}(S)$.

□432

3.5.3. Конформность преобразования (Монотонность 6:)

Общая теория монотонности требует конформности преобразования на выбранном классе LTS. Мы покажем конформность преобразования $\mathcal{T}_{\beta\gamma\delta}$ на классе всех LTS-спецификаций, тем самым, на любом его подклассе.

Утверждение 68: Пусть $C\subseteq Z$, $S\in LTS_{\beta\gamma\delta}(C)$ и в LTS $\mathcal{T}_{\beta\gamma\delta}(S)$ маршрут P имеет βγδ-трассу $\sigma\in traces_{\beta\gamma\delta}(P)$. Тогда:

- если P заканчивается в γ-состоянии, то в исходной LTS S найдётся мажорирующая βγδ-трасса $\lambda\in traces_{\beta\gamma\delta}(S)$ и $\sigma\preccurlyeq\lambda$;
- если P заканчивается в power-состоянии $[\mu]$ или $[\mu]t$, где $t\in !C_{\gamma\delta}$, то $\sigma\in traces_{\beta\gamma\delta}(S)$ и $[\mu]\subseteq[\sigma]$.

□433

Утверждение 69: **Условие Монотонность 6: выполнено:**
Преобразование $\mathcal{T}_{\beta\gamma\delta}$ конформно на классе всех LTS-спецификаций:
$\forall C\subseteq Z\ \forall S\in LTS_{\beta\gamma\delta}(C)\ \mathcal{T}_{\beta\gamma\delta}(S)\ ioco_{\beta\gamma\delta}\ S$.

□435

3.5.4. *S*-ветвящиеся и локально-конечно-ветвящиеся LTS.

$\mathfrak{C}=SBLTS_{\beta\gamma\delta}$	В качестве класса $\mathfrak{C}$ общей теории монотонности выберем класс *S*-ветвящихся LTS. В следующем разделе мы докажем мажорантность преобразования $\mathcal{T}_{\beta\gamma\delta}(S)$ на этом класе.

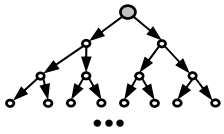
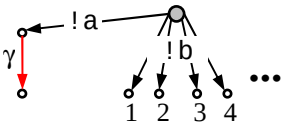
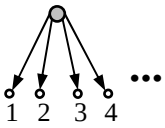
Выбор класса *S*-ветвящихся LTS определяется, прежде всего, потребностями алгоритмизации. Как мы установили выше (2.3.7), *S*-ветвление необходимо для алгоритмической генерации тестов. Ниже мы покажем, что для *S*-ветвящихся LTS можно алгоритмически выполнять преобразование $\mathcal{T}_{\beta\gamma\delta}$ и последующую композицию, получая в результате также *S*-ветвящиеся LTS. Преобразование, сохраняющее монотонность для класса всех LTS, имеет академический интерес, но, как мы увидим ниже, оно требует (для применения достаточных условий общей теории монотонности) другого мажорирования ϕ-трасс.

В этом разделе мы докажем ряд вспомогательных утверждений о *S*-ветвящихся и локально-конечно-ветвящихся LTS, а также одно вспомогательное, верное для любых LTS.

Утверждение 70: Пусть задан алфавит $C \subseteq Z$ и *S*-ветвящаяся LTS $\mathbf{S} \in \boldsymbol{SBLTS}_{\beta\gamma\delta}(C)$. Тогда множество состояний после любой конечной безопасной βγδ-трассы конечно: $\forall \sigma \in \boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})\ \mathbf{S}\ \boldsymbol{after}\ \sigma$ конечно.

□436

Примеры на Рис.91 показывают, что каждое из условий в Утверждение 70: необходимо (это условие нельзя опустить): конечность и безопасность βγδ-трассы, *S*-ветвимость LTS. Здесь в заголовке столбца указано нарушаемое условие.

<table>
<tr><th colspan="3">примеры необходимости условий в Утверждение 70:</th></tr>
<tr><td>1. βγδ-трасса конечная
  </td><td>2. βγδ-трасса безопасная
 
</td><td>3. LTS S-ветвящаяся
 
</td></tr>
<tr><td>все переходы по символу z.
бесконечная βγδ-трасса ⟨z,z,...⟩</td><td>βγδ-трасса ⟨!b⟩</td><td>все переходы по символу z.
βγδ-трасса ⟨z⟩</td></tr>
<tr><td colspan="3">Рис.91.</td></tr>
</table>

Утверждение 71: Пусть в алфавите $C \subseteq Z$ заданы *S*-ветвящаяся LTS $\mathbf{S}$, непустое множество трасс βδ-отказов $\mathbf{O}$, замкнутое по конкатенации, и безопасная βγδ-трасса $\sigma \in \boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})$ такая, что $\forall \mathbf{o}_i \in \mathbf{O}\ \sigma \cdot \mathbf{o}_i \in \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$. Тогда $\exists \mathbf{o} \in \mathbf{O}\ \mathbf{S}\ \boldsymbol{after}\ \sigma \cdot \mathbf{o} = \cap(\mathbf{S}\ \boldsymbol{after}\ \{\sigma\} \cdot \mathbf{O}) \neq \varnothing$.[65]

□436

Множество $\mathbf{O}$ необходимо должно быть непусто, так как $\cap(\mathbf{S}\ \boldsymbol{after}\ \{\sigma\} \cdot \varnothing) = \varnothing$. Необходимость остальных условий (эти условия нельзя опускать) Утверждение 71: показана на Рис.92. Здесь в заголовке столбца указано нарушаемое условие.

[65] $\cap(\mathbf{S}\ \boldsymbol{after}\ \{\sigma\} \cdot \mathbf{O}) = \cap(\{\mathbf{S}\ \boldsymbol{after}\ \sigma \cdot \mathbf{o}_i \mid \mathbf{o}_i \in \mathbf{O}\})$.

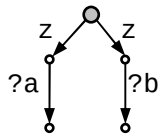
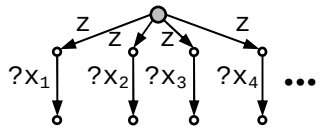
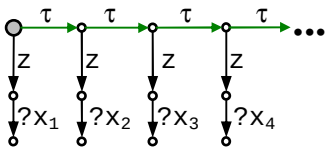
примеры необходимости условий в Утверждение 71:		
1. замкнутость по конкатенации $O=\{\{?b\},\{?a\}\}$	2. *S*-ветвимость $z\in C_\tau$, $O=\{\{?x_i\}\mid i=1,2,...\}^*$	3. βγδ-трасса безопасная $O=\{\{?x_i\}\mid i=1,2,...\}^*$
		

Рис.92.

Утверждение 72: Пусть заданы базовый алфавит $C\subseteq Z$, спецификация $S\in LTS_{\beta\gamma\delta}(C)$ и конформная реализация $I\in\mathfrak{I}(S)$. Пусть также для трасс βδ-отказов $o_1,o_2\in\beta\delta(C)^*$ имеются βγδ-трассы $\mu\cdot o_1,\mu\cdot o_2\in safe_{\beta\gamma\delta}(S)$ и $\mu\cdot o_1\cdot o_2\in traces_{\beta\gamma\delta}(I)$. Тогда $\mu\cdot o_1\cdot o_2\in safe_{\beta\gamma\delta}(S)$.

□437

3.5.5. Мажорантность преобразования (Монотонность 7:)

Структура раздела:
- Вспомогательные утверждения о свойствах преобразования.
- Основное утверждение о свойствах преобразования.
- Конечно-мажорантность.
- Мажорантность преобразования (Монотонность 7:).

Вспомогательные утверждения о свойствах преобразования.

Утверждение 73: Пусть $C\subseteq Z$ и $S\in LTS_{\beta\gamma\delta}(C)$.
1. $\forall S\in LTS_{\beta\gamma\delta}$ преобразованная LTS $\mathcal{T}_{\beta\gamma\delta}(S)$ γ-однородна;
2. Если LTS S *S*-ветвящаяся, то для каждого power-состояния вида $[\mu]$ или $[\mu]t$, множество состояний $[\mu]$ конечно.
3. Преобразованная LTS сохраняет *S*-ветвимость, если число реакций в алфавите конечно: $\forall S\in SBLTS_{\beta\gamma\delta}$ $\mathcal{T}_{\beta\gamma\delta}(S)\in SBLTS_{\beta\gamma\delta}$.

□438

Если алфавит реакций бесконечен, то из power-состояния v может выходить бесконечное число переходов по разным реакциям. А тогда при сингуляризации оно «расщепляется» на бесконечное число состояний - по одному на каждую реакцию из $init(v)$. Можно заметить, что для конечности сингуляризации достаточно было бы потребовать не конечности алфавита реакций, а конечности $init(v)$. Но тогда нужно было бы иначе определить алгоритмическое задание LTS: вместо итератора реакций и итератора переходов по каждой реакции нужен итератор переходов по всем реакциям.

Сохранение *S*-ветвимости гарантирует, что сужение преобразования $\mathcal{T}_{\beta\gamma\delta}$ на поддомен $\mathfrak{C}=\boldsymbol{SBLTS}_{\beta\gamma\delta}$ удовлетворяет требованию общей теории монотонности $\mathcal{T}\colon \mathfrak{C}\to\mathfrak{C}$.

Утверждение 74: Для каждой *S*-ветвящейся LTS **S** с конечным числом реакций преобразованная LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ локально-конечно-ветвящаяся и τ-ограниченная.

□438

Свойство локально-конечно-ветвимости преобразованной LTS нам потребуется для доказательства мажорантности преобразования. Свойство τ-ограниченности нам потребуется для алгоритмизации преобразования.

Основное утверждение о свойствах преобразования.

Сейчас мы докажем утверждение, описывающее основные свойства преобразования $\mathcal{T}_{\beta\gamma\delta}$, из которых будет следовать его мажорантность.

Пусть $C\subseteq Z$ и $\mathbf{S}\in\boldsymbol{SBLTS}_{\beta\gamma\delta}(C)$. Если **μ** является нормальной ϕ-трассой некоторой конформной реализации $\mathbf{I}\in\mathfrak{I}(\mathbf{S})$, то существует такой маршрут P LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ и такая ϕ-трасса **σ** этого маршрута, которая мажорирует ϕ-трассу **μ**. Если маршрут заканчивается в power-состоянии $\omega(P)=[\mu]$ или $\omega(P)=[\mu]t$, где $t\in !C_{\gamma\delta}$, то имеет место α-мажорирование ϕ-трасс, βγδ-трасса μ порождается ϕ-трассами **μ** и **σ**. Если **μ** заканчивается на ϕ-символ o, то для каждой реакции !y, которой **μ** может продолжаться в конформной реализации (то есть, !y не принадлежит ***ref***-множеству ϕ-символа o), среди таких маршрутов найдётся такой маршрут P, что либо во множестве [μ] реакция !y опасна, либо реакция безопасна, а маршрут заканчивается в power-состоянии $\omega(P)=[\mu]!y$. Если **μ** пуста или заканчивается на стимул или реакцию, то $\omega(P)=[\mu]$.

Запишем это утверждение формально, помечая для удобства термы логического выражения:

Утверждение 75: Пусть $C\subseteq Z$ и $\mathbf{S}\in\boldsymbol{SBLTS}_{\beta\gamma\delta}(C)$.
$\forall\mathbf{I}\in\mathfrak{I}(\mathbf{S})\ \forall\boldsymbol{\mu}\in\boldsymbol{traces}_{\phi}(\mathbf{I})$, **μ** нормальна, $\forall o\in\Phi(C)\ \forall !y\in C$
$\exists P\ \exists\boldsymbol{\sigma}\ \exists\mu\ \exists t\in !C_{\gamma\delta}$

1: $P\in\boldsymbol{runs}_{\circ}\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$

2: & $\boldsymbol{\sigma}\in\boldsymbol{traces}_{\phi}(P)$

3: & $\boldsymbol{\mu}\preccurlyeq\boldsymbol{\sigma}$

$$
\begin{array}{ll}
4: & \&\ (\quad (\omega(P)=[\mu]\ \vee\ \omega(P)=[\mu]t) \\
5: & \quad\ \Rightarrow\ \ \boldsymbol{\mu}\preccurlyeq^{\alpha}\boldsymbol{\sigma} \\
6: & \quad\ \&\ \mu\in\xi(\boldsymbol{\mu})\cap\xi(\boldsymbol{\sigma}) \\
7: & \quad\ \&\ \mu\in \boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S}) \\
8: & \quad\ \&\ (\quad \boldsymbol{\mu}\neq\epsilon\ \&\ \boldsymbol{\mu}(|\boldsymbol{\mu}|)=o\ \&\ !y\notin o_r\ \&\ \mu\cdot\langle !y\rangle\in \boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S}) \\
 & \qquad\ \Rightarrow\ \omega(P)=[\mu]!y\ ) \\
9: & \quad\ \&\ (\ (\boldsymbol{\mu}=\epsilon\ \vee\ \boldsymbol{\mu}(|\boldsymbol{\mu}|)\in C)\ \Rightarrow\ \omega(P)=[\mu]\ ) \\
 &).
\end{array}
$$

□439

Конечно-мажорантность.

Определение 120: Будем говорить, что LTS-спецификация *конечно-мажорантна*, если множество её конечных ϕ-трасс мажорирует множество конечных ϕ-трасс любой конформной реализации. Для подкласса $\mathfrak{C}\subseteq \boldsymbol{LTS}_{\beta\gamma\delta}$ будем говорить, что преобразование $\mathcal{T}:\mathfrak{C}\to\mathfrak{C}$ *конечно-мажорантно* (на классе $\mathfrak{C}$), если для каждой LTS-спецификации $\mathbf{S}\in\mathfrak{C}$ преобразованная спецификация $\mathcal{T}(\mathbf{S})$ конечно-мажорантна: $\forall \mathbf{S}\in\mathfrak{C}\ \ \cup_{\circ}\boldsymbol{traces}_{\phi\circ}\mathfrak{I}(\mathbf{S})\ \preccurlyeq\ \boldsymbol{traces}_{\phi\circ}\mathcal{T}(\mathbf{S})$.

□

Утверждение 76: Преобразование $\mathcal{T}_{\beta\gamma\delta}$ конечно-мажорантно на классе *S*-ветвящихся LTS-спецификаций $\boldsymbol{SBLTS}_{\beta\gamma\delta}$.

□451

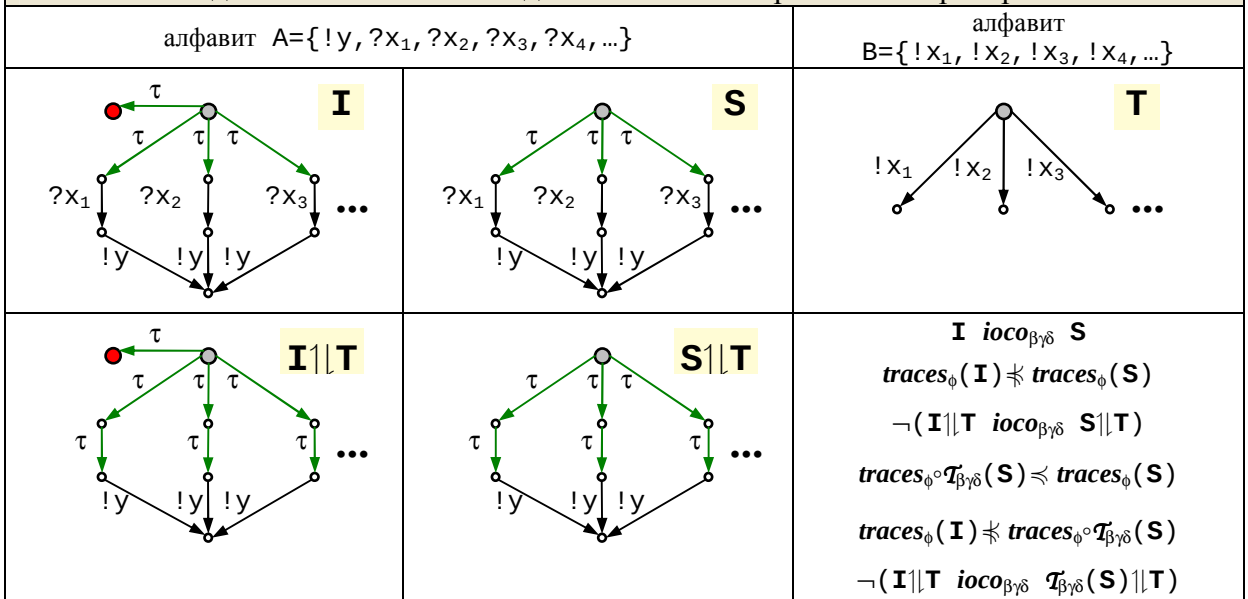
Требование *S*-ветвимости необходимо (его нельзя опустить) для конечно-мажорантности преобразования, что показывается примером на Рис.93. Здесь в конформной реализации $\mathbf{I}$ после пустой трассы есть (красное) состояние с ϕ-символом $(C,\varnothing)$, а в спецификации $\mathbf{S}$ такого состояния нет. Любая конечная последовательность отказов, порождаемая ϕ-символом $(C,\varnothing)$, порождается также одним из ϕ-символов спецификации $(C\backslash\{!y,?x_i\},\varnothing)$, однако для любого i нет мажорирования $(C,\varnothing)\not\preccurlyeq(C\backslash\{!y,?x_i\},\varnothing)$. Тем самым, ϕ-трасса конформной реализации $\langle(C,\varnothing)\rangle$ не мажорируется никакой ϕ-трассой спецификации. Заметим, что преобразованная спецификация $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ отличается от исходной только добавлением переходов по всем стимулам из начального нестабильного состояния, что не меняет вышеизложенного. Заметим, что преобразованная спецификация $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ отличается от исходной добавлением после пустой трассы состояний, но из каждого добавленного состояния есть переходы по

стимулам. Поэтому вышеизложенное справедливо и для преобразованной спецификации.

Из-за отсутствия мажорирования возникает нарушение левомонотонности (несохранение соответствия при асинхронном тестировании) даже после $\mathcal{T}_{\beta\gamma\delta}$-преобразования спецификации (не среды). Например, для среды $\mathbf{T}$ композиция $\mathbf{I}\|_{\downarrow}\mathbf{T}$ не конформна композиции $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})\|_{\downarrow}\mathbf{T}$: в композиции $\mathbf{I}\|_{\downarrow}\mathbf{T}$ имеется «лишняя» стационарность: $\langle\delta\rangle\in tt(\Sigma\grave{}\,)\cap\mathbf{I}\grave{}$, но $\langle\delta\rangle\notin\Sigma\grave{}$, где $\mathbf{I}\grave{}=traces_{\beta\gamma\delta}(\mathbf{I}\|_{\downarrow}\mathbf{T})$ и $\Sigma\grave{}=traces_{\beta\gamma\delta}(\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})\|_{\downarrow}\mathbf{T})$.

В этом примере LTS $\mathbf{T}$ существенно несингулярна, чего не бывает после преобразования $\mathcal{T}_{\beta\gamma\delta}$. Мы не знаем примера нарушения монотонности, когда, в отличие от левомонотонности, оба операнда подвергаются $\mathcal{T}_{\beta\gamma\delta}$-преобразованию. Есть гипотеза, что это преобразование монотонно на классе всех LTS-спецификаций, а не только *S*-ветвящихся. Пример показывает, что, если это так, то требуется другое, более сложное, мажорирование ϕ-трасс, если мы хотим применять условия монотонности общей теории. Возможно также, что монотонность придётся доказывать, не используя этих условий, поскольку они всего лишь достаточны. Однако, имея в виду цель алгоритмизации преобразования, композиции и генерации тестов, нам достаточно рассматривать класс *S*-ветвящихся LTS.

Необходимость *S*-ветвимости для конечно-мажорантности преобразования



$\mathbf{I}\ ioco_{\beta\gamma\delta}\ \mathbf{S}$

$traces_{\phi}(\mathbf{I})\not\preccurlyeq traces_{\phi}(\mathbf{S})$

$\neg(\mathbf{I}\|_{\downarrow}\mathbf{T}\ ioco_{\beta\gamma\delta}\ \mathbf{S}\|_{\downarrow}\mathbf{T})$

$traces_{\phi}\circ\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})\preccurlyeq traces_{\phi}(\mathbf{S})$

$traces_{\phi}(\mathbf{I})\not\preccurlyeq traces_{\phi}\circ\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$

$\neg(\mathbf{I}\|_{\downarrow}\mathbf{T}\ ioco_{\beta\gamma\delta}\ \mathcal{T}_{\beta\gamma\delta}(\mathbf{S})\|_{\downarrow}\mathbf{T})$

Рис.93.

Мажорантность преобразования (Монотонность 7:).

Преобразование $\tau_{\beta\gamma\delta}$ не мажорантно на классе всех LTS-спецификаций, поскольку оно даже не конечно-мажорантно на этом классе. Тем не менее, на подклассе *S*-ветвящихся LTS-спецификаций преобразование конечно-мажорантно. Кроме того, мы показали, что преобразование $\tau_{\beta\gamma\delta}$ преобразует *S*-ветвящуюся LTS-спецификацию в локально-конечно-ветвящуюся LTS-спецификацию. Нам надо доказать мажорантность $\tau_{\beta\gamma\delta}$ на подклассе *S*-ветвящихся LTS-спецификаций. Вместо этого мы докажем более общее утверждение: любая конечно-мажорантная локально-конечно-ветвящаяся LTS-спецификация, (в частности, S-ветвящаяся LTS-спецификаций после преобразования $\tau_{\beta\gamma\delta}$), мажорантна.

Утверждение 77: Если локально-конечно-ветвящаяся спецификация конечно-мажорантна, то она мажорантна.

□452

Нарушение локально-конечно-ветвимости может привести (но не обязательно приводит) к тому, что конечно-мажорантная спецификация не будет мажорантной. В примере на Рис.94, конформная реализация **K** имеет бесконечную ϕ-трассу $\{!y\}^{\infty}$, а спецификация **P** не содержит эту бесконечную ϕ-трассу, но содержит все её конечные префиксы $\{!y\}^{*}$. Эта спецификация конечно-мажорантна, но не мажорантна. Нарушение мажорантности приводит к несохранению соответствия при композиции. Композиция спецификаций $\mathbf{P}\|_{\downarrow}\mathbf{L}$ безопасна, а композиция конформных реализаций $\mathbf{K}\|_{\downarrow}\mathbf{L}$ опасна, так как её начальное состояние дивергентно. Тем самым, не выполнена гипотеза о безопасности. Правда, спецификация **P** не является $\tau_{\beta\gamma\delta}$-преобразованной, а после $\tau_{\beta\gamma\delta}$-преобразования в ней как раз появляется бесконечная ϕ-трасса $\{!y\}^{\infty}$. Поэтому вопрос о монотонности преобразования $\tau_{\beta\gamma\delta}$ на классе всех LTS-спецификаций остаётся открытым.

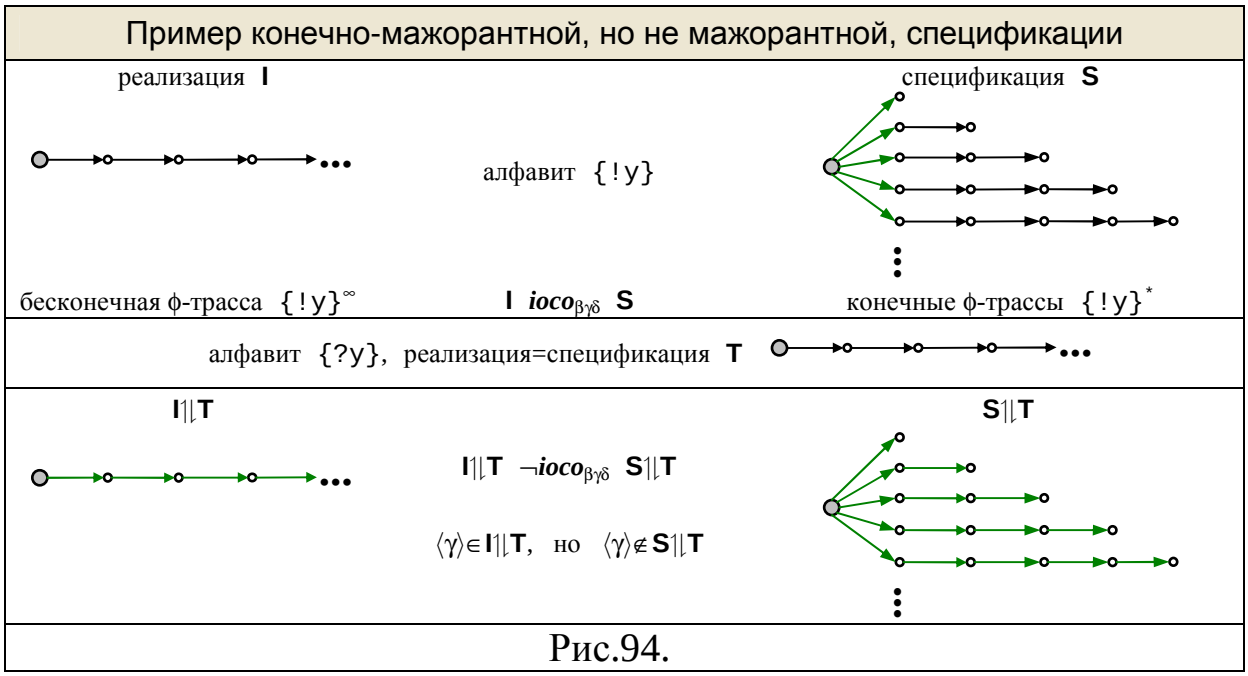


Рис.94.

Утверждение 78: **Условие Монотонность 7: выполнено на классе *S*-ветвящихся LTS-спецификаций:** преобразование $\mathcal{T}_{\beta\gamma\delta}$ мажорантно на классе $\boldsymbol{SBLTS}_{\beta\gamma\delta}$ с конечным алфавитом реакций:

$$\forall \mathbf{S} \in \boldsymbol{SBLTS}_{\beta\gamma\delta} \quad \cup\circ \boldsymbol{traces}_{\phi}^{\omega} \circ \mathcal{I}(\mathbf{S}) \preccurlyeq \boldsymbol{traces}_{\phi}^{\omega} \circ \mathcal{T}_{\beta\gamma\delta}(\mathbf{S}).$$

□454

3.5.6. Монотонность преобразования

Утверждение 79: Соответствие $\boldsymbol{ioco}_{\beta\gamma\delta}$ $\mathcal{T}_{\beta\gamma\delta}$-левомонотонно и $\mathcal{T}_{\beta\gamma\delta}$-монотонно на классе *S*-ветвящихся LTS-спецификаций с конечным числом реакций.

□454

Из монотонности преобразования следует его инвариантность. Поэтому теперь мы можем доказать утверждение аналогичное Утверждение 67: о достижимости в преобразованной LTS её power-состояний по соответствующим βγδ-трассам, но безопасным не только в исходной, но и в преобразованной LTS.

Утверждение 80: Если LTS **S** S-ветвящаяся с конечным числом реакций, то каждое power-состояние LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ вида $[\mu]$ или $[\mu]t$, где $t \in !C_{\gamma\delta}$, достижимо в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ по βγδ-трассе μ, и эта βγδ-трасса безопасна в $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$, то есть $\mu \in \boldsymbol{safe}_{\beta\gamma\delta} \circ \mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$.

□455

3.5.7. Оптимизация преобразования

Для преобразования $\mathcal{T}_{\beta\gamma\delta}$ существует тривиальная оптимизация. Пусть $C \subseteq Z$, $\mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$ и $\mathbf{T} = \mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$. Напомним, что в каждом состоянии LTS $\mathbf{T}$ определено не более одного перехода по каждому стимулу или реакции/ Поэтому мы можем удалить дублирующие переходы по стимулам и реакциям из нестабильного состояния t в случае, когда переходы по этим же стимулам и реакциям и в те же постсостояния имеются в стабильных состояниях $t_1, t_2, \ldots$, в которые мы попадаем из t через τ-переходы. Получится LTS $\mathbf{T1} = \boldsymbol{opt1}(\mathbf{T}) = LTS(V_{\mathbf{T}}, C_\gamma, E_{\mathbf{T}} \setminus E, t_0)$, где E – наименьшее множество, порождаемое следующими правилами вывода:

$$\forall t, t_1, t` \in \boldsymbol{der}(\mathbf{T})\ \forall z \in C$$
$$t \text{—}\tau\!\rightarrow t_1\ \&\ t \text{—}z\!\rightarrow t`\ \&\ t_1 \text{—}z\!\rightarrow t` \vdash t \text{—}z\!\rightarrow t`.$$

Очевидно, что при такой оптимизации ***opt1*** множество ϕ-трасс не изменяется.

Следующая оптимизация заключается в удалении «лишних» нестабильных состояний из LTS $\mathbf{T1}$. Если из достижимого нестабильного состояния t выходит ровно один τ-переход в стабильное состояние t_1 и не выходят переходы по стимулам и реакциям, то состояния t и t_1 можно "склеить": удалить τ-переход $t \text{—}\tau\!\rightarrow t_1$, а все переходы в t заменить на переходы в t_1. Получится LTS $\mathbf{T2} = \boldsymbol{opt1}(\mathbf{T1}) = LTS(V_{\mathbf{T1}}, C_\gamma, (E_{\mathbf{T1}} \setminus E_1) \cup E_2, t_0)$, где E_1 – наименьшее множество, порождаемое следующими правилами вывода:

$$\forall t, t_1 \in \boldsymbol{der}(\mathbf{T1})$$
$$t \text{—}\tau\!\rightarrow t_1\ \&\ \forall t_2 \neq t_1\ t \text{—}\tau\!\not\rightarrow t_2\ \&\ \forall z \in C\ t \text{—}z\!\not\rightarrow \vdash t \text{—}\tau\!\rightarrow t_1,$$

а E_2 – наименьшее множество, порождаемое следующими правилами вывода:

$$\forall t, t_1, t` \in \boldsymbol{der}(\mathbf{T1})\ \forall u \in C$$
$$t \text{—}\tau\!\rightarrow t_1\ \&\ \forall t_2 \neq t_1\ t \text{—}\tau\!\not\rightarrow t_2\ \&\ \forall z \in C\ t \text{—}z\!\not\rightarrow$$
$$\&\ t` \text{—}u\!\rightarrow t \vdash t` \text{—}u\!\rightarrow t_1.$$

Очевидно, что при такой оптимизации ***opt2*** множество ϕ-трасс также не изменяется.

В дальнейшем, для удобства, мы будем в некоторых примерах использовать оптимизированное преобразование $\boldsymbol{opt2} \circ \boldsymbol{opt1} \circ \mathcal{T}_{\beta\gamma\delta}$, которое, допуская вольность речи, будем по-прежнему обозначать $\mathcal{T}_{\beta\gamma\delta}$. Такое преобразование тоже $\boldsymbol{ioco}_{\beta\gamma\delta}$-монотонно.

Глава 3.6. Общий случай: Алгоритмизация

Структура главы:
1. Замкнутость по алгоритмическому преобразованию $\mathcal{T}_{\beta\gamma\delta}$
2. Алгоритмизация преобразования $\mathcal{T}_{\beta\gamma\delta}$ для конечного алфавита
3. Алгоритмическое преобразование $\mathcal{T}_{\beta\gamma\delta}$ при многократной композиции
4. Алфавит и ветвление при композиции LTS
5. Проблема дивергенции
6. Слабо-регулярность и $\mathcal{T}_{\beta\gamma\delta}$-преобразование. Сильно-регулярность
7. Двойная LTS и $\mathcal{W}_{\beta\gamma\delta}$-преобразование.
8. Алгоритмизация композиции $\mathcal{T}_{\beta\gamma\delta}$-преобразованных LTS
9. Алгоритмизация композиции $\mathcal{W}_{\beta\gamma\delta}$-преобразованных LTS

В этой главе исследуются алгоритмические вопросы преобразования $\mathcal{T}_{\beta\gamma\delta}$ и композиции преобразованных LTS-спецификаций. Основная проблема связана с дивергенцией в композиции LTS, которая не наследуется из дивергенции того или иного операнда, а возникает как результат бесконечной цепочки синхронных переходов. Мы налагаем на исходные LTS-спецификации специальные ограничения, которые позволяют «обнаруживать» такую дивергенцию в композиции $\mathcal{T}_{\beta\gamma\delta}$-преобразованных LTS алгоритмическим путём. Далее предлагается модифицированное преобразование $\mathcal{W}_{\beta\gamma\delta}$, которое также является ***ioco***$_{\beta\gamma\delta}$-монотонным, но предъявляет к исходным спецификациям меньшие требования.

Мы будем считать, что LTS-спецификации алгоритмически заданы теми же способами, которые годятся для генерации тестов по этим LTS. В разделе 2.3.7 эти способы обозначены как **`L1`** и **`L2`**, и отличаются друг от друга тем, определяются ли переходы только по безопасным стимулам и реакциям (**`L1`**) или по всем стимулам и реакциям (**`L2`**). В обоих случаях конечен алфавит реакций, LTS безопасны и *S*-ветвятся (**`L1`**) или *S*-τ-ограничены (**`L2`**).

3.6.1. Замкнутость по алгоритмическому преобразованию $\mathcal{T}_{\beta\gamma\delta}$

Структура раздела:
- Алфавит.
- Ветвление и дивергенция.

Алфавит.

В процессе преобразования нам нужно для каждой безопасной βγδ-трассы μ определить, является ли полным множество $[\mu]$ состояний после трассы. Полнота, в частности, означает наличие перехода по каждому стимулу хотя

бы в одном состоянии из $[\mu]$. Очевидно, что за конечное время определить полноту множества $[\mu]$ можно только в том случае, когда множество стимулов конечно.

Кроме того, мы должны проводить τ-переходы $[\mu]\text{—}\tau\rightarrow[\mu\cdot o]!y$, $[\mu]\text{—}\tau\rightarrow[\mu\cdot o]\gamma$ или $[\mu]\text{—}\tau\rightarrow[\mu\cdot o]\delta$ для каждой трассы o, при которой βγδ-трасса $\mu\cdot o$ остаётся безопасной. При бесконечном числе стимулов число таких трасс отказов даже без повторения отказов бесконечно, но число подмножеств конечного множества $[\mu]$ конечно. Поэтому нам нужно было бы перебирать эти конечные подмножества (достаточно перебирать подмножества стабильных состояний) и сравнивать их по порождаемым ими множествам отказов. Однако такое сравнение невозможно сделать при задании LTS способом **L1** или **L2**, если число стимулов (и, следовательно, их блокировок) бесконечно. Поэтому и в этом случае нам также требуется конечность числа стимулов.[66]

Поскольку алгоритмическое задание LTS требует также конечности множества реакций, мы получаем LTS с конечным алфавитом.

Поскольку преобразование $\mathcal{T}_{\beta\gamma\delta}$ сохраняет алфавит, конечность алфавита сохраняется и в преобразованной LTS.

Ветвление и дивергенция.

Если из *S*-τ-ограниченности[67] LTS следует её *S*-ветвимость (Утверждение 47:), то, вообще говоря, из τ-ограниченности[68] не следует конечно-ветвимость. Действительно, число переходов из достижимого состояния по опасному символу может быть не конечно, а только перечислимо, если в одном из таких состояний начинается τ-маршрут с самопересечением или заканчивающийся в состоянии с γ-переходом.

---

[66] Выбор не всех, а только части, подмножеств является оптимизацией. Если эту оптимизацию не делать, не требуется конечность числа стимулов. Однако она все равно нужна для определения полноты множества состояний после βγδ-трассы.

[67] *S*-τ-ограниченность означает, что τ-ограничено начальное состояние и множество постсостояний $s`$ переходов $s\text{—}z\rightarrow s`$ для каждого *S*-достижимого состояния s и каждого стимула или реакции z (Определение 73:).

[68] τ-ограниченность означает τ-ограниченность каждого достижимого состояния (Определение 72:).

По Утверждение 74:, если число реакций конечно, то LTS $\mathcal{T}_{\beta\gamma\delta}$(**S**) локально-конечно-ветвящаяся и τ-ограниченная. Для конечного алфавита локально-конечно-ветвимость совпадает с конечно-ветвимостью. Конечная LTS, очевидно, всегда τ-ограничена.

Утверждение 81: Класс *S*-ветвящихся и подкласс *S*-τ-ограниченных LTS в конечном алфавите замкнуты по преобразованию $\mathcal{T}_{\beta\gamma\delta}$, причём результат преобразования – это конечно-ветвящиеся τ-ограниченные LTS (Рис.95). Класс конечных LTS в конечном алфавите также замкнут по преобразованию $\mathcal{T}_{\beta\gamma\delta}$.

□455

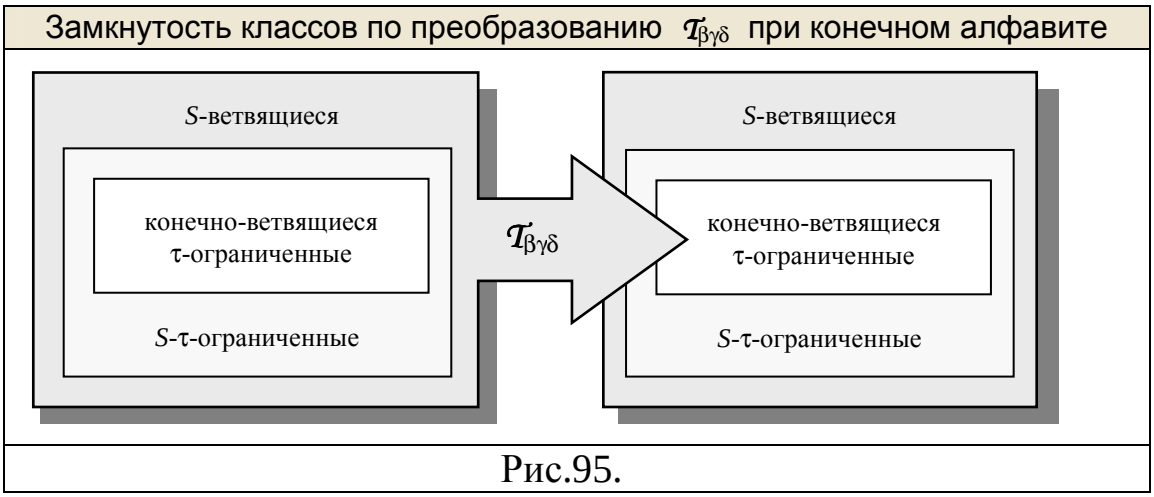


Рис.95.

3.6.2. Алгоритмизация преобразования $\mathcal{T}_{\beta\gamma\delta}$ для конечного алфавита

Структура раздела:

- Алгоритмическое преобразование $\mathcal{T}_{\beta\gamma\delta}$:L1→L1 для конечного алфавита.
- Алгоритмическое преобразование L12L2:L1→L2 для преобразованного операнда $\mathcal{T}\beta\gamma\delta(S)$.
- Алгоритмические преобразования $\mathcal{T}_{\beta\gamma\delta}$:L1→L2, $\mathcal{T}_{\beta\gamma\delta}$:L2→L1, $\mathcal{T}_{\beta\gamma\delta}$:L2→L2 для конечного алфавита.

Алгоритмическое преобразование $\mathcal{T}_{\beta\gamma\delta}$:L1→L1 для конечного алфавита.

Утверждение 82: Существует алгоритм, определённый на классе **L1**-LTS с конечным базовым алфавитом и возвращающий для каждой LTS **S** из его домена **L1**-LTS $\mathcal{T}_{\beta\gamma\delta}$(**S**).

□455

Алгоритмическое преобразование L12L2:L1→L2 для преобразованного операнда $\mathcal{T}_{\beta\gamma\delta}$(S).

Алгоритмы **L1** ничего не говорят о постсостояниях переходов по разрушающим стимулам и реакциям из *S*-достижимых состояний, а алгоритмы **L2**, напротив, такие постсостояния перечисляют. Поэтому в общем случае невозможно преобразовать алгоритмы **L1** в алгоритмы **L2**.

Однако, для LTS $\mathcal{T}_{\beta\gamma\delta}$(**S**) известно, что все переходы по стимулам и реакциям, опасным в состоянии, разрушающие в этом состоянии и ведут в стандартное состояние γ, в котором определена только γ-петля, а все остальные состояния *S*-достижимы (Утверждение 80:). Поэтому преобразование из **L1**-представления в **L2**-представление делается тривиально. Более того, алгоритмы, задающие преобразованную **L2**-LTS, можно определить во всех достижимых состояниях.

Определение 121: *Всюду-заданной* **L2**-LTS будем называть такую **L2**-LTS, в которой зелёными состояниями (состояниями, в которых заданы оракул **Gamma2** и итераторы **Extend2** и **Tau2**) являются все достижимые состояния.

□

Утверждение 83: Существует алгоритм, который для каждой **L1**-LTS вида $\mathcal{T}_{\beta\gamma\delta}$(**S**) с конечным базовым алфавитом возвращает всюду-заданную **L2**-LTS $\mathcal{T}_{\beta\gamma\delta}$(**S**).

□458

Алгоритмические преобразования $\mathcal{T}_{\beta\gamma\delta}$:L1→L2, $\mathcal{T}_{\beta\gamma\delta}$:L2→L1, $\mathcal{T}_{\beta\gamma\delta}$:L2→L2 для конечного алфавита.

Если алфавит конечен, то мы имеем алгоритмические преобразования **L2**2**L1**:**L2**→**L1** (Утверждение 48:), $\mathcal{T}_{\beta\gamma\delta}$:**L1**→**L1** и, для операнда $\mathcal{T}_{\beta\gamma\delta}$(**S**), **L1**2**L2**:**L1**→**L2**. Тем самым, мы имеем все возможные алгоритмические преобразования: $\mathcal{T}_{\beta\gamma\delta}$:**L1**→**L1**, $\mathcal{T}_{\beta\gamma\delta}$:**L1**→**L2**, $\mathcal{T}_{\beta\gamma\delta}$:**L2**→**L1**, $\mathcal{T}_{\beta\gamma\delta}$:**L2**→**L2** (Рис.96).

Алгоритмизация преобразования $\mathcal{T}_{\beta\gamma\delta}$ при конечном алфавите

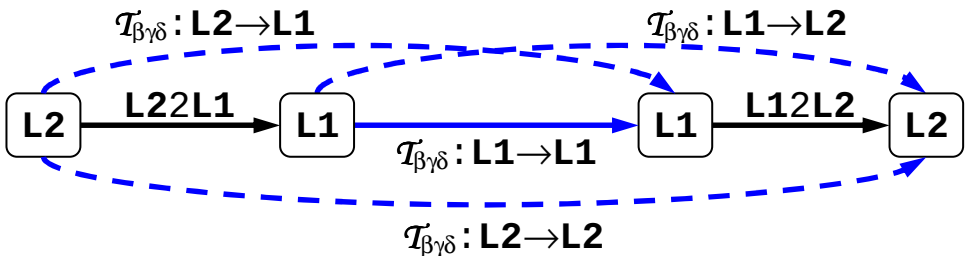


Рис.96.

3.6.3. Алгоритмическое преобразование $\mathcal{T}_{\beta\gamma\delta}$ при многократной композиции

Композиция преобразованных LTS, вообще говоря, может не быть преобразованной LTS (её нельзя получить преобразованием $\mathcal{T}_{\beta\gamma\delta}$ из какой-нибудь LTS). Достаточно рассмотреть свойство сингулярности состояний. В примере на Рис.97 (строка **1**) видно, что это свойство может нарушиться при композиции преобразованных LTS.

Поэтому, если мы хотим получить преобразованную спецификацию системы из двух компонентов, нам нужно преобразовать компоненты, скомпоновать их, а потом преобразовать полученную композицию.

Нарушение сингулярности при композиции преобразованных LTS
(используется оптимизированное преобразование $\mathcal{T}_{\beta\gamma\delta}$)

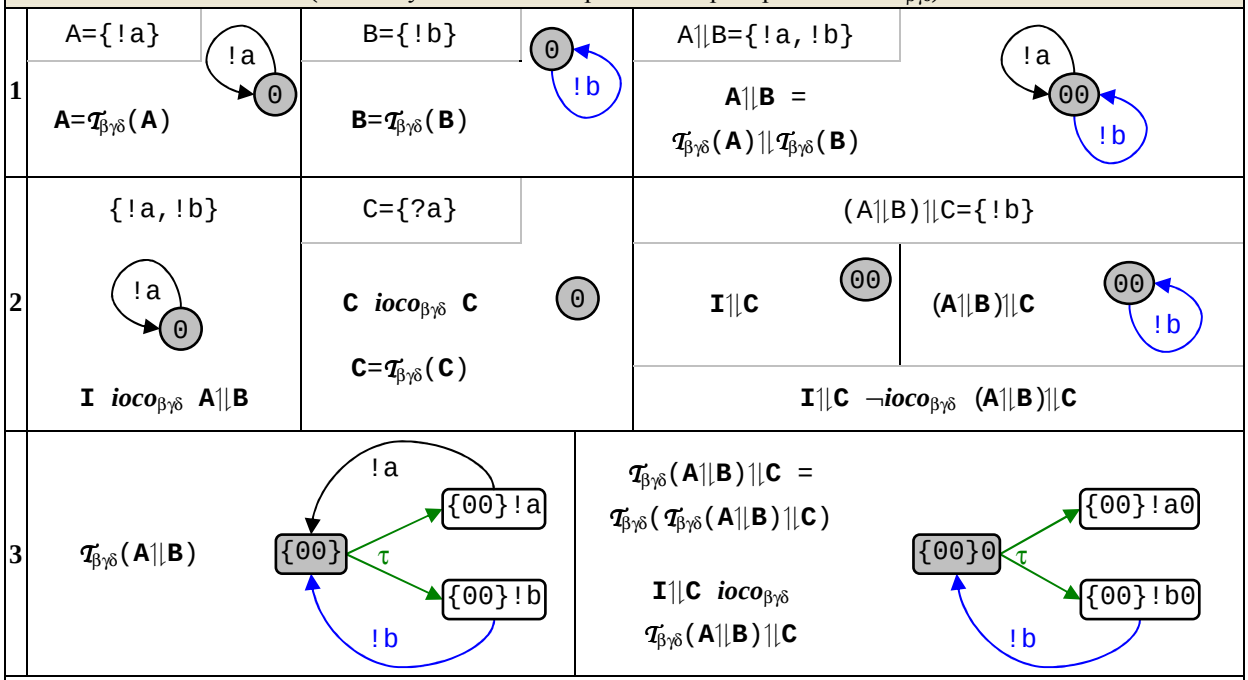


Рис.97.

Что меняется при преобразовании композиции преобразованных спецификаций, то есть чем отличаются $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S_1})\wr\mathcal{T}_{\beta\gamma\delta}(\mathbf{S_2})$ и $\mathcal{T}_{\beta\gamma\delta}(\mathcal{T}_{\beta\gamma\delta}(\mathbf{S_1})\wr\mathcal{T}_{\beta\gamma\delta}(\mathbf{S_2}))$?

Мы доказали, что ф-трассы композиции $\mathbf{S_{12}}=\mathcal{T}_{\beta\gamma\delta}(\mathbf{S_1})\wr\mathcal{T}_{\beta\gamma\delta}(\mathbf{S_2})$ мажорируют все ф-трассы всех композиций конформных реализаций, то есть ф-трассы композиций $\mathbf{I_1}\wr\mathbf{I_2}$, где $\mathbf{I_1}$ ***ioco***$_{\beta\gamma\delta}$ $\mathbf{S_1}$ и $\mathbf{I_2}$ ***ioco***$_{\beta\gamma\delta}$ $\mathbf{S_2}$. Иными словами, композиция $\mathbf{S_{12}}$ является косой композицией спецификаций $\mathbf{S_1}$ и $\mathbf{S_2}$. Отсюда следует, что нам не требуется преобразование композиции $\mathbf{S_{12}}$ для дальнейшей композиции со спецификацией $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S_3})$, если мы рассматриваем только те реализации, конформные $\mathbf{S_{12}}$, которые могут быть представлены как композиция $\mathbf{I_1}\wr\mathbf{I_2}$ реализаций, конформных спецификациям $\mathbf{S_1}$ и $\mathbf{S_2}$.

Иными словами, если схема компоновки, то есть число компонентов n и порядок их композиции, считается фиксированным, то есть реализации и спецификации компонуются по одной схеме, то для построения спецификации системы, задаваемой спецификациями компонентов $\mathbf{S_1},\mathbf{S_2},\ldots,\mathbf{S_n}$, достаточно преобразования спецификаций компонентов и далее их композиции по этой схеме компонновки. Для $\mathbf{I_1}$ ***ioco***$_{\beta\gamma\delta}$ $\mathbf{S_1}$, $\mathbf{I_2}$ ***ioco***$_{\beta\gamma\delta}$ $\mathbf{S_2}$, ..., $\mathbf{I_n}$ ***ioco***$_{\beta\gamma\delta}$ $\mathbf{S_n}$ всегда будет $\mathbf{I_1}\wr\mathbf{I_2}\wr\ldots\wr\mathbf{I_n}$ ***ioco***$_{\beta\gamma\delta}$ $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S_1})\wr\mathcal{T}_{\beta\gamma\delta}(\mathbf{S_2})\wr\ldots\wr\mathcal{T}_{\beta\gamma\delta}(\mathbf{S_n})$, где скобки, определяющие порядок композиции[69], расставлены одинаково для реализаций и спецификаций в соответствии с данной схемой компоновки.

Однако, если схема компоновки реализаций не обязательно совпадает со схемой компоновки спецификаций, могут быть и другие реализации, конформные спецификации системы. Вообще, спецификации $\mathbf{S_{12}}$ могут быть конформны композиции реализаций, не конформных спецификациям $\mathbf{S_1}$ и $\mathbf{S_2}$, или реализация может состоять не из двух, а из одного или больше, чем двух, компонентов. В этом случае дальнейшая композиция спецификаций без $\mathcal{T}_{\beta\gamma\delta}$-преобразования "промежуточных" результатов предшествующих композиций может привести к нарушению монотонности. Пример на Рис.97 (строка **2**). Поэтому для дальнейшей композиции нам требуется преобразовать композицию, то есть рассматривать спецификацию $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S_{12}})$ (на Рис.97 строка **3**).

---

[69] Скобки нужны, поскольку композиция LTS, вообще говоря, не ассоциативна.

Аналогичный пример можно привести для нарушения свойства sγ-однородности (Рис.98).

Нарушение sγ-однородности при композиции преобразованных LTS
(используется оптимизированное преобразование $\mathcal{T}_{\beta\gamma\delta}$)

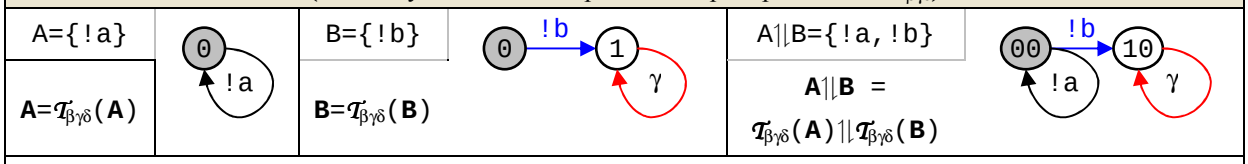


Рис.98.

В общем случае композиция спецификаций выполняется по схеме компоновки спецификаций с помощью оператора $.\|\mathcal{T}_{\beta\gamma\delta}\|. = \mathcal{T}_{\beta\gamma\delta}(.)\|\|\mathcal{T}_{\beta\gamma\delta}(.)$, то есть как $\mathbf{S_1}\|\mathcal{T}_{\beta\gamma\delta}\|\mathbf{S_2}\|\mathcal{T}_{\beta\gamma\delta}\|...\|\mathcal{T}_{\beta\gamma\delta}\|\mathbf{S_n}$.

3.6.4. Алфавит и ветвление при композиции LTS

Структура раздела:
- Алфавит.
- Ветвление.
- Локально-конечно-ветвящиеся LTS.

Алфавит.

Поскольку для алгоритмизации преобразования $\mathcal{T}_{\beta\gamma\delta}$ требуется конечность алфавита, композицию LTS мы также будем рассматривать только для LTS, имеющих конечные алфавиты. Очевидно, композиция LTS с конечными алфавитами A и B также имеет конечный алфавит $A\|\|B=A\setminus B\cup B\setminus A$.

Ветвление.

В общем случае композиция *S*-ветвящихся LTS не является *S*-ветвящейся. Пример приведён на Рис.99. Здесь LTS **A** и **B** *S*-ветвящиеся, но их композиция не является *S*-ветвящейся. В состоянии c LTS **A** было определено бесконечное (возможно, несчётное) число переходов по реакции !y, но это не нарушало *S*-ветвимости, так как состояние c не было *S*-достижимым. После композиции в состоянии ce, по-прежнему, бесконечное число переходов по реакции !y, но теперь это состояние достижимо по безопасной βγδ-трассе {?x}!y, что нарушает *S*-ветвимость композиционной LTS. На этом же примере показано, что, если компоновать $\mathcal{T}_{\beta\gamma\delta}$-преобразованные LTS, которые являются конечно-ветвящимися, то композиция будет конечно-ветвящейся и, следовательно, *S*-ветвящейся. Это оказывается верным и в общем случае композиции конечно-ветвящихся LTS.

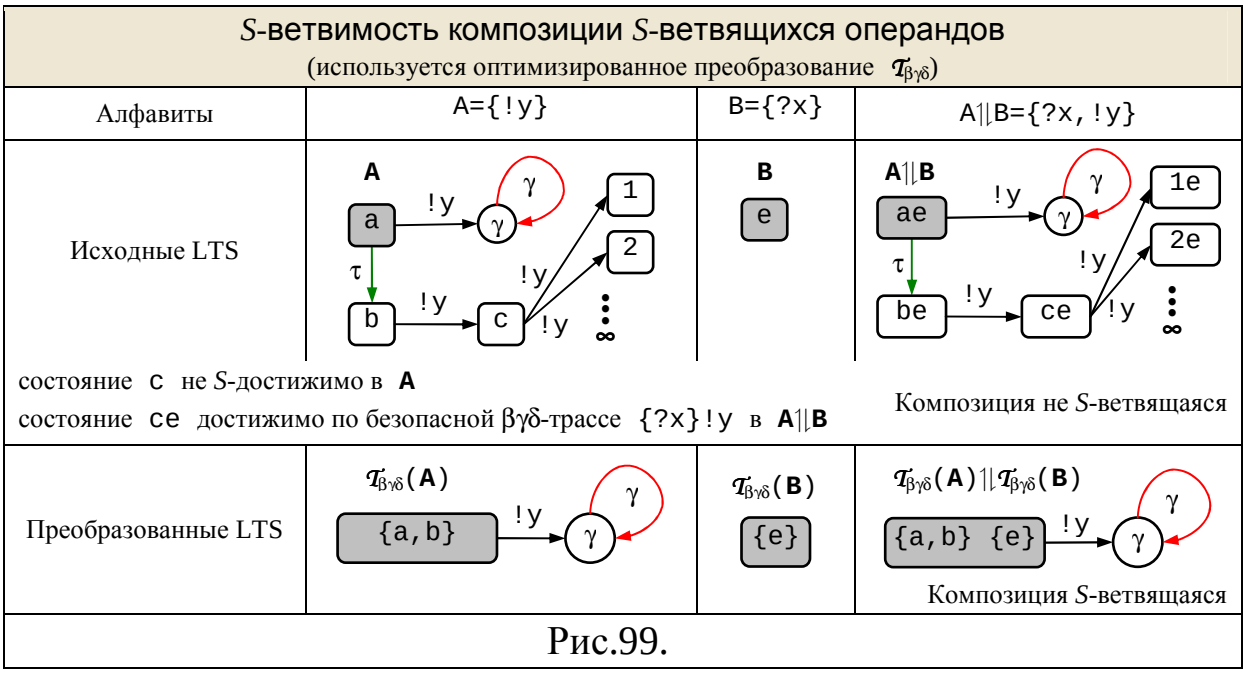


Рис.99.

Локально-конечно-ветвящиеся LTS.

Мы будем изучать более общий случай композиции LTS, в которых конечно не общее число переходов из состояния (конечно-ветвящиеся LTS), а число переходов по каждому символу (локально-конечно-ветвящиеся LTS). Локально-конечно-ветвящаяся LTS в конечном алфавите конечно-ветвится.

Утверждение 84: Класс локально-конечно-ветвящихся LTS с конечным алфавитом реакций замкнут по композиции LTS.

□459

Как следствие, класс конечно-ветвящихся LTS с конечными алфавитами замкнут по композиции LTS. Как было показано выше, для LTS в конечном алфавите $\mathcal{T}_{\beta\gamma\delta}$-преобразованная LTS конечно-ветвящаяся. Поэтому композиция $\mathcal{T}_{\beta\gamma\delta}$-преобразованных LTS также конечно-ветвящаяся.

3.6.5. Проблема дивергенции

Структура раздела:

- τ-ограниченности операндов недостаточно для *S*-τ-ограниченоести композиции.
- Префикс-регулярности операндов недостаточно для *S*-τ-ограниченоести композиции.
- Слабо-регулярность и замкнутость по композиции.
- Слабо-регулярность операндов не необходима.

Основной проблемой алгоритмизации композиции LTS является дивергенция, порождаемая бесконечной цепочкой синхронных переходов. Такую дивергенцию можно обнаружить алгоритмически, если композиционная LTS *S*-τ-ограничена. В этом разделе исследуются те ограничения, которые мы должны наложить на исходные LTS, чтобы композиция $\mathcal{T}_{\beta\gamma\delta}$-преобразованных LTS была *S*-τ-ограниченной.

τ-ограниченности операндов недостаточно для *S*-τ-ограниченеости композиции.

К сожалению, даже τ-ограниченности операндов недостаточно для *S*-τ-ограниченности композиции, что показывает пример на Рис.100.

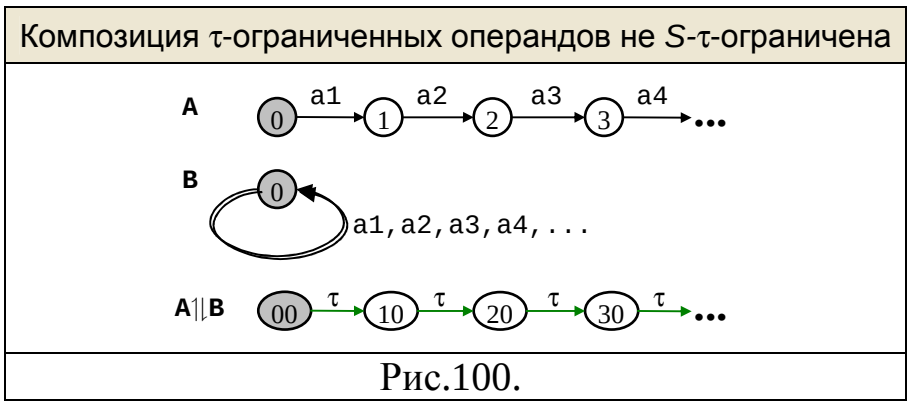


Рис.100.

Если мы не налагаем никаких ограничений на алфавиты компонуемых LTS, кроме их конечности, то для любой LTS **A**, в которой есть бесконечный маршрут, начинающийся в достижимом состоянии a, найдётся такая LTS **B** и такое её состояние b, что в композиции **A||B** состояние ab достижимо, и в нём начинается бесконечный τ-маршрут. Пример тот же самый (Рис.100), но только состояние a – это одно из состояний 0,1,2,... в LTS **A**, а состояние b – это состояние 0 в LTS **B**.

Само по себе наличие бесконечного τ-маршрута не противоречит *S*-τ-ограниченности, если хотя бы из одного состояния этого маршрута по τ-переходам можно достичь τ-цикла или γ-перехода. Сейчас мы будем искать те ограничения, которые нужно наложить на LTS-операнды, чтобы композиционная LTS была *S*-τ-ограниченной.

Нас будет интересовать композиция только таких LTS, которые либо являются результатом $\mathcal{T}_{\beta\gamma\delta}$-преобразования, либо результатом композиции, быть может, многократной, таких $\mathcal{T}_{\beta\gamma\delta}$-преобразованных LTS. Как уже показано выше, $\mathcal{T}_{\beta\gamma\delta}$-преобразованная LTS τ-ограничена (Утверждение 81:). Мы покажем, что ограничения, которые нужно дополнительно наложить на

$\tau_{\beta\gamma\delta}$-преобразованные LTS для того, чтобы их композиция была *S*-τ-ограниченной, сохраняются при композиции таких LTS и включают τ-ограниченность.

Префикс-регулярности операндов недостаточно для *S*-τ-ограниченеости композиции.

Сначала по аналогии с τ-ограниченностью потребуем, чтобы LTS-операнд содержал только такие бесконечные маршруты, трассы которых имеют конечный префикс, являющийся трассой некоторого маршрута с самопересечением (достижимый τ-цикл) или заканчивающегося в состоянии, где определён γ-переход. Иными словами, у каждой бесконечной трассы[70] либо должно быть ответвление по γ (трасса разрушающая), либо должен быть конечный префикс, который является трассой некоторого маршрута с самопересечением.

Целью обнаружения дивергенции является либо проверка безопасности LTS, либо проверка безопасности стимула или реакции в состоянии. Поэтому трассы и маршруты должны рассматриваться для начального состояния и любого множества состояний. Мы усилим это требование и будем рассматривать каждое достижимое состояние по отдельности. Учитывая, что речь идёт о самопересекающихся маршрутах, трасса которых – конечный префикс бесконечной трассы, такие LTS можно назвать префикс-регулярными.

Состояние $s \in V_s$ будем называть *префикс-регулярным*, если для каждой бесконечной неразрушающей трассы σ, начинающейся в s, существует конечный самопересекающийся маршрут, начинающийся в s и имеющий в качестве трассы префикс σ. Будем говорить, что LTS **S** *префикс-регулярна*, если префикс-регулярны все её достижимые состояния.

Итак, мы могли бы потребовать, чтобы при композиции LTS-операнды были τ-ограниченными и префикс-регулярными. Однако пример на Рис.101 показывает, что этого требования тоже недостаточно.

---

[70] Можно было бы говорить о бесконечных βγδ-трассах, имеющих бесконечное число вхождений стимулов и реакций, то есть не отказов.

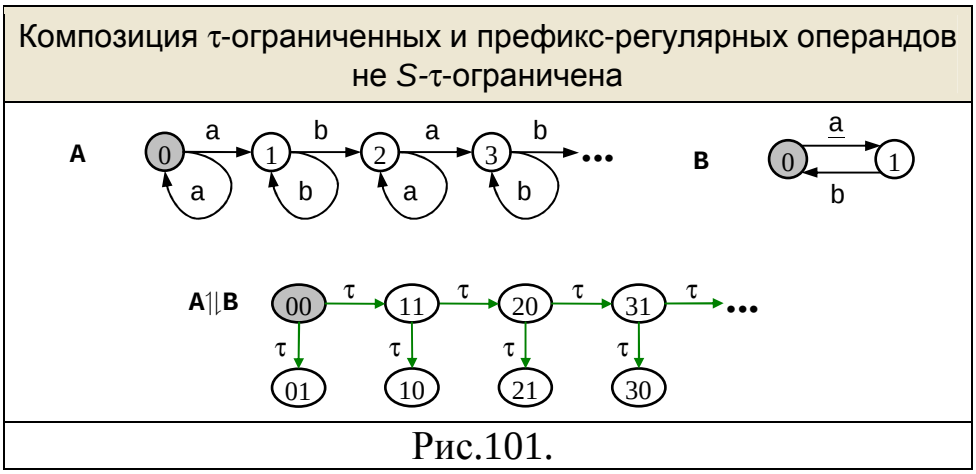


Рис.101.

Причина в том, что циклический маршрут в операнде вовсе не обязательно приводит к появлению циклического маршрута композиции, если компонуемый с ним маршрут другого операнда не циклический.

Слабо-регулярность и замкнутость по композиции.

Таким образом, нам нужно требование более сильное, чем префикс-регулярность: для бесконечной неразрушающей трассы должен найтись такой бесконечный маршрут с этой трассой, который "укладывается" в конечную под-LTS, то есть проходит через конечное число состояний. Тогда для любых двух компонуемых бесконечных трасс операндов найдутся два циклических маршрута в этих операндах, которые имеют эти трассы.

Определение 122: Маршрут R будем называть *регулярным*, если он проходит через конечное множество состояний: $|\boldsymbol{Im}\circ\boldsymbol{pre}\circ R|\neq\infty$.

□

Определение 123: Состояние s будем называть *слабо-регулярным*, если для каждой бесконечной неразрушающей трассы (трассы без γ-ответвлений) λ, начинающейся в s, существует регулярный маршрут, начинающийся в s и имеющий трассу λ.

□

Определение 124: Пусть $C\subseteq Z$ и $\mathbf{S}\in\boldsymbol{LTS}_{\beta\gamma\delta}(C)$. Будем говорить, что LTS $\mathbf{S}$ *слабо-регулярна*, если слабо-регулярно каждое её достижимое состояние.

□

Замечание: Для фиксированных конечных алфавитов $A,B\subseteq Z$ требование слабо-регулярности можно было бы соответствующим образом ослабить: нам достаточно рассматривать бесконечные трассы только в синхронном подалфавите: в алфавите $A\cap\underline{B}\subseteq A$ для LTS $\mathbf{A}\in\boldsymbol{LTS}_{\beta\gamma\delta}(A)$, и в алфавите $\underline{A}\cap B\subseteq B$ для LTS $\mathbf{B}\in\boldsymbol{LTS}_{\beta\gamma\delta}(B)$. Такие LTS можно было бы называть $A\cap\underline{B}$-слабо-регулярными и $B\cap\underline{A}$-слабо-регулярными, соответственно. Однако при

композиции алфавит меняется, и композиция **A**⫛**B**, имеющая алфавит A⫛B, не обязательно будет A⫛B∩C-слабо-регулярной или C∩A⫛B-слабо-регулярной для третьего алфавита C. Иными словами, при многократной композиции мы не будем иметь замкнутых по композиции классов, определяемых на базе такой ослабленной слабо-регулярности.

Утверждение 85: Класс локально-конечно-ветвящихся, τ-ограниченных и слабо-регулярных LTS с конечными алфавитами реакций замкнут по композиции LTS (Рис.102).

□460

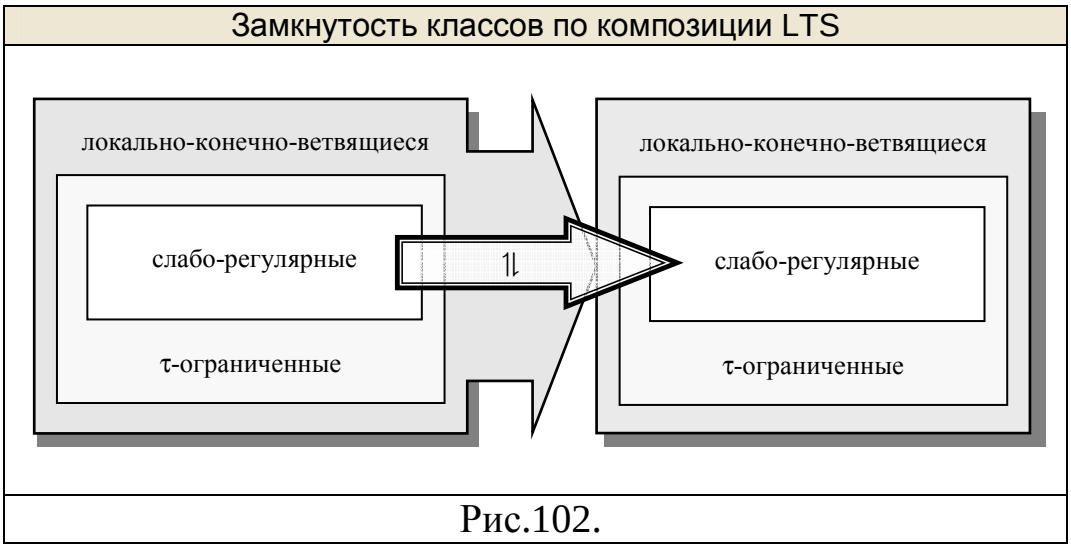


Рис.102.

Следствие: класс конечно-ветвящихся, τ-ограниченных и слабо-регулярных LTS с конечными алфавитами замкнут по композиции LTS.

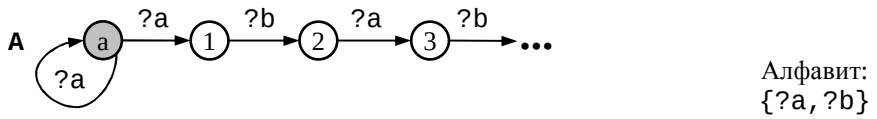
Слабо-регулярность операндов не необходима.

Можно было бы искать необходимое и достаточное условие, при выполнении которого в операндах их композиция оказывается *S*-τ-ограниченной. Под необходимостью здесь имеется в виду следующее: если в LTS не выполнено это условие, то найдётся такая другая LTS, что их композиция не *S*-τ-ограничена. В этом смысле условие τ-ограниченности и слабо-регулярности не является необходимым.

Причина этого в следующем. Пусть в состоянии a начинается бесконечная неразрушающая трасса μ, которая не "укладывается" в конечную под-LTS: каждый маршрут с этой трассой, начинающийся в a, проходит через бесконечное множество состояний. Чтобы при композиции такой маршрут превратился в τ-маршрут, нужно в парной LTS в некотором состоянии b определить маршрут с неразрушающей противоположной трассой μ. Тогда

все стимулы и реакции трассы μ станут синхронными. Однако может оказаться, что из-за этого невозможно получить состояние ab, достижимое по безопасной βγδ-трассе плюс один переход по стимулу или реакции (пример на Рис.103). Другой вариант: каждый раз, когда состояние ab достижимо таким образом, вместе с ним достигается такое состояние a`b, что в состоянии a` трасса μ "укладывается" в конечную под-LTS. А тогда требование *S*-τ-ограниченности композиции не налагает никаких ограничений на бесконечные τ-маршруты, начинающиеся в ab.

Композиция не слабо-регулярных операндов *S*-τ-ограничена



Алфавит:
{?a,?b}

Состояние a не слабо-регулярно.
Если бесконечный маршрут, начинающийся в a, при композиции превращается в τ-маршрут, то композиция опасна, но в состоянии a есть τ-цикл

Рис.103.

Необходимое и достаточное условие должно быть "промежуточным" между префикс-регулярностью и слабо-регулярностью. Однако представляется, что такое условие может оказаться "диким": слишком сложным и трудно проверяемым. В то же время класс τ-ограниченных и слабо-регулярных LTS достаточно широк: он включает не только все конечные LTS, но и очень многие бесконечные LTS, в том числе и LTS с маршрутами, проходящими через бесконечные множества состояний.

3.6.6. Слабо-регулярность и $\mathcal{T}_{\beta\gamma\delta}$-преобразование. Сильно-регулярность

Слабо-регулярность – это требование к операнду композиции. Однако первичные элементы такой композиции – это не исходные LTS, а результаты их $\mathcal{T}_{\beta\gamma\delta}$-преобразования. К сожалению, преобразование $\mathcal{T}_{\beta\gamma\delta}$ обладает неприятным свойством: оно может не только не строить слабо-регулярную LTS при не слабо-регулярном операнде, но даже не сохранять слабо-регулярность операнда. Пример на Рис.104.

Пример несохранения слабо-регулярности при $\mathcal{T}_{\beta\gamma\delta}$-преобразовании.
(используется оптимизированное преобразование $\mathcal{T}_{\beta\gamma\delta}$)

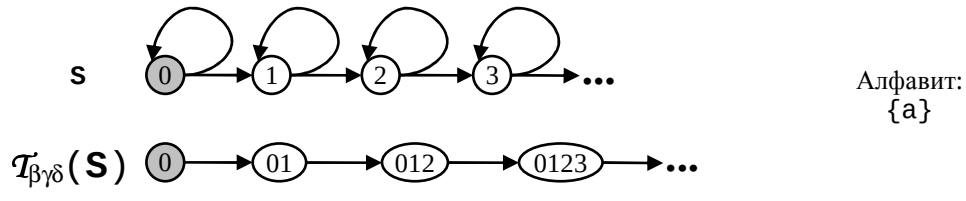


Рис.104.

Заметим, что эта проблема возникает только в том случае, когда исходная LTS безопасна, поскольку в противном случае $\mathcal{T}_{\beta\gamma\delta}$-преобразованная LTS состоит только из одного γ-состояния с единственным переходом γ-петлёй.

Таким образом, нам нужно переформулировать условие применительно к исходным LTS, то есть найти условие, которому должны удовлетворять исходные LTS, чтобы соответствующие им $\mathcal{T}_{\beta\gamma\delta}$-преобразованные LTS были слабо-регулярными. Такое условие мы назовём сильно-регулярностью.

Определение 125: βγδ-трассу будем называть *регулярной*, если регулярен каждый маршрут, имеющий эту βγδ-трассу.

□

Определение 126: Будем говорить, что LTS *сильно-регулярна*, если для каждой бесконечной безопасной βγδ-трассы $\mu\cdot\lambda$, где $\lambda\in C_\gamma^*$ (трасса – не содержит βδ-отказов), существует регулярная безопасная βγδ-трасса $\mu\cdot\sigma$, где $\sigma_\downarrow C=\lambda$.

□

Утверждение 86: Пусть $C\subseteq Z$ и $S\in SBLTS_{\beta\gamma\delta}(C)$. Для того, чтобы преобразованная LTS $\mathcal{T}_{\beta\gamma\delta}(S)$ была слабо-регулярной, необходимо и достаточно, чтобы LTS S была сильно-регулярной.

□462

3.6.7. Двойная LTS и $\mathcal{W}_{\beta\gamma\delta}$-преобразование.

Структура раздела:

- Двойная LTS: преобразование $\mathcal{W}_{\beta\gamma\delta}$.
- Монотонность и левомонотонность $\mathcal{W}_{\beta\gamma\delta}$-преобразования на классе *S*-ветвящихся LTS-спецификаций.
- *S*-регулярность и замкнутость по $\mathcal{W}_{\beta\gamma\delta}$-преобразованию.
- Алгоритмизация $\mathcal{W}_{\beta\gamma\delta}$-преобразования.

Здесь мы рассмотрим другое решение проблемы несохранения слабо-регулярности при $\mathcal{T}_{\beta\gamma\delta}$-преобразовании. Вместо того, чтобы требовать от исходной LTS сильно-регулярности, которая $\mathcal{T}_{\beta\gamma\delta}$-преобразованием превращается в слабо-регулярность, мы определим $\mathcal{W}_{\beta\gamma\delta}$-преобразование, сохраняющее слабую регулярность и *S*-τ-ограниченность. Более того, мы ослаблим требование слабой регулярности исходной LTS, введя *S*-регулярность.

Двойная LTS: преобразование $\mathcal{W}_{\beta\gamma\delta}$.

Идея $\mathcal{W}_{\beta\gamma\delta}$-преобразования заключается в том, чтобы «объединить» исходную и $\mathcal{T}_{\beta\gamma\delta}$-преобразованную LTS, добавив τ-переходы из каждого нестабильного power-состояния $[\mu]$ в принадлежащие ему состояния исходной LTS $s\in[\mu]$. В такой двойной LTS $\mathcal{T}_{\beta\gamma\delta}$-составляющая обеспечит мажорирование ϕ-трасс конформных реализаций, а исходная LTS обеспечит нужные свойства регулярности и *S*-τ-ограниченности.

Определение 127: Определим двойную LTS $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$, составленную из исходной LTS $\mathbf{S}$ и $\mathcal{T}_{\beta\gamma\delta}$-преобразованной LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ (Рис.105). Из каждого нестабильного power-состояния $[\mu]$ определим τ-переходы $[\mu]\text{—}\tau\rightarrow s$ во все состояния исходной LTS $s\in[\mu]$. В качестве начального состояния LTS $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ выберем начальное power-состояние $[\epsilon]$.

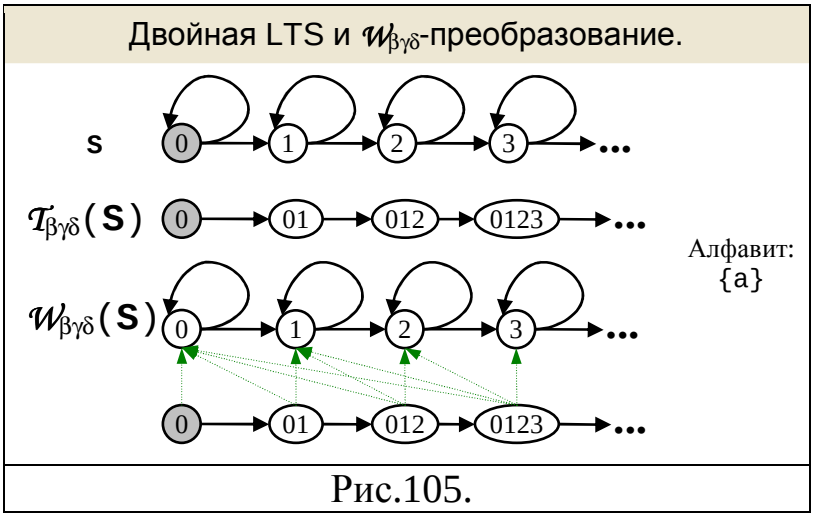


Рис.105.

Монотонность и левомонотонность $\mathcal{W}_{\beta\gamma\delta}$-преобразования на классе *S*-ветвящихся LTS-спецификаций.

Мы покажем, что на классе *S*-ветвящихся LTS-спецификаций $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ и $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ эквивалентны по мажорированию βγδ-трасс и ϕ-трасс.

Мажорирование βγδ-трасс совпадает с отношением $\boldsymbol{ioco}_{\beta\gamma\delta}$, которое транзитивно. Поэтому инвариантность $\mathcal{T}_{\beta\gamma\delta}$-преобразования влечёт инвариантность $\mathcal{W}_{\beta\gamma\delta}$-преобразования. Мажорирование ϕ-трасс транзитивно. Поэтому мажорантность $\mathcal{T}_{\beta\gamma\delta}$-преобразования и влечёт мажорантность $\mathcal{W}_{\beta\gamma\delta}$-преобразование. Тем самым $\mathcal{W}_{\beta\gamma\delta}$-преобразование $\boldsymbol{ioco}_{\beta\gamma\delta}$-левомонотонно и $\boldsymbol{ioco}_{\beta\gamma\delta}$-монотонно.

Утверждение 87: На классе *S*-ветвящихся LTS-спецификаций с конечным числом реакций результаты $\mathcal{T}_{\beta\gamma\delta}$- и $\mathcal{W}_{\beta\gamma\delta}$-преобразований эквивалентны по мажорированию βγδ-трасс и ϕ-трасс.

□466

Утверждение 88: Отношение $\boldsymbol{ioco}_{\beta\gamma\delta}$ $\mathcal{W}_{\beta\gamma\delta}$-левомонотонно и $\mathcal{W}_{\beta\gamma\delta}$-монотонно на классе *S*-ветвящихся LTS-спецификаций с конечным числом реакций.

□468

_S_-регулярность и замкнутость по $\mathcal{W}_{\beta\gamma\delta}$-преобразованию.

Определение 128: Состояние `s` будем называть *S-регулярным*, если для каждой бесконечной трассы λ, начинающейся в `s` и проходящей только через *S*-достижимые состояния, существует регулярный маршрут, начинающийся в `s` и имеющий трассу λ.

□

Заметим, что такая трасса неразрушающая, поскольку в противном случае она проходила бы через состояние, в котором есть трасса ⟨γ⟩, а такое состояние не может быть *S*-достижимым. Однако такая трасса не обязана быть безопасной в `s`, например, трасса `!y,!y,...` в начальном состоянии LTS на Рис.106.

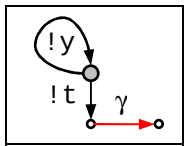


Рис.106.

Определение 129: Пусть $C \subseteq Z$ и $\mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$. Будем говорить, что LTS **S** *S-регулярна*, если *S*-регулярно каждое её *S*-достижимое состояние.

□

Утверждение 89: Преобразование $\mathcal{W}_{\beta\gamma\delta}$ сохраняет свойства *S*-τ-ограниченности и *S*-регулярности для LTS с конечным алфавитом реакций.

□468

Алгоритмизация $\mathcal{W}_{\beta\gamma\delta}$-преобразования.

Поскольку преобразование $\mathcal{T}_{\beta\gamma\delta}$ является частью преобразования $\mathcal{W}_{\beta\gamma\delta}$, а для алгоритмизации преобразования $\mathcal{T}_{\beta\gamma\delta}$ требуется конечность алфавита, для алгоритмизации преобразования $\mathcal{W}_{\beta\gamma\delta}$ также требуется конечность алфавита.

Утверждение 90: Существует алгоритм, который для каждой **L2**-LTS **S** с конечным базовым алфавитом возвращает **L2**-LTS $\mathcal{W}_{\beta\gamma\delta}$(**S**).

□471

3.6.8. Алгоритмизация композиции $\mathcal{T}_{\beta\gamma\delta}$-преобразованных LTS

Структура раздела:
- Алгоритмическая композиция L2-LTS.
- L1-LTS.

Алгоритмическая композиция L2-LTS.

Выше мы установили (Утверждение 81: и Утверждение 86:), что, если исходная LTS **S** в конечном алфавите *S*-τ-ограниченная и сильно-регулярная, то преобразованная LTS $\mathcal{T}_{\beta\gamma\delta}$(**S**) тоже в конечном алфавите, конечно-ветвящаяся, τ-ограниченная и слабо-регулярная.

Кроме этого, мы установили (Утверждение 82: и Утверждение 83:), что существует алгоритмическое преобразование **L1**-LTS **S** во всюду-заданную **L2**-LTS $\mathcal{T}_{\beta\gamma\delta}$(**S**).

Также выше показано (следствие из Утверждение 85:), что класс конечно-ветвящихся, τ-ограниченных и слабо-регулярных LTS замкнут по композиции LTS.

Нам осталось рассмотреть алгоритмизацию композиции $\mathcal{T}_{\beta\gamma\delta}$-преобразованных **L2**-LTS. Хотя такие LTS конечно-ветвящиеся, мы рассмотрим более общий случай локально-конечно-ветвящихся **L2**-LTS.

Для таких LTS небольшая проблема возникает при построении итератора реакций композиционного алфавита A⅂⌊B в том случае, когда алфавит стимулов одного из операндов бесконечен, например, ?A. Для каждой реакции !y∈!B нам нужно проверить, является она синхронной или асинхронной, то есть ?y∈?A или ?y∉?A. Тем самым, требуется

разрешимость алфавита стимулов ?A относительно множества !B. Заметим, что число реакций конечно и, следовательно, множество !B также конечно, однако, это не гарантирует требуемую разрешимость.

Для того, чтобы такая гарантия была при композиции любых LTS рассматриваемого класса, достаточно потребовать разрешимость алфавита стимулов ?A относительно всего универсума стимулов ?Z (заметим, что ?Z={?}×W и !Z={!}×W, что обеспечивается для каждой реакции наличие в ?Z стимула, противоположного ей: ?Z=!Z). Для алгоритмического задания LTS нам, естественно, требуется *оракул стимулов*, разрешающий алфавит стимулов LTS относительно универсума стимулов.

Определение 130: Будем говорить, что для LTS в алфавите C задан *оракул стимулов*, если задан алгоритм $\mathbf{isX}_C$, разрешающий алфавит стимулов ?C относительно универсума стимулов ?Z: для ?x∈?Z $\mathbf{isX}_C$(?x) = ?x∈?C. □

Утверждение 91: Существует алгоритм, который для любых двух всюду-заданных **L2**-LTS с оракулами стимулов строит всюду-заданную **L2**-композицию с оракулом стимулов.

□472

Очевидно, для конечного алфавита C оракул стимулов $\mathbf{isX}_C$ строится тривиально по итератору стимулов $\mathbf{X}_C$. Поэтому существует алгоритм, который для любых двух всюду-заданных **L2**-LTS в конечных алфавитах строит всюду-заданную **L2**-композицию.

Теперь мы можем подвести итоги алгоритмизации преобразования $\mathcal{T}_{\beta\gamma\delta}$ и последующей композиции в виде требований к исходной спецификации и свойств результата преобразования и композиции:

A	$\mathcal{T}_{\beta\gamma\delta}$(**A**)	$\mathcal{T}_{\beta\gamma\delta}(\mathbf{A_1}) \| \mathcal{T}_{\beta\gamma\delta}(\mathbf{A_2})$
конечность алфавита		
S-ветвимость	конечная ветвимость	
S-τ-ограниченность	τ-ограниченность	
сильная регулярность	слабая регулярность	
L2	всюду-заданная **L2**	

В каждой строке более сильное требование выделено серым цветом фона. При этих ограничениях результаты преобразований и композиции удовлетворяют одним и тем же ограничениям.

L1-LTS.

Поскольку у нас всегда есть алгоритмическое преобразование **L2**2**L1**:**L2**→**L1**, мы можем на любом этапе многократной композиции его выполнить. Однако результат такого преобразования, вообще говоря, не может быть преобразован обратно в **L2**-LTS. Причина в нарушении γ-однородности: в отличие от $\mathcal{T}_{\beta\gamma\delta}$-преобразованной LTS, композиция LTS может быть не γ-однородной.

Покажем, что нарушение γ-однородности препятствует преобразованию **L1**2**L2**. Пусть в LTS **A** в *S*-достижимом состоянии a определены разрушающие переходы по реакциям $!y_1,...,!y_n$ (n>0) и хотя бы один неразрушающий переход по реакции !t. Рассмотрим другую LTS **B**, в алфавите которой есть стимулы $?y_1,...,?y_n$, но нет стимула ?t, и которая состоит из одного начального состояния b_0 без переходов. При композиции **A**|↓**B** достижимое состояние ab_0 будет *S*-достижимо, в нём не будет разрушающих переходов по реакциям, но должен быть определён неразрушающий переход по асинхронной реакции !t. Однако, если LTS **A** задана способом **L1**, то нам неизвестно постсостояние перехода a—!t→, поэтому мы не сможем определить постсостояние перехода ab_0—!t→.

Таким образом, преобразование **L2**2**L1** имеет смысл делать только после завершения композиции, когда результат композиции будет использоваться только для верификации конформности.

3.6.9. Алгоритмизация композиции $\mathcal{W}_{\beta\gamma\delta}$-преобразованных LTS

Структура раздела:
- Две проблемы композиции.
- Раскрашенные LTS и *C*-преобразование.
- Как обойтись без раскраски состояний?

Две проблемы композиции.

В отличие от $\mathcal{T}_{\beta\gamma\delta}$-преобразования алгоритм $\mathcal{W}_{\beta\gamma\delta}$-преобразования в общем случае не строит всюду-заданную **L2**-LTS, поскольку $\mathcal{W}_{\beta\gamma\delta}$(**S**) включает как под-LTS исходную **L2**-LTS **S**. Также мы установили (Утверждение 89:), что $\mathcal{W}_{\beta\gamma\delta}$-преобразование, сохраняет *S*-τ-ограниченность и *S*-регулярность. Однако, в отличие от LTS $\mathcal{T}_{\beta\gamma\delta}$(**S**) LTS $\mathcal{W}_{\beta\gamma\delta}$(**S**) в общем случае не является конечно-ветвящейся, τ-ограниченной и слабо-регулярной, поскольку содержит как свою часть исходную **L2**-LTS **S**.

Всё это создаёт две проблемы при композиции: проблему *S*-достижимости и проблему *S*-τ-ограниченности. Эти проблемы связаны с тем, что для алгоритмической проверки конформности по композиции спецификаций требуется выполнение следующих условий: а) в *S*-достижимых состояниях композиции алгоритмически определены все переходы, то есть *S*-достижимые состояния зелёные, и 2) сама композиция *S*-τ-ограниченная.

Рассмотрим композиционное состояние ab в LTS **A**⫴**B**.

Проблема *S*-достижимости: состояние ab может оказаться *S*-достижимым (зелёным), а одно из состояний-операндов a или b *S*+-достижимым (жёлтым). Пример на Рис.99 (исходные LTS).

Для верификации конформности нам было необходимо и достаточно следующего условия: зелёные состояния исходных LTS те и только те состояния, которые *S*-достижимы. Если состояние ab *S*-достижимо, то, в силу необходимости условия верификации конформности, оно должно быть зелёным, что возможно при алгоритмической композиции только в том случае, когда оба операнда a и b зелёные. Иначе мы не можем построить алгоритмы в состоянии ab по алгоритмам в операндах a и b. Однако, если мы используем условие верификации конформности для исходных LTS как необходимое и достаточное, то некоторое состояние a или b может оказаться не зелёным, а жёлтым (*S*+-достижимым).

Проблема *S*-τ-ограниченности: начальное состояние или множество постсостояний переходов ab—z→ из *S*-достижимого (зелёного) состояния ab по стимулу или реакции z может оказаться не τ-ограниченным.

Отчасти эта проблема также является простым следствием проблемы *S*-достижимости. Однако даже в том случае, когда все состояния ab, a и b *S*-достижимы (зелёные), множество {a`b`|ab—z→a`b`} может оказаться не *S*-τ-ограниченным.

Например, переход ab—z→a`b наследует асинхронный переход из первого операнда a—z→a`. Состояние a` *S*+-достижимо (жёлтое) и поэтому число переходов из него по каждому символу t не регламентировано. Может оказаться, что существует неперечислимое множество переходов вида a`—t→. Если в другом состоянии b

определён переход $b\text{—}\underline{t}\rightarrow$, то число τ-переходов из состояния $a`b$ будет неперечислимым.

Другой пример: существует перечислимое, но бесконечное, множество переходов вида $a`\text{—}t\rightarrow$. Пусть, для определённости, все эти переходы, а также единственный переход $b\text{—}\underline{t}\rightarrow$, ведут терминальные состояния. Тогда в состоянии $a`b$ дерево τ-маршрутов не содержит маршрутов с самопересечением и состояний с γ-переходом, но будет бесконечным «веером» τ-переходов.

Раскрашенные LTS и $\mathcal{C}$-преобразование.

Все эти проблемы возникают по одной причине: в LTS $\mathcal{W}_{\beta\gamma\delta}(\mathbf{s})$ могут быть *S*+-достижимые (жёлтые) состояния. Попробуем от них избавиться с помощью преобразования $\mathcal{C}$:

1. Мы оставим только *S*-достижимые (зелёные) состояния и состояние γ. Каждый переход $s\text{—}z\rightarrow s`$ из *S*-достижимого (зелёного) состояния $s$ в *S*+-достижимое (жёлтое) или *S*+γ-достижимое (красное) состояние $s`$ удалим, добавив переход $s\text{—}z\rightarrow\gamma$.

   После этого получится τ-ограниченная и слабо-регулярная LTS.

2. Однако она может остаться *S*-ветвящейся, но не конечно-ветвящейся. Это происходит из-за того, что в *S*-достижимом (зелёном) состоянии s, кроме перехода $s\text{—}z\rightarrow\gamma$ (бывшего ранее или добавленного), остались переходы $s\text{—}z\rightarrow s`$ в *S*-достижимые (зелёные) состояния, которых может быть бесконечное число. Такие переходы мы также удалим.

В результате мы получим конечно-ветвящуюся, τ-ограниченную и слабо-регулярную LTS $\mathcal{C}\circ\mathcal{W}_{\beta\gamma\delta}(\mathbf{s})$, то есть обладающую теми же свойствами, что LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{s})$. Отличие $\mathcal{C}\circ\mathcal{W}_{\beta\gamma\delta}(\mathbf{s})$ от $\mathcal{T}_{\beta\gamma\delta}(\mathbf{s})$ в том, что мы не требуем сильно-регулярности исходной LTS $\mathbf{s}$, а только её *S*-регулярности.

Мы покажем, что отношение $\boldsymbol{ioco}_{\beta\gamma\delta}$ $\mathcal{C}\circ\mathcal{W}_{\beta\gamma\delta}$-левомонотонно и $\mathcal{C}\circ\mathcal{W}_{\beta\gamma\delta}$-монотонно. Тем самым, теория алгоритмизации композиции, описанная для преобразования $\mathcal{T}_{\beta\gamma\delta}$, годится и для преобразования $\mathcal{C}\circ\mathcal{W}_{\beta\gamma\delta}$: класс конечно-ветвящихся, τ-ограниченных и слабо-регулярных LTS замкнут по

композиции LTS (Утверждение 85:, следствие), и композиция таких LTS алгоритмизуема (Утверждение 91:).

Само преобразование $\mathcal{C}\circ\mathcal{W}_{\beta\gamma\delta}$ мы опишем для LTS, в которых цвет состояния явно задан функцией цвета (*раскраской*). Будем считать, что раскраска *правильная*: зелёными состояниями объявлены *S*-достижимые состояния и только они. При этих условиях преобразование $\mathcal{C}\circ\mathcal{W}_{\beta\gamma\delta}$ алгоритмизуемо.

Определение 131: Пусть для LTS $\mathbf{S}$ задана функция цвета состояния $\boldsymbol{color}:V_{\mathbf{S}}\to\{\boldsymbol{зелёный},\boldsymbol{жёлтый},\boldsymbol{красный},\boldsymbol{чёрный}\}$, которую будем называть *раскраской*. Такую LTS будем называть *раскрашенной*. Если зелёные состояния – это *S*-достижимые состояния и только они, а остальные состояния определены по общему правилу на основе зелёных (*S*+-достижимые состояния жёлтые, *S*+γ-достижимые состояния красные), то такую раскраску будем называть *правильной*. LTS с правильной раскраской будем называть *правильно раскрашенной*.

□

Если LTS $\mathbf{S}$ раскрашена, то можно «дораскрасить» LTS $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$, объявив все power-состояния LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ зелёными, а состояние γ красным. Поскольку все power-состояния *S*-достижимы, такая раскраска LTS $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ будет правильной. Поэтому для раскрашенной исходной LTS $\mathbf{S}$ будем считать, что преобразование $\mathcal{W}_{\beta\gamma\delta}$ также строит раскрашенную LTS.

Определение 132: Для правильно раскрашенной LTS $\mathbf{S}=LTS(V_{\mathbf{S}},C_\gamma,E_{\mathbf{S}},s_0)$ определим LTS $\mathcal{C}(\mathbf{S})=LTS(V_{\mathbf{S}}\cup\{\gamma`\},C_\gamma,E,s_0)$, где, $\gamma`\notin V_{\mathbf{S}}$, а E – наименьшее множество, порождаемое следующими правилами вывода: $\forall s,s_1,s_2\in V_{\mathbf{S}}\ \forall u\in C_\tau\ \forall z\in C$

(1) $\vdash \gamma` — \gamma \to \gamma`$,

(2) $\neg\ \mathbf{S}\ \boldsymbol{safe} \vdash s_0 — \tau \to \gamma`$,

(3) $s — u \to s_1\ \&\ \boldsymbol{color}(s)=\boldsymbol{зелёный}$
$\&\ (\forall s_2\ s — u \to s_2 \Rightarrow \boldsymbol{color}(s_2)=\boldsymbol{зелёный}) \vdash s — u \to s_1$,

(4) $s — u \to s_1\ \&\ \boldsymbol{color}(s)=\boldsymbol{зелёный}$
$\&\ \exists s_2\ s — u \to s_2\ \&\ \boldsymbol{color}(s_2)\neq\boldsymbol{зелёный} \vdash s — u \to \gamma`$.

□

Утверждение 92: Пусть задана *S*-τ-ограниченная и *S*-регулярная правильно-раскрашенная LTS $\mathbf{S}$. Тогда LTS $\mathcal{C}(\mathbf{S})$ локально-конечно-ветвящаяся, τ-ограниченная и слабо-регулярная.

□474

Следствие: если LTS $\mathbf{S}$ определена в конечном алфавите, S-τ-ограниченная, S-регулярная и правильно-раскрашенная, то LTS $\mathcal{C}(\mathbf{S})$ конечно-ветвящаяся, τ-ограниченная и слабо-регулярная.

Утверждение 93: Пусть задана правильно-раскрашенная LTS $\mathbf{S}$. Тогда для любого достижимого состояния $s \neq \gamma`$ LTS $\mathcal{C}(\mathbf{S})$ имеет место: $\boldsymbol{init}_{\mathcal{C}(\mathbf{S})}(s) = \boldsymbol{init}_{\mathbf{S}}(s)$ и, если s стабильно, то $\phi_{\mathcal{C}(\mathbf{S})}(s)_r = \phi_{\mathbf{S}}(s)_r$ и $\phi_{\mathcal{C}(\mathbf{S})}(s)_g \supseteq \phi_{\mathbf{S}}(s)_g$ (в нижнем индексе мы указываем LTS, в которой определяется $\boldsymbol{init}$ и ϕ-символ состояния).

□475

Утверждение 94: Пусть задана правильно-раскрашенная LTS $\mathbf{S}$. Тогда:

1. $\mathcal{C}(\mathbf{S})$ $\boldsymbol{ioco}_{\beta\gamma\delta}$ $\mathbf{S}$,
2. $\boldsymbol{traces}_{\phi}^{\omega} \circ \mathcal{C}(\mathbf{S}) \succcurlyeq \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S})$.

□476

Утверждение 95: Соответствие $\boldsymbol{ioco}_{\beta\gamma\delta}$ $\mathcal{C} \circ \mathcal{W}_{\beta\gamma\delta}$-левомонотонно и $\mathcal{C} \circ \mathcal{W}_{\beta\gamma\delta}$-монотонно на классе правильно-раскрашенных S-ветвящихся LTS-спецификаций с конечным алфавитом реакций.

□477

Утверждение 96: Существует алгоритм, который для каждой правильно-раскрашенной $\mathbf{L2}$-LTS $\mathbf{S}$ возвращает всюду-заданную $\mathbf{L2}$-LTS $\mathcal{C}(\mathbf{S})$.

□478

Теперь мы можем подвести итоги алгоритмизации преобразования $\mathcal{C} \circ \mathcal{W}_{\beta\gamma\delta}$ и последующей композиции в виде требований к исходной спецификации и свойств результата преобразования и композиции.

$\mathbf{A}$	$\mathcal{C} \circ \mathcal{W}_{\beta\gamma\delta}(\mathbf{A})$	$\mathcal{C} \circ \mathcal{W}_{\beta\gamma\delta}(\mathbf{A_1}) \between \mathcal{C} \circ \mathcal{W}_{\beta\gamma\delta}(\mathbf{A_2})$
конечность алфавита		
S-ветвимость	конечная ветвимость	
S-τ-ограниченность	τ-ограниченность	
S-регулярность	слабая регулярность	
$\mathbf{L2}$	всюду-заданная $\mathbf{L2}$	

В каждой строке более сильное требование выделено серым цветом фона. При этих ограничениях результаты преобразований и композиции удовлетворяют одним и тем же ограничениям.

Как обойтись без раскраски состояний?

Покажем, что можно обойтись без функции цвета состояния (раскраски), если иметь в виду, что конечная цель – использование спецификации

композиционной системы для верификации конформности (быть может, после многократной композиции).

Проверка конформности основана на выборе безопасных βγδ-трасс спецификации. Для того, чтобы проверить безопасность βγδ-трассы σ, нужно проверить безопасность состояний всех маршрутов с βγδ-трассой σ, а также βγδ-трасс, ответвляющихся от σ по стимулу ?x при продолжении σ блокировкой стимула {?x}, или по реакции !y при продолжении σ другой реакцией !t или стационарностью δ. Такая проверка безопасности происходит, фактически, обходом дерева указанных маршрутов «по ширине».

Пусть в какой-то момент времени построено множество состояний после безопасного префикса μ<σ. Если следующий символ в σ – это блокировка стимула {?x}, то нужно выбрать подмножество состояний с этой блокировкой, а в остальных состояниях проверить безопасность состояний после всех переходов по стимулу z=?x. Если следующий символ в σ – это стационарность δ, то нужно выбрать подмножество стационарных состояний, а в остальных состояниях проверить безопасность состояний после всех переходов по каждой реакции z=!y. Если следующий символ в σ – это реакция !y, то нужно во всех состояниях проверить безопасность состояний после всех переходов по каждой реакции z=!t. Наконец, если следующий символ в σ – это стимул ?x, то нужно проверить безопасность состояний после всех переходов по ?x.

В любом случае сначала определяются переходы по базовому символу z, а потом проверяется безопасность постсостояний этих переходов.

Проверка безопасности состояний выполняется как τ-замыкание множества состояний с контролем по разрушению. Если спецификация – это результат композиции, то τ-замыкание можно делать в два этапа. На первом этапе строится замыкание по асинхронным τ-переходам. Если разрушение не обнаружено, то это означает, что состояния-операнды зелёные, и тогда на втором этапе можно строить синхронные τ-переходы. После построения синхронных τ-переходов снова работает первый этап асинхронного τ-замыкания постсостояний этих синхронных τ-переходов. Затем опять строятся новые синхронные τ-переходы. И так далее. Общая проверка безопасности заканчивается либо обнаружением разрушения, либо тогда, когда больше нет ни асинхронных, ни синхронных τ-переходов.

При многократной композиции асинхронное τ-замыкание для каждого состояния-операнда строится аналогичным образом, если это состояние, в свою очередь, является композицией состояний. В итоге мы строим и анализируем τ-замыкание состояний исходных LTS, а синхронные τ-переходы некоторого уровня композиции делаем только после того, как убедимся, что на предыдущем уровне композиции не обнаруживается разрушение.

При таком способе работы нам не нужна функция цвета, поскольку синхронные τ-переходы будут строиться только в том случае, когда гарантированно все состояния-операнды *S*-достижимы.

Для этого мы должны иметь два итератора **Tau2**: для асинхронных и для синхронных τ-переходов. Это эквивалентно другому варианту: оставляем один итератор **Tau2**, но выполняем асинхронное τ-замыкание в оракуле **Gamma2**. Обычно для композиционного состояния ab оракул возвращает ***true***, если в a или в b определён *γ-переход*:
Gamma2(ab) = **Gamma2**(a)∨**Gamma2**(b) = a—γ→ ∨ b—γ→.
Теперь предлагается возвращать ***true***, если a или b опасно:
Gamma2(ab) = ¬ (a ***safe*** & b ***safe***).

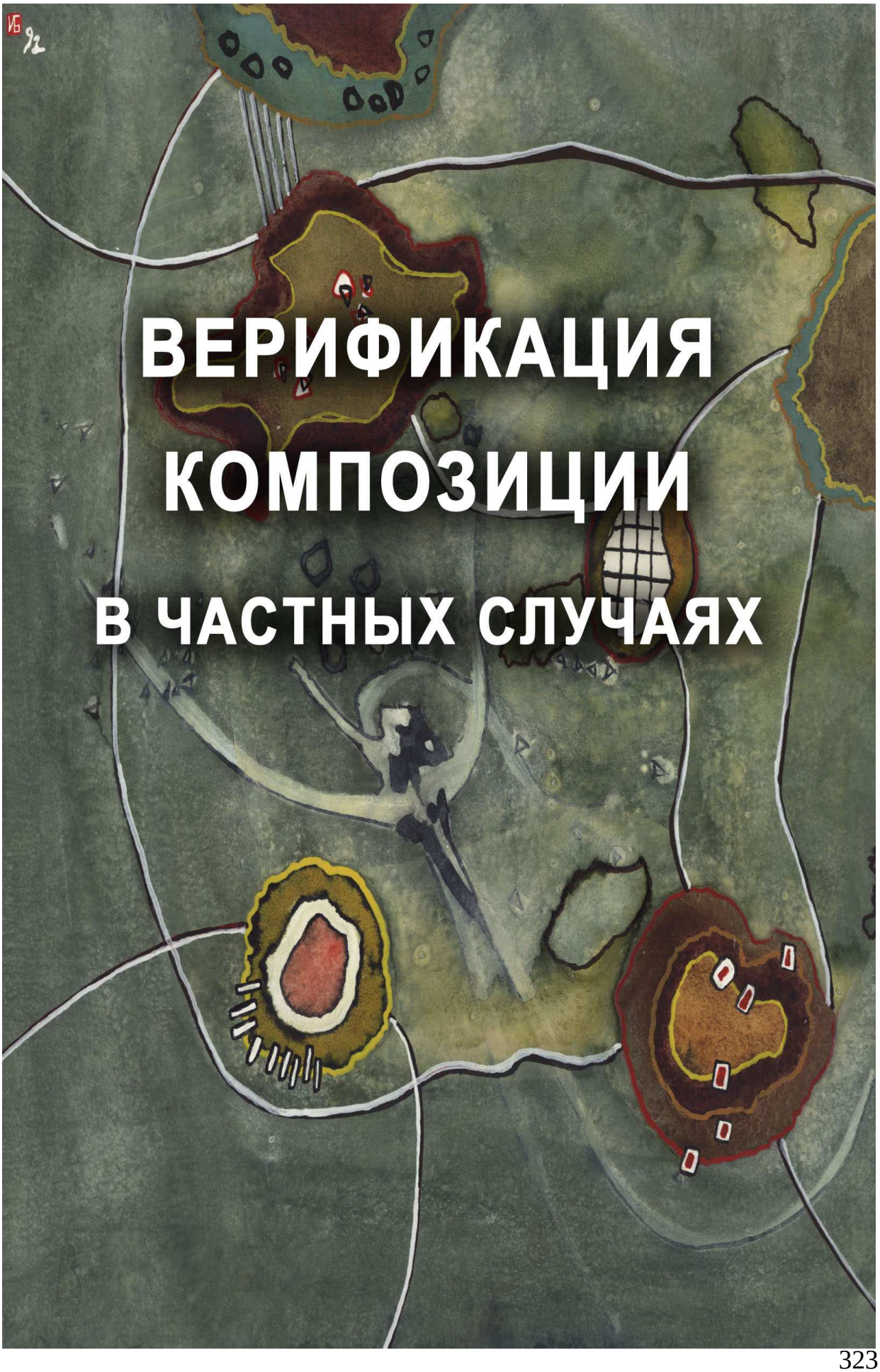
При таком способе работы итератор **Extend2** будет строиться только в *S*-достижимом состоянии композиции и только в тот момент времени, когда мы попадаем в это состояние по безопасной βγδ-трассе. До этого момента времени мы также можем попадать в это состояние, но итератор **Extend2** не используется. В то же время, если все состояния-операнды исходных LTS *S*-достижимы, мы используем их итераторы **Extend2** для того, чтобы строить синхронные τ-переходы, если они нам нужны. Эти синхронные τ-переходы оказываются нужными только в том случае, когда по асинхронным τ-переходам разрушение недостижимо. Иными словами, если модифицированный оракул **Gamma2** сообщил, что разрушение недостижимо по асинхронным τ-переходам во всех интересующих нас состояниях, то мы строим итераторы **Tau2**, что позволяет проверять синхронные τ-переходы.

Все *S*-достижимые состоянии композиции $\mathcal{W}_{\beta\gamma\delta}$-преобразованных спецификаций – это в точности все *S*-достижимые состояния композиции $\mathcal{C}^{\circ}\mathcal{W}_{\beta\gamma\delta}$-преобразованных спецификаций. Каждое из таких состояний является композицией отличных от γ` состояний $\mathcal{C}^{\circ}\mathcal{W}_{\beta\gamma\delta}$-преобразованных спецификаций. В каждом таком состоянии s алгоритмически

определяются все переходы. Если переход s—z→s` «безопасен», то есть принадлежит хотя бы одному маршруту с безопасной βγδ-трассой композиции $\mathcal{W}_{\beta\gamma\delta}$-преобразованных спецификаций, то он будет в композиции $\mathcal{C}\circ\mathcal{W}_{\beta\gamma\delta}$-преобразованных спецификаций. С другой стороны, если в композиции $\mathcal{C}\circ\mathcal{W}_{\beta\gamma\delta}$-преобразованных спецификаций имеется переход s—z→γ`, то символ z в композиции $\mathcal{W}_{\beta\gamma\delta}$-преобразованных спецификаций опасен после любой безопасной βγδ-трассы, заканчивающейся в состоянии s. В то же время в композиции $\mathcal{C}\circ\mathcal{W}_{\beta\gamma\delta}$-преобразованных спецификаций может быть переход s—z→s` для символа z, опасного в композиции $\mathcal{W}_{\beta\gamma\delta}$-преобразованных спецификаций после каждой безопасной βγδ-трассы, заканчивающейся в состоянии s.

Иными словами, композиция $\mathcal{C}\circ\mathcal{W}_{\beta\gamma\delta}$-преобразованных спецификаций определяет те алгоритмы в состояниях, которые достаточны для верификации конформности. В *S*-достижимом состоянии нам нужны переходы по каждому символу, по которому такой переход возникает при композиции $\mathcal{W}_{\beta\gamma\delta}$-преобразованных спецификаций. Для «безопасных» переходов их постсостояния должны быть такими же, как при композиции $\mathcal{W}_{\beta\gamma\delta}$-преобразованных спецификаций. Для остальных переходов постсостояния вообще несущественны, если при этом сохраняются *S*-достижимые состояния. Гарантированный результат будет в том случае, когда все такие переходы ведут в красные состояния (состояния с γ-переходом). В композиции $\mathcal{C}\circ\mathcal{W}_{\beta\gamma\delta}$-преобразованных спецификаций хотя и не все такие переходы ведут в красные состояния, тем не менее, тех, которые ведут в красные состояния, достаточно для сохранения множества *S*-достижимых состояний.

Резюмируя, заметим, что, если спецификация композиционной системы используется только для верификации конформности, то преобразование $\mathcal{C}$ можно считать чисто теоретической конструкцией, моделирующей использование $\mathcal{W}_{\beta\gamma\delta}$-преобразованной спецификации в композиции для последующей верификации конформности.


ВЕРИФИКАЦИЯ
КОМПОЗИЦИИ
В ЧАСТНЫХ СЛУЧАЯХ

Часть 4. Верификация композиции в частных случаях

Структура части:
1. Спецификации без разрушения
2. Спецификации без блокировок
3. Спецификации без блокировок и без разрушения

В этой части рассматриваются три частных класса LTS. Для каждого из этих классов определяется своё отношение мажорирования ϕ-трасс и своё преобразование спецификаций. Эти преобразования оказываются проще преобразования, определённого для общего случая. Доказываются выполнение свойств мажорирования, требуемых достаточными условиями монотонности, и монотонность преобразований. Рассматриваются проблемы алгоритмизации преобразований и композиции преобразованных спецификаций.

Для наглядности, будем называть βγδ-трассы LTS этих трёх классов 1) βδ-трассами, 2) γδ-трассами и 3) δ-трассами[71]. Также мы будем использовать соответствующие обозначения, опуская нижние индексы 1) "$_{\gamma}$", 2) "$_{\beta}$" или 3) "$_{\beta}$" и "$_{\gamma}$" в идентификаторах. Например, 1) $\boldsymbol{LTS}_{\beta\delta}$, $\boldsymbol{traces}_{\beta\delta}$, $\boldsymbol{safe}_{\beta\delta}$, 2) $\boldsymbol{LTS}_{\gamma\delta}$, $\boldsymbol{traces}_{\gamma\delta}$, $\boldsymbol{safe}_{\gamma\delta}$ и 3), $\boldsymbol{LTS}_{\delta}$, $\boldsymbol{traces}_{\delta}$, $\boldsymbol{safe}_{\delta}$ и т.п.

[71] δ-трассы – это то же самое, что трассы с задержками (Suspension Traces).

Глава 4.1. Спецификации без разрушения

Структура главы:
1. Мажорирование ϕ-трасс без разрушения
2. Преобразование $\mathcal{T}_{\beta\delta}$
3. Алгоритмизация

Отсутствие разрушения упрощает определение мажорирования ϕ-трасс. Во-первых, отсутствует γ-мажорирование, есть только α-мажорирование. Во-вторых, ***gamma***-множество любого ϕ-символа пусто, поэтому в качестве ϕ-символа можно использовать просто ***ref***-множество. Из-за этого мажорирование ϕ-символов сводится к их вложенности.

Точно также при определении преобразования нам не нужно вводить состояние γ, и не нужно добавлять переходы по опасным реакциям, поскольку таких нет. В остальном преобразование остаётся тем же и основано на построении power-LTS.

4.1.1. Мажорирование ϕ-трасс без разрушения

$\preccurlyeq$ для ϕ-трасс	Определим мажорирование ϕ-трасс общей теории монотонности как упрощённый вариант мажорирования ϕ-трасс общего случая LTS с блокировками и разрушением.

Определение 133: **Мажорирование ϕ-трасс без разрушения.** Пусть задан алфавит $C \subseteq Z$.

1) Определим отношение мажорирования базовых символов и ϕ-символов без разрушения:

   $\forall a,b \in C_\gamma \qquad a \preccurlyeq b =_{def} a=b,$

   $\forall a,b \in \Phi(C) \quad a \preccurlyeq b =_{def} a_r \subseteq b_r.$

2) Определим отношение мажорирования ϕ-трасс без разрушения (в одном алфавите C):

   $\forall \boldsymbol{\mu},\boldsymbol{\sigma} \in \boldsymbol{\Phi}^\omega(C) \cap C_\phi^{\ \omega}$

   $\boldsymbol{\mu} \preccurlyeq \boldsymbol{\sigma} =_{def} |\boldsymbol{\mu}|=|\boldsymbol{\sigma}| \ \&\ \forall i \in [0..|\boldsymbol{\mu}|]\ \boldsymbol{\mu}(i) \preccurlyeq \boldsymbol{\sigma}(i).$

□

Свойства мажорирования ϕ-трасс в общем случае, которые требуются достаточными условиями монотонности, переносятся на частный случай LTS без разрушения: мажорирование ϕ-трасс является предпорядком, в частности, рефлексивно (Монотонность 8:), генеративно (Монотонность 4:) и композиционно (Монотонность 5:).

На самом деле нам даже не нужны ни определение мажорирования ϕ-трасс без разрушения, ни его свойства. Наша цель – определить преобразование и доказать его (лево-) монотонность. Поскольку мы будем определять преобразование для LTS без разрушения как частный случай общего преобразования $\mathcal{T}_{\beta\gamma\delta}$, свойства монотонности будут выполнены автоматически.

4.1.2. Преобразование $\mathcal{T}_{\beta\delta}$

$\mathcal{T}=\mathcal{T}_{\beta\delta}$	Определим преобразование $\mathcal{T}$ общей теории монотонности как преобразование $\mathcal{T}_{\beta\delta}$ – упрощённый вариант преобразования $\mathcal{T}_{\beta\gamma\delta}$. Докажем его левомонотонность и монотонность.

Поскольку в спецификациях нет разрушения, при определении их преобразования нам не нужно вводить состояние γ, и не нужно добавлять переходы по опасным реакциям, поскольку таких нет. В остальном преобразование остаётся тем же и основано на построении power-LTS.

Определение 134: Определим преобразование $\mathcal{T}_{\beta\delta}: \boldsymbol{LTS}_{\beta\delta} \to \boldsymbol{LTS}_{\beta\delta}$.
Для $C \subseteq Z$, $\mathbf{S}=\mathrm{LTS}(V_S, C, E_S, s_0)$ и $\Sigma=\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$ $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})=\mathrm{LTS}(V, C, E, [\epsilon])$, где $V = [\boldsymbol{T}(\Sigma)] \cup ([\boldsymbol{S}(\Sigma)] \times !C_\delta\})$, а E – наименьшее множество, порождаемое следующими правилами вывода:

$\forall \mu \in \boldsymbol{T}(\Sigma)\ \forall !y \in C\ \forall ?x \in C\ \forall \mathbf{o} \in \beta\delta(C)^*$

(**T**!**T**)	$\mu\cdot\langle !y\rangle \in \boldsymbol{T}(\Sigma)$	$\vdash [\mu] \text{—} !y \to [\mu\cdot\langle !y\rangle]$;
(**T**?**T**)	$\mu\cdot\langle ?x\rangle \in \boldsymbol{T}(\Sigma)$	$\vdash [\mu] \text{—} ?x \to [\mu\cdot\langle ?x\rangle]$;
(**S**!**T**)	$\mu\cdot\mathbf{o} \in \boldsymbol{S}(\Sigma)$ & $\mu\cdot\mathbf{o}\cdot\langle !y\rangle \in \boldsymbol{T}(\Sigma)$	$\vdash [\mu] \text{—} \tau \to [\mu\cdot\mathbf{o}]!y$ $[\mu\cdot\mathbf{o}]!y \text{—} !y \to [\mu\cdot\mathbf{o}\cdot\langle !y\rangle]$;
(**S**?**T**)	—"— —"— & $\mu\cdot\mathbf{o}\cdot\langle ?x\rangle \in \boldsymbol{T}(\Sigma)$	$\vdash [\mu\cdot\mathbf{o}]!y \text{—} ?x \to [\mu\cdot\mathbf{o}\cdot\langle ?x\rangle]$;
(**D**)	$\mu\cdot\mathbf{o} \in \boldsymbol{D}(\Sigma) \cap \boldsymbol{S}(\Sigma)$	$\vdash [\mu] \text{—} \tau \to [\mu\cdot\mathbf{o}]\delta$;
(**D**?**T**)	—"— & $\mu\cdot\mathbf{o}\cdot\langle ?x\rangle \in \boldsymbol{T}(\Sigma)$	$\vdash [\mu\cdot\mathbf{o}]\delta \text{—} ?x \to [\mu\cdot\mathbf{o}\cdot\langle ?x\rangle]$.

□

Очевидно, для LTS без разрушения $\mathbf{S}$ LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ и $\mathcal{T}_{\beta\delta}(\mathbf{S})$ равны. Поэтому (лево-) монотонность преобразования $\mathcal{T}_{\beta\delta}$ непосредственно вытекает из монотонности преобразования $\mathcal{T}_{\beta\gamma\delta}$.

Утверждение 97: Соответствие $\boldsymbol{ioco}_{\beta\gamma\delta}$ $\mathcal{T}_{\beta\delta}$-левомонотонно и $\mathcal{T}_{\beta\delta}$-монотонно на классе *S*-ветвящихся LTS-спецификаций без разрушения с конечным числом реакций.

□478

Необходимость *S*-ветвимости показана на Рис.94. Необходимость преобразования $\mathcal{T}_{\beta\delta}$ для LTS без разрушения показывается теми же примерами, что изображены на Рис.81, Рис.82, Рис.83, Рис.84, Рис.85, Рис.86 и Рис.87.

4.1.3. Алгоритмизация

Структура раздела:

- Замкнутость по преобразованию $\mathcal{T}_{\beta\delta}$
- Алгоритмизация преобразования $\mathcal{T}_{\beta\delta}$ для конечного алфавита
- Незамкнутость по композиции
- Алгоритмизация композиции LTS

Мы будем считать, что LTS-спецификации алгоритмически заданы теми же способами, которые годятся для генерации тестов по этим LTS. В разделе 2.3.7 эти способы обозначены как **L1** и **L2**, и отличаются друг от друга тем, определяются ли переходы только по неразрушающим стимулам и реакциям (**L1**) или по всем стимулам и реакциям (**L2**). В обоих случаях конечен алфавит реакций, LTS безопасны и *S*-ветвятся (**L1**) или *S*-τ-ограничены (**L2**).

Замкнутость по преобразованию $\mathcal{T}_{\beta\delta}$.

Очевидно, при отсутствии разрушения *S*-ветвимость совпадает с конечной ветвимостью, *S*-τ-ограниченность совпадает с τ-ограниченностью. Более того, τ-ограниченность означает конечность дерева τ-маршрутов, начинающихся в одном достижимом состоянии. Утверждение о замкнутости преобразования для общего случая соответствующим образом переносится на случай спецификаций без разрушения.

Утверждение 98: Класс конечно-ветвящихся и подкласс конечно-ветвящихся τ-ограниченных LTS без разрушения в конечном алфавите замкнуты по преобразованию $\mathcal{T}_{\beta\delta}$. Класс конечных LTS в конечном алфавите также замкнут по преобразованию $\mathcal{T}_{\beta\delta}$.

□479

Алгоритмизация преобразования $\mathcal{T}_{\beta\delta}$ для конечного алфавита.

Алгоритмы преобразования LTS общего случая (Утверждение 82:, Утверждение 83:) легко переносятся на случай отсутствия разрушения. Также, как в общем случае, требуется конечность алфавита.

Незамкнутость по композиции.

Класс LTS без разрушения не замкнут по композиции. Действительно, если в LTS без разрушения есть бесконечный маршрут, то всегда можно найти такую LTS без разрушения, в которой есть бесконечный маршрут с противоположной (по операции подчёркивание) трассой. В композиции этих LTS будет бесконечный τ-маршрут, порождаемый бесконечной цепочкой синхронных переходов. Тем самым в композиции возникнет дивергенция, которую мы моделируем разрушением, то есть композиция этих LTS выйдет за пределы класса LTS без разрушения.

Отсюда следует, что для многократной композиции не имеет смысла рассматривать LTS без разрушения.

Алгоритмизация композиции LTS.

Поскольку **L2**-заданная LTS без разрушения, очевидно, является всюду-заданной, мы можем использовать обычный алгоритм композиции всюду-заданных LTS в конечных алфавитах (Утверждение 91:). Разумеется, как и в общем случае, требуется наличие оракула стимулов, который разрешает множество стимулов LTS относительно всего универсума стимулов.

Рассуждения в общем случае о преобразовании при многократной композиции (3.6.3) остаются справедливыми для спецификаций без блокировок и преобразования $\mathcal{T}_{\beta\delta}$.

Глава 4.2. Спецификации без блокировок

Структура главы:
1. Определение мажорирования ϕ-трасс
2. Мажорирование ϕ-трасс предпорядок (Монотонность 8:)
3. Генеративность мажорирования ϕ-трасс (Монотонность 4:)
4. Композиционность мажорирования ϕ-трасс (Монотонность 5:)
5. Преобразование $\mathcal{T}_{\gamma\delta}$
6. Конформность преобразования $\mathcal{T}_{\gamma\delta}$ (Монотонность 6:)
7. Мажорантность преобразования $\mathcal{T}_{\gamma\delta}$ (Монотонность 7:)
8. Монотонность преобразования $\mathcal{T}_{\gamma\delta}$
9. Алгоритмизация

Отсутствие блокировок позволяет резко упростить преобразование спецификаций. Нам не нужно создавать всё многообразие ϕ-символов, которые могут быть в конформных реализациях. Иными словами, нам не требуется построение power-LTS.

Действительно, в любой спецификации без блокировок ***ref***-множество ϕ-символа либо содержит все реакции (стационарность), либо пусто. Но такие ϕ-символы уже есть в исходной спецификации.

Что касается ***gamma***-множества ϕ-символа, то оно предназначено для двух целей. Во-первых, устанавливается соответствие разрушающего стимула в спецификации и блокировки этого стимула в реализации. Однако достаточно, чтобы такой разрушающий стимул в спецификации был определён хотя бы в одном состоянии после ϕ-трассы независимо от других разрушающих стимулов, соответствующих другим блокировкам реализационного состояния. Такое состояние в спецификации может быть либо любым, в частности, нестабильным (то есть без ϕ-символа), либо стационарным. А это уже есть в исходной спецификации. Во-вторых, устанавливается соответствие разрушающей реакции в спецификации и стационарности в реализации. Однако достаточно, чтобы разрушающая реакция была определена хотя бы в одном – любом, в частности, нестабильном (то есть без ϕ-символа) – состоянии после ϕ-трассы, независимо от наличия в реализационном состоянии блокировок стимулов. А это также уже есть в исходной спецификации.

Также нам не нужна сингуляризация, поскольку, если в состоянии определены переходы по реакциям, оно нестационарно и, при отсутствии блокировок, в нём нет никаких βδ-отказов, которые следовало бы учитывать перед той или иной реакцией.

В результате остаётся единственная цель преобразования: обеспечение γ-однородности.

Мы определим специальное мажорирование ϕ-трасс и специальное преобразование на классе *S*-ветвящихся LTS-спецификаций без блокировок и докажем относящиеся к ним условия монотонности 5÷8. На самом деле, отсутствие блокировок нам потребуется только для одного условия – композиционности мажорирования (Монотонность 5:), а *S*-ветвимость, как и в общем случае, – только для мажорантности преобразования (Монотонность 7:).

Как говорилось выше (2.5.3, Расширения семантики *ioco*.), сужение отношения $\boldsymbol{ioco}_{\beta\gamma\delta}$ на класс пополненных LTS-спецификаций без блокировок (разрушение допускается) эквивалентно отношению ***ioco*** с расширенными доменами реализаций (допускаются любые LTS) и исходных спецификаций (допускается разрушение). После пополнения исходной спецификации в ней не остаётся блокировок, но разрушение остаётся (если оно было) и может появиться дополнительное.

4.2.1. Определение мажорирования ϕ-трасс

Структура раздела:
- Сильное мажорирование βγδ-трасс (конечных и бесконечных).
- ξ^{ω}-оператор (порождение конечных и бесконечных βγδ-трасс).
- Мажорирование ϕ-трасс.

$\preccurlyeq$ для ϕ-трасс	Определим мажорирование ϕ-трасс для случая LTS-спецификаций без блокировок и в следующих разделах докажем выполнение условий Монотонность 8:,Монотонность 4: и Монотонность 5:.

Мажорирование ϕ-трасс для спецификаций без блокировок существенно отличается от общего случая. Вместе с тем, хотя блокировки не допускаются в спецификации, они могут быть в реализации, если соответствующие стимулы разрушающие в спецификации.

Поэтому мы должны в мажорировании выразить это соответствие стимулов, разрушающих в спецификации, и их блокировок в реализации. Это соответствие, вообще говоря, не выражается через мажорирование ϕ-символов, поскольку разные стимулы, соответствующие блокировкам одного реализационного состояния, могут быть «разбросаны» по разным (в том числе, нестабильным) состояниям спецификации после ϕ-трассы, то есть

они разрушающие в разных состояниях спецификации. Нужно только учесть, что в реализации блокировка стимула может быть в стационарном состоянии, а тогда этот стимул должен быть разрушающим также в стационарном состоянии спецификации.

Аналогично стационарности в некотором состоянии реализации может соответствовать в спецификации не только стационарность, но и разрушающая реакция, определённая в каком-нибудь (в том числе, нестабильном) состоянии спецификации независимо от того, в этом или каких-то других состояниях определены разрушающие стимулы, соответствующие блокировкам этого реализационного состояния.

В результате мажорирование ϕ-трасс придётся определять как отношение «один к многим». В эти «многие» войдут ϕ-трассы, заканчивающиеся стимулом и разрушением для каждой блокировки в ***ref***-множестве реализационного ϕ-символа. Также стационарности, порождаемой реализационным ϕ-символом, может соответствовать не только стационарность, порождаемая спецификационным ϕ-символом, но и продолжение ϕ-трассы какой-нибудь реакцией и разрушением.

В целом такое мажорирование ϕ-трасс «один к многим» сводится к мажорированию множеств βγδ-трасс, порождаемых мажорируемой ϕ-трассой и всеми ϕ-трассами мажорирующего множества ϕ-трасс. Однако это мажорирование βγδ-трасс – *более сильное* отношение, чем обычное мажорирование βγδ-трасс (Определение 97:): мы должны запретить случай, когда мажорируемая βγδ-трасса продолжается реакцией, а мажорирующая – *другой* реакцией и далее разрушением. Нужно, чтобы мажорирующая βγδ-трасса продолжалась *той же самой* реакцией и далее разрушением, аналогично продолжению стимулом.

Ещё одно отличие связано с тем, что для определения мажорирования бесконечных ϕ-трасс нам придётся использовать бесконечные βγδ-трассы и их сильное мажорирование. Теперь уже бесконечная ϕ-трасса должна порождать бесконечные βγδ-трассы, а не пустое множество, как мы определили выше (Определение 97:). Мы введём новый оператор $\xi^{\omega}(\boldsymbol{\sigma})$, определяющий множество всех βγδ-трасс, порождаемых ϕ-трассой $\boldsymbol{\sigma}$.

Отношение мажорирования «один к многим» естественно распространяется на «многие к многим».

Сильное мажорирование βγδ-трасс (конечных и бесконечных).

Определение 135: Пусть заданы алфавит $C \subseteq Z$ и две (конечные или бесконечные) βγδ-трассы $\sigma, \mu \in C_{\beta\gamma\delta}^{\omega}$.

- Будем говорить, что σ *сильно мажорирует* μ (μ *сильно мажорируется* σ), и обозначать $\mu \preccurlyeq\!\preccurlyeq \sigma$, если A) βγδ-трассы σ и μ совпадают, или B) имеет место *сильное γ-мажорирование*: βγδ-трасса σ конечна и заканчивается разрушением, и либо B1) $\sigma = \langle\gamma\rangle$, либо B2) её префикс π до предпоследнего символа (стимула или реакции) a не включительно совпадает с префиксом βγδ-трассы μ, продолжаемый в μ либо B21) βδ-отказом $\beta\delta(a)$, либо B22) тем же символом a:[72]

  $\mu \preccurlyeq\!\preccurlyeq \sigma =_{def} \mu=\sigma \vee \mu \preccurlyeq\!\preccurlyeq^{\gamma} \sigma$,

  $\mu \preccurlyeq\!\preccurlyeq^{\gamma} \sigma =_{def} \sigma=\langle\gamma\rangle \vee \exists \pi\ \exists a \in C\ \sigma = \pi\cdot\langle a,\gamma\rangle \ \&\ (\pi\cdot\langle\beta\delta(a)\rangle \le \mu \vee \pi\cdot\langle a\rangle \le \mu)$.
- Распространим сильное мажорирование βγδ-трасс на множества βγδ-трасс: $\forall A, B \subseteq C_{\beta\gamma\delta}^{\omega}\ A \preccurlyeq\!\preccurlyeq B =_{def} \forall a \in A\ \exists b \in B\ a \preccurlyeq\!\preccurlyeq b$.

□

Это определение сильного γ-мажорирования βγδ-трасс, очевидно, эквивалентно следующему: мажоранта σ ответвляется от мажорируемой βγδ-трассы μ либо по разрушению, либо по стимулу и разрушению при продолжении μ блокировкой этого стимула, либо по реакции и разрушению при продолжении μ стационарностью:

$\mu \preccurlyeq\!\preccurlyeq^{\gamma} \sigma =_{def} \exists \pi \le \mu\ \sigma = \pi\cdot\langle\gamma\rangle \vee \exists a \in C\ \sigma = \pi\cdot\langle a,\gamma\rangle \ \&\ \pi\cdot\langle\beta\delta(a)\rangle \le \mu$.

Замечание: Поскольку здесь рассматривается случай спецификаций без блокировок, мы могли бы усилить отношение сильного мажорирования βγδ-трасс, запретив мажорирование блокировки $\{?x\}$ блокировкой $\{?x\}$, оставив только γ-мажорирование $\langle ?x,\gamma\rangle$. Формально, для этого нужно было бы изменить мажорирование по равенству: вместо $\mu=\sigma$ нужно $\mu\uparrow\beta(C)=\mu=\sigma$. Все достаточные условия монотонности останутся справедливыми для класса LTS-спецификаций без блокировок. Отличие будет лишь в том, что, кроме композиционности мажорирования ϕ-трасс (Монотонность 5:), отсутствие блокировок в спецификации потребуется

---

[72] При обычном (не сильном) мажорировании конечных βγδ-трасс в случае B22) требуется лишь, чтобы μ продолжалась символом $b \in \beta\delta(a)$. Если a стимул, то $\beta\delta(a)=\{a\}$ как при сильном мажорировании. Но, если a реакция, то $\beta\delta(a)=\delta$, и b любая реакция, не необязательно a.

также для рефлексивности сильного мажорирования βγδ-трасс и, следовательно, рефлексивности мажорирования ϕ-трасс (Монотонность 8:), а также для конечной мажорантности и, следовательно, мажорантности преобразования (Монотонность 7:).

Утверждение 99: Конечная βγδ-трасса сильно мажорируется только конечной βγδ-трассой.

□479

Утверждение 100: Сильное мажорирование конечных βγδ-трасс влечёт обычное мажорирование этих βγδ-трасс:

$\forall C \subseteq Z \ \forall \mu,\lambda \in C_{\beta\gamma\delta}^{*} \ \mu \preccurlyeq\preccurlyeq \lambda \ \Rightarrow \ \mu \preccurlyeq \lambda.$

□479

Утверждение 101: Сильное мажорирование βγδ-трасс и множеств βγδ-трасс являются предпорядками (рефлексивны и транзитивны).

□479

ξ^{ω}-оператор (порождение конечных и бесконечных βγδ-трасс).

Оператор $\xi^{\omega}(\boldsymbol{\sigma})$ аналогичен оператору $\xi(\boldsymbol{\sigma})$ за тем исключением, что для бесконечной ϕ-трассы $\boldsymbol{\sigma}$ определяет не пустое множество βγδ-трасс, а бесконечную последовательность конкатенаций множеств $\xi_{\circ}\boldsymbol{\sigma}(i)$, где $i=1,2,...$.

Заметим, что ϕ-трасса и порождаемая ею βγδ-трасса имеют одну и ту же подтрассу базовых символов. Поэтому, если бесконечная ϕ-трасса имеет бесконечную подтрассу базовых символов, то порождаемая βγδ-трасса также будет бесконечной. Бесконечная ϕ-трасса с конечной подтрассой базовых символов – это ϕ-трасса, завершающаяся бесконечным повторением одного и того же ϕ-символа. Поскольку ϕ-символ порождает все возможные конечные последовательности βδ-отказов, вложенных в его ***ref***-множество, в том числе пустую последовательность, такая ϕ-трасса порождает как бесконечные βγδ-трассы, завершающиеся бесконечным постфиксом βδ-отказов, так и конечные βγδ-трассы. В то же время конечная ϕ-трасса $\boldsymbol{\sigma}$ порождает только конечные βγδ-трассы и для неё $\xi^{\omega}(\boldsymbol{\sigma})=\xi(\boldsymbol{\sigma})$.

Определение 136: Определим ξ^{ω}-оператор, строящий по ϕ-трассе множество *порождаемых* ею βγδ-трасс, как конечных, так и бесконечных.
Для $C \subseteq Z$ и $\boldsymbol{\sigma} \in \boldsymbol{\Phi}^{\omega}(C)$

$\xi^{\omega}(\sigma) =_{def} \cdot(\xi_{\circ}\sigma)$.[73]

□

Мажорирование ϕ-трасс.

Определим мажорирование ϕ-трасс «один к многим» как сильное мажорирование множеств βγδ-трасс, порождаемых мажорируемой ϕ-трассой и всеми ϕ-трассами мажорирующего множества ϕ-трасс. Далее распространим мажорирование «один к многим» на мажорирование множеств ϕ-трасс, то есть отношение «многие к многим», потребовав, чтобы любая ϕ-трасса из мажорируемого множества сильно мажорировалась («один к многим») мажорирующим множеством. Наконец, определим эквивалентность LTS по мажорированию ϕ-трасс как взаимное мажорирование их множеств ϕ-трасс.

Определение 137: Пусть задан алфавит $C \subseteq Z$.

- Мажорирование «один к многим»: ϕ-трасса мажорируется множеством ϕ-трасс, если множество порождаемых ею βγδ-трасс сильно мажорируется множеством βγδ-трасс, порождаемых всеми ϕ-трассами множества:

  $\forall \mu \in \Phi^{\omega}(C)\ \forall \Sigma \subseteq \Phi^{\omega}(C)\ \ \mu \preccurlyeq \Sigma =_{def} \xi^{\omega}(\mu) \preccurlyeq\preccurlyeq \cup_{\circ}\xi^{\omega}(\Sigma)$.

- Распространим мажорирование «один к многим» на случай «многие к многим»:

  $\forall M, \Sigma \subseteq \Phi^{\omega}(C)\ \ M \preccurlyeq \Sigma =_{def} \forall \mu \in M\ \ \mu \preccurlyeq \Sigma$.

- Две LTS-модели A и B будем называть эквивалентными по мажорированию ϕ-трасс, если множества ϕ-трасс этих LTS мажорируют друг друга: $traces_{\phi}^{\omega}(A) \preccurlyeq traces_{\phi}^{\omega}(B)$ & $traces_{\phi}^{\omega}(B) \preccurlyeq traces_{\phi}^{\omega}(A)$.

□

Определение мажорирования множеств ϕ-трасс, очевидно, может быть записано также в следующей форме:

$\forall M, \Sigma \subseteq \Phi^{\omega}(C)\ \ M \preccurlyeq \Sigma =_{def} \cup_{\circ}\xi^{\omega}(M) \preccurlyeq\preccurlyeq \cup_{\circ}\xi^{\omega}(\Sigma)$.

4.2.2. Мажорирование ϕ-трасс предпорядок (Монотонность 8:)

Для общей теории монотонности требуется рефлексивность мажорирования множеств ϕ-трасс на выбранном классе $\mathfrak{C}$. Мы покажем рефлексивность и

[73] Конкатенация последовательности множеств последовательностей – это множество всех возможных конкатенаций последовательностей из этих множеств.

транзитивность мажорирования любых множеств ϕ-трасс, в том числе множеств ϕ-трасс любых LTS, а не только из классае $\mathfrak{C}$.

Утверждение 102: **Условие Монотонность 8: выполнено:** мажорирование множеств ϕ-трасс является предпорядком (рефлексивно и транзитивно).

□480

4.2.3. Генеративность мажорирования ϕ-трасс (Монотонность 4:)

Мы докажем генеративность мажорирования произвольных множеств ϕ-трасс, а не только множеств ϕ-трасс LTS, как это требуется в общей теории монотонности.

Утверждение 103: Мажорирование множеств ϕ-трасс генеративно:

$\forall \mathbf{A},\mathbf{B}\subseteq \mathbf{\Phi}^{\omega}(C)\ \ \mathbf{A}\preccurlyeq\mathbf{B} \ \Rightarrow\ \cup\circ\xi(\mathbf{A})\preccurlyeq\cup\circ\xi(\mathbf{B})$.

□481

4.2.4. Композиционность мажорирования ϕ-трасс (Монотонность 5:)

Композиционность мажорирования мы докажем на классе всех LTS-спецификаций без блокировок. Отсюда будет следовать требуемое свойство на подклассе *S*-ветвящихся LTS-спецификаций без блокировок.

Утверждение 104: **Условие Монотонность 5: выполнено:** мажорирование ϕ-трасс композиционно на классе LTS-спецификаций без блокировок:

$\forall \mathbf{I_1},\mathbf{I_2}\in \boldsymbol{LTS}_{\beta\gamma\delta}\ \ \forall \mathbf{S_1},\mathbf{S_2}\in \boldsymbol{LTS}_{\gamma\delta}$

$\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{I_1})\preccurlyeq\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S_1})\ \ \&\ \ \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{I_2})\preccurlyeq\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S_2})$

$\Rightarrow\ \ \cup(\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{I_1})\between\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{I_2}))\preccurlyeq\cup(\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S_1})\between\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S_2}))$.

□481

4.2.5. Преобразование $\mathcal{T}_{\gamma\delta}$

$\mathcal{T}=\mathcal{T}_{\gamma\delta}$	Определим преобразование $\mathcal{T}$ общей теории монотонности как преобразование $\mathcal{T}_{\gamma\delta}$ и докажем в последующих разделах выполнение условий Монотонность 6: и Монотонность 7:.

Прежде всего покажем, что $\boldsymbol{ioco}_{\beta\gamma\delta}$ не монотонно, то есть тождественное преобразование не монотонно. Немонотонность возникает из-за того, что в

спецификации после общей безопасной βγδ-трассы реакции могут быть опасны, но βγδ-трасса не продолжается некоторой реакцией или такое продолжение не разрушающее (изображено пунктиром). В конформной реализации βγδ-трасса может продолжаться этой реакцией без разрушения. Пример на Рис.88.

Поэтому мы должны так преобразовать спецификацию, чтобы продолжить безопасную βγδ-трассу, после которой реакции опасны, всеми реакциями и далее разрушением. Очевидно, для этого достаточно в каждом состоянии, в котором определён переход по *некоторой* реакции с дальнейшим разрушением, добавить переходы по *каждой* реакции с дальнейшим разрушением. Для этого мы в каждом состоянии s, в котором есть переход по реакции $s—!y→s`$ в состояние $s`$, где есть βγδ-трасса $\langle\gamma\rangle$, добавим переход по каждой $!t$ реакции в специальное γ-состояние $s—!t→\gamma$.

Определение 138: Определим преобразование $\mathcal{T}_{\gamma\delta}:\boldsymbol{LTS}_{\beta\gamma\delta}\to\boldsymbol{LTS}_{\beta\gamma\delta}$. Для $C\subseteq Z$ и $\mathbf{S}=LTS(V_S,C_\gamma,E_S,s_0)$ определим $\mathcal{T}_{\gamma\delta}(\mathbf{S})=LTS(V_S\cup\{\gamma\},C_\gamma,E_S\cup E,s_0)$, где $\gamma\notin V_S$, а E – наименьшее множество, порождаемое следующими правилами вывода:

$\forall s,s`\in V_S \quad \forall !y,!z\in C$

$$\vdash \gamma—\gamma→\gamma,$$

$$s—!y→s` \ \&\ \neg\ s` \ \textit{safe} \quad \vdash s—!z→\gamma.$$

□

Утверждение 105: Преобразование $\mathcal{T}_{\gamma\delta}$ строит γ-однородную LTS для любой исходной LTS.

□486

4.2.6. Конформность преобразования $\mathcal{T}_{\gamma\delta}$ (Монотонность 6:)

Согласно общей теории монотонности, нам надо доказать конформность преобразования на классе *S*-ветвящихся LTS-спецификаций без блокировок, но мы докажем более сильное утверждение об инвариантности преобразования на классе всех LTS-спецификаций.

Утверждение 106: **Условие Монотонность 6: выполнено:** Преобразование $\mathcal{T}_{\gamma\delta}$ инвариантно на классе всех LTS-спецификаций:
$\forall\mathbf{S}\in\boldsymbol{LTS}_{\beta\gamma\delta}\ \ \mathcal{T}_{\gamma\delta}(\mathbf{S})\ \sim_{\boldsymbol{ioco}_{\beta\gamma\delta}}\ \mathbf{S}$.

□487

4.2.7. Мажорантность преобразования $\mathcal{T}_{\gamma\delta}$ (Монотонность 7:)

Согласно общей теории монотонности, нам надо доказать мажорантность преобразования на классе *S*-ветвящихся LTS-спецификаций без блокировок, но мы докажем более сильное утверждение о мажорантности преобразования на классе *S*-ветвящихся LTS-спецификаций, в том числе и тех, в которых могут быть блокировки.

Утверждение 107: Обычное и сильное мажорирование множеств конечных βγδ-трасс одной LTS множеством конечных βγδ-трасс другой LTS совпадают, если мажорирующая LTS γ-однородна:

$\forall \mathbf{I,S} \in \boldsymbol{LTS}_{\beta\gamma\delta}\ \mathbf{S}$ γ-однородна $\Rightarrow\ (\mathbf{I} \preccurlyeq \boldsymbol{\Sigma} \Leftrightarrow \mathbf{I} \preccurlyeq\preccurlyeq \boldsymbol{\Sigma})$, где

$\mathbf{I}=\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{I})$ и $\boldsymbol{\Sigma}=\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$.

□487

Утверждение 108: Любая γ-однородная LTS-спецификация конечно-мажорантна (для мажорирования ϕ-трасс по Определение 137:):

$\forall \mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}\ \mathbf{S}$ γ-однородна $\Rightarrow \cup_{\circ} \boldsymbol{traces}_{\phi} \circ \boldsymbol{\mathcal{I}}(\mathbf{S}) \preccurlyeq \boldsymbol{traces}_{\phi}(\mathbf{S})$.

□487

Утверждение 109: Любая γ-однородная *S*-ветвящаяся конечно-мажорантная LTS-спецификаций мажорантна (для мажорирования ϕ-трасс по Определение 137:):

$\forall \mathbf{S} \in \boldsymbol{SBLTS}_{\beta\gamma\delta}\ \cup_{\circ} \boldsymbol{traces}_{\phi}^{\omega} \circ \boldsymbol{\mathcal{I}}(\mathbf{S}) \preccurlyeq \boldsymbol{traces}_{\phi}^{\omega} \circ \mathcal{T}_{\gamma\delta}(\mathbf{S})$.

□488

Утверждение 110: **Условие Монотонность 7: выполнено:** преобразование $\mathcal{T}_{\gamma\delta}$ мажорантно (для мажорирования ϕ-трасс по Определение 137:) на классе *S*-ветвящихся LTS-спецификаций:

$\forall \mathbf{S} \in \boldsymbol{SBLTS}_{\beta\gamma\delta}\ \cup_{\circ} \boldsymbol{traces}_{\phi}^{\omega} \circ \boldsymbol{\mathcal{I}}(\mathbf{S}) \preccurlyeq \boldsymbol{traces}_{\phi}^{\omega} \circ \mathcal{T}_{\gamma\delta}(\mathbf{S})$.

□491

4.2.8. Монотонность преобразования $\mathcal{T}_{\gamma\delta}$

Утверждение 111: Соответствие $\boldsymbol{ioco}_{\beta\gamma\delta}$ $\mathcal{T}_{\gamma\delta}$-левомонотонно и $\mathcal{T}_{\gamma\delta}$-монотонно на классе *S*-ветвящихся LTS-спецификаций без блокировок.

□492

Необходимость *S*-ветвимости показана на Рис.94.

4.2.9. Алгоритмизация

Структура раздела:

- Замкнутость по преобразованию $\mathcal{T}_{\gamma\delta}$

- Алгоритмизация преобразования $\mathcal{T}_{\gamma\delta}$
- Алгоритмизация композиции $\mathcal{T}_{\gamma\delta}$-преобразованных LTS

Мы будем считать, что LTS-спецификации алгоритмически заданы способом **L2**, который годится для генерации тестов по этим LTS (2.3.7). Переходы определяются по всем стимулам и реакциям. Алфавит реакций конечен, LTS безопасны и *S*-τ-ограничены.

Замкнутость по преобразованию $\mathcal{T}_{\gamma\delta}$.

В отличие от общего случая для LTS без блокировок преобразование $\mathcal{T}_{\gamma\delta}$ не требует конечности алфавита стимулов.

Результат $\mathcal{T}_{\gamma\delta}$-преобразования, в отличие от преобразования $\mathcal{T}_{\beta\gamma\delta}$ и аналогично преобразованию $\mathcal{W}_{\beta\gamma\delta}$, будет конечно-ветвимым, τ-ограниченным и слабо-регулярным только в том случае, когда этими свойствами обладала исходная LTS. В то же время, аналогично преобразованию $\mathcal{W}_{\beta\gamma\delta}$, нам будет достаточно не конечно-ветвимости, τ-ограниченности и слабо-регулярности, а только *S*-τ-ограниченности и *S*-регулярности исходной спецификации, которые сохраняются преобразованием $\mathcal{T}_{\gamma\delta}$.

Утверждение 112: Преобразование $\mathcal{T}_{\gamma\delta}$ сохраняет свойства *S*-τ-ограниченности и *S*-регулярности.

□492

Алгоритмизация преобразования $\mathcal{T}_{\gamma\delta}$.

Преобразование $\mathcal{T}_{\gamma\delta}$ для **L2**-LTS алгоритмизуемо тривиальным способом. Если в состоянии есть хотя бы один разрушающий переход по реакции, то по всем реакциям определяем переходы в состояние γ. Это можно сделать за конечное время, поскольку число реакций конечно, а проверка разрушаемости реакции также выполняется за конечное время в силу *S*-τ-ограниченности LTS.

Утверждение 113: Существует алгоритм, определённый на классе **L2**-LTS и возвращающий для каждой LTS **S** из его домена **L2**-LTS $\mathcal{T}_{\gamma\delta}(\mathbf{S})$.

□492

Рассуждения в общем случае о преобразовании при многократной композиции (3.6.3) остаются справедливыми для спецификаций без блокировок и преобразования $\mathcal{T}_{\gamma\delta}$.

Алгоритмизация композиции $\mathcal{T}_{\gamma\delta}$-преобразованных LTS.

Две проблемы композиции, о которых мы говорили в связи с $\mathcal{W}_{\beta\gamma\delta}$-преобразованием (3.6.9), актуальны и для случая LTS без блокировок и преобразования $\mathcal{T}_{\gamma\delta}$. Решение этих проблемы точно такое же: использование правильно раскрашенных исходных LTS и преобразования $\mathcal{C}$.

В 3.6.9 мы доказали утверждения, касающиеся свойств раскрашенных LTS и преобразования $\mathcal{C}$ безотносительно к мажорированию ϕ-трасс: Утверждение 92:, Утверждение 93:, Утверждение 94: (п.1) и Утверждение 96: об алгоритмизации преобразования $\mathcal{C}$. Эти утверждения остаются верными и в частном случае LTS без блокировок.

Здесь для LTS без блокировок и мажорирования ϕ-трасс по Определение 137: мы докажем утверждения, аналогичные Утверждение 94: (п.2) и Утверждение 95:, касающиеся мажорирования ϕ-трасс и монотонности в связи с преобразованием $\mathcal{C}$.

Утверждение 114: Пусть задана правильно-раскрашенная LTS без блокировок $\mathbf{S}$. Тогда $\boldsymbol{traces}_{\phi}^{\omega}\circ\mathcal{C}(\mathbf{S})\succcurlyeq\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S})$.

□493

Утверждение 115: Соответствие $\boldsymbol{ioco}_{\beta\gamma\delta}$ $\mathcal{C}\circ\mathcal{T}_{\gamma\delta}$-левомонотонно и $\mathcal{C}\circ\mathcal{T}_{\gamma\delta}$-монотонно на классе правильно-раскрашенных *S*-ветвящихся LTS-спецификаций без блокировок с конечным алфавитом реакций.

□494

Преобразование $\mathcal{C}\circ\mathcal{W}_{\beta\gamma\delta}$ даёт конечно-ветвящуюся LTS, так как алфавит спецификации был конечным. Преобразование $\mathcal{C}\circ\mathcal{T}_{\gamma\delta}$ даёт локально-конечно-ветвящуюся LTS, так как алфавит стимулов спецификации может быть не конечным, а только перечислимым. Однако Утверждение 85: и Утверждение 91: доказаны для более общего случая локально-конечно-ветвящихся LTS. Нам требуются лишь оракулы стимулов исходных LTS. Заметим, что оракулы стимулов LTS $\mathbf{S}$, $\mathcal{T}_{\gamma\delta}(\mathbf{S})$ и $\mathcal{C}\circ\mathcal{T}_{\gamma\delta}(\mathbf{S})$ совпадают (поскольку совпадают их алфавиты).

Композиция LTS без блокировок, очевидно, также не имеет блокировок.

Теперь мы можем подвести итоги алгоритмизации преобразования $\mathcal{C}\circ\mathcal{T}_{\beta\gamma\delta}$ и последующей композиции в виде требований к исходной спецификации и свойств результата преобразования и композиции.

$\mathbf{A}$	$\mathcal{C}^{\circ}\mathcal{T}_{\gamma\delta}(\mathbf{A})$	$\mathcal{C}^{\circ}\mathcal{T}_{\gamma\delta}(\mathbf{A_1}) \upharpoonright\!\downarrow \mathcal{C}^{\circ}\mathcal{T}_{\gamma\delta}(\mathbf{A_2})$
конечность алфавита реакций, отсутствие блокировок		
S-ветвимость	конечная ветвимость	
S-τ-ограниченность	τ-ограниченность	
S-регулярность	слабая регулярность	
L2 с оракулом стимулов	всюду-заданная **L2** с оракулом стимулов	

Рассуждения в 3.6.9 «Как обойтись без раскраски состояний?» остаются справедливыми и для случая LTS без блокировок.

Глава 4.3. Спецификации без блокировок и без разрушения

Структура главы:
1. Монотонность
2. Алгоритмизация

$\preccurlyeq$ для ϕ-трасс	Мажорирование ϕ-трасс общей теории монотонности будем понимать как упрощённую версию мажоривания ϕ-трасс, определённого для спецификаций без блокировок. Это сводится к вложенности множеств (как конечных, так и бесконечных) βγδ-трасс.
$\tau=\epsilon$	Определим преобразование τ общей теории монотонности как тождественное преобразование.

Спецификация без блокировок и разрушения является частным случаем спецификации без блокировок. Поэтому здесь можно применять преобразование $\tau_{\gamma\delta}$. При отсутствии разрушения это преобразование становится тождественным. Тем самым, отношение $\boldsymbol{ioco}_{\beta\gamma\delta}$ монотонно и левомонотонно на этом классе LTS-спецификаций.

Как говорилось выше (2.5.3, Расширения семантики *ioco*.), сужение отношения $\boldsymbol{ioco}_{\beta\gamma\delta}$ на класс пополненных LTS-спецификаций без блокировок и без разрушения эквивалентно нерасширенному отношению ***ioco*** в том случае, когда в исходной спецификации каждая δ-трасса продолжается каждым стимулом. Если исходная спецификация не полностью определена по стимулам, то после пополнения в ней не остаётся блокировок (и не появляются разрушения).

4.3.1. Монотонность

Утверждение 116: Соответствие $\boldsymbol{ioco}_{\beta\gamma\delta}$ левомонотонно и монотонно на классе *S*-ветвящихся LTS-спецификаций без блокировок и без разрушения.

□494

4.3.2. Алгоритмизация

Алгоритмизация преобразования: Преобразование для LTS без блокировок и разрушения не требуется.

Алгоритмизация композиции: Поскольку **L2**-заданная LTS без разрушения, очевидно, является всюду-заданной, мы можем использовать обычный алгоритм композиции всюду-заданных LTS (Утверждение 91:). Разумеется,

как и в общем случае, требуется наличие оракула стимулов, который разрешает множество стимулов LTS относительно всего универсума стимулов.

Замкнутость по композиции: Как было сказано выше (4.1.3), класс LTS без разрушения не замкнут по композиции. Отсюда следует, что для многократной композиции не имеет смысла рассматривать LTS без блокировок и разрушения. Это является веской причиной использовать вместо отношения ***ioco*** его расширение на произвольные LTS-реализации и спецификации с разрушением (2.5.3, Расширения семантики *ioco*.).


ИТОГИ
И
ПЕРСПЕКТИВЫ

Итоги и перспективы

1. Основные результаты

Понятие разрушения.

Введено понятие *разрушения*, моделирующего любое нежелательное поведение, то есть поведение, которого следует избегать при тестировании. В частности, реальное разрушение системы. Дивергенцию предлагается моделировать разрушением, поскольку она, как правило, неотличима от отказа и может рассматриваться как нежелательное поведение.

Машина тестирования с ограниченным управлением.

Для формализации тестового эксперимента вводится разновидность машины тестирования с ограниченным управлением (ограниченным набором множеств разрешаемых действий). Для систем с блокировками стимулов, стационарностью и разрушением конкретизируется машина тестирования как $\beta\gamma\delta$-машина.

Трассовая модель и отношение $ioco_{\beta\gamma\delta}$.

Для систем с блокировками, стационарностью и разрушением определяется явная трассовая модель ($\beta\gamma\delta$-модель) и в каком-то смысле эквивалентные подмодели безопасных и финальных трасс.

На уровне трассовой модели для систем с блокировками, стационарностью и разрушением определяются отношение $\boldsymbol{ioco}_{\beta\gamma\delta}$, полный тестовый набор и его генерация.

Предложены три способа алгоритмического задания трассовой спецификации и алгоритмы генерации полного тестового набора.

LTS-модель и отношение $ioco_{\beta\gamma\delta}$.

Для систем с блокировками, стационарностью и разрушением определяется LTS-модель и показывается её эквивалентность (с точки зрения соответствия) трассовой модели. Соответственно, для LTS определяются отношение $\boldsymbol{ioco}_{\beta\gamma\delta}$, полный тестовый набор и его генерация.

Предложены два способа алгоритмического задания LTS-спецификации и алгоритмы генерации полного тестового набора.

Сравнение семантик отношений *ioco* и $ioco_{\beta\gamma\delta}$.

Проведено сравнение семантик отношений ***ioco*** и $ioco_{\beta\gamma\delta}$, показано их совпадение на общем домене реализаций (без блокировок и разрушения) и спецификаций (с учётом соответствующего пополнения). Отношение ***ioco*** расширено на класс реализаций с блокировками и разрушением и спецификаций с разрушением, семантика такого расширения совпадает с семантикой отношения $ioco_{\beta\gamma\delta}$ на тех же доменах реализации и спецификации.

Для этой цели предложено гамма-пополнение спецификации, отличающееся от известного демонического пополнения сохранением информации о неспецифицированных стимулах, что позволяет избегать излишних проверок при асинхронном тестировании.

Верификация композиции.

Введено понятие косой композиции спецификаций, которая позволяет решать две важные задачи: 1) проверка правильности декомпозиции спецификации системы (соотношение спецификаций системы и компонентов), 2) генерация спецификации системы по спецификациям компонентов, в частности, с целью дальнейшей генерации системных тестов.

Построена общая теория монотонности соответствий, основанных на трассах наблюдений, и предложены восемь достаточных условий монотонности.

Введён новый вид трасс – ϕ-трассы, которые, вообще говоря, не являются трассами наблюдений, но отражают ту часть структуры LTS, которая необходима и достаточна для определения композиции с точки зрения соответствия $ioco_{\beta\gamma\delta}$.

Предложены два монотонных преобразования спецификаций $\mathcal{T}_{\beta\gamma\delta}$ и $\mathcal{W}_{\beta\gamma\delta}$ (отличающиеся требованиями к исходным спецификациям). Описаны алгоритмы построения этих преобразований и композиции преобразованных LTS.

Для важных частных случаев LTS без блокировок и/или разрушения предложены упрощённые версии монотонного преобразования.

2. Направления дальнейших исследований

Проблема полноты тестирования.

Полный тестовый набор, как правило, бесконечен и может быть только перечислим. Для практического применения нужно 1) на основе реализационных гипотез и тестовых возможностей определить критерий конечного значимого тестового набора и 2) построить фильтр для отбора тестов такого набора при перечислении полного тестового набора или алгоритм построения такого набора «напрямую».

Одним из универсальных способов решения этих проблем является факторизация спецификации. Однако здесь возникает дополнительная проблема композиции: как связана факторизация компонентов с факторизацией системы? Иными словами, нужно ли при факторизации компонентов заново выполнять преобразование факторизованных спецификаций компонентов и их композицию или можно «напрямую» факторизовать косую композицию исходных спецификаций компонентов.

Проблемы асинхронного тестирования.

В общем виде эта проблема для отношения $\boldsymbol{ioco}_{\beta\gamma\delta}$ решена в настоящей работе. Тем не менее, по аналогии с частными случаями подклассов LTS, когда монотонное преобразование упрощается, можно ожидать упрощения решения этой проблемы для различных частных видов сред взаимодействия.

При асинхронном тестировании наблюдаемая трасса не есть, вообще говоря, трасса, которую проходит реализация. Поэтому мы не можем сравнивать наблюдаемую трассу с трассами спецификации. Вместе с тем существует множество гипотетических трасс, то есть трасс, которые могла бы пройти спецификация при данной наблюдаемой трассе. Тест выносит вердикт ***true***, если это множество не пусто. Конечно, это множество есть множество трасс косой композиции спецификации со средой, но оно может быть вычислено «на лету», зная устройство среды. Способ такого вычисления и есть способ решения проблемы асинхронного тестирования для той или иной среды. Класс «не слишком сложных» сред, для которых такой способ легко определить и алгоритмизовать, достаточно широк и включает многие практические используемые среды взаимодействия.

Смежные проблемы поддержки дизайна систем.

Помимо решённой в данном исследовании проблемы верификации композиции, важное значение имеют следующие две проблемы.

Ослабление требований к компонентам: по спецификациям компонентов и корректной (в частности, самой сильной) спецификации системы сгенерировать ослабленные спецификации компонентов. Корень этой проблемы в том, что спецификация компонента, вообще говоря, является избыточной для его функционирования в конкретном месте данной системы.

Усиление требований к компонентам (декомпозиция системных требований): по спецификации системы и ослабленным (в частности, отсутствующим) спецификациям компонентов сгенерировать полные спецификации компонентов (в рамках данной системы).

Распространение теории на случаи различных тестовых возможностей и реализационных гипотез.

В отличие от вышеперечисленных проблем, связанных с использованием предложенной теории соответствия на практике или с решением смежных проблем (тоже важных на практике), можно предложить следующие другие направления исследований.

Отказы общего вида. Соответствие и композиция для систем с другими (кроме блокировки стимулов и стационарности) видами отказов. Здесь может рассматриваться машина тестирования с ограниченным управлением общего вида. В частности, в связи с проблемами «стимулы на выбор» и «реакции на выбор».

Приоритеты. Соответствие и композиция для систем с приоритетами. Здесь есть проблемы как описания таких систем в трассовой и LTS-модели, так и генерации тестов и, особенно, композиции систем с приоритетами.

Дивергенция. Соответствие и композиция для систем с дивергенцией, уже не интерпретируемой как нежелательное поведение (разрушение). Отчасти эта проблема решается через приоритеты, но имеет и самостоятельное значение. При отсутствии приоритетов можно рассматривать λ-наблюдение «дивергенция или отказ», которое можно проверять по спецификации, а также во многих случаях уточнять как «дивергенция» или как «отказ» с последующей проверкой по спецификации.

θ-действие. Такое действие в реализации (и в спецификации) описывает переход к «альтернативному» поведению, когда другие поведения невозможны. Хотя θ-действие может моделироваться приоритетами, во многих частных случаях оно удобнее в теории и на практике. Например, ε-действие как переход при отсутствии стимулов.

Запрет на торможение реакций как важный частный случай ограниченных тестовых возможностей. Здесь требует переопределение соответствия $\boldsymbol{ioco}_{\beta\gamma\delta}$ и соответствующая модификация генерации тестов и композиции.


ЛИТЕРАТУРА

Литература

[Bern91] G. Bernot. Testing against formal specifications: A theoretical view. In S. Abramsky and T.S.E. Maibaum, editors, TAPSOFT'91, Volume 2, pp. 99-119. Lecture Notes in Computer Science 494, Springer-Verlag, 1991.

[BGNV05] Andreas Blass, Yuri Gurevich, Lev Nachmanson, and Margus Veanes. Play to Test Microsoft Research, Technical Report MSR-TR-2005-04, January 2005. 5th International Workshop on Formal Approaches to Testing of Software (FATES 2005). Edinburgh, July 2005.

[BK05-1] Бурдонов И. Б., Косачев А. С. «Тестирование компонентов распределенной системы.» Труды Всероссийской научной конференции «Научный сервис в сети ИНТЕРНЕТ», Изд-во МГУ, 2005, стр.63-65.

[BK05-2] Бурдонов И. Б., Косачев А. С. «Верификация композиции распределенной системы.» Труды Всероссийской научной конференции «Научный сервис в сети ИНТЕРНЕТ», Изд-во МГУ, 2005, стр.67-69.

[BKK00] И. Б. Бурдонов, А. С. Косачев, В. В. Кулямин. Использование конечных автоматов для тестирования программ. "Программирование". 2000. No. 2. стр.12-28.

[BKK03] И. Б. Бурдонов, А. С. Косачев, В. В. Кулямин. «Асинхронные автоматы: классификация и тестирование.» Труды ИСП РАН, т. 4, 2003, с. 7-84.

[BKK03-1] И. Б. Бурдонов, А. С. Косачев, В. В. Кулямин. Неизбыточные алгоритмы обхода ориентированных графов. Детерминированнный случай. "Программирование". 2003. No. 5.

[BKK04] И. Б. Бурдонов, А. С. Косачев, В. В. Кулямин. Неизбыточные алгоритмы обхода ориентированных графов. Недетерминированнный случай. "Программирование". 2004. No. 1.

[BKK06] I. Bourdonov, A. Kossatchev, and V. Kuliamin. Formal Conformance Testing of Systems with Refused Inputs and

Forbidden Actions. Proc. of MBT 2006, Vienna, Austria, March 2006.

[BP94] Gregor V. Bochmann , Alexandre Petrenko. Protocol testing: review of methods and relevance for software testing, Proceedings of the 1994 international symposium on Software testing and analysis, pp.109-124, August 17-19, 1994, Seattle, Washington, United States.

[BRT03r] M. van der Bijl, A. Rensink, J. Tretmans. Component Based Testing with ioco. CTIT Technical Report TR-CTIT-03-34, University of Twente, 2003.

[BRT03] Machiel van der Bijl, Arend Rensink, Jan Tretmans. Compositional testing with ioco. In Formal Approaches to Software Testing: Third International Workshop, FATES 2003, Montreal, Quebec, Canada, October 6th, 2003. Editors: Alexandre Petrenko, Andreas Ulrich ISBN: 3-540-20894-1. LNCS volume 2931, Springer, pp. 86-100.

[BZ83] Daniel Brand and Pitro Zafiropulo. On communicating finite-state machines. J.ACM, 30(2):323–342, 1983.

[FB92] S. Fujiwara and G. von Bochmann. Testing Nondeterministic Finite State Machine with Fault Coverage. IFIP Transactions, Proceedings of IFIP TC6 Fourth International Workshop on Protocol Test Systems, 1991, Ed. by Jan Kroon, Rudolf J. Hei-jink, and Ed Brinksma, 1992, North-Holland, pp. 267-280.

[Glab90] R.J. van Glabbeek. The linear time – branching time spectrum. In J.C.M. Baeten and J.W. Klop, editors, *CONCUR'90*, Lecture Notes in Computer Science 458, Springer-Verlag, 1990, pp 278–297.

[Glab93] van Glabbeek, R. J. The linear time - branching time spectrum II; the semantics of sequential processes with silent moves. Proceedings CONCUR '93, Hildesheim, Germany, August 1993 (E. Best, ed.), LNCS 715, Springer-Verlag, 1993, pp. 66-81.

[Heer98] L. Heerink. Ins and Outs in Refusal Testing. PhD thesis, University of Twente, Enschede, The Netherlands, 1998.

[Henn64] F. C. Hennie. Fault detecting experiments for sequential circuits. Proc. 5-th Ann. Symp. Switching Circuit Theory and Logical Design, pp. 95–110, 1964.

[HMU02] Д. Хопкрофт, Р. Мотвани, Д. Ульман. Введение в теорию автоматов, языков и вычислений, 2-е изд.. М., «Вильямс», 2002.

[Hoare85] C.A.R. Hoare. Communicating Sequential Processes. Prentice-Hall, 1985.

[Holz91] Gerard J. Holzmann. Design And Validation Of Computer Protocols. Prentice-Hall, 1991.

[HP04] J. L. Huo, A. Petrenko. On Testing Partially Specified IOTS through Lossless Queues. Proc. of TestCom 2004, LNCS 2978, pp. 76–94, Springer-Verlag, 2004.

[HT97] L. Heerink, J. Tretmans. *Refusal Testing for Classes of Transition Systems with inputs and Outputs.* In T. Mizuno, N. Shiratori, T. Higashino, A. Togashi, eds. Formal Description Techniques and Protocol Specification, Testing and Verification. Chapman & Hill, 1997.

[ISO95] Revised Working Draft on “Framework: Formal Methods in Conformance Testing”, JTC1/SC21/WG1/Project 54/1 // ISO Interim Meeting / ITU-T on, Paris, 1995.

[JJTV99] C. Jard, T. Jéron, L. Tanguy, C. Viho. Remote testing can be as powerful as local testing, in Formal methods for protocol engineering and distributed systems, FORTE XII/ PSTV XIX' 99, Beijing, China, J. Wu, S. Chanson, Q. Gao (eds.), pp. 25-40, October 1999.

[KKPPB03] Victor V. Kuliamin, Alexander S.Kossatchev, Alexander K. Petrenko, Nick V. Pakoulin, Igor B. Bourdonov. Integration of Functional and Timed Testing of Real-Time and Concurrent Systems. Perspectives of System Informatics // LNCS. No. 2890. 2003. pp. 450-461.

[KPKB03] В.В.Кулямин, А.К.Петренко, А.С.Косачев, И.Б.Бурдонов. Подход UniTesK к разработке тестов. "Программирование", 2003, №6, стр.25-43. V.V. Kuliamin, A.K. Petrenko, A.S. Kossatchev, and I.B. Burdonov. The UniTesK Approach to Designing Test Suites. Programming and Computer Software, 2003, 6, p.310.

[König] D.König. Über eine Schlusswaise aus dem Endlichen ins Unendliche. Acta Litt. Ac. Sci. Hung. Fran. Josep. No 3. 1927. pp. 121-130. См. также К.Куратовский, А.Мостовский. Теория множеств. М.«Мир», 1970.

[LG05] G. Lestiennes, M.-C. Gaudel. Test de systemes reactifs non receptifs. Journal Europeen des Systemes Automatises, Modelisation des Systemes Reactifs, pp. 255–270. Hermes, 2005.

[LT87] Nancy A. Lynch and Mark R. Tuttle. Hierarchical correctness proofs for distributed algorithms. In Proceedings of the 6th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, pp. 137-151, August 1987.

[LY96] D. Lee and M. Yannakakis. Principles and Methods of Testing Finite State Machines—A Survey, Proceedings of the IEEE 84, No. 8, 1090–1123, 1996.

[Miln80] R. Milner. A Calculus of Communicating Systems, LNCS, vol. 92, Springer-Verlag, 1980.

[Miln81] R. Milner. Modal characterisation of observable machine behaviour. In G. Astesiano & C. Bohm, editors: Proceedings CAAP 81, LNCS 112, Springer, pp. 25-34.

[Miln89] R. Milner. Communication and Concurrency. Prentice-Hall, 1989.

[Phal94] M. Phalippou. Relations d`Implantation et Hypotheses de Test sur des Automates a Entrees et Sorties. PhD thesis, L`Universite de Bordeaux I, France, 1994.

[PYB96] A. Petrenko, N. Yevtushenko, G. von Bochmann. Testing deterministic implementations from nondeterministic FSM specifications. Selected proceedings of the IFIP TC6 9-th international workshop on Testing of communicating systems, September 1996.

[PYH03] A. Petrenko, N. Yevtushenko and J.L. Huo. Testing Transition Systems with Input and Output Testers. In: Proc. IFIP TC6/WG6.1 15th Int. Conf. Testing of Communicating Systems, TestCom’2003, pp. 129-145. Sophia Antipolis, France, May 26-29, 2003.

[Seg93] Segala, R. Quiescence, fairness, testing and the notation of implementation (extended abstract). In LNCS, volume 715. Proceedings of the 4th International Conference on Concurrency Theory (CONCUR`93), Hildesheim, Germany, 1993.

[Ско90] Общая алгебра. Под ред. Л.А.Скорнякова. т.1. М. Наука, Физматлит, 1990.

[Tret92] J. Tretmans. A Formal Approach to Conformance Testing. PhD. Thesis, University of Twente, Enschede, The Netherlands, 1992.

[Tret96] Tretmans, J. Test Generation with Inputs, Outputs and Repetitive Quiescence. In: Software-Concepts and Tools, Vol. 17, Issue 3, 1996.

[Vaan91] F. Vaandrager. On the relationship between process algebra and Input/Output Automata. In Logic in Computer Science, pp. 387-398. Sixth Annual IEEE Symposium, IEEE Computer Society Press, 1991.

[Vasil73] М. П. Василевский. Диагностика ошибок в автоматах. Кибернетика и системный анализ, т. 9, № 4, стр. 98–108, 1973.


ДОКАЗАТЕЛЬСТВА
УТВЕРЖДЕНИЙ

Приложение: Доказательства утверждений

1. Доказательство Утверждение 1:

Пусть задана βγδ-машина в алфавите $C \subseteq Z$, и пусть Σ – множество её трасс, ожидающихся с самого начала работы. Оно, очевидно, является βγδ-деревом, поскольку все его трассы – это βγδ-трассы, и, если ожидается наблюдение βγδ-трассы, то ожидается наблюдение и любого её префикса.

Повторим правила, определяющие работу машины тестирования (Определение 4:):

(Машина.1) После печати символа разрушения γ машина разрушена: больше ничего не печатается.

(Машина.2) После печати отказа $A \subseteq C$ машина не может напечатать никакое из разрешённых действий $a \in A$.

(Машина.3) Машина может напечатать отказ $A \subseteq C$ только в том случае, когда нажата кнопка “A”, а машина находится в стабильном состоянии.

(Машина.4) Если машина находится в стабильном состоянии, нажата кнопка “A”, $A \subseteq C$, и хотя бы одно из разрешённых кнопкой действий $a \in A$ машина может напечатать, то машина обязана напечатать одно из этих разрешённых действий и отжать кнопку “A”.

(Машина.5) Если машина находится в стабильном состоянии, нажата кнопка “A”, $A \subseteq C$, но ни одно из разрешённых этой кнопкой действий $a \in A$ машина не может напечатать, то машина обязана напечатать отказ A и отжать кнопку “A”.

(Машина.6) После печати отказа машина остаётся в том же самом стабильном состоянии: если машина перешла в стабильное состояние s, то любую трассу, которую она может напечатать после печати отказа $A \subseteq C$, она может напечатать и без печати отказа. Формально: $\Sigma(s,A)=\Sigma(s)$.

(Машина.7) Машина не изменяется в процессе работы, то есть, множество трасс, которые можно ожидать после печати любой трассы σ, равно Σ ***after*** σ, где Σ – множество трасс, ожидающихся с самого начала работы.

(Машина.8) Если машина не разрушена, она через конечное ограниченное время либо разрушается, либо переходит в стабильное состояние (моделирование дивергенции разрушением).

(Машина.9) Машина может печатать только те действия $a \in C$, которые разрешены нажатой кнопкой “A”, то есть $a \in A$.

Прежде всего, из свойства (Машина.7) непосредственно следует, что могут наблюдаться те и только те трассы, которые ожидаются с самого начала работы, то есть, множество трасс $\mathbf{\Sigma}$. Поэтому множество трасс машины совпадает с $\mathbf{\Sigma}$.

βγδ-допустимость (βγδ1). Следует из свойства (Машина.1).

βγδ-замкнутость (βγδ4). Покажем, что это следует из свойств (Машина.3,6,7).

Пусть $\mu\cdot\langle o\rangle\cdot\lambda\in\mathbf{\Sigma}$, где o – отказ. Тогда $\langle o\rangle\cdot\lambda\in(\mathbf{\Sigma}\ \textbf{\textit{after}}\ \mu)$ и, по свойству (Машина.7), после трассы μ может наблюдаться трасса $\langle o\rangle\cdot\lambda$. В частности, может наблюдаться отказ o. Тогда, по свойству (Машина.3), после трассы μ машина может оказаться в стабильном состоянии s, после чего возможно наблюдение отказа o и далее трассы λ.

a) *Удаление отказа*. Имеем $\lambda\in\mathbf{\Sigma}(s,o)$.
По свойству (Машина.6), $\mathbf{\Sigma}(s)=\mathbf{\Sigma}(s,o)$.
Поэтому $\lambda\in\mathbf{\Sigma}(s,o) \Rightarrow \lambda\in\mathbf{\Sigma}(s)$.
Тем самым, трасса λ может наблюдаться сразу после трассы μ.
По свойству (Машина.7), $\lambda\in(\mathbf{\Sigma}\ \textbf{\textit{after}}\ \mu)$, то есть $\mu\cdot\lambda\in\mathbf{\Sigma}$.

b) *Повторение отказа*. Имеем $\langle o\rangle\cdot\lambda\in\mathbf{\Sigma}(s)$.
По свойству (Машина.6), $\mathbf{\Sigma}(s)=\mathbf{\Sigma}(s,o)$.
Поэтому $\langle o\rangle\cdot\lambda\in\mathbf{\Sigma}(s) \Rightarrow \langle o\rangle\cdot\lambda\in\mathbf{\Sigma}(s,o) \Rightarrow \langle o\rangle\cdot\langle o\rangle\cdot\lambda\in\mathbf{\Sigma}(s)$.
Тем самым, трасса $\langle o\rangle\cdot\langle o\rangle\cdot\lambda$ может наблюдаться сразу после трассы μ.
По свойству (Машина.7), $\langle o\rangle\cdot\langle o\rangle\cdot\lambda\in(\mathbf{\Sigma}\ \textbf{\textit{after}}\ \mu)$, то есть $\mu\cdot\langle o\rangle\cdot\langle o\rangle\cdot\lambda\in\mathbf{\Sigma}$.

c) *Перестановка соседних отказов*. Пусть $\lambda=\langle o`\rangle\cdot\lambda`$, где $o`$ – отказ.
Тогда $\langle o`\rangle\cdot\lambda`\in\mathbf{\Sigma}(s,o)$.
По свойству (Машина.6), $\mathbf{\Sigma}(s)=\mathbf{\Sigma}(s,o)$.
Отсюда $\langle o`\rangle\cdot\lambda`\in\mathbf{\Sigma}(s,o) \Rightarrow \langle o`\rangle\cdot\lambda`\in\mathbf{\Sigma}(s) \Rightarrow \lambda`\in\mathbf{\Sigma}(s,o`)$.
По свойству (Машина.6), $\mathbf{\Sigma}(s,o`)=\mathbf{\Sigma}(s)=\mathbf{\Sigma}(s,o)$.
Поэтому $\lambda`\in\mathbf{\Sigma}(s,o`) \Rightarrow \lambda`\in\mathbf{\Sigma}(s,o) \Rightarrow \langle o\rangle\cdot\lambda`\in\mathbf{\Sigma}(s)$
$\Rightarrow\langle o\rangle\cdot\lambda`\in\mathbf{\Sigma}(s,o`) \Rightarrow \langle o`\rangle\cdot\langle o\rangle\cdot\lambda`\in\mathbf{\Sigma}(s)$.

Тем самым, трасса $\langle o`\rangle\cdot\langle o\rangle\cdot\lambda`$ может наблюдаться сразу после трассы μ.

По свойству (Машина.7), $\langle o`\rangle\cdot\langle o\rangle\cdot\lambda`\in(\Sigma\ \textit{after}\ \mu)$, то есть $\mu\cdot\langle o`\rangle\cdot\langle o\rangle\cdot\lambda`\in\Sigma$.

βγδ-согласованность (βγδ2). Покажем, что это следует из свойств (Машина.2,3,6,7).

Поскольку βγδ-замкнутость следует из свойств (Машина.3,6,7), для доказательства βγδ-согласованности (βγδ2) достаточно показать выполнение свойства ослабленной βγδ-согласованности. Если $\mu\cdot\langle o\rangle\in\Sigma$, где o – отказ, значит, по βγδ-замкнутости, $\mu\cdot\langle o\rangle\cdot\langle o\rangle\in\Sigma$, то есть $\langle o\rangle\in(\Sigma\ \textit{after}\ \mu\cdot\langle o\rangle)$. По свойству (Машина.7), после наблюдения трассы $\mu\cdot\langle o\rangle$ может снова наблюдаться отказ o.

a) По свойству (Машина.3), после наблюдения $\mu\cdot\langle o\rangle$ машина не может выполнить действие разрушения γ, поскольку может наблюдаться отказ o.

   По свойству (Машина.7) $\langle\gamma\rangle\notin(\Sigma\ \textit{after}\ \mu\cdot\langle o\rangle)$, то есть $\mu\cdot\langle o\rangle\cdot\langle\gamma\rangle\notin\Sigma$.

b) Если $o=\{?x\}$, значит, по свойству (Машина.2), после наблюдения $\mu\cdot\langle\{?x\}\rangle$ машина не может принять стимул $?x$, поскольку может наблюдаться его блокировка $\{?x\}$, то есть, не может ожидаться трасса $\langle?x\rangle$.

   По свойству (Машина.7) $\langle?x\rangle\notin(\Sigma\ \textit{after}\ \mu\cdot\langle\{?x\}\rangle)$, то есть $\mu\cdot\langle\{?x\}\rangle\cdot\langle?x\rangle\notin\Sigma$.

c) Аналогично, если $o=\delta$, значит, по свойству (Машина.2), после наблюдения $\mu\cdot\langle\delta\rangle$ машина не может выдать ни одну из реакций $!y$, поскольку может наблюдаться δ, то есть, не может ожидаться ни одна из трасс $\langle!y\rangle$.

   По свойству (Машина.7), для любой реакции $!y$ имеет место $\langle!y\rangle\notin(\Sigma\ \textit{after}\ \mu\cdot\langle\delta\rangle)$, то есть $\mu\cdot\langle\delta\rangle\cdot\langle!y\rangle\notin\Sigma$.

βγδ-конвергентность (βγδ3). Покажем, что это следует из свойств (Машина.3,4,5,7,8,9).

Если трасса не содержит символ γ, значит машина не разрушена. Поэтому, если трасса не продолжается символом γ, значит машина после этой трассы не может выполнить действие разрушения и, по свойству (Машина.8), у неё нет дивергенции: через конечное время машина перейдёт в стабильное состояние.

Если до перехода машины в стабильное состояние никакая кнопка не нажата, машина, по свойству (Машина.3) не может напечатать отказ, а по свойству (Машина.9) не может напечатать внешнее действие. Значит машина ничего не печатает. Если после этого нажимается кнопка "A", машина, по свойствам (Машина.4) и (Машина.5) должна либо напечатать отказ A, либо напечатать действие, которое, по свойству (Машина.9), должно принадлежать A.

Если кнопка "A" нажимается до перехода машины в стабильное состояние, то, по свойствам (Машина.3) и (Машина.9) машина либо печатает действие, принадлежащее A, либо сначала переходит в стабильное состояние, где, как сказано выше, либо печатает отказ A, либо печатает действие, принадлежащее A.

В любом случае при нажатии кнопки стимула ?x машина печатает стимул ?x или его блокировку {?x}, а при нажатии кнопки всех реакций машина печатает какую-нибудь реакцию !y или стационарность δ. Таким образом, если трасса μ не заканчивается и не продолжается символом γ, то после неё наблюдается каждый стимул или его блокировка, а также какая-нибудь реакция или стационарность. По свойству (Машина.7), это и означает конвергентность трассы μ.

<u>$\beta\gamma\delta$-полнота ($\beta\gamma\delta$5)</u>. Покажем, что это следует из свойств (Машина.3,4,5,6,7).
Пусть $\mu\cdot\langle o\rangle\cdot\lambda\in\boldsymbol{\Sigma}$, где o – отказ. Тогда $\langle o\rangle\cdot\lambda\in(\boldsymbol{\Sigma}\ \textbf{\textit{after}}\ \mu)$ и, по свойству (Машина.7), после трассы μ может наблюдаться трасса $\langle o\rangle\cdot\lambda$. В частности, может наблюдаться отказ o. Тогда, по свойству (Машина.3), после трассы μ машина может оказаться в стабильном состоянии s, после чего возможно наблюдение отказа o и далее трассы λ: $\lambda\in\boldsymbol{\Sigma}(s,o)$.

a) Если среди трасс $\boldsymbol{\Sigma}(s)$ нет трассы $\langle ?x\rangle$, значит, по свойству (Машина.4), в стабильном состоянии s при нажатой кнопке стимула ?x машина не может принять стимул ?x.

Но тогда, по свойству (Машина.5), в состоянии s при нажатой кнопке стимула ?x наблюдается блокировка $\langle\{?x\}\rangle\in\mathbf{\Sigma}(s)$.

По свойству (Машина.6), $\mathbf{\Sigma}(s,o)=\mathbf{\Sigma}(s)=\mathbf{\Sigma}(s,\{?x\})$.

Поэтому $\lambda\in\mathbf{\Sigma}(s,o) \Rightarrow \lambda\in\mathbf{\Sigma}(s,\{?x\}) \Rightarrow \langle\{?x\}\rangle\cdot\lambda\in\mathbf{\Sigma}(s)$

$\Rightarrow\langle\{?x\}\rangle\cdot\lambda\in\mathbf{\Sigma}(s,o) \Rightarrow \langle o\rangle\cdot\langle\{?x\}\rangle\cdot\lambda\in\mathbf{\Sigma}(s)$.

Тем самым, трасса $\langle o\rangle\cdot\langle\{?x\}\rangle\cdot\lambda$ может наблюдаться сразу после трассы μ.

По свойству (Машина.7), $\langle o\rangle\cdot\langle\{?x\}\rangle\cdot\lambda\in(\mathbf{\Sigma}\ \textbf{\textit{after}}\ \mu)$, то есть $\mu\cdot\langle o\rangle\cdot\langle\{?x\}\rangle\cdot\lambda\in\mathbf{\Sigma}$.

b) Аналогично, если среди трасс $\mathbf{\Sigma}(s)$ нет ни одной трассы $\langle !y\rangle$, значит, по свойству (Машина.4), в стабильном состоянии s при нажатой кнопке приёма всех реакций машина не может выдать ни одну реакцию !y.

Но тогда, по свойству (Машина.5), в состоянии s при нажатой кнопке приёма всех реакций наблюдается стационарность $\langle\delta\rangle\in\mathbf{\Sigma}(s)$.

По свойству (Машина.6), $\mathbf{\Sigma}(s,o)=\mathbf{\Sigma}(s)=\mathbf{\Sigma}(s,\delta)$.

Поэтому $\lambda\in\mathbf{\Sigma}(s,o) \Rightarrow \lambda\in\mathbf{\Sigma}(s,\delta) \Rightarrow \langle\delta\rangle\cdot\lambda\in\mathbf{\Sigma}(s)$

$\Rightarrow\langle\delta\rangle\cdot\lambda\in\mathbf{\Sigma}(s,o) \Rightarrow \langle o\rangle\cdot\langle\delta\rangle\cdot\lambda\in\mathbf{\Sigma}(s)$.

Тем самым, трасса $\langle o\rangle\cdot\langle\delta\rangle\cdot\lambda$ может наблюдаться сразу после трассы μ.

По свойству (Машина.7), $\langle o\rangle\cdot\langle\delta\rangle\cdot\lambda\in(\mathbf{\Sigma}\ \textbf{\textit{after}}\ \mu)$, то есть $\mu\cdot\langle o\rangle\cdot\langle\delta\rangle\cdot\lambda\in\mathbf{\Sigma}$.

2. Доказательство Утверждение 2:

Если $\mathbf{T}$ является βγδ-моделью, а $\langle\gamma\rangle\notin\mathbf{T}$, то пустая трасса не продолжается разрушением и поэтому конвергентна, что влечёт $\mathbf{T}\downarrow$.

Покажем, что из условия $\langle\gamma\rangle\in\mathbf{T}\ \vee\ \mathbf{T}\downarrow$ следует, что $\mathbf{T}$ является βγδ-моделью.

Любая трасса из $\mathbf{T}$, отличная от $\langle\gamma\rangle$, имеет вид $\langle z\rangle\cdot\sigma\in\mathbf{T}$, где $z\in C$.

βγδ-допустимость (βγδ1): Пустая трасса допустима. Поскольку в квази-βγδ-модели разрушение ничем не продолжается, трасса $\langle\gamma\rangle$ также допустима.

Достаточно показать, что трасса $\langle z\rangle\cdot\sigma$ допустима, а это следует из допустимости трассы σ в βγδ-модели $\mathbf{T}$ ***after*** $\langle z\rangle$.

βγδ-согласованность (βγδ2): Пустая трасса и трасса $\langle\gamma\rangle$ согласованы, а согласованность трассы $\langle z\rangle\cdot\sigma\in\mathbf{T}$ следует из согласованности трассы $\sigma\in\mathbf{T}$ в βγδ-модели $\mathbf{T}$ ***after*** $\langle z\rangle$.

βγδ-конвергентность (βγδ3): Конвергентность трассы $\langle z\rangle\cdot\sigma\in\mathbf{T}$ следует из конвергентности трассы $\sigma\in\mathbf{T}$ в βγδ-модели $\mathbf{T}$ ***after*** $\langle z\rangle$. Пустая трасса должна быть конвергентна, только если нет трассы $\langle\gamma\rangle$, а в этом случае $\mathbf{T}$ конвергентно, что влечёт конвергентность пустой трассы.

βγδ-замкнутость (βγδ4): $\boldsymbol{DRT}(\epsilon)=\{\epsilon\}$, $\boldsymbol{DRT}(\langle\gamma\rangle)=\{\langle\gamma\rangle\}$.
Для трассы $\langle z\rangle\cdot\sigma\in\mathbf{T}$, по βγδ-замкнутости βγδ-модели $\mathbf{T}$ ***after*** $\langle z\rangle$, $\boldsymbol{DRT}(\sigma)\subseteq(\mathbf{T}$ ***after*** $\langle z\rangle)$.
Следовательно, $\boldsymbol{DRT}(\langle z\rangle\cdot\sigma)=\{\langle z\rangle\}\cdot\boldsymbol{DRT}(\sigma)\subseteq\{\langle z\rangle\}\cdot(\mathbf{T}$ ***after*** $\langle z\rangle)\subseteq\mathbf{T}$.

βγδ-полнота (βγδ5): $\boldsymbol{Ins}(\epsilon)=\{\epsilon\}$, $\boldsymbol{Ins}(\langle\gamma\rangle)=\{\langle\gamma\rangle\}$.
Выполнение этого свойства для трассы $\langle z\rangle\cdot\sigma\cdot\lambda\in\mathbf{T}$, где σ заканчивается отказом, следует из его выполнения для трассы $\sigma\cdot\lambda$ в βγδ-модели $\mathbf{T}$ ***after*** $\langle z\rangle$.

3. Доказательство Утверждение 3:

Сначала покажем, что $O^*\subseteq\mathbf{T}$.

Будем вести доказательство от противного.

Пусть есть трасса $o\in O^*\setminus\mathbf{T}$.
Выберем трассу o минимальной по длине среди всех таких трасс.
Трасса o не пуста, так как пустая трасса принадлежит любому дереву.
Тогда трассу можно представить в виде $o=o`\cdot\langle o\rangle$, где трасса $o`\in O^*\cap\mathbf{T}$, а отказ $o\in O\setminus\boldsymbol{head}(\mathbf{T}$ ***after*** $o`)$.
По свойству (стаб.1), $\mathbf{T}$ ***after*** $o` = \mathbf{T}$.

Следовательно, $o \in O \setminus \mathit{head}(T) = (\mathit{head}(T) \cap \beta\delta(C)) \setminus \mathit{head}(T) = \varnothing$, чего быть не может.

Теперь покажем, что любая трасса o отказов из T принадлежит O^*.
Будем вести доказательство от противного.

Пусть есть трасса $o \in T \setminus O^*$.
Выберем трассу o минимальной по длине среди всех таких трасс.
Тогда трассу можно представить в виде $o = o` \cdot \langle o \rangle$, где $o` \in O^* \cap T$, а отказ $o \in (\mathit{head}(T \ \mathit{after}\ o`) \cap \beta\delta(C)) \setminus O$.
По свойству (стаб.1), $T \ \mathit{after}\ o` = T$.
Следовательно

$$o \in (\mathit{head}(T) \cap \beta\delta(C)) \setminus O$$
$$= ((\mathit{head}(T) \cap \beta\delta(C)) \setminus (\mathit{head}(T) \cap \beta\delta(C)) = \varnothing),$$

чего быть не может.

Покажем, что $T \subseteq O^* \cdot T_C$.

Любая трасса из T может быть представлена в виде $o \cdot \lambda \in T$, где $o \in \beta\delta(C)^*$, а трасса λ не начинается с отказа.
Поскольку любая трасса o отказов из T принадлежит O^*, $o \in O^*$.
$o \cdot \lambda \in T$ влечёт $\lambda \in (T \ \mathit{after}\ o)$.
По свойству (стаб.1), $T \ \mathit{after}\ o = T$.
Поэтому $\lambda \in T$ и, поскольку λ не начинается с отказа, $\lambda \in T_C$.
Тем самым, $o \cdot \lambda \in O^* \cdot T_C$.

Покажем, что $O^* \cdot T_C \subseteq T$.

Пусть трасса $o \cdot \lambda \in O^* \cdot T_C$, где $o \in O^*$, $\lambda \in T_C$.
Поскольку $O^* \subseteq T$, то $o \in T$.
Поскольку $\lambda \in T_C$, а $T_C \subseteq T$, то $\lambda \in T$.
По свойству (стаб.1), $T \ \mathit{after}\ o = T$.
Значит, $\lambda \in (T \ \mathit{after}\ o)$.
Тем самым, $o \cdot \lambda \in T$.

4. Доказательство Утверждение 4:

βγδ-допустимость (βγδ1): Если трасса $\mu \cdot \sigma \in \Sigma$ не содержит γ «внутри», то очевидно, её постфикс $\sigma \in (\Sigma \ \mathit{after}\ \mu)$ также не содержит γ «внутри».

βγδ-согласованность (βγδ2): Если в трассе $\mu\cdot\sigma\in\mathbf{\Sigma}$ за блокировкой стимула не следует этот стимул, а за стационарностью не следует реакция, то, очевидно, это верно и для её постфикса $\sigma\in(\mathbf{\Sigma}\ \textbf{\textit{after}}\ \mu)$.

βγδ-конвергентность (βγδ3): Если трасса $\sigma\in(\mathbf{\Sigma}\ \textbf{\textit{after}}\ \mu)$ не заканчивается и не продолжается разрушением в дереве $\mathbf{\Sigma}\ \textbf{\textit{after}}\ \mu$, то, очевидно, трасса $\mu\cdot\sigma\in\mathbf{\Sigma}$ не заканчивается и не продолжается разрушением в дереве $\mathbf{\Sigma}$. По βγδ-конвергентности дерева $\mathbf{\Sigma}$, трасса $\mu\cdot\sigma\in\mathbf{\Sigma}$ продолжается в дереве $\mathbf{\Sigma}$ каждым стимулом или его блокировкой, а также какой-нибудь реакцией или стационарностью. Но тогда, очевидно, это верно и для её постфикса $\sigma\in(\mathbf{\Sigma}\ \textbf{\textit{after}}\ \mu)$ в дереве $\mathbf{\Sigma}\ \textbf{\textit{after}}\ \mu$.

βγδ-замкнутость (βγδ4): Для любой трассы $\sigma\in(\mathbf{\Sigma}\ \textbf{\textit{after}}\ \mu)$ имеет место: $\{\mu\}\cdot\textbf{\textit{DRT}}(\sigma)\subseteq\textbf{\textit{DRT}}(\mu\cdot\sigma)$. Поскольку $\mu\cdot\sigma\in\mathbf{\Sigma}$, по βγδ-замкнутости дерева $\mathbf{\Sigma}$, $\textbf{\textit{DRT}}(\mu\cdot\sigma)\subseteq\mathbf{\Sigma}$. Поэтому $\{\mu\}\cdot\textbf{\textit{DRT}}(\sigma)\subseteq\mathbf{\Sigma}$, что влечёт $\textbf{\textit{DRT}}(\sigma)\subseteq(\mathbf{\Sigma}\ \textbf{\textit{after}}\ \mu)$.

βγδ-полнота (βγδ5): Пусть трасса $\sigma\cdot\lambda\in(\mathbf{\Sigma}\ \textbf{\textit{after}}\ \mu)$, где σ заканчивается отказом. Тогда трасса $\mu\cdot\sigma$ в дереве $\mathbf{\Sigma}$ также заканчивается отказом. По βγδ-полноте дерева $\mathbf{\Sigma}$, если трасса $\mu\cdot\sigma$ в дереве $\mathbf{\Sigma}$ a) не продолжается стимулом $?x$ или b) не продолжается никакой реакций $!y$, то между трассами $\mu\cdot\sigma$ и λ можно вставить, оставаясь в рамках дерева $\mathbf{\Sigma}$, a) блокировку стимула $\{?x\}$ или b) стационарность δ. Но тогда, очевидно, это можно сделать между трассами σ и λ, оставаясь в рамках дерева $\mathbf{\Sigma}\ \textbf{\textit{after}}\ \mu$.

5. Доказательство Утверждение 5:

Пусть для $C\subseteq Z$ дерево $\mathbf{\Sigma}\in\textbf{\textit{MODEL}}_{\beta\gamma\delta}(C)$.
Наибольшее контрстабильное поддерево состоит из всех трасс $\mathbf{\Sigma}$, начинающихся не с отказов:
$\mathbf{\Sigma}_C=\{\lambda\in\mathbf{\Sigma}\mid\lambda\neq\epsilon\ \Rightarrow\ \lambda(1)\in C_\gamma\}=\{\epsilon\}\cup\{\langle z\rangle\cdot\lambda`\in\mathbf{\Sigma}\mid z\in C_\gamma\ \&\ \lambda`\in C_{\beta\gamma\delta}{}^*\}$.

1. Сначала покажем, что $\mathbf{\Sigma}_C$ после каждого стимула или реакции является βγδ-моделью.

   Очевидно, $\forall z\in C_\gamma\ \mathbf{\Sigma}_C\ \textbf{\textit{after}}\ \langle z\rangle = \mathbf{\Sigma}\ \textbf{\textit{after}}\ \langle z\rangle$.

   По Утверждение 4:, если $z\neq\gamma$, то $\mathbf{\Sigma}\ \textbf{\textit{after}}\ \langle z\rangle$ является βγδ-моделью.

   Следовательно, $\mathbf{\Sigma}_C\ \textbf{\textit{after}}\ \langle z\rangle$ является βγδ-моделью.

2. Теперь покажем, что в $\mathbf{\Sigma}_c$ разрушение ничем не продолжается.

Очевидно, $\langle\gamma\rangle\cdot\lambda\in\mathbf{\Sigma}_c \Rightarrow \langle\gamma\rangle\cdot\lambda\in\mathbf{\Sigma}$.

По βγδ-допустимости дерева $\mathbf{\Sigma}$, $\langle\gamma\rangle\cdot\lambda\in\mathbf{\Sigma} \Rightarrow \lambda=\epsilon$.

Следовательно, $\langle\gamma\rangle\cdot\lambda\in\mathbf{\Sigma}_c \Rightarrow \lambda=\epsilon$.

Из 1 и 2 следует, что $\mathbf{\Sigma}_c$ является квази-βγδ-моделью.

6. Доказательство Утверждение 6:

Если $O=\varnothing$, то ***head***$(\mathbf{T})$ содержит все стимулы и , по крайней мере, одну реакцию.
Следовательно, $\mathbf{T}\downarrow$.
Поскольку $\gamma\notin$ ***head***$(\mathbf{T})$, по Утверждение 2:, $\mathbf{T}\in$ ***MODEL***$_{\beta\gamma\delta}(C)$.
Если $O=\varnothing$, то $O^*\cdot\mathbf{T}=\mathbf{T}$ и, тем самым, утверждение доказано.

Пусть $O\neq\varnothing$. Тогда любая трасса из $O^*\cdot\mathbf{T}$ имеет вид $o\in O^*$ или $o\cdot\langle z\rangle\cdot\sigma$, где $z\in C$ и $\langle z\rangle\cdot\sigma\in\mathbf{T}$.

βγδ-допустимость (βγδ1): Трасса o, как любая трасса отказов, допустима. По βγδ-допустимости дерева $\mathbf{T}$ ***after*** $\langle z\rangle$, трасса σ не содержит γ «внутри». Тогда конкатенация $o\cdot\langle z\rangle\cdot\sigma$ также не содержит γ «внутри».

βγδ-согласованность (βγδ2): Трасса o, как любая трасса отказов, согласована. Поскольку дерево $\mathbf{T}$ ***after*** $\langle z\rangle$ βγδ-согласовано, трасса σ согласована. Тем самым, конкатенация $o\cdot\langle z\rangle\cdot\sigma$ согласована.

βγδ-конвергентность (βγδ3): Трасса o конвергентна, поскольку ***head***$(O^*\cdot\mathbf{T}$ ***after*** $o) = O\cup$***head***$(\mathbf{T})$, а это множество, по определению O, содержит каждый стимул или его блокировку и какую-нибудь реакцию или стационарность. Поскольку $\mathbf{T}$ ***after*** $\langle z\rangle$ βγδ-модель, трасса σ, не заканчивающаяся и не продолжающаяся разрушением, конвергентна. Тем самым, конкатенация $o\cdot\langle z\rangle\cdot\sigma$ конвергентна.

βγδ-замкнутость (βγδ4). ***DRT***$(o)\subseteq O^*\subseteq O^*\cdot\mathbf{T}$.
По βγδ-замкнутости $\mathbf{T}$ ***after*** $\langle z\rangle$, ***DRT***$(o\cdot\langle z\rangle\cdot\sigma) =$
DRT$(o)\cdot\{\langle z\rangle\}\cdot$***DRT***$(\sigma) \subseteq O^*\cdot\{\langle z\rangle\}\cdot(\mathbf{T}$ ***after*** $\langle z\rangle)\subseteq O^*\cdot\mathbf{T}$.

βγδ-полнота (βγδ5). По определению O, трасса o не продолжается стимулом $?x$ или не продолжается никакой реакцией тогда и только тогда, когда $\{?x\}\in O$ или $\delta\in O$. В этом случае для любой трассы $o\cdot\lambda\in O^*\cdot T$ трасса $o\cdot\langle\{?x\}\rangle\cdot\lambda\in O^*\cdot T$ или $o\cdot\langle\delta\rangle\cdot\lambda\in O^*\cdot T$.

Поскольку дерево T ***after*** $\langle z\rangle$ βγδ-полно, полнота в нём его трассы σ влечёт выполнение этого свойства для трассы $o\cdot\langle z\rangle\cdot\sigma$ в дереве $O^*\cdot T$.

7. Доказательство Утверждение 7:

По Утверждение 5:, наибольшее контрстабильное дерево Σ_c является квази-βγδ-моделью.

Пусть трасса $o\cdot\lambda\in\Sigma$, где o – непустая трасса отказов, а трасса λ не начинается с отказа: $o=RefH(o\cdot\lambda)\neq\epsilon$. Для этой трассы мы построим максимальное стабильное поддерево $T\subseteq\Sigma$, содержащее эту трассу $o\cdot\lambda\in T$, и такое, что его наибольшее контрстабильное поддерево совпадает с наибольшим контрстабильным поддеревом дерева Σ после трассы отказов o: $T_c=(\Sigma\ \textit{after}\ o)_c$. Заметим, что, по свойству βγδ-согласованности, $(\Sigma\ \textit{after}\ o)_c$ не содержит трассу $\langle\gamma\rangle$. (см. Рис.107)

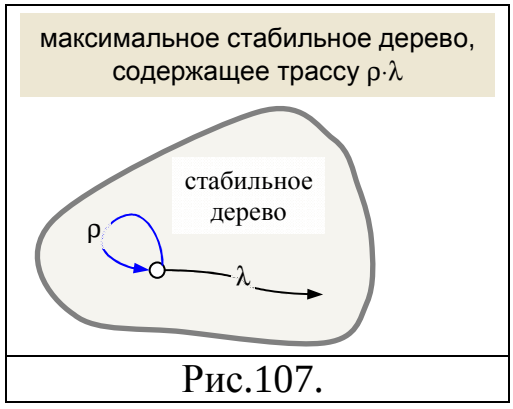


Рис.107.

Обозначим $O=Im(o)$. По свойству βγδ-замкнутости (βγδ4) дерева Σ, любая трасса μ, продолжающая в Σ трассу o, продолжает в Σ каждую трассу из O^*: $O^*\cdot(\Sigma\ \textit{after}\ o)\subseteq\Sigma$.

Обозначим $T(O)=O^*\cdot(\Sigma\ \textit{after}\ o)$. Очевидно, $o\cdot\lambda\in T(O)$. По построению, $T(O)$ – дерево. Имеем $T(O)_c=(O^*\cdot(\Sigma\ \textit{after}\ o))_c=(\Sigma\ \textit{after}\ o)_c$. По построению, $\forall o`\in O^*\ T(O)\ \textit{after}\ o` = T(O)$, то есть, почти выполнено свойство (стаб.1): оно может быть не выполнено только для трассы отказов $o`\notin O^*$.

Так как o – непустая трасса отказов, то, по βγδ-согласованности дерева Σ, трасса o не продолжается разрушением. А тогда, по βγδ-конвергентности дерева Σ, трасса o продолжается каждым стимулом или его блокировкой,

и продолжается какой-нибудь реакцией или δ. Отсюда следуют следующие два свойства для O^*:

· $\forall ?x \in C\ O^* \cdot \{\langle ?x \rangle\} \subseteq \mathbf{T}(O) \vee O^* \cdot \{\langle \{?x\} \rangle\} \subseteq \mathbf{T}(O)$;

· $\exists !y \in C\ O^* \cdot \{\langle !y \rangle\} \subseteq \mathbf{T}(O) \vee O^* \cdot \{\langle \delta \rangle\} \subseteq \mathbf{T}(O)$.

Поскольку $\mathbf{T}(O)$ ***after*** $o = \mathbf{T}(O)$, эти два свойства эквивалентны следующим двум свойствам:

· $\forall ?x \in C\ \langle ?x \rangle \in \mathbf{T}(O) \vee \langle \{?x\} \rangle \in \mathbf{T}(O)$;

· $\exists !y \in C\ \langle !y \rangle \in \mathbf{T}(O) \vee \langle \delta \rangle \in \mathbf{T}(O)$.

Поскольку $\mathbf{T}(O)$ ***after*** $o = \mathbf{T}(O)$, а трасса o не продолжается разрушением, $\langle \gamma \rangle \notin \mathbf{T}(O)$.

Таким образом, дерево $\mathbf{T}(O)$ конвергентно, то есть, выполнено свойство (стаб.2).

Мы видим, что дерево $\mathbf{T}(O)$ содержит исходную трассу $o \cdot \lambda$, $\mathbf{T}(O)_c = (\mathbf{\Sigma}$ ***after*** $o)_c$, и $\mathbf{T}(O)$ «почти стабильно»: в нём может быть не выполнено только свойство (стаб.3), а свойство (стаб.1) может быть выполнено частично: оно может быть не выполнено только для трасс отказов, не принадлежащих O^*.

Теперь преобразуем $\mathbf{T}(O)$ в стабильное дерево с сохранением остальных свойств. Для этого сначала определим «недостающие» отказы, которые можно добавить в O без изменения самого дерева (и, тем самым, без нарушения его свойств). (см. Рис.108)

«Недостающие» блокировки – это блокировки таких стимулов $?x \in C$, что $\{?x\} \notin O$, а трассы отказов из O не продолжаются стимулами $?x$ и продолжаются их блокировками $\{?x\}$:

$$\beta_{miss}(O) = \{\ \{?x\} \mid\ ?x \in C\ \ \&\ \{?x\} \notin O\ \ \&\ \langle ?x \rangle \notin \mathbf{T}(O)\ \ \&\ \langle \{?x\} \rangle \in \mathbf{T}(O)\ \}.$$

Аналогично, стационарность δ «недостающая», если $\delta \notin O$, а трассы отказов из O не продолжаются реакциями и продолжаются δ:

$$\delta_{miss}(O) = \{\ \delta \mid \delta \notin O\ \&\ \forall !y \in !C\ \ \langle !y \rangle \notin \mathbf{T}(O)\ \&\ \langle \delta \rangle \in \mathbf{T}(O)\ \}.$$

Определим $O` = O \cup \beta_{miss}(O) \cup \delta_{miss}(O)$.
По свойству $\beta\gamma\delta$-полноты ($\beta\gamma\delta 5$) дерева $\mathbf{\Sigma}$, $\mathbf{T}(O`) = \mathbf{T}(O)$.
(Заметим, что $O`^* = \cup_\circ$***Ins***$(O^*) = \cup_\circ$***DRTIns***$(O^*) =$ ***DRTIns***(o) в дереве $\mathbf{\Sigma}$.)

Для $\mathbf{T}(O`)$, по-прежнему, $o{\cdot}\lambda\in\mathbf{T}(O`)$, $\mathbf{T}(O`)_c=(\boldsymbol{\Sigma}\ \textit{after}\ o)_c$, выполнено свойство (стаб.2) и почти выполнено свойство (стаб.1): $\forall o`\in O`^*\ \mathbf{T}(O`)\ \textit{after}\ o` = \mathbf{T}(O`)$.
Но для $\mathbf{T}(O`)$ множества «недостающих» отказов $\beta_{miss}(O`)$ и $\delta_{miss}(O`)$ пусты.

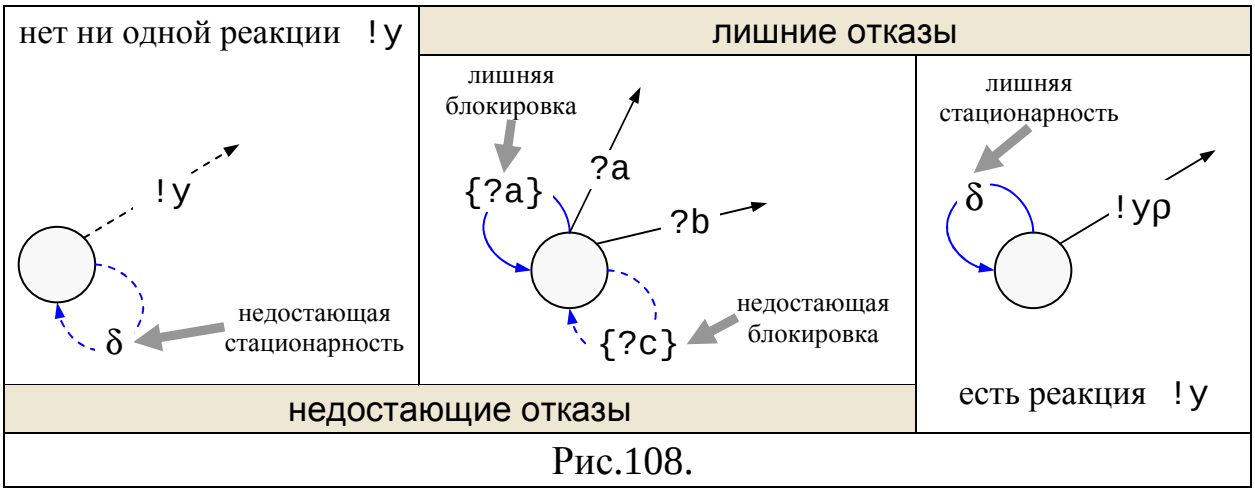


Рис.108.

Дерево $\mathbf{T}(O`)$ может содержать «лишние» отказы, из-за которых нарушается свойство (стаб.3). «Лишняя» трасса – это любая трасса, начинающаяся с «лишнего» отказа. (см. Рис.108)

«Лишняя» блокировка – это блокировка такого стимула $?x\in C$, что трассы отказов из $O`^*$ продолжаются как этим стимулом $?x$, так и его блокировкой $\{?x\}$. По свойству βγδ-согласованности (βγδ2) дерева $\boldsymbol{\Sigma}$, такая «лишняя» блокировка $\{?x\}\notin O`$. Определим множество «лишних» трасс для «лишних» блокировок:

$$\mathbf{T}\beta(O`) = \{\ \langle\{?x\}\rangle{\cdot}\mu\ \mid\ ?x\in C\ \&\ \{?x\}\notin O`\ \&\ \langle ?x\rangle\in\mathbf{T}(O`)\ \&\ \langle\{?x\}\rangle\in\mathbf{T}(O`)\ \&\ \mu\in C_{\beta\gamma\delta}{}^*\ \}.$$

Аналогично стационарность δ «лишняя», если трассы отказов из $O`^*$ продолжаются как реакциями, так и стационарностью $\delta$. По свойству βγδ-согласованности (βγδ2) дерева $\boldsymbol{\Sigma}$, такая «лишняя» стационарность $\delta\notin O`$. Определим множество «лишних» трасс для «лишней» стационарности:

$$\mathbf{T}\delta(O`) = \{\ \langle\delta\rangle{\cdot}\mu\ \mid\ \delta\notin O`\ \&\ \exists!y\in!C\ \langle!y\rangle\in\mathbf{T}(O`)\ \&\ \langle\delta\rangle\in\mathbf{T}(O`)\ \&\ \mu\in C_{\beta\gamma\delta}{}^*\ \}.$$

Удалим продолжение $O`^*$ «лишними» трассами:
$\mathbf{T}=\mathbf{T}(O`)\setminus(O`^*{\cdot}(\mathbf{T}\beta(O`)\cup\mathbf{T}\delta(O`)))$.

Трасса $o \cdot \lambda$ не удаляется, так как λ начинается не с отказа, а o содержит только такие отказы, которые есть в $O`$, в то время как удаляемая трасса начинается с последовательности отказов, содержащей «лишний» отказ, который не принадлежит $O`$. Таким образом $o \cdot \lambda \in \mathbf{T}$.

Множество $\mathbf{T}$ является деревом по построению.

Поскольку удаляемые трассы начинаются с отказа, остаётся верным $\mathbf{T}_c = (\Sigma \ \textit{after} \ o)_c$.

Поскольку мы удаляем множество трасс вида $O`^* \cdot \ldots$, свойство (стаб.1) продолжает почти выполняться: $\forall o \in O`^* \ \mathbf{T} \ \textit{after} \ o = \mathbf{T}$. Более того, поскольку удалённые «лишние» отказы не принадлежали $O`$, теперь любая трасса отказов из $\mathbf{T}$ содержится в $O`^*$. Поэтому свойство (стаб.1) выполняется полностью:
$\forall o \in \mathbf{T} \cap (\beta\delta(C)^*) \ \mathbf{T} \ \textit{after} \ o = \mathbf{T}$.

Каждая удаляемая трасса имеет вид $o` \cdot \langle \{?x\} \rangle \cdot \ldots$, где $o` \in O`^*$, $\{?x\} \notin O`$ и $\langle ?x \rangle \in \mathbf{T}$, или вид $o` \cdot \langle \delta \rangle \cdot \ldots$, где $o` \in O`^*$, $\delta \notin O`$ и $\exists ! y \in C \ \langle !y \rangle \in \mathbf{T}$.
Иными словами, удаляются блокировки стимулов, если есть сами стимулы, и удаляется стационарность, если есть хотя бы одна реакция. Поэтому свойство конвергентности (стаб.2) сохраняется.

Наконец, теперь у нас нет «лишних» трасс, нарушающих свойство (стаб.3).

Итак, $\mathbf{T}$ – стабильное дерево, содержащее исходную трассу $o \cdot \lambda$ и $\mathbf{T}_c = (\Sigma \ \textit{after} \ o)_c$.

Теперь покажем, что дерево $\mathbf{T}$ является βγδ-моделью.

По Утверждение 4:, $\Sigma \ \textit{after} \ o$ является βγδ-моделью. По Утверждение 5:, $(\Sigma \ \textit{after} \ o)_c$ является квази-βγδ-моделью. По Утверждение 3:, $\mathbf{T} = O`^* \cdot \mathbf{T}_c$. Поскольку $\mathbf{T}$ стабильное дерево, для конкатенации $\mathbf{T} = O`^* \cdot \mathbf{T}_c$ выполнено условие
$O` = \{\{?x\} \mid ?x \in C \setminus \textit{head}(\mathbf{T}_c)\} \cup \{\delta \mid !C \cap \textit{head}(\mathbf{T}_c) = \varnothing\}$.
Поскольку $\mathbf{T}_c = (\Sigma \ \textit{after} \ o)_c$, $\mathbf{T}_c$ является квази-βγδ-моделью и не содержит трассы $\langle \gamma \rangle$. Тем самым, выполнены условия Утверждение 6: для $O`^* \cdot \mathbf{T}_c$ и $\mathbf{T}$ является βγδ-моделью.

Теперь мы можем построить для каждой трассы из Σ, начинающейся с отказов, стабильную βγδ-модель, содержащую эту трассу. Множество таких βγδ-моделей обозначим Σ_s. Теперь, очевидно, $\Sigma=\Sigma_c\cup(\cup(\Sigma_s))$, что и требовалось доказать.

8. Доказательство Утверждение 8:

Утверждение непосредственно следует из определения операций ***DRTIns***.

Можно также заметить, что в доказательстве Утверждение 7: для каждой трассы $o\cdot\lambda\in\Sigma$, где o – непустая трасса отказов, а трасса λ не начинается с отказа: $o=\mathit{RefH}(o\cdot\lambda)\neq\epsilon$, мы строили дерево T со свойствами:

$\mathit{DRTIns}(o)=O`^*$,
$T_c=(\Sigma\ \mathit{after}\ o)_c$,
$O` = \{\{?x\}|?x\in C\setminus\mathit{head}(T_c)\} \cup \{\delta|!C\cap\mathit{head}(T_c)=\varnothing\}$.

Напомним, что $\mathit{head}(T_c)=C\cap\mathit{head}(T)$.

Отсюда, для $\lambda=\epsilon$, мы также имеем

$$\mathit{DRTIns}(o) = (\{ \{?x\} \mid ?x\in C\setminus\mathit{head}(\Sigma\ \mathit{after}\ o) \} \cup \{ \delta \mid !C\cap\mathit{head}(\Sigma\ \mathit{after}\ o)=\varnothing \})^*.$$

9. Доказательство Утверждение 9:

Пусть в алфавите $C\subseteq Z$ задана модель безопасных трасс T. Обозначим её пополнение до конвергентности через Σ. Очевидно, оно является деревом трасс.

Сохранение безопасных трасс непосредственно следует из определения пополнения до конвергентности: $\mathit{safe}(\Sigma)=T$.

Нам надо показать, что дерево Σ удовлетворяет всем требованиям βγδ-модели, кроме, быть может, βγδ-замкнутости.

<u>βγδ-допустимость.</u>
В T нет разрушения, а в добавленных трассах разрушение может быть только последним символом.

<u>βγδ-согласованность.</u>
Трассы T согласованы. Добавленная трасса $\sigma\cdot\langle?x\rangle$ или $\sigma\cdot\langle!y\rangle$ может быть не согласована только тогда, когда $\sigma=\mu\cdot\langle\{?x\}\rangle\cdot o$ или $\sigma=\mu\cdot\langle\delta\rangle\cdot o$, соответственно, где $o\in\beta\delta(C)^*$. Но тогда, по условию добавления трассы

$\sigma\cdot\langle ?x\rangle$ или $\sigma\cdot\langle !y\rangle$ должно быть $\mu\cdot\langle\{?x\}\rangle\cdot o\uparrow\{?x\}$ или $\mu\cdot\langle\delta\rangle\cdot o\uparrow\delta$, соответственно. По βδ-конвергентности **T**, в этом случае $\mu\uparrow\{?x\}$ или $\mu\uparrow\delta$, что, очевидно, не верно. Следовательно, добавленные трассы вида $\sigma\cdot\langle ?x\rangle$ и $\sigma\cdot\langle !y\rangle$ согласованы. А тогда добавленные трассы вида $\sigma\cdot\langle ?x,\gamma\rangle$ и $\sigma\cdot\langle !y,\gamma\rangle$ также согласованы.

βγδ-конвергентность.
Из условия добавления трасс следует, что после добавления трассы **T** конвергентны. Добавленные трассы заканчиваются или продолжаются разрушением.

βγδ-полнота.

Пусть трасса $\mu\cdot\langle l\rangle\cdot\lambda\in\mathbf{T}$ имеет префикс $\mu\cdot\langle l\rangle$, заканчивающий отказом l, и этот префикс в **T** продолжается отказом r, но не продолжается каким-либо символом, принадлежащим r. Тогда, по βδ-полноте **T**, требуемое свойство выполнено: отказ r можно вставить в трассу после префикса $\mu\cdot\langle l,r\rangle\cdot\lambda\in\mathbf{T}$. Очевидно, это будет верно и для добавленных трасс, когда λ продолжена $\langle ?x\rangle$, $\langle ?x,\gamma\rangle$, $\langle !y\rangle$ или $\langle !y,\gamma\rangle$.

Пусть теперь трасса $\mu\cdot\langle l\rangle\cdot\lambda\in\mathbf{T}$ имеет префикс $\mu\cdot\langle l\rangle$, заканчивающий отказом l, и этот префикс в **T** не продолжается ни отказом r, ни каким-либо символом, принадлежащим r. Тогда, очевидно, $\mu\cdot\langle l\rangle\uparrow r$, и после пополнения префикс $\mu\cdot\langle l\rangle$ будет продолжаться некоторым стимулом или реакцией $z\in r$. Очевидно, и в этом случае требуемое свойство будет верно и для добавленных трасс, когда λ продолжена $\langle ?x\rangle$, $\langle ?x,\gamma\rangle$, $\langle !y\rangle$ или $\langle !y,\gamma\rangle$.

10. **Доказательство Утверждение 10:**

Пусть в алфавите $C\subseteq Z$ задано дерево трасс **Σ**, удовлетворяющее всем свойствам βγδ-модели, кроме, быть может, βγδ-замкнутости, с множеством безопасных трасс $\mathbf{T}=\boldsymbol{safe}(\mathbf{\Sigma})$.

Покажем, что замыкание по ***DRT***-операциям сохраняет все остальные свойства βγδ-модели.

βγδ-допустимость.

DRT-операции не меняют подтрассы базовых символов, следовательно, разрушение остаётся последним символом трасс.

βγδ-согласованность.

Следует из того, что ***DRT***-операции не меняют подтрассы базовых символов и добавляют только те отказы, которые уже были в последовательности отказов между двумя базовыми символами (до или после всех базовых символов).

βγδ-конвергентность.

Если бы трасса $\sigma` \in \boldsymbol{DRT}(\sigma)$ в $\cup_\circ \boldsymbol{DRT}(\boldsymbol{\Sigma})$ не заканчивалась и не продолжалась разрушением, а также была бы o-дивергентна, то, очевидно, это было бы верно и для трассы σ в $\boldsymbol{\Sigma}$, что противоречит βγδ-конвергентности $\boldsymbol{\Sigma}$.

βγδ-полнота.

Пусть трасса $\sigma` \in \cup_\circ \boldsymbol{DRT}(\boldsymbol{\Sigma})$ заканчивается отказом и в $\cup_\circ \boldsymbol{DRT}(\boldsymbol{\Sigma})$ не продолжается стимулом $?x$ (не продолжается реакциями), и существует трасса $\sigma` \cdot \lambda` \in \cup_\circ \boldsymbol{DRT}(\boldsymbol{\Sigma})$. Представим трассу $\lambda`$ в виде $\lambda` = o \cdot \mu`$, где $o = \boldsymbol{RefH}(\lambda`)$. Тогда $\sigma` \cdot \lambda` = \sigma` \cdot o \cdot \mu`$, где трасса $\mu`$ не начинается с отказа. Тогда трасса $\sigma` \cdot o \in \cup_\circ \boldsymbol{DRT}(\boldsymbol{\Sigma})$ также заканчивается отказом и в $\cup_\circ \boldsymbol{DRT}(\boldsymbol{\Sigma})$ не продолжается стимулом $?x$ (не продолжается реакциями). Поскольку трасса $\mu`$ не начинается с отказа, существует такая трасса $\sigma \cdot \mu \in \boldsymbol{\Sigma}$, что $\sigma` \cdot o \in \boldsymbol{DRT}(\sigma)$ и $\mu` \in \boldsymbol{DRT}(\mu)$. Очевидно, трасса σ заканчивается отказом и в $\boldsymbol{\Sigma}$ не продолжается стимулом $?x$ (не продолжается реакциями). Тогда, по βγδ-полноте $\boldsymbol{\Sigma}$, $\sigma \cdot \langle o \rangle \cdot \mu \in \boldsymbol{\Sigma}$, где $o = \{?x\}$ $(o = \delta)$.

Поэтому $\sigma` \cdot o \cdot \langle o \rangle \cdot \mu` \in \cup_\circ \boldsymbol{DRT}(\boldsymbol{\Sigma}) \Rightarrow \sigma` \cdot \langle o \rangle \cdot o \cdot \mu` \in \cup_\circ \boldsymbol{DRT}(\boldsymbol{\Sigma})$

$\Rightarrow \sigma` \cdot \langle o \rangle \cdot \lambda` \in \cup_\circ \boldsymbol{DRT}(\boldsymbol{\Sigma})$.

Таким образом, $\cup_\circ \boldsymbol{DRT}(\boldsymbol{\Sigma})$ является βγδ-моделью.

Нам осталось показать, что замыкание по ***DRT***-операциям сохраняет множество безопасных трасс $\mathbf{T} = \boldsymbol{safe}_\circ \cup_\circ \boldsymbol{DRT}(\boldsymbol{\Sigma})$.

Сначала покажем, что старые безопасные трассы остаются безопасными: $\mathbf{T} \subseteq \boldsymbol{safe}_\circ \cup_\circ \boldsymbol{DRT}(\boldsymbol{\Sigma})$.

Допустим обратное: существует трасса $\sigma \in \mathbf{T}$, но $\sigma \notin \boldsymbol{safe}_\circ \cup_\circ \boldsymbol{DRT}(\boldsymbol{\Sigma})$.

Тогда либо существует трасса $\mu\cdot\langle ?x\rangle\le\sigma$ или $\mu\cdot\langle\{?x\}\rangle\le\sigma$ и трасса $\mu\cdot\langle ?x,\gamma\rangle\in\cup_\circ\boldsymbol{DRT}(\boldsymbol{\Sigma})$, либо существует трасса $\mu\cdot\langle !y\rangle\le\sigma$ или $\mu\cdot\langle\delta\rangle\le\sigma$ и трасса $\mu\cdot\langle !z,\gamma\rangle\in\cup_\circ\boldsymbol{DRT}(\boldsymbol{\Sigma})$ для некоторой реакции $!z\in C$. Заметим, что $\mu\le\sigma$ и $\sigma\in\mathbf{T}$ влечёт $\mu\in\mathbf{T}$. Поскольку перед и после разрушения не может быть отказов, должна существовать трасса $\lambda\cdot\langle ?x,\gamma\rangle\in\boldsymbol{\Sigma}$ или, соответственно, $\lambda\cdot\langle !z,\gamma\rangle\in\boldsymbol{\Sigma}$, и $\mu\in\boldsymbol{DRT}(\lambda)$. Очевидно, $\lambda\in\mathbf{T}$. Но тогда в дереве $\mathbf{T}$ $\lambda\uparrow\{?x\}$ или, соответственно, $\lambda\uparrow\delta$. Отсюда, по βδ-конвергентности $\mathbf{T}$, $\mu\uparrow\{?x\}$ или, соответственно, $\mu\uparrow\delta$. Следовательно, $\mu\cdot\langle ?x,\gamma\rangle\in\boldsymbol{\Sigma}$ или, соответственно, $\mu\cdot\langle !z,\gamma\rangle\in\boldsymbol{\Sigma}$. Отсюда $\sigma\notin\boldsymbol{safe}(\boldsymbol{\Sigma})=\mathbf{T}$.

Мы пришли к противоречию, и значит $\mathbf{T}\subseteq\boldsymbol{safe}_\circ\cup_\circ\boldsymbol{DRT}(\boldsymbol{\Sigma})$.

Теперь покажем, что новых безопасных трасс не появляется: $\mathbf{T}\supseteq\boldsymbol{safe}_\circ\cup_\circ\boldsymbol{DRT}(\boldsymbol{\Sigma})$.

Допустим обратное: существует трасса $\mu\in\boldsymbol{safe}_\circ\cup_\circ\boldsymbol{DRT}(\boldsymbol{\Sigma})$, но $\mu\notin\mathbf{T}$. Тогда существует такая трасса $\sigma\in\boldsymbol{\Sigma}$, что $\mu\in\boldsymbol{DRT}(\sigma)$. Поскольку добавленные трассы, то есть трассы из $\boldsymbol{\Sigma}\backslash\mathbf{T}$, заканчиваются или продолжаются разрушением, должно быть $\sigma\in\mathbf{T}$. По βδ-замкнутости $\mathbf{T}$, $\exists\lambda\ \exists u\ \lambda\cdot\langle u\rangle\le\mu\ \&\ \lambda\in\mathbf{T}\ \&\ \lambda\uparrow u$. Но тогда $\exists z\ \lambda\cdot\langle z,\gamma\rangle\in\boldsymbol{\Sigma}$. Следовательно, трасса μ не безопасна в $\boldsymbol{\Sigma}$ и, тем самым, в $\cup_\circ\boldsymbol{DRT}(\boldsymbol{\Sigma})$.

Мы пришли к противоречию, и значит $\mathbf{T}\supseteq\boldsymbol{safe}_\circ\cup_\circ\boldsymbol{DRT}(\boldsymbol{\Sigma})$.

Мы доказали, что $\mathbf{T}=\boldsymbol{safe}_\circ\cup_\circ\boldsymbol{DRT}(\boldsymbol{\Sigma})$.

11. Доказательство Утверждение 11:

Необходимость. Рассмотрим βγδ-модель $\boldsymbol{\Sigma}\in\boldsymbol{MODEL}_{\beta\gamma\delta}(C)$ и покажем, что подмножество её безопасных трасс $\mathbf{T}=\boldsymbol{safe}(\boldsymbol{\Sigma})$ удовлетворяет всем βδ-свойствам модели безопасных трасс.

βδ-стационарность.
Если реакций нет, то нет разрушающих реакций и поэтому стационарность безопасна после любой безопасной трассы. А тогда, по βγδ-конвергентности $\boldsymbol{\Sigma}$, продолжение безопасной трассы

стационарностью принадлежит $\mathbf{\Sigma}$ и, следовательно, является безопасной трассой в $\mathbf{\Sigma}$.

βδ-согласованность.
Безопасные трассы согласованы как и все трассы $\mathbf{\Sigma}$ (согласованность есть свойство отдельно взятой трассы).

βδ-конвергентность.
Пусть трасса $\sigma,\mu\in\mathbf{T}$ и $\sigma\in\boldsymbol{DRT}(\mu)$. Если μ o-дивергентна в $\mathbf{T}$, то для некоторого $z\in o$ будет $\mu\cdot\langle z,\gamma\rangle\in\mathbf{\Sigma}$. А тогда, в силу βγδ-замкнутости $\mathbf{\Sigma}$, будет $\sigma\cdot\langle z,\gamma\rangle\in\mathbf{\Sigma}$, то есть σ o-дивергентна в $\mathbf{T}$. Поэтому $\sigma\downarrow o$ в $\mathbf{T}$ влечёт $\mu\downarrow o$ в $\mathbf{T}$.

βδ-замкнутость.
По βγδ-замкнутости $\mathbf{\Sigma}$, для каждой безопасной трассы σ трасса $\mu\in\boldsymbol{DRT}(\sigma)$ принадлежит $\mathbf{\Sigma}$. Если $\mu\notin\mathbf{T}$, то есть не безопасна, то найдётся такой её префикс $\lambda\cdot\langle u\rangle\le\mu$, что $\lambda\cdot\langle z,\gamma\rangle\in\mathbf{\Sigma}$, где $z=u$, если u стимул или реакция, или $z\in u$, если u отказ. А тогда трасса λ u-дивергентна в поддереве безопасных трасс $\mathbf{T}$.

βδ-полнота.
По βγδ-полноте $\mathbf{\Sigma}$, $\mu\cdot\langle l,r\rangle\cdot\lambda\in\mathbf{\Sigma}$. Поскольку трасса $\mu\cdot\langle l\rangle\cdot\lambda$ безопасна, трасса $\mu\cdot\langle l,r\rangle\cdot\lambda$ может быть опасной только в том случае, когда в $\mathbf{\Sigma}$ отказ r опасен после префикса $\mu\cdot\langle l\rangle$, но именно этот случай исключён в условии βδ-полноты, поскольку $\mu\cdot\langle l,r\rangle\in\mathbf{T}$.

Достаточность непосредственно следует из Утверждение 9: и Утверждение 10:.

12. **Доказательство Утверждение 12:**

Пусть $C\subseteq Z$ и $\mathbf{\Sigma}\in\boldsymbol{Trees}(C_{\beta\gamma\delta})$. Нам надо показать, что $\boldsymbol{safe}\circ\boldsymbol{final}(\mathbf{\Sigma})=\boldsymbol{safe}(\mathbf{\Sigma})$.

Вложенность в одну сторону $\boldsymbol{safe}\circ\boldsymbol{final}(\mathbf{\Sigma})\subseteq\boldsymbol{safe}(\mathbf{\Sigma})$ следует из определения безопасных трасс: трасса, безопасная в дереве, безопасна в поддереве, которому она принадлежит.

Докажем обратную вложенность. Пусть трасса $\sigma \in \mathit{safe}(\Sigma)$. Тогда $\sigma \in \mathit{final}(\Sigma)$. Пусть трасса не безопасна в поддереве $\sigma \notin \mathit{safe} \circ \mathit{final}(\Sigma)$. Тогда существует трасса $\mu \cdot \langle z, \gamma \rangle \in \mathit{final}(\Sigma)$, где $\mu \in \mathit{safe} \circ \mathit{final}(\Sigma)$, $z \in C$ и $\sigma = \mu \cdot \langle z \rangle$ или $\sigma = \mu \cdot \langle z, \gamma \rangle$. Тогда $\mu \cdot \langle z, \gamma \rangle \in \Sigma$ и $\mu \in \mathit{safe}(\Sigma)$. Следовательно, $\sigma \notin \mathit{safe}(\Sigma)$. Поэтому, если $\sigma \in \mathit{safe}(\Sigma)$, то $\sigma \in \mathit{safe} \circ \mathit{final}(\Sigma)$.

13. Доказательство Утверждение 13:

Пусть $C \subseteq Z$ и $\Sigma \in \mathit{Trees}(C_{\beta\gamma\delta})$. Нам надо показать, что $\mathit{final} \circ \mathit{final}(\Sigma) = \mathit{final}(\Sigma)$.

Вложенность в одну сторону $\mathit{final} \circ \mathit{final}(\Sigma) \subseteq \mathit{final}(\Sigma)$ следует из вложенности $\mathit{safe} \circ \mathit{final}(\Sigma) \subseteq \mathit{safe}(\Sigma)$ и определения финальных трасс: трасса, финальная в дереве, финальна в поддереве, которому она принадлежит.

Докажем обратную вложенность. Пусть трасса $\sigma \in \mathit{final}(\Sigma)$. Если трасса безопасна в дереве $\sigma \in \mathit{safe}(\Sigma)$, то она тем более безопасна в поддереве $\sigma \in \mathit{safe} \circ \mathit{final}(\Sigma)$ и, следовательно, $\sigma \in \mathit{final} \circ \mathit{final}(\Sigma)$. В противном случае существует трасса $\mu \cdot \langle z, \gamma \rangle \in \Sigma$, где $\mu \in \mathit{safe}(\Sigma)$, $z \in C$ и $\sigma = \mu \cdot \langle z \rangle$ или $\sigma = \mu \cdot \langle z, \gamma \rangle$. Тогда $\mu \cdot \langle z, \gamma \rangle \in \mathit{final}(\Sigma)$ и $\mu \in \mathit{safe} \circ \mathit{final}(\Sigma)$. Следовательно, $\mu \cdot \langle z, \gamma \rangle \in \mathit{final} \circ \mathit{final}(\Sigma)$ и $\mu \cdot \langle z \rangle \in \mathit{final} \circ \mathit{final}(\Sigma)$, что влечёт $\sigma \in \mathit{final} \circ \mathit{final}(\Sigma)$.

14. Доказательство Утверждение 14:

По определению финальных трасс, модель финальных трасс является одним из пополнений до конвергентности её подмодели безопасных трасс. По Утверждение 9:, модель финальных трасс обладает всеми свойствами βγδ-модели, кроме, быть может, βγδ-замкнутости. Поэтому, по Утверждение 10:, модель финальных трасс может быть достроена до βγδ-модели с помощью ***DRT***-замыкания с сохранением множества безопасных трасс. Покажем, что при ***DRT***-замыкании множество финальных трасс тоже сохраняется.

Трасса, которая была финальна до замыкания, остаётся финальной, поскольку она является префиксом либо безопасной трассы, либо безопасной трассы, продолженной стимулом или реакцией и далее разрушением, а безопасные трассы сохраняются.

Покажем, что не появляется новых финальных трасс. Действительно, трассы, которые стали финальными безопасными трассами после замыкания, были такими и до замыкания. Трасса, которая до замыкания была финальной, но не безопасной, имеет вид $\sigma\cdot\langle z,\gamma\rangle$ или $\sigma\cdot\langle z\rangle$, где трасса σ безопасна, $z\in C$, а $\sigma\cdot\langle z,\gamma\rangle$ финальна. По определению ***DRT***-операций, добавляются трассы $\boldsymbol{DRT}(\sigma\cdot\langle z,\gamma\rangle)=\boldsymbol{DRT}(\sigma)\cdot\langle z,\gamma\rangle$ и $\boldsymbol{DRT}(\sigma\cdot\langle z\rangle)=\boldsymbol{DRT}(\sigma)\cdot\langle z\rangle$. Если трасса $\mu\in\boldsymbol{DRT}(\sigma)$ безопасна, то трассы $\mu\cdot\langle z,\gamma\rangle$ и $\mu\cdot\langle z\rangle$ и до замыкания были финальны. В противном случае, трассы $\mu\cdot\langle z,\gamma\rangle$ и $\mu\cdot\langle z\rangle$ не финальны после замыкания.

15. Доказательство Утверждение 15:

Необходимость. Рассмотрим $\beta\gamma\delta$-модель $\boldsymbol{\Sigma}\in\boldsymbol{MODEL}_{\beta\gamma\delta}(C)$ и покажем, что подмножество её финальных трасс $\mathbf{T}=\boldsymbol{final}(\boldsymbol{\Sigma})$ удовлетворяет всем свойствам модели финальных трасс.

Финальность означает $\boldsymbol{final}\circ\boldsymbol{final}(\boldsymbol{\Sigma})=\boldsymbol{final}(\boldsymbol{\Sigma})$, что доказано в Утверждение 13:.

Конвергентность. Если финальная трасса не безопасна в $\boldsymbol{\Sigma}$, она заканчивается разрушением или продолжается разрушением в $\boldsymbol{\Sigma}$ и, следовательно, в $\boldsymbol{final}(\boldsymbol{\Sigma})$. Такая трасса не обязана быть конвергентной. Если же трасса безопасна $\sigma\in\boldsymbol{safe}(\boldsymbol{\Sigma})$, то в $\boldsymbol{\Sigma}$ она конвергентна: для любого отказа $o\in\beta\delta(C)$ либо $\sigma\cdot\langle o\rangle\in\boldsymbol{\Sigma}$, либо существует $z\in o$ такой, что $\sigma\cdot\langle z\rangle\in\boldsymbol{\Sigma}$. Если такое продолжение безопасно, то оно также принадлежит $\boldsymbol{final}(\boldsymbol{\Sigma})$, и трасса σ конвергентна в $\boldsymbol{final}(\boldsymbol{\Sigma})$. Если такое продолжение не безопасно, существует $z`\in o$ такой, что $\sigma\cdot\langle z`,\gamma\rangle\in\boldsymbol{\Sigma}$. Но тогда $\sigma\cdot\langle z`,\gamma\rangle\in\boldsymbol{final}(\boldsymbol{\Sigma})$, и трасса σ также конвергентна в $\boldsymbol{final}(\boldsymbol{\Sigma})$.

Безопасность. По Утверждение 12:, $\boldsymbol{safe}\circ\boldsymbol{final}(\boldsymbol{\Sigma})=\boldsymbol{safe}(\boldsymbol{\Sigma})$. По Утверждение 11:, $\boldsymbol{safe}\circ\boldsymbol{MODEL}_{\beta\gamma\delta}(C)=\boldsymbol{MODEL}_{\text{safe}}(C)$. Следовательно, $\boldsymbol{safe}\circ\boldsymbol{final}(\boldsymbol{\Sigma})\in\boldsymbol{MODEL}_{\text{safe}}(C)$, что и требуется.

Достаточность непосредственно следует из Утверждение 14:.

16. Доказательство Утверждение 16:

Поскольку βγδ-модели замкнуты по ***DRT***-операциям, а *F*-изорморфные βγδ-модели имеют одно и то же множество финальных трасс, все такие βγδ-модели содержат ***DRT***-замыкание модели финальных трасс. По Утверждение 15:, ***DRT***-замыкание модели финальных трасс является βγδ-моделью с тем же множеством финальных трасс. Поэтому пересечение *F*-изорморфных βγδ-моделей совпадает с ***DRT***-замыканием модели финальных трасс и является βγδ-моделью.

17. Доказательство Утверждение 17:

Две *S*-изоморфные модели финальных трасс отличаются только тем, что в одной модели безопасная трасса σ продолжается до трассы $\sigma\cdot\langle !y,\gamma\rangle$, а в другой модели такого продолжения нет, но зато есть трасса $\sigma\cdot\langle !t,\gamma\rangle$. Объединяя все такие модели, мы получаем, очевидно, γ-однородную модель финальных трасс с тем же множеством безопасных трасс.

Очевидно, по модели безопасных трасс γ-однородная модель финальных трасс с тем же множеством безопасных трасс строится однозначно как γ-однородное пополнение до конвергентности.

18. Доказательство Утверждение 18:

Все части утверждения, кроме последней, непосредственно следуют из определения тестовых трасс. Часть 5 доказывается примерами на Рис.109 .

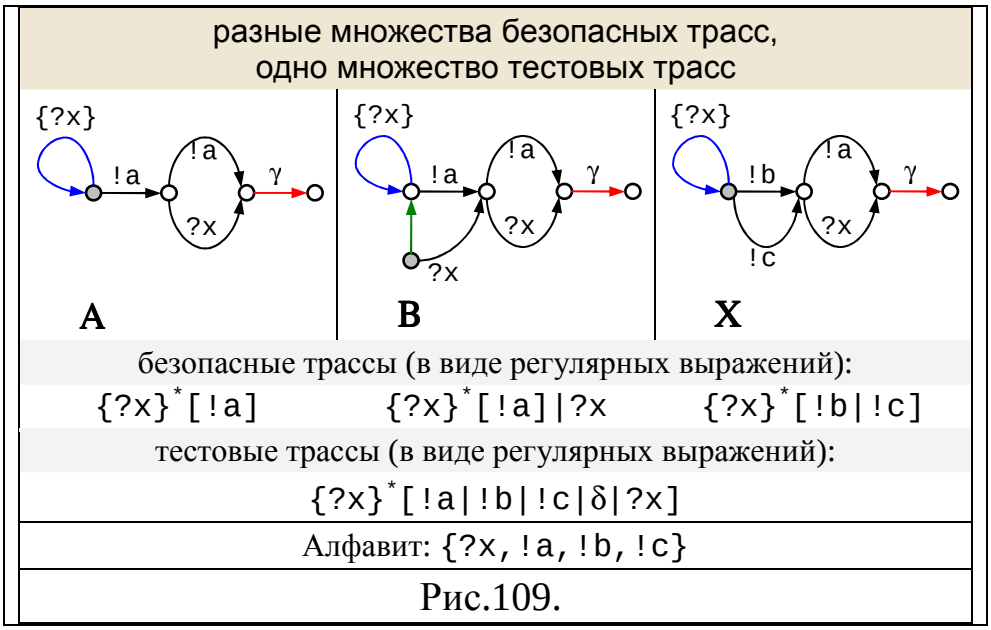


Рис.109.

19. **Доказательство Утверждение 19:**

Сначала докажем, что из $\mathbf{I}$ ***safe for*** $\mathbf{\Sigma}$ следует выполнение условий 1,2.

Условие 1. Пустая трасса есть в любой βγδ-модели, поэтому $\epsilon\in\mathbf{I}$. Если спецификация безопасна, то $\epsilon\in\boldsymbol{tt}(\mathbf{\Sigma})$. Следовательно, $\epsilon\in\boldsymbol{tt}(\mathbf{\Sigma})\cap\mathbf{I}$, что влечёт $\epsilon\in\boldsymbol{tt}(\mathbf{I})$, то есть реализация безопасна.

Условие 2. Пусть $\sigma\in\boldsymbol{safe}(\mathbf{\Sigma})\cap\boldsymbol{safe}(\mathbf{I})$ и $u\in\boldsymbol{safesymbols}(\mathbf{\Sigma}\ \boldsymbol{after}\ \sigma)$. Тогда $\sigma\cdot\langle u\rangle\in\boldsymbol{tt}(\mathbf{\Sigma})$.

Если $\sigma\cdot\langle u\rangle\in\mathbf{I}$, то $\sigma\cdot\langle u\rangle\in\boldsymbol{tt}(\mathbf{\Sigma})\cap\mathbf{I}$, что влечёт $\sigma\cdot\langle u\rangle\in\boldsymbol{tt}(\mathbf{I})$ и, следовательно, $u\in\boldsymbol{safesymbols}(\mathbf{I}\ \boldsymbol{after}\ \sigma)$.

Пусть $\sigma\cdot\langle u\rangle\notin\mathbf{I}$.

Если $u=?x$ стимул, то $\sigma\cdot\langle ?x\rangle\notin\mathbf{I}$ влечёт $\sigma\cdot\langle ?x,\gamma\rangle\notin\mathbf{I}$ и, следовательно, $?x\in\boldsymbol{safesymbols}(\mathbf{I}\ \boldsymbol{after}\ \sigma)$.

Если $u=\{?x\}$ блокировка стимула, то, по βγδ-конвергентности $\mathbf{I}$, $\sigma\cdot\langle\{?x\}\rangle\notin\mathbf{I}$ влечёт $\sigma\cdot\langle ?x\rangle\in\mathbf{I}$. По Утверждение 18:, из $\sigma\cdot\langle\{?x\}\rangle\in\boldsymbol{tt}(\mathbf{\Sigma})$ следует $\sigma\cdot\langle ?x\rangle\in\boldsymbol{tt}(\mathbf{\Sigma})$. Тогда $\sigma\cdot\langle ?x\rangle\in\mathbf{I}\cap\boldsymbol{tt}(\mathbf{\Sigma})$, что влечёт $\sigma\cdot\langle ?x\rangle\in\boldsymbol{tt}(\mathbf{I})$. По Утверждение 18:, $\sigma\cdot\langle\{?x\}\rangle\in\boldsymbol{tt}(\mathbf{I})$ и, значит, $\{?x\}\in\boldsymbol{safesymbols}(\mathbf{I}\ \boldsymbol{after}\ \sigma)$.

Если $u=!y$ реакция, то, по βγδ-конвергентности $\mathbf{I}$, $\sigma\cdot\langle !y\rangle\notin\mathbf{I}$ влечёт либо $\exists !y`\ \sigma\cdot\langle !y`\rangle\in\mathbf{I}$, либо $\sigma\cdot\langle\delta\rangle\in\mathbf{I}$. По Утверждение 18:, из $\sigma\cdot\langle !y\rangle\in\boldsymbol{tt}(\mathbf{\Sigma})$ следует, что для любой реакции $\sigma\cdot\langle !y`\rangle\in\boldsymbol{tt}(\mathbf{\Sigma})$ и $\sigma\cdot\langle\delta\rangle\in\boldsymbol{tt}(\mathbf{\Sigma})$. Тогда, соответственно, либо $\sigma\cdot\langle !y`\rangle\in\mathbf{I}\cap\boldsymbol{tt}(\mathbf{\Sigma})$, либо $\sigma\cdot\langle\delta\rangle\in\mathbf{I}\cap\boldsymbol{tt}(\mathbf{\Sigma})$, что влечёт, соответственно, либо $\sigma\cdot\langle !y`\rangle\in\boldsymbol{tt}(\mathbf{I})$, либо $\sigma\cdot\langle\delta\rangle\in\boldsymbol{tt}(\mathbf{I})$. А это, по Утверждение 18:, влечёт $\sigma\cdot\langle !y\rangle\in\boldsymbol{tt}(\mathbf{I})$ и, следовательно, $!y\in\boldsymbol{safesymbols}(\mathbf{I}\ \boldsymbol{after}\ \sigma)$.

Если $u=\delta$ стационарность, то, по $\beta\gamma\delta$-конвергентности $\mathbf{I}$, $\exists!y\ \sigma\cdot\langle!y\rangle\in\mathbf{I}$. По Утверждение 18:, из $\sigma\cdot\langle\delta\rangle\in\boldsymbol{tt}(\boldsymbol{\Sigma})$ следует, что для любой реакции $\sigma\cdot\langle!y\rangle\in\boldsymbol{tt}(\boldsymbol{\Sigma})$. Тогда $\sigma\cdot\langle!y\rangle\in\mathbf{I}\cap\boldsymbol{tt}(\boldsymbol{\Sigma})$, что влечёт $\sigma\cdot\langle!y\rangle\in\boldsymbol{tt}(\mathbf{I})$. А это, по Утверждение 18:, влечёт $\sigma\cdot\langle\delta\rangle\in\boldsymbol{tt}(\mathbf{I})$ и, следовательно, $\delta\in\boldsymbol{safesymbols}(\mathbf{I}\ \boldsymbol{after}\ \sigma)$.

Теперь докажем, что из условий 1,2 следует $\mathbf{I}\ \boldsymbol{safe\ for}\ \boldsymbol{\Sigma}$.

Сначала докажем, что $\epsilon\in\boldsymbol{tt}(\boldsymbol{\Sigma})\cap\mathbf{I}\ \Rightarrow\ \epsilon\in\boldsymbol{tt}(\mathbf{I})$. Пустая трасса есть в любой $\beta\gamma\delta$-модели, поэтому $\epsilon\in\mathbf{I}$. Если также $\epsilon\in\boldsymbol{tt}(\boldsymbol{\Sigma})$, то спецификация безопасна и, по условию 1, реализация безопасна, что влечёт $\epsilon\in\boldsymbol{tt}(\mathbf{I})$.

Допустим для некоторой непустой трассы требуемое свойство не выполнено. Выберем минимальную такую трассу, которая, по доказанному выше, не пуста и может быть представлена в виде $\sigma\cdot\langle u\rangle\in(\boldsymbol{tt}(\boldsymbol{\Sigma})\cap\mathbf{I})\setminus\boldsymbol{tt}(\mathbf{I})$, где $\sigma\in\boldsymbol{tt}(\boldsymbol{\Sigma})\cap\mathbf{I}\cap\boldsymbol{tt}(\mathbf{I})$. По определению $\boldsymbol{tt}$, из $\sigma\cdot\langle u\rangle\in\boldsymbol{tt}(\boldsymbol{\Sigma})$ следует $u\in\boldsymbol{safesymbols}(\boldsymbol{\Sigma}\ \boldsymbol{after}\ \sigma)$ и $\sigma\in\boldsymbol{safe}(\boldsymbol{\Sigma})$. Поскольку $\sigma\in\boldsymbol{tt}(\boldsymbol{\Sigma})\cap\mathbf{I}\cap\boldsymbol{tt}(\mathbf{I})$ влечёт $\sigma\in\mathbf{I}\cap\boldsymbol{tt}(\mathbf{I})$, то, по Утверждение 18:, это влечёт $\sigma\in\boldsymbol{safe}(\mathbf{I})$. По условию 2, $\sigma\cdot\langle u\rangle\in\boldsymbol{tt}(\boldsymbol{\Sigma})$ и $\sigma\in\boldsymbol{safe}(\mathbf{I})$ влечёт $u\in\boldsymbol{safesymbols}(\mathbf{I}\ \boldsymbol{after}\ \sigma)$. Но тогда, поскольку $\sigma\in\boldsymbol{safe}(\mathbf{I})$, $\sigma\cdot\langle u\rangle\in\boldsymbol{tt}(\mathbf{I})$, что противоречит предположению. Мы пришли к противоречию, и на этом доказательство завершается.

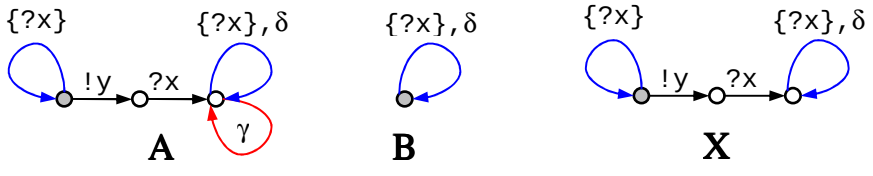
20. Доказательство Утверждение 20:

1) Отношение ***safe for***.

Рефлексивность непосредственно следует из определения
$\mathbf{A}\ \boldsymbol{safe\ for}\ \mathbf{A} = \boldsymbol{tt}(\mathbf{A})\cap\mathbf{A}\ \subseteq\ \boldsymbol{tt}(\mathbf{A}) = \boldsymbol{true}$.

Нетранзитивность показывается примером порождающих графов на Рис.110.

нетранзитивность отношения ***safe for***



C={?x,!y}
A ***safe for*** **B**
B ***safe for*** **X**
A ¬***safe for*** **X**

Рис.110.

2) Отношение $\boldsymbol{ioco}_{\beta\gamma\delta}$.

Рефлексивность. Следует из определения
$\mathbf{A}\ \boldsymbol{ioco}_{\beta\gamma\delta}\ \mathbf{A} = \boldsymbol{tt}(\mathbf{A})\cap\mathbf{A} \subseteq \mathbf{A}\cap\boldsymbol{tt}(\mathbf{A}) = \boldsymbol{true}$.

Транзитивность.
Пусть $\mathbf{A}\ \boldsymbol{ioco}_{\beta\gamma\delta}\ \mathbf{B} = \boldsymbol{tt}(\mathbf{B})\cap\mathbf{A} \subseteq \mathbf{B}\cap\boldsymbol{tt}(\mathbf{A})$,
$\mathbf{B}\ \boldsymbol{ioco}_{\beta\gamma\delta}\ \mathbf{X} = \boldsymbol{tt}(\mathbf{X})\cap\mathbf{B} \subseteq \mathbf{X}\cap\boldsymbol{tt}(\mathbf{B})$.

Покажем, что $\mathbf{A}\ \boldsymbol{ioco}_{\beta\gamma\delta}\ \mathbf{X} = \boldsymbol{tt}(\mathbf{X})\cap\mathbf{A} \subseteq \mathbf{X}\cap\boldsymbol{tt}(\mathbf{A})$.

Если $\mathbf{X}$ саморазрушающаяся, то есть $\epsilon\notin\boldsymbol{tt}(\mathbf{X})$, то $\boldsymbol{tt}(\mathbf{X})=\varnothing$ и, поэтому $\boldsymbol{tt}(\mathbf{X})\cap\mathbf{A} = \varnothing \subseteq \mathbf{X}\cap\boldsymbol{tt}(\mathbf{A})$.

Теперь пусть $\mathbf{X}$ безопасна, то есть $\epsilon\in\boldsymbol{tt}(\mathbf{X})$.

Сначала докажем вспомогательное утверждение: $\boldsymbol{tt}(\mathbf{X})\cap\mathbf{A}\subseteq\boldsymbol{tt}(\mathbf{B})$.

Допустим обратное: существует $\sigma\in(\boldsymbol{tt}(\mathbf{X})\cap\mathbf{A})\setminus\boldsymbol{tt}(\mathbf{B})$.
Поскольку $\epsilon\in\boldsymbol{tt}(\mathbf{X})$, $\epsilon\in\mathbf{B}$ и $\boldsymbol{tt}(\mathbf{X})\cap\mathbf{B} \subseteq \mathbf{X}\cap\boldsymbol{tt}(\mathbf{B})$, то $\epsilon\in\boldsymbol{tt}(\mathbf{B})$.
Поскольку, $\epsilon\in\mathbf{B}$, а $\boldsymbol{safe}(\mathbf{B})=\boldsymbol{tt}(\mathbf{B})\cap\mathbf{B}$, имеем $\epsilon\in\boldsymbol{safe}(\mathbf{B})$.
Значит, трассу σ можно представить в виде $\sigma=\mu\cdot\langle u\rangle\cdot\lambda$, где $\mu\in\boldsymbol{safe}(\mathbf{B})$, $\mu\cdot\langle u\rangle\notin\boldsymbol{safe}(\mathbf{B})$.
Если бы $u\in\boldsymbol{safesymbols}(\mathbf{B}\ \boldsymbol{after}\ \mu)$, то, поскольку $\mu\cdot\langle u\rangle\in\mathbf{A}$ и $\mathbf{A}\ \boldsymbol{ioco}_{\beta\gamma\delta}\ \mathbf{B}$, было бы $\mu\cdot\langle u\rangle\in\mathbf{B}$ и, очевидно, $\mu\cdot\langle u\rangle\in\boldsymbol{safe}(\mathbf{B})$, что неверно.
Значит $u\notin\boldsymbol{safesymbols}(\mathbf{B}\ \boldsymbol{after}\ \mu)$.
$\sigma\in\boldsymbol{tt}(\mathbf{X})$ влечёт $\mu\in\boldsymbol{safe}(\mathbf{X})$ и $u\in\boldsymbol{safesymbols}(\mathbf{X}\ \boldsymbol{after}\ \mu)$.
А это вместе с $\mathbf{B}\ \boldsymbol{safe\ for}\ \mathbf{X}$, по Утверждение 19:, влечёт $u\in\boldsymbol{safesymbols}(\mathbf{B}\ \boldsymbol{after}\ \mu)$, что не верно.

Мы пришли к противоречию и, значит, $\sigma \in \mathit{tt}(\mathbf{B})$.

Теперь докажем, что $\mathbf{A}$ ***safe for*** $\mathbf{X}$: $\mathit{tt}(\mathbf{X}) \cap \mathbf{A} \subseteq \mathit{tt}(\mathbf{A})$.

Пусть $\sigma \in \mathit{tt}(\mathbf{X}) \cap \mathbf{A}$.
По вспомогательному утверждению, $\sigma \in \mathit{tt}(\mathbf{B})$.
Тогда $\sigma \in \mathit{tt}(\mathbf{B}) \cap \mathbf{A}$, а $\mathbf{A}$ ***safe for*** $\mathbf{B}$ влечёт $\mathit{tt}(\mathbf{B}) \cap \mathbf{A} \subseteq \mathit{tt}(\mathbf{A})$.
Значит, $\sigma \in \mathit{tt}(\mathbf{A})$, что и требовалось доказать.

Теперь докажем, что $\mathbf{A}$ $\mathit{ioco}_{\beta\gamma\delta}$ $\mathbf{X}$.

Пусть $\sigma \in \mathit{tt}(\mathbf{X}) \cap \mathbf{A}$. Нам осталось показать, что $\sigma \in \mathbf{X}$.
Действительно, по вспомогательному утверждению, $\sigma \in \mathit{tt}(\mathbf{B})$.
Отсюда, поскольку $\mathbf{A}$ $\mathit{ioco}_{\beta\gamma\delta}$ $\mathbf{B}$ и $\sigma \in \mathbf{A}$, получаем $\sigma \in \mathbf{B}$.
Поскольку $\sigma \in \mathit{tt}(\mathbf{X})$ и $\mathbf{B}$ $\mathit{ioco}_{\beta\gamma\delta}$ $\mathbf{X}$, имеем $\sigma \in \mathbf{X}$, что и требовалось доказать.

21. Доказательство Утверждение 21:

Если одна из βγδ-моделей саморазрушающаяся, то из $\mathbf{I} \sim_{\mathit{ioco}_{\beta\gamma\delta}} \mathbf{\Sigma}$ следует, по Утверждение 19:, что вторая βγδ-модель также саморазрушающаяся. Тем самым, множества их безопасных трасс пусты и, значит, равны.

Если, наоборот, множества безопасных трасс обеих βγδ-моделей пусты, то эти βγδ-модели саморазрушающиеся и, следовательно, $\mathit{ioco}_{\beta\gamma\delta}$-эквивалентны.

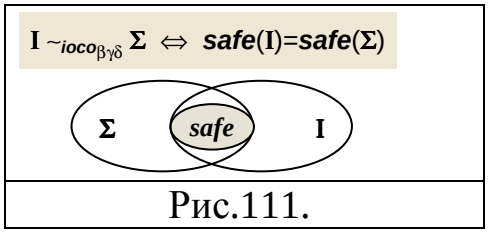


Рис.111.

Пусть обе βγδ-модели безопасны (Рис.111).

Докажем прямую импликацию $\mathbf{I} \sim_{\mathit{ioco}_{\beta\gamma\delta}} \mathbf{\Sigma} \Rightarrow \mathit{safe}(\mathbf{I}) = \mathit{safe}(\mathbf{\Sigma})$.

Пусть $\mathbf{I} \sim_{\mathit{ioco}_{\beta\gamma\delta}} \mathbf{\Sigma}$.
В силу симметрии, достаточно показать, что $\mathit{safe}(\mathbf{I}) \subseteq \mathit{safe}(\mathbf{\Sigma})$.
Поскольку обе спецификации безопасны, $\epsilon \in \mathit{safe}(\mathbf{I}) \cap \mathit{safe}(\mathbf{\Sigma})$.
Поэтому достаточно показать, что, если $\sigma \cdot \langle u \rangle \in \mathit{safe}(\mathbf{I})$, где $\sigma \in \mathit{safe}(\mathbf{I}) \cap \mathit{safe}(\mathbf{\Sigma})$, то $\sigma \cdot \langle u \rangle \in \mathit{safe}(\mathbf{\Sigma})$.
Очевидно, $u \in \mathit{safesymbols}(\mathbf{I}\ \mathit{after}\ \sigma)$.
Тогда $\mathbf{\Sigma}$ ***safe for*** $\mathbf{I}$ влечёт $u \in \mathit{safesymbols}(\mathbf{\Sigma}\ \mathit{after}\ \sigma)$.

Но тогда $\mathbf{I}$ $\boldsymbol{ioco}_{\beta\gamma\delta}$ $\mathbf{\Sigma}$ влечёт $\sigma\cdot\langle u\rangle\in\mathbf{\Sigma}$ и, тем самым, $\sigma\cdot\langle u\rangle\in\boldsymbol{safe}(\mathbf{\Sigma})$, что и требовалось показать.

Докажем обратную импликацию $\mathbf{I}\sim_{\boldsymbol{ioco}_{\beta\gamma\delta}}\mathbf{\Sigma}\Leftarrow\boldsymbol{safe}(\mathbf{I})=\boldsymbol{safe}(\mathbf{\Sigma})$.

Пусть $\boldsymbol{safe}(\mathbf{I})=\boldsymbol{safe}(\mathbf{\Sigma})$. Тогда, по Утверждение 18:, $\boldsymbol{tt}(\mathbf{I})=\boldsymbol{tt}(\mathbf{\Sigma})$. Отсюда, с одной стороны, $\boldsymbol{tt}(\mathbf{\Sigma})\cap\mathbf{I}=\boldsymbol{tt}(\mathbf{I})\cap\mathbf{I}=\boldsymbol{safe}(\mathbf{I})$, и, с другой стороны, $\mathbf{\Sigma}\cap\boldsymbol{tt}(\mathbf{I})=\mathbf{\Sigma}\cap\boldsymbol{tt}(\mathbf{\Sigma})=\boldsymbol{safe}(\mathbf{\Sigma})$. Поскольку $\boldsymbol{safe}(\mathbf{I})=\boldsymbol{safe}(\mathbf{\Sigma})$, имеем: $\boldsymbol{tt}(\mathbf{\Sigma})\cap\mathbf{I}=\mathbf{\Sigma}\cap\boldsymbol{tt}(\mathbf{I})$, то есть $\mathbf{I}\sim_{\boldsymbol{ioco}_{\beta\gamma\delta}}\mathbf{\Sigma}$.

22. **Доказательство Утверждение 22:**

1) Это легко видеть на Рис.32. Здесь множеству $\boldsymbol{tt}(\mathbf{\Sigma})\cap\mathbf{I}$ соответствует четыре варианта (tC÷tF); гипотеза о безопасности запрещает варианты (tC÷tD), а условие вложенности безопасных трасс – вариант tE; в результате остаётся только вариант конформности – вариант tF.

2) Как видно на Рис.32, из четырёх вариантов (tC÷tF) условие вложенности трасс запрещает варианты tC и tE. Нам нужно показать, что вариант tD невозможен: если трасса безопасна в спецификации и имеется в реализации, то она безопасна в реализации: $\boldsymbol{safe}(\mathbf{\Sigma})\cap\mathbf{I}\subseteq\boldsymbol{safe}(\mathbf{I})$. Допустим утверждение не верно, и существует трасса $\sigma\in(\boldsymbol{safe}(\mathbf{\Sigma})\cap\mathbf{I})\setminus\boldsymbol{safe}(\mathbf{I})$. Выберем такую трассу минимальной длины. Если $\sigma=\epsilon$ пустая трасса, то $\sigma\notin\boldsymbol{safe}(\mathbf{I})$ влечёт $\langle\gamma\rangle\in\mathbf{I}$. Но тогда, поскольку $\mathbf{I}\subseteq\mathbf{\Sigma}$, имеем $\langle\gamma\rangle\in\mathbf{\Sigma}$, то есть $\sigma\notin\boldsymbol{safe}(\mathbf{\Sigma})$, что противоречит допущению. Если трасса σ не пуста, её можно представить в виде $\sigma=\mu\cdot\langle u\rangle$, где $\mu\in\boldsymbol{safe}(\mathbf{\Sigma})\cap\boldsymbol{safe}(\mathbf{I})$. Отсюда, поскольку, $\sigma\notin\boldsymbol{safe}(\mathbf{I})$, то $\mu\cdot\langle z\rangle\cdot\langle\gamma\rangle\in\mathbf{I}$, где $z=u$, если u стимул или реакция, и $z\in u$, если u отказ. Но тогда, поскольку $\mathbf{I}\subseteq\mathbf{\Sigma}$, имеем $\mu\cdot\langle z\rangle\cdot\langle\gamma\rangle\in\mathbf{\Sigma}$, то есть символ u опасен в спецификации после трассы μ. Отсюда $\sigma\notin\boldsymbol{safe}(\mathbf{\Sigma})$, что противоречит допущению.

23. **Доказательство Утверждение 23:**

$\boldsymbol{final}(\mathbf{I})=\boldsymbol{final}(\mathbf{\Sigma})$
$\Rightarrow$ (* по определению ***safe*** *)
$\boldsymbol{safe}\circ\boldsymbol{final}(\mathbf{I})=\boldsymbol{safe}\circ\boldsymbol{final}(\mathbf{\Sigma})$
$\Rightarrow$ (* по Утверждение 12: *)
$\boldsymbol{safe}(\mathbf{I})=\boldsymbol{safe}(\mathbf{\Sigma})$
$\Rightarrow$ (* по Утверждение 21: *)

$\mathbf{I} \sim_{\boldsymbol{ioco}_{\beta\gamma\delta}} \mathbf{\Sigma}$.

24. Доказательство Утверждение 24:

1) Отсутствие в реализации разрушающих стимулов автоматически делает её отвечающей гипотезе о безопасности для любой спецификации.

2) Учитывая Утверждение 22:, нам достаточно доказать импликацию $\mathbf{I}\ \boldsymbol{ioco}_{\beta\gamma\delta}\ \mathbf{\Sigma} \Rightarrow \mathbf{I} \subseteq \mathbf{\Sigma}$. Допустим утверждение не верно, и в реализации существует трасса σ, которой нет в спецификации. Поскольку пустая трасса безопасна в спецификации без разрушения, трассу σ можно представить в виде $\sigma = \mu \cdot \langle u \rangle \in \mathbf{I}$, где $\mu \in \boldsymbol{safe}(\mathbf{\Sigma})$. Поскольку в спецификации без разрушения любой символ безопасен после любой трассы, u безопасен в спецификации после μ. Тогда, по условию конформности, должно быть $\mu \cdot \langle u \rangle \in \mathbf{\Sigma}$, что противоречит допущению.

25. Доказательство Утверждение 25:

Поскольку все тесты значимые, достаточно показать, что $\mathbf{TT}$ – исчерпывающий.

Мы покажем, что, если реализация $\mathbf{I} \in \boldsymbol{safeIMPL}(\mathbf{\Sigma})$ не соответствует спецификации, то найдётся такой тест $\mathbf{T} \in \mathbf{TT}$, который эта реализация не проходит.

По определению $\boldsymbol{ioco}_{\beta\gamma\delta}$, $\exists \sigma \in \boldsymbol{safe}(\mathbf{\Sigma})\ \exists u \in \boldsymbol{safesymbols}(\mathbf{\Sigma}\ \boldsymbol{after}\ \sigma)$ такой, что $\sigma \cdot \langle u \rangle \in \mathbf{I}$, но $\sigma \cdot \langle u \rangle \notin \mathbf{\Sigma}$.

Поскольку трасса σ безопасна, она конвергентна.

Обозначим: если $u = ?x$, то $u` = \{?x\}$,
если $u = \{?x\}$, то $u` = ?x$,
если $u \in !c_\delta$, то $u` \in !c_\delta$ – любое такое, что
$\sigma \cdot \langle u` \rangle \in \boldsymbol{safe}(\mathbf{\Sigma})$ (хотя бы одно такое $u` \neq u$ должно найтись).

В любом случае $\sigma \cdot \langle u` \rangle \in \boldsymbol{safe}(\mathbf{\Sigma})$.

А тогда $\exists \mathbf{T} \in \mathbf{TT}\ \ \sigma \cdot \langle u` \rangle \in \mathbf{T}$.

По способу построения тестовых трасс, также $\sigma \cdot \langle u \rangle \in \mathbf{T}$.

По строго-значимости теста, $\sigma \cdot \langle u \rangle \in \boldsymbol{fail}(\mathbf{T})$.

Но тогда реализация $\mathbf{I}$ не проходит тест $\mathbf{T}$.

26. **Доказательство Утверждение 26:**

1) По построению, $\boldsymbol{Test}(\mathbf{A})$ является управляемым деревом тестовых трасс с заданным вердиктом.

   Поскольку $\mathbf{A}$ нётерово множество, $\boldsymbol{Test}(\mathbf{A})$ также будет нётеровым и, следовательно, тестом.

   Докажем, что это значимый тест.

   Мы покажем, что, если реализация $\mathbf{I} \in \boldsymbol{safeIMPL}(\mathbf{\Sigma})$ не проходит тест $\boldsymbol{Test}(\mathbf{A})$, то есть, $\exists \sigma \in \mathbf{I} \cap \boldsymbol{fail} \circ \boldsymbol{Test}(\mathbf{A})$, то реализация не соответствует спецификации.

   По определению вердикта для $\boldsymbol{Test}(\mathbf{A})$, $\sigma \notin \mathbf{\Sigma}$.
   Возьмём трассу μ, равную трассе σ, укороченной на последний символ: $\sigma = \mu \cdot \langle u \rangle$.
   Заметим, что такое укорачивание нельзя сделать только для $\sigma = \epsilon$, но $\epsilon \in \mathbf{\Sigma}$.
   По определению тестовой трассы,
   $\mu \in \boldsymbol{safe}(\mathbf{\Sigma})$ и $u \in \boldsymbol{safesymbols}(\mathbf{\Sigma}\ \boldsymbol{after}\ \mu)$.
   Поскольку $\mu \cdot \langle u \rangle \in \mathbf{I}$, но $\mu \cdot \langle u \rangle \notin \mathbf{\Sigma}$, не верно, что $\mathbf{I}\ \boldsymbol{ioco}_{\beta\gamma\delta}\ \mathbf{\Sigma}$, то есть $\boldsymbol{Test}(\mathbf{A})$ значимый тест.

   Строго-значимость следует из определения вердикта для $\boldsymbol{Test}(\mathbf{A})$.

2) Очевидно, что для строго-значимого теста $\mathbf{T}$ множество его ***pass***-трасс и их префиксов является нётеровым ($\mathbf{T}$ нётерово) полууправляемым ($\mathbf{T}$ управляемое) поддеревом безопасных ($\mathbf{T}$ строго-значимое) трасс.

   Тестовая трасса либо безопасна, либо продолжает безопасную трассу безопасным символом.

   Поэтому для того, чтобы доказать равенство $\mathbf{T} = \boldsymbol{Test} \circ \cup \circ \boldsymbol{Tree} \circ \boldsymbol{pass}(\mathbf{T})$, достаточно показать, что тест вместе с любой ***fail***-трассой $\mu \cdot \langle u \rangle$ содержит трассу $\mu \cdot \langle u` \rangle$, являющуюся префиксом некоторой ***pass***-трассы.

   Предположим, это неверно.

По определению тестовой трассы, трасса μ безопасна, а символ u безопасен после μ, а тогда, по $\beta\gamma\delta$-конвергентности спецификации, найдётся такой символ $u`$, что $\mu\cdot\langle u`\rangle\in\mathbf{\Sigma}$.

Отсюда, по строгой значимости теста, $\mu\cdot\langle u`\rangle$ является ***pass***-трассой или немаксимальна.

По предположению, $\mu\cdot\langle u`\rangle$ немаксимальна.

Тогда, по нётеровости теста, в нём существует максимальная трасса $\mu_1\cdot\langle z_1\rangle>\mu\cdot\langle u`\rangle$, и, следовательно, $\mu<\mu_1$.

По предположению, $\mu_1\cdot\langle z_1\rangle$ должна быть ***fail***-трассой.

Применяя это рассуждение к трассе $\mu_1\cdot\langle z_1\rangle$, получаем ***fail***-трассу $\mu_2\cdot\langle z_2\rangle$, где $\mu_1<\mu_2$.

Повторяя это рассуждение дальше, получаем бесконечно-возрастающую цепочку трасс теста $\mu<\mu_1<\mu_2<...$, что противоречит нётеровости теста.

27. Доказательство Утверждение 27:

По Утверждение 26:, $\mathbf{TT(AA)}$ – строго-значимый тестовый набор.
Поскольку каждый тест содержит безопасные трассы, из которых он построен, из $\cup(\mathbf{AA})=\boldsymbol{safe}(\mathbf{\Sigma})$ следует $\cup(\mathbf{TT(AA)})\supseteq\boldsymbol{safe}(\mathbf{\Sigma})$.
Поэтому, по Утверждение 25:, этот тестовый набор полон.

28. Доказательство Утверждение 28:

В процессе перечисления безопасных трасс мы на каждом n-ом шаге имеем множество $\mathbf{\Sigma}_n$ из n безопасных трасс. На основе него тривиально строится множество всех искомых деревьев, в которых все трассы являются трассами из $\mathbf{\Sigma}_n$. Очевидно, для каждого искомого дерева найдётся такое n, что мы построим это дерево на n-ом шаге. Для перечисления искомых деревьев достаточно на каждом шаге строить все возможные искомые деревья и отфильтровывать те, которые уже были построены на предыдущем шаге.

29. Доказательство Утверждение 29:

Пусть $C\subseteq Z$ и $\mathbf{\Sigma}\in\mathbf{T0}(C)$.
Сначала построим итерацию безопасных трасс.

О безопасности спецификации мы узнаём с помощью оракула $\mathbf{Safe}_\Sigma()$. Если спецификация опасна, то безопасных трасс нет.

Пусть теперь спецификация безопасна. Тогда её пустая трасса безопасна.

Далее для каждой безопасной трассы σ итератор $\mathbf{Safehead}_{\Sigma}(\sigma)$ задаёт нумерацию безопасных символов, которыми трасса продолжается.
Заметим, что номер символа определяется не только самим символом, но и трассой σ.
Индексом безопасной трассы назовём сумму номеров её символов:
номер i-ого символа $\sigma(i)$ определяется непосредственно предшествующим ему префиксом трассы $\sigma[1..i-1]$.
Очевидно, что множество безопасных трасс с данным индексом конечно, и его можно перебрать алгоритмически, используя итератор $\mathbf{Safehead}_{\Sigma}(\sigma)$.
Тогда итерация безопасных трасс реализуется двумя вложенными циклами: во внешнем цикле перечисляем индекс, а во внутреннем – трассы с этим индексом; индекс 0 имеет пустая трасса.

По Утверждение 28:, мы можем построить алгоритм перечисления всех конечных полууправляемых деревьев безопасных трасс.

По Утверждение 26:, из нётерова полууправляемого дерева безопасных трасс $\mathbf{A}$ можно построить строго-значимый тест $\mathit{Test}(\mathbf{A})$, и любой строго-значимый тест может быть построен таким образом.

Очевидно, что тест $\mathit{Test}(\mathbf{A})$ будет иметь конечное число немаксимальных трасс тогда и только тогда, когда дерево $\mathbf{A}$ конечно.

Алгоритм теста $\mathit{Test}(\mathbf{A})$ работает следующим образом.

Если $\mathbf{A}$ содержит только одну пустую трассу, выдаётся вердикт ***pass*** (без тестирования).
Иначе, после немаксимальной трассы $\sigma \in \mathbf{A}$, начиная с пустой трассы, нажимается кнопка машины тестирования «послать стимул $?x$», если σ ***semi-send in*** $\mathbf{A}$ и $\mathit{head}(\mathbf{A}\ \mathit{after}\ \sigma) \subseteq \{?x, \{?x\}\}$, или кнопка «принять все реакции», если σ ***semi-wait in*** $\mathbf{A}$.
Далее после получения символа u проверяется его правильность с помощью оракула $\mathbf{SafeExtend}_{\Sigma}(\sigma, u)$.
Если символ неправильный, выносится вердикт ***fail***.
Если символ правильный, то проверяется $\sigma \cdot \langle u \rangle \in \mathbf{A}$; эта проверка алгоритмизуема, поскольку $\mathbf{A}$ конечно.

Если $\sigma\cdot\langle u\rangle\in \mathbf{A}$, действия повторяются для трассы $\sigma\cdot\langle u\rangle$ (начиная с нажатия кнопки), иначе – выносится вердикт ***pass***.

Тем самым, имея алгоритм перечисления всех конечных полууправляемых деревьев безопасных трасс, мы получаем алгоритм перечисления всех строго-значимых тестов с конечным числом немаксимальных трасс.

30. **Доказательство Утверждение 30:**

Пусть $C\subseteq Z$ и $\Sigma\in \mathbf{T1}(C)$.
Построим алгоритмы, задающие спецификацию способом **T0**: оракул $\mathbf{Safe}_\Sigma()$, итератор $\mathbf{Safehead}_\Sigma(\sigma)$ и оракул $\mathbf{SafeExtend}_\Sigma(\sigma,u)$ (Определение 33:). Оракул $\mathbf{Safe}_\Sigma()$, очевидно, такой же, как для способа **T0**, а оракул $\mathbf{SafeExtend}_\Sigma(\sigma,u)$ совпадает с оракулом $\mathbf{Extend1}_\Sigma(\sigma,u)$. Тем самым выполнено условие конечности теста.

Построим итератор $\mathbf{Safehead}_\Sigma(\sigma)$.

Итерируем конечное множество реакций с помощью итератора $\mathbf{Y}_C$, и проверяем каждую реакцию $!y$ с помощью оракула $\mathbf{Gamma1}_\Sigma(\sigma,!y)$. Если хотя бы одна реакция оказалась разрушающей, то все реакции и стационарность опасны, и мы их не перечисляем. В противном случае перечисляем все реакции и стационарность, которые продолжают трассу, что проверяется оракулом $\mathbf{Extend1}_\Sigma(\sigma,u)$.

Далее итерируем стимулы с помощью итератора $\mathbf{X}_C$, и проверяем каждый стимул $?x$ с помощью оракула $\mathbf{Gamma1}_\Sigma(\sigma,?x)$. Если стимул оказался разрушающим, то сам стимул и его блокировка опасны, и мы их не перечисляем, переходя к следующему стимулу, если такой окажется. В противном случае перечисляем стимул $?x$ и его блокировку $\{?x\}$, которые продолжают трассу, что проверяется оракулом $\mathbf{Extend1}_\Sigma(\sigma,u)$, и переходим к следующему стимулу, если такой окажется.

Таким образом, у нас выполнены условия Утверждение 29:, и, тем самым, утверждение доказано.

31. **Доказательство Утверждение 31:**

Пусть $\Sigma\in \boldsymbol{MODEL}_{\beta\gamma\delta}(C)$ и $T_2=\boldsymbol{tfinal}(\Sigma)$.
Обозначим $\Sigma_2 = \boldsymbol{CR}\circ\cup\circ\boldsymbol{DRT}(T_2)$.

Сначала покажем, что $\mathbf{\Sigma}_2$ является βγδ-моделью с той же подмоделью трансфинальных трасс $\mathbf{T}_2$, то есть $\mathbf{\Sigma}_2 \in \boldsymbol{E}(\mathbf{T}_2)$, что эквивалентно $\boldsymbol{tfinal}(\mathbf{\Sigma}_2)=\mathbf{T}_2$.

В подмодели трансфинальных трасс $\mathbf{T}_2$ неконвергентны только трансбезопасные трассы: такие трассы ничем не продолжаются. Продолжим их разрушением: $\mathbf{T}_{2\gamma} = \boldsymbol{final}(\mathbf{\Sigma}) \cup (\boldsymbol{tsafe}(\mathbf{\Sigma})\cdot\{\langle\gamma\rangle\})$. Очевидно, мы получим модель финальных трасс. Как показано в Утверждение 14:, её ***DRT***-замыкание $\mathbf{\Sigma}_{2\gamma}=\cup_\circ \boldsymbol{DRT}(\mathbf{T}_{2\gamma})$ является βγδ-моделью с тем же множеством финальных трасс. Если теперь заменить те разрушения, которые мы добавили, на все возможные трассы отказов, мы, очевидно, также получим βγδ-модель, совпадающую с $\mathbf{\Sigma}_2$. Мы показали, что $\mathbf{\Sigma}_2$ является βγδ-моделью.

Трасса $\sigma\cdot\langle !y\rangle$ трансбезопасная, если реакция $!y$ опасна, но неразрушающая, после безопасной трассы σ, то есть нет трассы $\sigma\cdot\langle !y,\gamma\rangle$, но есть трасса $\sigma\cdot\langle !t,\gamma\rangle$ для некоторой реакции $!t$. Следовательно любая трасса из $\boldsymbol{DRT}(\sigma)$ продолжалась трассой $\langle !t,\gamma\rangle$. Значит, реакция $!y$ опасна после любой трассы из $\boldsymbol{DRT}(\sigma)$. Отсюда следует, что безопасные трассы $\mathbf{\Sigma}_2$ те же самые, что в $\mathbf{\Sigma}_{2\gamma}$, $\mathbf{T}_{2\gamma}$ и $\mathbf{T}_2$. Кроме того, в $\mathbf{\Sigma}_2$ и $\mathbf{T}_2$ совпадают трансбезопасные трассы. Финальные, но опасные, трассы $\mathbf{\Sigma}_2$ имеют вид $\sigma\cdot\langle z,\gamma\rangle$ или $\sigma\cdot\langle z\rangle$, где трасса σ безопасна, а символ z разрушающий после σ. Поэтому финальные, но опасные, трассы $\mathbf{\Sigma}_2$ же самые, что в $\mathbf{T}_2$. Тем самым, все финальные трассы $\mathbf{\Sigma}_2$ же самые, что в $\mathbf{T}_2$. Таким образом, мы имеем $\boldsymbol{tfinal}(\mathbf{\Sigma}_2)=\mathbf{T}_2$.

Покажем, что $\cap_\circ \boldsymbol{E}(\mathbf{T}_2)=\cup_\circ \boldsymbol{DRT}(\mathbf{T}_2)$.

Поскольку βγδ-модели замкнуты по ***DRT***-операциям, а все βγδ-модели из $\boldsymbol{E}(\mathbf{T}_2)$ имеют одну и ту же подмодель трансфинальных трасс $\mathbf{T}_2$, все такие βγδ-модели содержат ***DRT***-замыкание этой подмодели трансфинальных трасс $\cup_\circ \boldsymbol{DRT}(\mathbf{T}_2)$. С другой стороны, одна из таких моделей, $\mathbf{\Sigma}_2$, отличается от $\cup_\circ \boldsymbol{DRT}(\mathbf{T}_2)$ только продолжениями трансбезопасных трасс всеми трассами отказов, то есть трассами из $\beta\delta(C)^*$. Если мы заменим такие продолжения на продолжения всеми трассами из $C^1\cdot\beta\delta(C)^*$, мы, очевидно, также получим βγδ-модель $\mathbf{\Sigma}_{2+}$ с

той же подмоделью трансфинальных трасс $\mathbf{T}_2$. Поскольку $\mathbf{\Sigma}_2 \cap \mathbf{\Sigma}_{2+} = \cup_\circ \boldsymbol{DRT}(\mathbf{T}_2)$, имеем $\cap_\circ \boldsymbol{E}(\mathbf{T}_2) = \cup_\circ \boldsymbol{DRT}(\mathbf{T}_2)$.

32. Доказательство Утверждение 32:

Пусть $C \subseteq Z$ и $\mathbf{\Sigma} \in \mathbf{T2}(C)$.
Построим алгоритмы, задающие спецификацию способом **T1**. С учётом алгоритмов, заданных для **T2**-спецификации, нам нужно построить оракулы $\mathbf{Safe}_\Sigma()$, $\mathbf{Gamma1}_\Sigma(\sigma, z)$ и $\mathbf{Extend1}_\Sigma(\sigma, u)$. В качестве оракула $\mathbf{Extend1}_\Sigma(\sigma, u)$, очевидно, годится оракул $\mathbf{Extend2}_\Sigma(\sigma, u)$, определённый на большем домене (не только для безопасных стимулов и реакций). Тем самым выполнено условие конечности теста.

Очевидно, оракул $\mathbf{Safe}_\Sigma() = \neg\mathbf{Gamma2}_\Sigma(\epsilon)$.

Построим оракул $\mathbf{Gamma1}_\Sigma(\sigma, z)$.

Для каждого стимула или реакции z мы можем проверить с помощью оракула $\mathbf{Extend2}_\Sigma(\sigma, z)$, продолжается трасса символом z или нет. Если трасса не продолжается символом z, то символ z неразрушающий после трассы. Если трасса продолжается символом z, то с помощью оракула $\mathbf{Gamma2}_\Sigma(\sigma \cdot \langle z \rangle)$ мы можем узнать, следует ли за этим символом разрушение или нет.

Таким образом, у нас выполнены условия Утверждение 30:, и, тем самым, требуемое утверждение доказано.

33. Доказательство Утверждение 33:

Пусть $C \subseteq Z$ и $\mathbf{\Sigma} \in \mathbf{T1}(C)$.
Построим алгоритмы, задающие спецификацию способом **T2**. С учётом алгоритмов, заданных для **T1**-спецификации, нам нужно построить оракулы $\mathbf{Extend2}_\Sigma(\sigma, u)$ и оракул $\mathbf{Gamma2}_\Sigma(\sigma, z)$.

Построим итератор $\mathbf{Extend2}_\Sigma(\sigma, u)$.

С помощью оракула $\mathbf{Gamma1}_\Sigma(\sigma, ?x)$ мы можем проверить, является ли стимул $?x$ разрушающим после безопасной трассы σ.

Если стимул разрушающий, полагаем $\mathbf{Extend2}_\Sigma(\sigma, ?x) = \textit{true}$.

В противном случае стимул $u=?x$ и его блокировка $u=\{?x\}$ безопасны после σ, и с помощью оракула $\mathbf{Extend1}_\Sigma(\sigma,u)$ мы можем проверить, продолжается ли трасса стимулом или его блокировкой: $\mathbf{Extend2}_\Sigma(\sigma,u) = \mathbf{Extend1}_\Sigma(\sigma,u)$.

С помощью оракула $\mathbf{Gamma1}_\Sigma(\sigma,!y)$ мы можем проверить, является ли реакция $!y$ разрушающей после безопасной трассы σ. Поскольку число реакция конечно, используя итератор $\mathbf{Y}_C$, мы можем проверить безопасность всех реакций после σ.

Если реакции опасны, для каждой реакции $!y$ полагаем $\mathbf{Extend2}_\Sigma(\sigma,!y) = \mathbf{Gamma1}_\Sigma(\sigma,!y)$.

В противном случае каждая реакция $u=!y$ и стационарность $u=\delta$ безопасны после σ, и $\mathbf{Extend2}_\Sigma(\sigma,u) = \mathbf{Extend1}_\Sigma(\sigma,u)$.

Построим оракул $\mathbf{Gamma2}_\Sigma(\sigma,z)$.

Для пустой трассы, очевидно, $\mathbf{Gamma2}_\Sigma(\epsilon) = \neg\mathbf{Safe}_\Sigma()$.

Для безопасной трассы σ и стимула или реакции z таких, что $\mathbf{Extend2}_\Sigma(\sigma,z)=\mathit{true}$, $\mathbf{Gamma2}_\Sigma(\sigma\cdot\langle z\rangle) = \mathbf{Gamma1}_\Sigma(\sigma,z)$.

34. Доказательство Утверждение 34:

Утверждение непосредственно следует из определения объединения LTS.

35. Доказательство Утверждение 35:

Нам нужно показать, что выполняются правила, определяющие работу машины тестирования (Определение 4:). Пусть стабильное состояние машины – это стабильное состояние LTS $\mathbf{M}$.

(Машина.1) После печати символа разрушения γ машина разрушена: больше ничего не печатается.

В LTS $\mathbf{M}$ γ-переход ведёт в терминальное состояние, а, по правилу LTS 4, символ γ печатается только при проходе такого γ-перехода.

(Машина.2) После печати отказа $A\subseteq C$ машина не может напечатать никакое из разрешённых действий $a\in A$.

В LTS M, по правилу LTS 4, отказ A печатается при прохождении перехода-петли, помеченного этим отказом A, а это происходит в стабильном состоянии, где нет перехода по действию $a \in A$. Следовательно, по правилам LTS 3,4, далее невозможна печать действия $a \in A$.

(Машина.3) Машина может напечатать отказ $A \subseteq C$ только в том случае, когда нажата кнопка “A”, а машина находится в стабильном состоянии.

Печать отказа, по правилам LTS 3,4, происходит при прохождении перехода-петли, помеченного этим отказом, а это происходит в стабильном состоянии.

(Машина.4) Если машина находится в стабильном состоянии, нажата кнопка “A”, $A \subseteq C$, и хотя бы одно из разрешённых кнопкой действий $a \in A$ машина может напечатать, то машина обязана напечатать одно из этих разрешённых действий и отжать кнопку “A”.

В стабильном состоянии, в котором есть переход по действию $a \in A$, нет перехода по отказу A. Поэтому, по правилам LTS 3,4, если машина может и напечатать действие $a \in A$, то она обязана это сделать.

(Машина.5) Если машина находится в стабильном состоянии, нажата кнопка “A”, $A \subseteq C$, но ни одно из разрешённых этой кнопкой действий $a \in A$ машина не может напечатать, то машина обязана напечатать отказ A и отжать кнопку “A”.

В стабильном состоянии, в котором нет переходов ни по одному действию $a \in A$, есть переход по отказу A. Поэтому, по правилам LTS 3,4, если машина может не может напечатать никакого действия $a \in A$, то она обязана напечатать отказ A.

(Машина.6) После печати отказа машина остаётся в том же самом стабильном состоянии: если машина перешла в стабильное состояние s, то любую трассу, которую она может напечатать после печати отказа $A \subseteq C$, она может напечатать и без печати отказа. Формально: $\Sigma(s,A)=\Sigma(s)$.

По правилу LTS 4, отказ печатается при прохождении перехода-петли, а в этом случае состояние не меняется.

(Машина.7) Машина не изменяется в процессе работы, то есть, множество трасс, которые можно ожидать после печати любой трассы σ, равно Σ ***after*** σ, где Σ – множество трасс, ожидающихся с самого начала работы.

Это следует из того, что в процессе работы не меняется сама LTS M.

(Машина.8) Если машина не разрушена, она через конечное ограниченное время либо разрушается, либо переходит в стабильное состояние (моделирование дивергенции разрушением).

Если кнопка не нажата, по LTS 2, машина переходит в стабильное состояние или в состояние γ. Поскольку в состояние γ можно попасть только по γ-переходу, машина разрушается.

Если нажата кнопка, то, по LTS 3, машина выполняет переход по символу $u \neq \tau$. Если $u=\gamma$, то машина разрушается. В противном случае, по LTS 4, кнопка отжимается. Если машина в этот момент находится в нестабильном состоянии, то, по предыдущему случаю, машина разрушается или переходит в стабильное состояние.

(Машина.9) Машина может печатать только те действия $a \in C$, которые разрешены нажатой кнопкой "A", то есть $a \in A$.

Это следует из правил LTS 3 и LTS 4.

36. **Доказательство Утверждение 36:**

<u>βγδ-допустимость</u> непосредственно следует из способа построения βγδ-трасс LTS: переход по разрушению ведёт в терминальное гамма-состояние.

<u>βγδ-согласованность</u> непосредственно следует из способа построения βγδ-трасс LTS: петли отказов рисуются в стабильных состояниях (из них не ведут γ-переходы) при отсутствии соответствующих переходов из этих состояний (переходов по стимулу при блокировке стимула и переходов по всем реакциям при стационарности) и, следовательно, после переходов по отказам.

<u>βγδ-конвергентность</u>. Для того, чтобы βγδ-трасса LTS σ была конвергентна, достаточно, чтобы она заканчивалась хотя бы в одном стабильном состоянии. Если σ заканчивается только в нестабильных состояниях, то, очевидно, из каждого такого состояния выходит либо переход по разрушению, либо бесконечный маршрут τ-переходов, то есть

это состояние дивергентно. В таком случае σ, по построению $\beta\gamma\delta$-трасс LTS, продолжается разрушением.

$\beta\gamma\delta$-замкнутость (замкнутость по ***DRT***-операциям) также непосредственно следует из способа построения $\beta\gamma\delta$-трасс LTS: отказы моделируются петлями в стабильных состояниях.

$\beta\gamma\delta$-полнота (замкнутость по ***Ins***-операции). Пусть $\beta\gamma\delta$-трасса σ заканчивается отказом и a) не продолжается некоторым стимулом $?x$ или b) не продолжается никакими реакциями. Такая трасса в LTS заканчивается в стабильных состояниях, в каждом из которых a) нет перехода по этому стимулу $?x$ или b) нет переходов по реакциям. Следовательно, в каждом из этих состояний определена a) петля-блокировка $\{?x\}$ или b) петля по стационарности δ. Если какая-то трасса λ продолжает трассу σ, то она начинается в одном из этих состояний. Следовательно, при проходе через такое состояние возможна трасса a) $\sigma\cdot\langle\{?x\}\rangle\cdot\lambda$ или b) $\sigma\cdot\langle\delta\rangle\cdot\lambda$.

37. **Доказательство Утверждение 37:**

Состояние $\mu\cdot\langle\gamma\rangle\in\mathbf{\Sigma}$ стабильно, поскольку в нём не определяются никакие переходы (по $\beta\gamma\delta$-допустимости, $\beta\gamma\delta$-трасса, заканчивающаяся на разрушение, ничем не продолжается).

Состояние вида $\mu\cdot o\in\mathbf{\Sigma}$, где $o\neq\epsilon$ и $o\in\beta\delta(C)^*$, стабильно: из него нет τ-перехода по правилам вывода, и из него нет γ-перехода, поскольку после $\beta\delta$-отказа не может быть разрушения по $\beta\gamma\delta$-согласованности модели. В этом состоянии определён переход по стимулу или реакции $z\in C$ тогда и только тогда, когда $\mu\cdot o\cdot\langle z\rangle\in\mathbf{\Sigma}$, что эквивалентно $z\in\mathit{head}(\mathbf{\Sigma}\ \mathit{after}\ \mu\cdot o)$.

Нам осталось показать, что состояние $\mu\in\mathbf{\Sigma}$, не заканчивающееся разрушением или $\beta\delta$-отказом, нестабильно или стабильно, но не порождает $\beta\delta$-отказов. Если μ продолжается в $\mathbf{\Sigma}$ хотя бы одним $\beta\delta$-отказом, по правилу вывода (2), в нём определяется τ-переход и оно нестабильно. В противном случае, по $\beta\gamma\delta$-конвергентности, μ продолжается в $\mathbf{\Sigma}$ каждым стимулом и хотя бы одной реакцией, то есть не порождает $\beta\delta$-отказов.

38. **Доказательство Утверждение 38:**

Обозначим $\mathbf{M}=\{\mu\in C_{\beta\gamma\delta}{}^* \mid \mathit{RefT}(\mu)=\epsilon\}$ – множество $\beta\gamma\delta$-трасс, не заканчивающих на $\beta\delta$-отказ.

Заметим, что в LTS $\mathbf{S}=\boldsymbol{T}_{\beta\gamma\delta}\boldsymbol{2L}_{\gamma}(\mathbf{\Sigma})$ нет дивергентных состояний, так как τ-переход, определяемый правилом вывода (2), ведёт в состояние $\mu\cdot\mathbf{o}$, из которого не ведут τ-переходы (поскольку $\mathbf{o}\neq\epsilon$).

Доказательство будем вести индукцией по длине маршрута S.

Очевидно, что для пустого маршрута утверждение верно:
$\omega((\epsilon,\epsilon))=\epsilon$, $\boldsymbol{traces}_{\beta\gamma\delta}(\epsilon)=\{\epsilon\}=\boldsymbol{DRTIns}(\epsilon)$.

Пусть утверждение верно для маршрута S, заканчивающегося в состоянии $\omega(S)=\mu\in\mathbf{\Sigma}\cap\mathbf{M}$: $\boldsymbol{traces}_{\beta\gamma\delta}(S)=\boldsymbol{DRTIns}(\mu)$.

Рассмотрим возможные продолжения S1 маршрута S.

$\mu\text{—}z\rightarrow\mu\cdot\langle z\rangle$, $z\in C_{\gamma}$.

По правилу вывода (1), $\mu\cdot\langle z\rangle\in\mathbf{\Sigma}$.

Имеем: $\omega(S1)=\mu\cdot\langle z\rangle$,

$\boldsymbol{traces}_{\beta\gamma\delta}(S1) = \boldsymbol{traces}_{\beta\gamma\delta}(S)\cdot\{\langle z\rangle\}$

$= \boldsymbol{DRTIns}(\mu)\cdot\{\langle z\rangle\} = \boldsymbol{DRTIns}(\mu\cdot\langle z\rangle)$.

$\mu\text{—}\tau\rightarrow\mu\cdot\mathbf{o}$, $\mathbf{o}\neq\epsilon$, $\mathbf{o}\in\beta\delta(C)^{*}$.

По правилу вывода (2), $\mu\cdot\mathbf{o}\in\mathbf{\Sigma}$.
Имеем $\omega(S1)=\mu\cdot\mathbf{o}$.
По определению βδ,
$\beta\delta(\mu\cdot\mathbf{o})=\{\{?x\}\mid ?x\in C\setminus\boldsymbol{init}(\mu\cdot\mathbf{o})\}\cup\{\delta\mid !C\cap\boldsymbol{init}(\mu\cdot\mathbf{o})=\varnothing\}$.
По утверждению о стабильности (Утверждение 37:),
$\boldsymbol{init}(\mu\cdot\mathbf{o})=\boldsymbol{head}(\mathbf{\Sigma}\ \boldsymbol{after}\ \mu\cdot\mathbf{o})\cap C$.
Поэтому

$$\begin{aligned}\beta\delta(\mu\cdot\mathbf{o}) &= \{\{?x\}\mid ?x\in ?C\setminus\boldsymbol{head}(\mathbf{\Sigma}\ \boldsymbol{after}\ \mu\cdot\mathbf{o})\}\\ &\quad\cup\{\ \delta\ \mid !C\cap\boldsymbol{head}(\mathbf{\Sigma}\ \boldsymbol{after}\ \mu\cdot\mathbf{o})=\varnothing\}\\ &= \{\{?x\}\mid ?x\in ?C\setminus\boldsymbol{head}((\mathbf{\Sigma}\ \boldsymbol{after}\ \mu)\ \boldsymbol{after}\ \mathbf{o})\}\\ &\quad\cup\{\ \delta\ \mid !C\cap\boldsymbol{head}((\mathbf{\Sigma}\ \boldsymbol{after}\ \mu)\ \boldsymbol{after}\ \mathbf{o})=\varnothing\}.\end{aligned}$$

По Утверждение 8:, $(\beta\delta(\mu\cdot\mathbf{o}))^{*}=\boldsymbol{DRTIns}_{(\mathbf{\Sigma}\ \boldsymbol{after}\ \mu)}(\mathbf{o})$.
Имеем:
$\boldsymbol{traces}_{\beta\gamma\delta}(S1)$
$= \boldsymbol{traces}_{\beta\gamma\delta}(S)\cdot(\beta\delta(\mu\cdot\mathbf{o}))^{*}$
$= \boldsymbol{traces}_{\beta\gamma\delta}(S)\cdot\boldsymbol{DRTIns}_{(\mathbf{\Sigma}\ \boldsymbol{after}\ \mu)}(\mathbf{o})$

$= \mathit{DRTIns}(\mu)\cdot\mathit{DRTIns}_{(\Sigma\ \mathit{after}\ \mu)}(o)$

$=$ (* поскольку трасса μ не заканчивается на отказ *)

$= \mathit{DRTIns}(\mu\cdot o)$.

$\mu\text{—}\tau\rightarrow\mu\cdot o\text{—}z\rightarrow\mu\cdot o\cdot\langle z\rangle$, $z\in C$, $o\neq\epsilon$, $o\in\beta\delta(C)^*$.

По правилам вывода (2) и (1), $\mu\cdot o\cdot\langle z\rangle\in\Sigma$.

Имеем $\omega(S1)=\mu\cdot o\cdot\langle z\rangle$.

Учитывая утверждение для маршрута, заканчивающегося в состоянии $\mu\cdot o$, имеем:

$\mathit{traces}_{\beta\gamma\delta}(S1)=\mathit{DRTIns}(\mu\cdot o)\cdot\{\langle z\rangle\}=\mathit{DRTIns}(\mu\cdot o\cdot\langle z\rangle)$.

39. Доказательство Утверждение 39:

Строгая конвергентность следует из того, что τ-переход, определяемый правилом вывода (2), ведёт в состояние $\mu\cdot o$, из которого не ведут τ-переходы (поскольку $o\neq\epsilon$).

Обозначим $M=\{\mu\in C_{\beta\gamma\delta}^{\ *}\mid \mathit{RefT}(\mu)=\epsilon\}$ – множество $\beta\gamma\delta$-трасс, не заканчивающих на $\beta\delta$-отказ.

1) Докажем вложенность $\Sigma\subseteq\mathit{traces}_{\beta\gamma\delta}(S)$.

Нам достаточно показать, что для каждой трассы $\mu\in\Sigma$ имеет место $(S\ \mathit{after}\ \mu)\neq\varnothing$. Мы будем доказывать более сильное утверждение: $\mu\in(S\ \mathit{after}\ \mu)$.

Доказательство будем вести индукцией по длине трассы $\mu\in\Sigma$.

Поскольку начальное состояние LTS S это пустая трасса,
$\epsilon\in(S\ \mathit{after}\ \epsilon)$.

Пусть утверждение верно для трассы $\mu\in\Sigma\cap M$.

Рассмотрим возможные продолжения трассы.

$\mu\cdot\langle z\rangle\in\Sigma$, где $z\in C_\gamma$.

По правилу вывода (1), имеется переход $\mu\text{—}z\rightarrow\mu\cdot\langle z\rangle$.

Поэтому $\mu\in(S\ \mathit{after}\ \mu)$ влечёт $\mu\cdot\langle z\rangle\in(S\ \mathit{after}\ \mu\cdot\langle z\rangle)$.

$\mu\cdot o\in\Sigma$, где $o\in\beta\delta(C)^*$ и $o\neq\epsilon$.

По правилу вывода (2), имеется переход $\mu\text{—}\tau\rightarrow\mu\cdot o$.

По утверждению о стабильности (Утверждение 37:),

$\mathit{init}(\mu\cdot o)=\mathit{head}(\Sigma\ \mathit{after}\ \mu\cdot o)\cap C$.

Если $\{?x\}\in\mathit{Im}(o)$, то, по $\beta\gamma\delta$-согласованности модели, $?x\notin\mathit{head}(\Sigma\ \mathit{after}\ \mu\cdot o)$ и, следовательно, $?x\notin\mathit{init}(\mu\cdot o)$, то есть $\{?x\}\in\beta\delta(\mu\cdot o)$.

Если $\delta\in\mathit{Im}(o)$, то, по $\beta\gamma\delta$-согласованности модели,

$\forall!y\in!C\ \ !y\notin\mathit{head}(\Sigma\ \mathit{after}\ \mu\cdot o)$ и, следовательно,

$\forall!y\in!C\ \ !y\notin\mathit{init}(\mu\cdot o)$, то есть $\delta\in\beta\delta(\mu\cdot o)$.

Таким образом, $\mathit{Im}(o)\subseteq\beta\delta(\mu\cdot o)$.

Поэтому $\mu\in(S\ \mathit{after}\ \mu)$ влечёт $\mu\cdot o\in(S\ \mathit{after}\ \mu\cdot o)$.

$\mu\cdot o\cdot\langle z\rangle\in\Sigma$, где $o\in\beta\delta(C)^*$, $o\neq\epsilon$ и $z\in C$

(трассы $\mu\cdot o\cdot\langle\gamma\rangle$ не может быть по $\beta\gamma\delta$-согласованности модели).

Из доказанного случая для трассы $\mu\cdot o$ следует $\mu\cdot o\in(S\ \mathit{after}\ \mu\cdot o)$.

По правилу вывода (1), имеется переход $\mu\cdot o\text{—}z\rightarrow\mu\cdot o\cdot\langle z\rangle$.

Поэтому $\mu\cdot o\cdot\langle z\rangle\in(S\ \mathit{after}\ \mu\cdot o\cdot\langle z\rangle)$.

2) Теперь докажем вложенность $\Sigma\supseteq\mathit{traces}_{\beta\gamma\delta}(S)$.

Согласно правилам вывода, каждый маршрут заканчивается в состоянии, которым является некоторая $\beta\gamma\delta$-трасса Σ.

Имеем $\mathit{traces}_{\beta\gamma\delta}(S)=\cup_\circ\mathit{traces}_{\beta\gamma\delta}{}_\circ\mathit{runs}(S)$.

По Утверждение 38:, для каждого маршрута $S\in\mathit{runs}(S)$ его $\beta\gamma\delta$-трассы $\mathit{traces}_{\beta\gamma\delta}(S)=\mathit{DRTIns}(\omega(S))$.

Поэтому $\mathit{traces}_{\beta\gamma\delta}(S)=\cup_\circ\mathit{DRTIns}_\circ\omega_\circ\mathit{runs}(S)\subseteq\Sigma$.

Совокупность утверждений 1 и 2 доказывает общее утверждение.

40. Доказательство Утверждение 40:

Состояние $\{\epsilon\}$ стабильно, поскольку, по правилам вывода (1,2), в нём не определяются никакие переходы.

По правилам вывода (1,2), в состоянии **T** ***stable*** не определены τ- и γ-переходы (стабильное дерево не содержит γ-трассы) и определён переход по стимулу или реакции $z \in C$ тогда и только тогда, когда $z \in$ ***head***(**T**).

Нам осталось показать, что состояние **T** ***unstable***, отличное от $\{\epsilon\}$, нестабильно.

Если **T** содержит γ-трассу, то, по правилу вывода (1), в состоянии **T** определяется γ-переход и оно нестабильно.

Пусть **T** не содержит γ-трассу.

Если в разложении $\mathbf{T}=\mathbf{T}_c \cup (\cup(\mathbf{T}_s))$ множество стабильных деревьев $\mathbf{T}_s$ не пусто, по правилу вывода (2), в состоянии **T** определяется τ-переход и оно нестабильно.

Если в разложении $\mathbf{T}=\mathbf{T}_c \cup (\cup(\mathbf{T}_s))$ множество $\mathbf{T}_s=\varnothing$, дерево $\mathbf{T}=\mathbf{T}_c$, то есть контрстабильно. Поскольку **T** βγδ-модель и не содержит γ-трассу, пустая трасса в нём конвергентна. А тогда, по определению конвергентности дерева, **T** конвергентно. Контрстабильное конвергентное дерево стабильно, что противоречит условию **T** ***unstable***.

41. Доказательство Утверждение 41:

Строгая конвергентность следует из того, что τ-переход, определяемый правилом вывода (2), ведёт в состояние **T`**, из которого не ведут τ-переходы (поскольку дерево **T`** ***stable***).

Мы будем ниже рассматривать только достижимые состояния LTS.

1) Докажем вложенность $\Sigma \subseteq \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$.

A. Сначала докажем утверждение 1) для трассы μ, которая не заканчивается на βδ-отказ.

Мы будем доказывать более сильный вариант утверждения A:
для каждой трассы $\mu\cdot\lambda \in \Sigma$, где μ не заканчивается на βδ-отказ, $\mu \in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$ и существует такое состояние $\mathbf{T} \in (\mathbf{S}\ \mathit{after}\ \mu)$, что $\lambda \in \mathbf{T}$.

Доказательство будем вести индукцией по числу стимулов, реакций и разрушений в трассе μ, то есть по длине проекции $\mu{\downarrow}C_\gamma$.

Для $\mu=\epsilon$ и любой трассы $\epsilon\cdot\lambda\in\Sigma$ пустая трасса $\epsilon\in\Sigma\cap traces_{\beta\gamma\delta}(S)$ и заканчивается в начальном состоянии $\Sigma\in(S\ after\ \epsilon)$, а $\lambda\in\Sigma$.

Пусть утверждение А верно для трассы μ и докажем его для трассы $\mu\cdot o\cdot\langle z\rangle\in\Sigma$, где $o\in\beta\delta(C)^*$ трасса отказов, а $z\in C_\gamma$ – базовый символ.

Пусть $\mu\cdot o\cdot\langle z\rangle\cdot\lambda\in\Sigma$.

По предположению индукции, $\mu\in traces_{\beta\gamma\delta}(S)$ и существует такое состояние $T\in(S\ after\ \mu)$, что $o\cdot\langle z\rangle\cdot\lambda\in T$.

Пусть $o=\epsilon$.

Тогда, по правилу вывода (1), трасса $\mu\cdot\langle z\rangle\in traces_{\beta\gamma\delta}(S)$, $(T\ after\ \langle z\rangle)\in(S\ after\ \mu\cdot\langle z\rangle)$ и $\lambda\in(T\ after\ \langle z\rangle)$.

Пусть $o\neq\epsilon$.

Если дерево T стабильно, обозначим $T`=T$ (тогда $T\Longrightarrow T`$). В противном случае, по правилу вывода (2), для некоторого $T`\in T_s$ выполняется $o\cdot\langle z\rangle\cdot\lambda\in T`$, и $T\text{—}\tau\rightarrow T`$ (тогда $T\Longrightarrow T`$).

По утверждению о стабильности (Утверждение 40:), $o\in traces_{\beta\gamma\delta}(T`)$.
А тогда
$\mu\in traces_{\beta\gamma\delta}(S)$ & $T\in(S\ after\ \mu)$ & $T\Longrightarrow T`$ & $o\in traces_{\beta\gamma\delta}(T`)$ влечёт $\mu\cdot o\in traces_{\beta\gamma\delta}(S)$ и $T`\in(S\ after\ \mu\cdot o)$.
Из $T`$ *stable* и $o\cdot\langle z\rangle\cdot\lambda\in T`$ следует $\langle z\rangle\cdot\lambda\in T`$ (правило стаб.1).
Отсюда, по правилу вывода (1), $\mu\cdot o\cdot\langle z\rangle\in traces_{\beta\gamma\delta}(S)$, $(T`\ after\ \langle z\rangle)\in(S\ after\ \mu\cdot o\cdot\langle z\rangle)$ и $\lambda\in(T`\ after\ \langle z\rangle)$.

Утверждение А доказано.

В. Теперь докажем первое утверждение для трассы, заканчивающейся $\beta\delta$-отказом.

Такая трасса имеет вид $\mu\cdot o\in\Sigma$, где μ не заканчивается на $\beta\delta$-отказ, а $o\in\beta\delta(C)^*$ непустая трасса отказов $o\neq\epsilon$.

Действительно, по утверждению А, $\mu\in \mathit{traces}_{\beta\gamma\delta}(S)$ и существует такое состояние $T\in(S\ \mathit{after}\ \mu)$, что $o\in T$.

Если дерево T стабильно, обозначим $T`=T$ (тогда $T\Rightarrow T`$). В противном случае, по правилу вывода (2), для некоторого $T`\in T_s$ выполняется $o\in T`$, и $T\text{—}\tau\rightarrow T`$ (тогда $T\Rightarrow T`$).
По утверждению о стабильности (Утверждение 40:), $o\in \mathit{traces}_{\beta\gamma\delta}(T`)$.
А тогда
$\mu\in \mathit{traces}_{\beta\gamma\delta}(S)\ \&\ T\in(S\ \mathit{after}\ \mu)\ \&\ T\Rightarrow T`\ \&\ o\in \mathit{traces}_{\beta\gamma\delta}(T`)$
влечёт $\mu\cdot o\in \mathit{traces}_{\beta\gamma\delta}(S)$.

Первое утверждение доказано.

2) Теперь докажем вложенность $\Sigma\supseteq \mathit{traces}_{\beta\gamma\delta}(S)$.

Поскольку $\mathit{traces}_{\beta\gamma\delta} = \mathit{trace}\circ\mathit{runs}_{\beta\gamma\delta}$, нам достаточно показать, что для каждого $\beta\gamma\delta$-маршрута $S\in \mathit{runs}_{\beta\gamma\delta}(S)$ его трасса $\sigma=\mathit{trace}(S)\in\Sigma$. Мы будем доказывать также, что $\omega(S)=T\subseteq(\Sigma\ \mathit{after}\ \sigma)$.

Доказательство будем вести индукцией по длине $\beta\gamma\delta$-маршрута S.

Очевидно, что для пустого маршрута второе утверждение верно:
$\epsilon=\mathit{trace}((\Sigma,\epsilon))\in\Sigma$ и $\omega((\Sigma,\epsilon))=\Sigma\subseteq(\Sigma\ \mathit{after}\ \epsilon)$.

Пусть утверждение верно для $\beta\gamma\delta$-маршрута S и докажем его для $\beta\gamma\delta$-маршрута $S`=S\cdot\langle e\rangle$. По определению $\beta\gamma\delta$-маршрута, S не содержит γ-перехода.

По предположению индукции,
$\sigma=\mathit{trace}(S)\in\Sigma$ и $\omega(S)=T\subseteq(\Sigma\ \mathit{after}\ \sigma)$.

Переход e порождён одним из правил вывода (1,2) или петлёй $\beta\delta$-отказа.

Для правила вывода (1), $\mathit{trace}(S`)=\sigma\cdot\langle z\rangle$, $\sigma\cdot\langle z\rangle\in\Sigma$ и
$\omega(S`)=(T\ \mathit{after}\ \langle z\rangle)\subseteq(\Sigma\ \mathit{after}\ \sigma\cdot\langle z\rangle)$.

Для правила вывода (2), $trace(S`)=\sigma$, $\sigma\in\Sigma$ и
$\omega(S`)=T\subseteq(\Sigma\ after\ \sigma)$.

Для петли по βδ-отказу o, $trace(S`)=\sigma\cdot\langle o\rangle$ и
$\omega(S`)=T\subseteq(\Sigma\ after\ \sigma)$.
Поскольку S не содержит γ-перехода, $T\neq\{\epsilon\}$.
Поэтому, по утверждению о стабильности (Утверждение 40:), дерево T стабильно и $init(T)=head(T)\cap C$.
По конвергентности стабильного дерева, $\langle o\rangle\in T$.
Поэтому $\sigma\cdot\langle o\rangle\in\Sigma$.
По свойству βγδ-замкнутости, $T=(T\ after\ \langle o\rangle)\subseteq(\Sigma\ after\ \sigma\cdot\langle o\rangle)$.

Второе утверждение доказано.
Учитывая первое утверждение, общее утверждение доказано.

42. Доказательство Утверждение 42:

Непосредственно следует из Утверждение 36: и Утверждение 39: (или Утверждение 41:).

43. Доказательство Утверждение 43:

1) непосредственно следует из определения преобразований ***LTest*** и $L_\theta 2L_{\beta\delta}$, а 2) следует из 1) и определения LTS-теста.

44. Доказательство Утверждение 44:

Обозначим $T=LTest(T)$.

Пусть в композиции $I|\downarrow T$ $\exists it\in der(i_0t_0)$ такое, что $t=fail$.

Это может быть тогда и только тогда, когда существует такая βγδ-трасса реализации $\sigma\in traces_{\beta\gamma\delta}(I)$, что в преобразованной тестовой LTS $L_\theta 2L_{\beta\delta}(T)$ есть такая же βγδ-трасса $\sigma\in traces\circ L_\theta 2L_{\beta\delta}(T)$, заканчивающаяся в состоянии t.
Последнее, по Утверждение 43:, эквивалентно $\sigma\in T$.
Отсюда, по определению ***passes*** для βγδ-деревьев и LTS, непосредственно следует доказываемое утверждение.

45. Доказательство Утверждение 45:

Пусть βγδ-трасса σ безопасна.

Сначала покажем, что в *S*-ветвящейся LTS множество $\mathbf{R}_\sigma$ маршрутов с βγδ-трассой σ конечно. В любом состоянии, достижимом по βγδ-трассе μ<σ, должно быть конечное число τ-переходов и переходов по стимулу или реакции[74] σ(|μ|+1), а в любом состоянии, достижимом по βγδ-трассе σ, должно быть конечное число τ-переходов. Тем самым, дерево $\mathbf{R}_{\le\sigma}=\cup\{\mathbf{R}_\mu \mid \mu\le\sigma\}$ конечно-ветвящееся. Это дерево не может содержать бесконечного маршрута, поскольку такой маршрут для конечной βγδ-трассы μ≤σ содержал бы бесконечный постфикс τ-переходов, а тогда была бы βγδ-трасса $\mu\cdot\langle\gamma\rangle$, что противоречит безопасности βγδ-трассы σ. Отсюда, по теореме Кёнига [KÖNIG], дерево $\mathbf{R}_{\le\sigma}$ как конечно-ветвящееся дерево, не содержащее бесконечно возрастающей цепочки маршрутов, конечно. Следовательно, конечно его подмножество $\mathbf{R}_\sigma$.

Теперь покажем, что, если для каждой безопасной трассы σ множество маршрутов $\mathbf{R}_\sigma$ конечно, то LTS *S*-ветвящаяся. Для каждого стимула или реакции z, безопасного после σ, трасса $\sigma\cdot\langle z\rangle$ тоже безопасна. Из конечности множества $\mathbf{R}_{\sigma\cdot\langle z\rangle}$ следует, что в каждом состоянии после трассы σ, должно быть определено конечное число τ-переходов и переходов по z. Тем самым, LTS *S*-ветвящаяся.

46. Доказательство Утверждение 46:

Пусть C⊆Z и S∈**L1**(C). Обозначим $\Sigma=\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$.

Построим алгоритмы, задающие трассовую спецификацию Σ способом **T1**. С учётом алгоритмов, заданных для **L1**-спецификации, нам нужно построить оракулы $\mathbf{Safe}_\Sigma()$, $\mathbf{Gamma1}_\Sigma(\sigma,z)$ и $\mathbf{Extend1}_\Sigma(\sigma,u)$.

По Утверждение 45:, дерево маршрутов с безопасной βγδ-трассой σ конечно. Отсюда множество состояний **S** ***after*** σ конечно. Опишем алгоритм построения этого множества.

Сначала рассмотрим построение τ-замыкания (замыкание по τ-маршрутам) любого конечного множества состояний, каждое из которых *S*-достижимо. Поскольку в каждом состоянии этого τ-замыкания

[74] Если σ(|μ|+1) отказ, то ему не соответствует переход в маршруте.

определено конечное число τ-переходов, это легко делается с помощью итератора $\mathbf{Tau1_S}(s)$.

Теперь рассмотрим случай пустой трассы $\sigma=\epsilon$. Очевидно, **S** ***after*** ϵ совпадает с τ-замыканием множества $\{s_0\}$, состоящего из одного начального состояния.

Пусть теперь у нас построено множество состояний **S** ***after*** σ. Построим множество состояний **S** ***after*** $\sigma\cdot\langle u\rangle$ для символа u, безопасного после σ.

Если $u=z$ стимул или реакция, то с помощью итератора $\mathbf{Extend1_S}(s,z)$ определяем множество постсостояний переходов из состояний **S** ***after*** σ по z. Это множество конечно, поскольку LTS *S*-ветвящаяся. Затем строим его τ-замыкание, то есть **S** ***after*** $\sigma\cdot\langle z\rangle$.

Если $u=\{?x\}$ блокировка, то с помощью итераторов $\mathbf{Extend1_S}(s,?x)$ и $\mathbf{Tau1_S}(s)$ определяем конечное подмножество стабильных состояний множества **S** ***after*** σ, в которых нет перехода по стимулу $?x$.

Если $u=\delta$ стационарность, то с помощью итераторов $\mathbf{Y_C}$ и $\mathbf{Tau1_S}(s)$ определяем конечное подмножество стационарных состояний множества **S** ***after*** σ, то есть стабильных состояний, в которых нет переходов по реакциям.

Далее: $\mathbf{Safe_\Sigma}() = \mathbf{Safe_S}()$.

Построим оракул $\mathbf{Gamma1_\Sigma}(\sigma,z)$.

Стимул или реакция z разрушающий после βγδ-трассы σ, если он разрушающий хотя бы в одном состоянии после этой βγδ-трассы.
Поэтому нам достаточно применить оракул $\mathbf{Gamma1_S}(s,z)$ в каждом состоянии $s\in$(**S** ***after*** σ).

Теперь построим оракул $\mathbf{Extend1_\Sigma}(\sigma,u)$.

Стимул $u=?x$ безопасен после βγδ-трассы, если он неразрушающий после неё, что проверяется уже построенным оракулом

Gamma1$_\Sigma$(σ,?x). Блокировка u={?x} безопасна после βγδ-трассы, если стимул ?x безопасен после неё.

Реакция u=!y или стационарность u=δ безопасны после βγδ-трассы, если все реакции неразрушающие после неё. Это можно проверить с помощью итератора **Y**$_C$ конечного множества реакций и уже построенного оракула **Gamma1**$_\Sigma$(σ,!y).

Безопасный стимул или реакция u=z продолжают безопасную βγδ-трассу σ, если переход по z определён хотя бы в одном из состояний **S** ***after*** σ, что проверяется итератором **Extend1**$_S$(s,z) (он должен выдать хотя бы одно постсостояние перехода по z).

Блокировка безопасного стимула u={?x} продолжает безопасную βγδ-трассу σ, если стимул ?x не определён хотя бы в одном из стабильных состояний **S** ***after*** σ. Стабильность проверяется итератором **Tau1**$_S$(s), а отсутствие перехода – итератором **Extend1**$_S$(s,z).

Стационарность u=δ проверяется отсутствием переходов по каждой реакции из конечного множества реакций (перечисляемым итератором **Y**$_C$) хотя бы в одном из стабильных (итератор **Tau1**$_S$(s)) состояний конечного множества **S** ***after*** σ.

Таким образом, у нас выполнены условия Утверждение 30:, и, тем самым, требуемое утверждение доказано.

47. Доказательство Утверждение 47:

Нам надо показать, что в каждом *S*-достижимом состоянии s определено конечное число 1) τ-переходов и 2) переходов по каждому стимулу или реакции z, безопасному после некоторой безопасной βγδ-трассы, заканчивающейся в состоянии s.

1) Состояние s либо начальное, либо является постсостоянием перехода s_1—z→s из *S*-достижимого состояния s_1. Кроме того, в s не может начинаться бесконечный τ-маршрут (нет дивергенции) или τ-маршрут, заканчивающийся в состоянии с γ-переходом. Следовательно, по *S*-τ-ограниченности, дерево τ-маршрутов, начинающихся в s, конечно. А тогда конечно и число τ-переходов из s.

2) Для перехода $s \text{—}z\rightarrow s_2$ постсостояние s_2 *S*-достижимо и, следовательно, в s_2 не может начинаться бесконечный τ-маршрут (нет дивергенции) или τ-маршрут, заканчивающийся в состоянии с γ-переходом. Следовательно, по *S*-τ-ограниченности, дерево τ-маршрутов, начинающихся во всех таких постсостояниях s_2, конечно. А тогда конечно и число переходов $s \text{—}z\rightarrow s_2$.

48. Доказательство Утверждение 48:

Пусть $C \subseteq Z$ и $S \in \mathbf{L2}(C)$.
Построим алгоритмы, задающие LTS-спецификацию **S** способом **L1**. С учётом алгоритмов, заданных для **L2**-спецификации, нам нужно построить оракулы $\mathbf{Safe}_S()$ и $\mathbf{Gamma1}_S(s,z)$ и итераторы $\mathbf{Extend1}_S(s,z)$ и $\mathbf{Tau1}_S(s)$.

Построим оракул $\mathbf{Safe}_S()$.

Мы будем использовать итераторы $\mathbf{Tau2}_S$ и $\mathbf{Gamma2}_S$. На каждом шаге у нас будет конечная LTS τ-замыкания **S`**, все состояния которой достижимы по τ-переходам из начального состояния (в начале одно начальное состояние). В каждом состоянии **S`**, где нет γ-перехода, задан итератор переходов по τ. Построение τ-замыкания состоит из последовательности циклов итерации. В начале цикла все состояния **S`** помечены. Мы будем применять τ-итераторы в помеченных состояниях (в каждом состоянии итератор применяется один раз, пока он не исчерпает все задаваемые им τ-переходы), одновременно снимая метки и добавляя новые τ-переходы и, быть может, новые состояния без пометок. После того как все помеченные состояния **S`** исчерпаны, заново размечаем все её состояния и повторяем цикл итерации (τ-итераторы в старых состояниях, не исчерпавшие все задаваемые ими τ-переходы, продолжают свою работу, а τ-итераторы в новых состояниях начинают свою работу с начала).

Одновременно с построением LTS **S`** мы будем контролировать разрушение для каждого добавляемого состояния с помощью оракула $\mathbf{Gamma2}_S$, и появление τ-циклов в **S`**. При обнаружении γ-перехода или τ-цикла, построение прекращается, и спецификация **S** считается опасной. Очевидно, это может быть обнаружено за конечное время. В противном случае, поскольку LTS **S** *S*-τ-ограничена, LTS τ-замыкания

S` конечна и, следовательно, процесс закончится за конечное время. Спецификация **S** считается безопасной.

Построим оракул **Gamma1**$_{\mathbf{S}}$(s,z), где $s \in$ (**S** ***after*** σ) и $\sigma \in$ ***safe***$_{\beta\gamma\delta}$(**S**).

Чтобы проверить разрушаемость стимула или реакции z в *S*-достижимом состоянии s, нужно проверить, что из постсостояния s` некоторого перехода s—z→s` достижим по τ-переходам γ-переход или дивергентное состояние.

Поскольку LTS *S*-τ-ограничена, множество таких переходов по z перечислимо, и, используя итератор **Extend2**$_{\mathbf{S}}$(s,z), мы можем их перечислять. В процессе перечисления мы будем проверять достижимость по τ-переходам γ-перехода или τ-цикла. Это делается аналогично последовательности циклов итерации для проверки безопасности спецификации (построение оракула **Safe**$_{\mathbf{S}}$), но только 1) в качестве начального состояния мы возьмём первое состояние s_1`, возвращаемое итератором **Extend2**$_{\mathbf{S}}$(s,z), и 2) после каждого цикла итерации мы будем добавлять в LTS τ-замыкания ещё одно состояние – следующее состояние, возвращаемое итератором **Extend2**$_{\mathbf{S}}$(s,z). Поскольку LTS *S*-τ-ограничена, этот процесс закончится через конечное время либо построением τ-замыкания множества постсостояний всех переходов из s по безопасному стимулу z, либо обнаружением γ-перехода или τ-цикла.

Итератор **Extend1**$_{\mathbf{S}}$(s,z), очевидно, совпадает с итератором **Extend2**$_{\mathbf{S}}$(s,z) на поддомене стимулов и реакций, безопасных после некоторой безопасной трассы, заканчивающейся в состоянии s.

Итератор **Tau1**$_{\mathbf{S}}$(s), очевидно, совпадает с итератором **Tau2**$_{\mathbf{S}}$(s) на поддомене *S*-достижимых состояний.

Таким образом, у нас выполнены условия Утверждение 46:, и, тем самым, требуемое утверждение доказано.

49. Доказательство Утверждение 49:

Любой добавленный переход из старого состояния s – это переход по стимулу s—?x→, по которому раньше не было перехода из s.

Поскольку старые состояния и переходы, а также начальное состояние, сохраняются, удалена может быть только такая βγδ-трасса исходной LTS, которая имеет вид $\mu\cdot\langle\{?x\}\rangle\cdot\lambda$, где βγδ-трасса μ сохраняется, и появляется (или она была раньше) βγδ-трасса $\mu\cdot\langle?x\rangle$.

Также любая добавленная βγδ-трасса имеет вид $\mu\cdot\langle?x\rangle\cdot\lambda$, где βγδ-трасса μ была раньше. Если состояние s нестабильно, трасса μ не заканчивается на отказ, а если стабильно, то в нём до пополнения блокировался стимул $?x$, и, следовательно, трасса μ продолжалась блокировкой $\{?x\}$.

50. **Доказательство Утверждение 50:**

Сначала покажем, что преобразования ***TLDemonic*** и ***TLGamma*** являются трассовыми ***TL***-пополнениями.

Для доказательства достаточно рассмотреть LTS $\mathbf{S1}=\boldsymbol{T_{\beta\gamma\delta}2L_{\gamma}}(\boldsymbol{\Sigma})$ и определить в её состояниях переходы по отсутствующим стимулам так, как это диктуется преобразованием $\boldsymbol{Comp_0}$; это преобразование обозначим $\boldsymbol{Scomp_0}$. Заметим, что преобразования $\boldsymbol{T_{\beta\gamma\delta}2L_{\gamma}}$ и $\boldsymbol{Scomp_0}$, вообще говоря, не алгоритмичны. Но нам это и не требуется: позже мы покажем алгоритмичность преобразования $\boldsymbol{Comp_0}$, а здесь нам нужно показать только, что оно является трассовым ***TL***-пополнением.

Состояние LTS $\mathbf{S1}$ – это βδ-трасса σ.
Удаляя блокировки, получаем δ-трассу $\mu=\sigma\uparrow\beta(C)$.
В состоянии σ определяем отсутствующий переход по стимулу $?x$, если преобразованием $\boldsymbol{Comp_0}$ δ-трасса μ продолжается этим стимулом по правилу вывода (1).
Этот переход проводим в (быть может, новое) состояние $\mu\cdot\langle?x\rangle$.
Переходы из новых состояний определяются так же, как это делается преобразованием $\boldsymbol{Comp_0}$.

Оставшиеся в LTS $\boldsymbol{Scomp_0}(\mathbf{S1})$ блокировки удаляем с помощью пополнения состояний $\boldsymbol{Scomp_d}$ или $\boldsymbol{Scomp_\gamma}$.
Получаем LTS $\mathbf{S2}=\boldsymbol{Scomp_d}\circ\boldsymbol{Scomp_0}(\mathbf{S1})$ или, соответственно, $\mathbf{S2}=\boldsymbol{Scomp_\gamma}\circ\boldsymbol{Scomp_0}(\mathbf{S1})$.

Состоянию μ LTS $\mathbf{S}=\boldsymbol{Scomp_d}\circ\boldsymbol{Comp_0}(\boldsymbol{\Sigma})$ или, соответственно, $\mathbf{S}=\boldsymbol{Scomp_\gamma}\circ\boldsymbol{Comp_0}(\boldsymbol{\Sigma})$ соответствует множество состояний (δ-трасс) σ LTS $\mathbf{S2}$, из которых δ-трасса μ получается удалением блокировок и повторных

стационарностей (стационарностей, следующих за стационарностью). Состояние μ вносит в δ-трассы LTS **S** то и только то, что после описанного пополнения состояний, будет вноситься соответствующими ему состояниями σ в δ-трассы LTS **S2**.

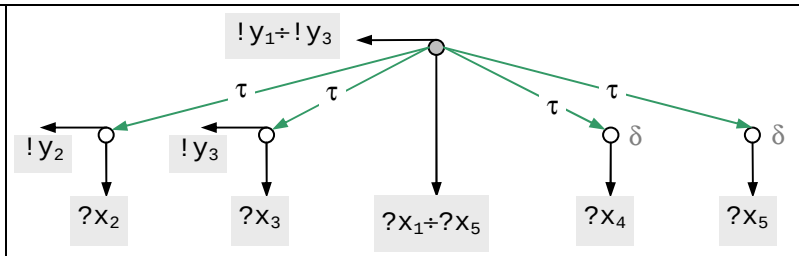
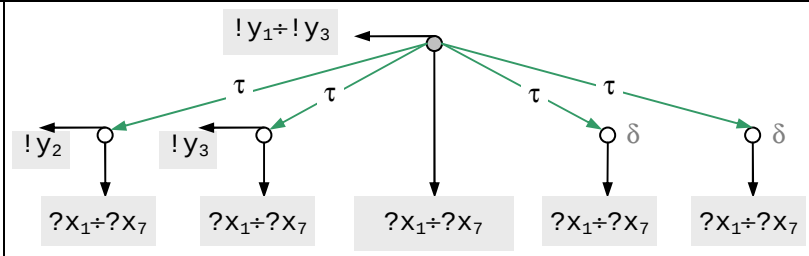
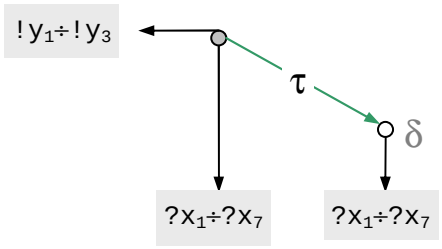
Иными словами, в LTS **S2** «веер» τ-переходов из нестабильного состояния σ в стабильные состояния вида $\sigma \cdot \mathbf{o}$, где $\mathbf{o}$ трасса отказов, разбивается на два «веера»:

1) τ-переходы в стационарные состояния и
2) τ-переходы в нестационарные состояния.

Все стационарные состояния можно без изменения множества $\beta\gamma\delta$-трасс слить в одно стационарное состояние, поскольку эти состояния порождают единственный отказ – стационарность.

Все нестационарные состояния также можно без изменения множества $\beta\gamma\delta$-трасс слить в одно нестабильное состояние вместе с началом τ-перехода – состоянием σ, поскольку в нестабильных состояниях нет отказов.

После такого слияния, выполненного по всей LTS **S2**, мы получим LTS **S** (Рис.112).

преобразования ***Demonic*** и ***Gamma*** как пополнения состояний LTS $T_{\beta\gamma\delta}2L_{\gamma}(\Sigma)$	
алфавит $C=\{?x_1,?x_2,?x_3,?x_4,?x_5,?x_6,?x_7,!y_1,!y_2,!y_3\}$	
S1=$T_{\beta\gamma\delta}2L_{\gamma}(\Sigma)$	
S2=$Scomp_{d}\circ Scomp_{0}$(**S1**) *или* **S2**=$Scomp_{\gamma}\circ Scomp_{0}$(**S1**)	
S=$Scomp_{d}\circ Comp_{0}(\Sigma)$ *или* **S**=$Scomp_{\gamma}\circ Comp_{0}(\Sigma)$	
Рис.112.	

Отсутствие блокировок в пополненных спецификациях ***TLDemonic***(Σ) и ***TLGamma***(Σ) непосредственно следует из того, что пополнения $Scomp_{d}$ и $Scomp_{\gamma}$ не оставляют блокировок.

Сохранение семантики отношения ***ioco*** следует из всего определения пополнений.

51. Доказательство Утверждение 51:

Для βδ-модели **T1** задаются алгоритмы: итератор стимулов $\mathbf{X}_C$, итератор реакций $\mathbf{Y}_C$, оракул разрушения $\mathbf{Gamma1}_{\Sigma}(\sigma,z)$ и оракул безопасного продолжения $\mathbf{Extend1}_{\Sigma}(\sigma,u)$. Поскольку модель без разрушения, оракул безопасности $\mathbf{Safe}_{\Sigma}()\equiv true$, а оракул безопасного продолжения $\mathbf{Extend1}_{\Sigma}(\sigma,u)$ применяется ко всем символам: стимулам, реакциям и отказам.

Результат алгоритмического пополнения $AComp_0$ – LTS-модель, заданная способом **L2**. Для этого способа нужно задать итераторы стимулов $\mathbf{X}_{C`}$ и

реакций $\mathbf{Y}_{C`}$, оракул разрушения **Gamma2**, итераторы переходов **Extend2** и **Tau2**.

Поскольку $C`=C\cup\{!\mathit{error}\}$, $\mathbf{X}_{C`}=\mathbf{X}_C$, а итератор реакций $\mathbf{Y}_{C`}$ сначала перечисляет реакцию $!\mathit{error}$, а потом использует итератор $\mathbf{Y}_C$ для перечисления остальных реакций.

При отсутствии разрушения оракул разрушения $\mathbf{Gamma2}_{\mathit{AComp}_0(\Sigma)}(s)\equiv\mathit{false}$.

Построим итератор $\mathbf{Extend2}_{\mathit{AComp}_0(\Sigma)}(s,z)$.

Параметрами итератора являются состояние s LTS $\mathit{AComp}_0(\Sigma)$ и стимул или реакция z. Итератор перечисляет постсостояния переходов по z из состояния s. Состояние s – это некоторая δ-трасса $s=\mu$. Переход $\mu\text{—}z\rightarrow\mu\cdot\langle z\rangle$ определяется правилами вывода (1), (2) и (4) и, если существует, то единственный. Согласно этим правилам нужно перебрать все δ-трассы $\mu`\in\boldsymbol{D}(\mu)\cap\boldsymbol{\Sigma}$.

Очевидно, множество $\boldsymbol{D}(\mu)$, то есть множество трасс, получаемых из μ удалением стационарностей, конечно и его легко алгоритмически перечислить. Для каждой перечисляемой трассы $\mu`$ нужно проверить её принадлежность модели $\boldsymbol{\Sigma}$, что легко сделать с помощью последовательного применения оракула $\mathbf{Extend2}_{\Sigma}(\mu`[1..i],\mu`(i+1))$. Таким образом мы можем перечислять трассы $\mu`\in\boldsymbol{D}(\mu)\cap\boldsymbol{\Sigma}$.

Пусть $z=?x$ стимул. Тогда, по правилу вывода (1), переход $\mu\text{—}?x\rightarrow\mu\cdot\langle ?x\rangle$ существует тогда и только тогда, когда найдётся хотя бы одна трасса $\mu`$, продолжаемая стимулом $?x$. Это проверяется последовательным применением оракула $\mathbf{Extend2}_{\Sigma}(\mu`,?x)$ к перечисляемым трассам $\mu`$. Итератор перечисляет единственное постсостояние $\mu\cdot\langle ?x\rangle$ или сообщает, что перечисляемое множество пусто.

Пусть $z=!y$ реакция. Тогда, по правилу вывода (2), переход $\mu\text{—}!y\rightarrow\mu\cdot\langle !y\rangle$ существует тогда и только тогда, когда все трассы $\mu`$ продолжаются реакцией $!y$. Это проверяется последовательным

применением оракула $\mathbf{Extend2}_{\Sigma}(\mu`,!y)$ к перечисляемым трассам $\mu`$. Итератор перечисляет единственное постсостояние $\mu\cdot\langle!y\rangle$ или сообщает, что перечисляемое множество пусто.

Пусть $z=!\boldsymbol{error}$ добавленная ошибочная реакция. Тогда, по правилу вывода (4), если нет реакции или стационарности, которыми продолжалась бы каждая трасса $\mu`$, то переход $\mu\text{—}!\boldsymbol{error}\rightarrow\mu\cdot\langle!\boldsymbol{error}\rangle$ существует. Это проверяется последовательным применением оракула $\mathbf{Extend2}_{\Sigma}(\mu`,z)$ к перечисляемым трассам $\mu`$ для z, пробегающего конечное множество реакций и стационарность. Итератор перечисляет единственное постсостояние $\mu\cdot\langle!\boldsymbol{error}\rangle$ или сообщает, что перечисляемое множество пусто.

Построим итератор $\mathbf{Tau2}_{\boldsymbol{AComp}_0(\Sigma)}(s)$.

Параметром итератора является состояние s LTS $\boldsymbol{AComp}_0(\Sigma)$. Итератор перечисляет постсостояния τ-переходов из состояния s. Состояние s – это некоторая δ-трасса $s=\mu$. Переход $\mu\text{—}\tau\rightarrow\mu\cdot\langle\delta\rangle$ определяется правилом вывода (3), и, если существует, то единственный. Согласно этому правилу нужно проверить, заканчивается или нет трасса μ стационарностью. Если трасса не заканчивается стационарностью, то нужно перебрать все δ-трассы $\mu`\in\boldsymbol{D}(\mu)\cap\boldsymbol{\Sigma}$, что можно сделать так, как описано выше, и проверить продолжение каждой трассы $\mu`$ стационарностью, что можно сделать с помощью оракула $\mathbf{Extend2}_{\Sigma}(\mu`,\delta)$. Итератор перечисляет единственное постсостояние $\mu\cdot\langle\delta\rangle$ или сообщает, что перечисляемое множество пусто.

52. Доказательство Утверждение 52:

Нам нужно доказать следующие утверждения:

1) Инвариантность преобразования: $\forall\mathbf{S}\in\mathfrak{C}\ \mathcal{T}(\mathbf{S})\ \sim_{rel}\ \mathbf{S}$.
2) Композиция преобразованных спецификаций является корректной спецификацией: композиция реализаций $\mathbf{I_1}$ и $\mathbf{I_2}$, конформных спецификациям $\mathbf{S_1}$ и $\mathbf{S_2}$, соответственно, конформна композиции преобразованных спецификаций $\mathcal{T}(\mathbf{S_1})$ и $\mathcal{T}(\mathbf{S_2})$:
   $\forall\mathbf{S_1},\mathbf{S_2}\in\mathfrak{C}\ \forall\mathbf{I_1}\in\mathfrak{I}(\mathbf{S_1})\ \forall\mathbf{I_2}\in\mathfrak{I}(\mathbf{S_2})\ \mathbf{I_1}\!\upharpoonright\!\downarrow\mathbf{I_2}\ \boldsymbol{rel}\ \mathcal{T}(\mathbf{S_1})\!\upharpoonright\!\downarrow\mathcal{T}(\mathbf{S_2})$.
3) Композиция преобразованных спецификаций является левокорректной спецификацией: композиция реализации $\mathbf{I_1}$, конформной спецификации

$\mathbf{S_1}$, с реализацией $\mathbf{I_2}$ конформна композиции преобразованной спецификации $\mathcal{T}(\mathbf{S_1})$ с той же реализацией $\mathbf{I_2}$:

$\forall \mathbf{S_1}, \mathbf{I_2} \in \mathfrak{C}\ \ \forall \mathbf{I_1} \in \mathfrak{I}(\mathbf{S_1})\ \ \mathbf{I_1} \| \mathbf{I_2}\ \ \textit{rel}\ \ \mathcal{T}(\mathbf{S_1}) \| \mathbf{I_2}$.

4) Композиция преобразованных спецификаций $\mathcal{T}(\mathbf{S_1})$ и $\mathcal{T}(\mathbf{S_2})$ является самой сильной корректной спецификацией:

$\forall \mathbf{S_1}, \mathbf{S_2} \in \mathfrak{C}\ \ \forall \mathbf{S} \in \mathfrak{M}$

$(\forall \mathbf{I_1} \in \mathfrak{I}(\mathbf{S_1})\ \ \forall \mathbf{I_2} \in \mathfrak{I}(\mathbf{S_2})\ \ \mathbf{I_1} \| \mathbf{I_2}\ \ \textit{rel}\ \ \mathbf{S}) \Rightarrow \mathcal{T}(\mathbf{S_1}) \| \mathcal{T}(\mathbf{S_2})\ \ \textit{rel}\ \ \mathbf{S}$.

5) Композиция преобразованной спецификации $\mathcal{T}(\mathbf{S_1})$ с реализацией $\mathbf{I_2}$ является самой сильной левокорректной спецификацией:

$\forall \mathbf{S_1}, \mathbf{I_2} \in \mathfrak{C}\ \ \forall \mathbf{S} \in \mathfrak{M}$

$(\forall \mathbf{I_1} \in \mathfrak{I}(\mathbf{S_1})\ \ \mathbf{I_1} \| \mathbf{I_2}\ \ \textit{rel}\ \ \mathbf{S}) \Rightarrow \mathcal{T}(\mathbf{S_1}) \| \mathbf{I_2}\ \ \textit{rel}\ \ \mathbf{S}$.

Доказательство утверждения 1).

По конформности преобразования $\mathcal{T}$ (Монотонность 6:),

$\mathcal{T}(\mathbf{S})\ \ \textit{rel}\ \ \mathbf{S}$.

Докажем обратное: $\mathbf{S}\ \ \textit{rel}\ \ \mathcal{T}(\mathbf{S})$.

По рефлексивности предпорядка,

$\mathbf{S}\ \ \textit{rel}\ \ \mathbf{S}$.

Тогда, по мажорантности преобразования $\mathcal{T}$ (Монотонность 7:),

$f(\mathbf{S}) \preccurlyeq f \circ \mathcal{T}(\mathbf{S})$.

Тогда, по генеративности мажорирования ϕ-трасс (Монотонность 4:),

$\cup \circ \xi \circ f(\mathbf{S}) \preccurlyeq \cup \circ \xi \circ f \circ \mathcal{T}(\mathbf{S})$.

Тогда, по генеративности ϕ-трасс (Монотонность 2:),

$g(\mathbf{S}) \preccurlyeq g \circ \mathcal{T}(\mathbf{S})$.

Тогда, по эквивалентности соответствия мажорированию βγδ-трасс (Монотонность 1:),

$\mathbf{S}\ \ \textit{rel}\ \ \mathcal{T}(\mathbf{S})$.

Утверждение 1) доказано.

Доказательство утверждения 2).

По мажорантности преобразования $\mathcal{T}$ (Монотонность 7:),

$f(\mathbf{I_1}) \preccurlyeq f \circ \mathcal{T}(\mathbf{S_1})$ и $f(\mathbf{I_2}) \preccurlyeq f \circ \mathcal{T}(\mathbf{S_2})$.

Тогда, по композиционности мажорирования ϕ-трасс (Монотонность 5:),

$\cup(f(\mathbf{I_1}) \| f(\mathbf{I_2})) \preccurlyeq \cup(f \circ \mathcal{T}(\mathbf{S_1}) \| f \circ \mathcal{T}(\mathbf{S_2}))$.

Тогда, по аддитивности ϕ-трасс (Монотонность 3:),

$f(\mathbf{I}_1 \wr\!\downarrow \mathbf{I}_2) \preccurlyeq f(\mathcal{T}(\mathbf{S}_1) \wr\!\downarrow \mathcal{T}(\mathbf{S}_2))$.

Тогда, по генеративности мажорирования ϕ-трасс (Монотонность 4:),

$\cup_\circ \xi_\circ f(\mathbf{I}_1 \wr\!\downarrow \mathbf{I}_2) \preccurlyeq \cup_\circ \xi_\circ f(\mathcal{T}(\mathbf{S}_1) \wr\!\downarrow \mathcal{T}(\mathbf{S}_2))$.

Тогда, по генеративности ϕ-трасс (Монотонность 2:),

$g(\mathbf{I}_1 \wr\!\downarrow \mathbf{I}_2) \preccurlyeq g(\mathcal{T}(\mathbf{S}_1) \wr\!\downarrow \mathcal{T}(\mathbf{S}_2))$.

Тогда, по эквивалентности соответствия мажорированию βγδ-трасс (Монотонность 1:),

$\mathbf{I}_1 \wr\!\downarrow \mathbf{I}_2$ ***rel*** $\mathcal{T}(\mathbf{S}_1) \wr\!\downarrow \mathcal{T}(\mathbf{S}_2)$.

Утверждение 2) доказано.

Доказательство утверждения 3).

По мажорантности преобразования $\mathcal{T}$ (Монотонность 7:),

$f(\mathbf{I}_1) \preccurlyeq f_\circ \mathcal{T}(\mathbf{S}_1)$.

По рефлексивности мажорирования ϕ-трасс (Монотонность 8:),

$f(\mathbf{I}_2) \preccurlyeq f(\mathbf{I}_2)$.

Тогда, по композиционности мажорирования ϕ-трасс (Монотонность 5:),

$\cup(f(\mathbf{I}_1) \wr\!\downarrow f(\mathbf{I}_2)) \preccurlyeq \cup(f_\circ \mathcal{T}(\mathbf{S}_1) \wr\!\downarrow f(\mathbf{I}_2))$.

Тогда, по аддитивности ϕ-трасс (Монотонность 3:),

$f(\mathbf{I}_1 \wr\!\downarrow \mathbf{I}_2) \preccurlyeq f(\mathcal{T}(\mathbf{S}_1) \wr\!\downarrow \mathbf{I}_2)$.

Тогда, по генеративности мажорирования ϕ-трасс (Монотонность 4:),

$\cup_\circ \xi_\circ f(\mathbf{I}_1 \wr\!\downarrow \mathbf{I}_2) \preccurlyeq \cup_\circ \xi_\circ f(\mathcal{T}(\mathbf{S}_1) \wr\!\downarrow \mathbf{I}_2)$.

Тогда, по генеративности ϕ-трасс (Монотонность 2:),

$g(\mathbf{I}_1 \wr\!\downarrow \mathbf{I}_2) \preccurlyeq g(\mathcal{T}(\mathbf{S}_1) \wr\!\downarrow \mathbf{I}_2)$.

Тогда, по эквивалентности соответствия мажорированию βγδ-трасс (Монотонность 1:),

$\mathbf{I}_1 \wr\!\downarrow \mathbf{I}_2$ ***rel*** $\mathcal{T}(\mathbf{S}_1) \wr\!\downarrow \mathbf{I}_2$.

Утверждение 3) доказано.

Доказательство утверждения 4).

По конформности преобразования $\mathcal{T}$ (Монотонность 6:),

$\mathcal{T}(\mathbf{S}_1)$ ***rel*** $\mathbf{S}_1$ и $\mathcal{T}(\mathbf{S}_2)$ ***rel*** $\mathbf{S}_2$.

Поэтому для корректной спецификации $\mathbf{S}$ имеем

$\mathcal{T}(\mathbf{S}_1) \wr\!\downarrow \mathcal{T}(\mathbf{S}_2)$ ***rel*** $\mathbf{S}$.

Утверждение 4) доказано.

Доказательство утверждения 5).

По конформности преобразования $\mathcal{T}$ (Монотонность 6:),
$\mathcal{T}(\mathbf{S_1})$ ***rel*** $\mathbf{S_1}$.
Поэтому для левокорректной спецификации $\mathbf{S}$ имеем
$\mathcal{T}(\mathbf{S_1})\restriction\mathbf{I_2}$ ***rel*** $\mathbf{S}$.
Утверждение 5) доказано.

Основное утверждение доказано.

53. **Доказательство Утверждение 53:**

1. Докажем, что безопасная βγδ-трасса μ мажорируется только сама собой.
Допустим, утверждение неверно. Тогда $\mu\preccurlyeq^{\gamma}\sigma$, что означает:

$$\sigma=\langle\gamma\rangle$$
$$\vee\ \exists\pi\ \exists a\in C\ \sigma=\pi\cdot\langle a,\gamma\rangle\ \&\ (\pi\cdot\langle\beta\delta(a)\rangle\le\mu\ \vee\ \exists b\in\beta\delta(a)\ \pi\cdot\langle b\rangle\le\mu).$$

Но это противоречит безопасности βγδ-трассы μ.

2. Докажем, что опасная βγδ-трасса μ имеет γ-мажоранту.
Опасность означает:

$$\langle\gamma\rangle\in\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$$
$$\vee\ \exists a\in C\ \exists\pi\cdot\langle a,\gamma\rangle\in\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})\ \pi\cdot\langle\beta\delta(a)\rangle\le\mu\ \vee\ \exists b\in\beta\delta(a)\ \pi\cdot\langle b\rangle\le\mu.$$

А тогда $\mu\preccurlyeq^{\gamma}\pi\cdot\langle a,\gamma\rangle$.

54. **Доказательство Утверждение 54:**

Обозначим: $\mathbf{I}=\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{I})$, $\mathbf{\Sigma}=\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$.

1. Сначала докажем прямую импликацию

$$\mathbf{I}\ \boldsymbol{ioco}_{\beta\gamma\delta}\ \mathbf{\Sigma}\ \Rightarrow\ \mathbf{I}\ \preccurlyeq\ \mathbf{\Sigma}\ =\ \forall\mu\in\mathbf{I}\ \exists\sigma\in\mathbf{\Sigma}\ \mu\preccurlyeq\sigma.$$

Допустим, утверждение не верно: существует βγδ-трасса $\mu\in\mathbf{I}$, которая не мажорируется ни одной βγδ-трассой из $\mathbf{\Sigma}$. Выберем такую βγδ-трассу μ минимальной длины. Возможны два варианта:

1.1. Пустая трасса $\mu=\epsilon$.

По определению мажорируемости (случай А), $\epsilon \preccurlyeq \epsilon$.

Также $\epsilon \in \mathbf{\Sigma}$ (ϵ принадлежит любой βγδ-модели).

Таким образом, $\mu = \epsilon \preccurlyeq \epsilon \in \mathbf{\Sigma}$, что противоречит допущению.

1.2. βγδ-трасса μ не пуста.

Тогда, в силу минимальности μ, она имеет вид $\mu = \kappa \cdot \langle u \rangle$, где $\kappa \preccurlyeq \sigma$ для некоторой βγδ-трассы $\sigma \in \mathbf{\Sigma}$. Далее возможны два варианта:

1.2.1. $\sigma \neq \kappa$.

По определению мажорируемости (случай В), возможны два случая:

1.2.1.1. Случай В1: $\sigma = \langle \gamma \rangle$. Тогда $\mu \preccurlyeq \sigma \in \mathbf{\Sigma}$, что противоречит допущению.

1.2.1.2. Случай В2: βγδ-трасса представима в виде $\sigma = \pi \cdot \langle a, \gamma \rangle$, где $a \in Z$ стимул или реакция.

Возможны два подслучая:

Подслучай В21: $\pi \cdot \langle \beta\delta(a) \rangle \leq \kappa < \mu$.

Подслучай В22: $\exists b \in \beta\delta(a)\ \ \pi \cdot \langle b \rangle \leq \kappa < \mu$.

В каждом из этих двух подслучаев оказывается, по определению мажорируемости, $\mu = \kappa \cdot \langle u \rangle \preccurlyeq \pi \cdot \langle a, \gamma \rangle \in \mathbf{\Sigma}$, что противоречит допущению.

1.2.2. $\sigma = \kappa$.

Возможны два варианта:

1.2.2.1. $\sigma \notin \mathit{safe}(\mathbf{\Sigma})$.

Тогда, по определению безопасной βγδ-трассы, возможны два случая:

1.2.2.1.1. Случай В1: $\sigma = \epsilon$.

Тогда $\langle \gamma \rangle \in \mathbf{\Sigma}$, и $\mu \preccurlyeq \langle \gamma \rangle \in \mathbf{\Sigma}$, что противоречит допущению.

1.2.2.1.2. Случай В2: у βγδ-трассы есть префикс $\pi \cdot \langle v \rangle \leq \sigma$ такой, что имеет место один из двух подслучаев:

Подслучай B21: $\exists a\in Z\ \ v=\beta\delta(a)\ \ \&\ \ \pi\cdot\langle a,\gamma\rangle\in\mathbf{\Sigma}$.

Подслучай B22: $\exists a\in Z\ \ v\in\beta\delta(a)\ \ \&\ \ \pi\cdot\langle a,\gamma\rangle\in\mathbf{\Sigma}$.

В каждом из этих двух подслучаев оказывается, по определению мажорируемости, $\mu=\kappa\cdot\langle u\rangle\preccurlyeq\pi\cdot\langle a,\gamma\rangle\in\mathbf{\Sigma}$, что противоречит допущению.

1.2.2.2. $\sigma\in\mathit{\mathbf{safe}}(\mathbf{\Sigma})$.
Возможны два случая:

1.2.2.2.1. Случай A: символ u безопасен в $\mathbf{\Sigma}$ после σ.
Тогда, по условию конформности, должно быть $\mu=\kappa\cdot\langle u\rangle=\sigma\cdot\langle u\rangle\in\mathbf{\Sigma}$.

Но, по определению мажорируемости, $\mu\preccurlyeq\mu$, что противоречит допущению.

1.2.2.2.2. Случай B2: символ u опасен в $\mathbf{\Sigma}$ после σ.
По определению безопасного символа, возможны два подслучая:

Подслучай B21: $\exists a\in Z\ \ u=\beta\delta(a)\ \ \&\ \ \pi\cdot\langle a,\gamma\rangle\in\mathbf{\Sigma}$.

Подслучай B22: $\exists a\in Z\ \ u\in\beta\delta(a)\ \ \&\ \ \pi\cdot\langle a,\gamma\rangle\in\mathbf{\Sigma}$.

В каждом из этих двух подслучаев оказывается, по определению мажорируемости, $\mu=\kappa\cdot\langle u\rangle\preccurlyeq\pi\cdot\langle a,\gamma\rangle\in\mathbf{\Sigma}$, что противоречит допущению.

Тем самым прямая импликация доказана.

2. Теперь докажем обратную импликацию

$\mathbf{I}\ \mathit{\mathbf{ioco}}_{\beta\gamma\delta}\ \mathbf{\Sigma}\ \Leftarrow\ \forall\mu\in\mathbf{I}\ \exists\sigma\in\mathbf{\Sigma}\ \ \mu\preccurlyeq\sigma\ =\ \mathbf{I}\ \preccurlyeq\ \mathbf{\Sigma}$.

Допустим, утверждение не верно: существует $\beta\gamma\delta$-трасса $\mu\in\mathit{\mathbf{tt}}(\mathbf{\Sigma})\cap\mathbf{I}$ такая, что $\mu\notin\mathbf{\Sigma}\cap\mathit{\mathbf{tt}}(\mathbf{I})$. Выберем такую $\beta\gamma\delta$-трассу μ минимальной длины. Далее возможны два варианта:

2.1. Пустая трасса $\mu=\epsilon$.
Возможны два варианта:

2.1.1. $\epsilon \notin \boldsymbol{tt}(\mathbf{I})$.

Тогда, по определению тестовых βγδ-трасс, $\langle\gamma\rangle \in \mathbf{I}$.

Поскольку, по определению мажорируемости, трасса $\langle\gamma\rangle$ мажорируется только сама собой, то $\langle\gamma\rangle \in \mathbf{\Sigma}$, что противоречит $\epsilon=\mu \in \boldsymbol{tt}(\mathbf{\Sigma})$.

2.1.2. $\epsilon \in \boldsymbol{tt}(\mathbf{I})$.

Поскольку пустая трасса есть в любой βγδ-модели, то $\epsilon \in \mathbf{\Sigma}$.

Таким образом, $\epsilon \in \mathbf{\Sigma} \cap \boldsymbol{tt}(\mathbf{I})$, что противоречит допущению.

2.2. βγδ-трасса μ не пуста.

Тогда, в силу минимальности μ, она имеет вид $\mu=\kappa\cdot\langle u\rangle$, где $\kappa \in \mathbf{\Sigma} \cap \boldsymbol{tt}(\mathbf{I})$.

Поскольку $\mu \in \boldsymbol{tt}(\mathbf{\Sigma})$, то её префикс $\kappa \in \boldsymbol{tt}(\mathbf{\Sigma})$.

Таким образом, $\kappa \in \boldsymbol{tt}(\mathbf{\Sigma}) \cap \mathbf{\Sigma} = \boldsymbol{safe}(\mathbf{\Sigma})$.

Далее возможны два варианта:

2.2.1. $\mu \notin \mathbf{\Sigma}$.

Тогда мажорирующая её βγδ-трасса $\sigma \in \mathbf{\Sigma}$ не совпадает с ней $\sigma \neq \mu$.

Следовательно, по определению мажорируемости (случай B), возможны два случая:

2.2.1.1. Случай B1: $\sigma=\langle\gamma\rangle$.

Но тогда $\boldsymbol{safe}(\mathbf{\Sigma})=\varnothing$, что противоречит $\kappa \in \boldsymbol{safe}(\mathbf{\Sigma})$.

2.2.1.2. Случай B2: $\sigma=\pi\cdot\langle a,\gamma\rangle$ и $\pi\cdot\langle v\rangle \le \mu=\kappa\cdot\langle u\rangle$, причём имеет место один из двух подслучаев:

Подслучай B21: $v=\beta\delta(a)$.
Подслучай B22: $v \in \beta\delta(a)$.

В каждом из этих двух подслучаев, по определению безопасной βγδ-трассы, оказывается, что символ v опасен после префикса π, что противоречит $\kappa \in \boldsymbol{safe}(\mathbf{\Sigma})$.

2.2.2. $\mu \in \mathbf{\Sigma}$.

Поскольку также $\mu\in \boldsymbol{tt}(\boldsymbol{\Sigma})$, то $\mu\in \boldsymbol{tt}(\boldsymbol{\Sigma})\cap\boldsymbol{\Sigma}=\boldsymbol{safe}(\boldsymbol{\Sigma})$.
Если допущение верно, то $\mu\in\boldsymbol{\Sigma}$ влечёт $\mu\notin \boldsymbol{tt}(\mathbf{I})$.

Поскольку $\mu=\kappa\cdot\langle u\rangle\in\mathbf{I}$, то её префикс $\kappa\in\mathbf{I}$.

Таким образом, $\kappa\in\mathbf{I}\cap\boldsymbol{tt}(\mathbf{I})=\boldsymbol{safe}(\mathbf{I})$.
Следовательно, по определению тестовых βγδ-трасс, символ u опасен в $\mathbf{I}$ после κ.
Но тогда, по определению безопасного символа, возможны два варианта:

1. $\exists z\in Z\ u=\beta\delta(z)\ \&\ \kappa\cdot\langle z,\gamma\rangle\in\mathbf{I}$.
2. $\exists z\in Z\ u\in\beta\delta(z)\ \&\ \kappa\cdot\langle z,\gamma\rangle\in\mathbf{I}$.

В обоих вариантах для βγδ-трассы $\kappa\cdot\langle z,\gamma\rangle$ должна найтись мажорирующая её βγδ-трасса $\pi\in\boldsymbol{\Sigma}$. Далее возможны два случая:

2.2.2.1. Случай A: $\pi=\kappa\cdot\langle z,\gamma\rangle$.

Но это противоречит $\mu=\kappa\cdot\langle u\rangle\in\boldsymbol{safe}(\boldsymbol{\Sigma})$.

2.2.2.2. Случай B1: $\pi=\langle\gamma\rangle$.

Но тогда $\boldsymbol{safe}(\boldsymbol{\Sigma})=\varnothing$, что противоречит $\mu\in\boldsymbol{safe}(\boldsymbol{\Sigma})$.

2.2.2.3. Случай B2: $\pi=\rho\cdot\langle a,\gamma\rangle$ и $\rho\cdot\langle v\rangle\le\kappa\cdot\langle z\rangle$, причём имеет место один из двух подслучаев:

Подслучай B21: $v=\beta\delta(a)$.
Подслучай B22: $v\in\beta\delta(a)$.

В каждом из этих двух подслучаев символ v опасен в $\boldsymbol{\Sigma}$ после βγδ-трассы ρ. Далее возможны два варианта:

2.2.2.3.1. $\rho\cdot\langle v\rangle=\kappa\cdot\langle z\rangle$.

В этом варианте, очевидно, $v=z$. Поэтому v стимул или реакция, и, следовательно, $\beta\delta(z)=\beta\delta(v)=\beta\delta(a)$.

Далее возможны два варианта:
a. $u=\beta\delta(z)=\beta\delta(a)$.
b. $u\in\beta\delta(z)=\beta\delta(a)$.

В каждом из этих двух вариантов u опасен в $\boldsymbol{\Sigma}$ после κ, что противоречит $\mu=\kappa\cdot\langle u\rangle\in\boldsymbol{safe}(\boldsymbol{\Sigma})$.

2.2.2.3.2. $\rho\cdot\langle v\rangle<\kappa\cdot\langle z\rangle$.

В этом варианте v опасен в $\boldsymbol{\Sigma}$ после ρ, что влечёт $\kappa\notin \boldsymbol{safe}(\boldsymbol{\Sigma})$. А это противоречит $\mu=\kappa\cdot\langle u\rangle\in \boldsymbol{safe}(\boldsymbol{\Sigma})$.

Тем самым обратная импликация тоже доказана.

55. Доказательство Утверждение 55:

1) Замкнутость ϕ-трасс LTS по взятию префикса следует из того, что ϕ-трассы LTS $\mathbf{S}$ – это трассы LTS $\boldsymbol{L_\gamma 2 L_{\phi\gamma}}(\mathbf{S})$, а трассы любой LTS обладают этим свойством.
2) Возможность удаления ϕ-символа следует из того, что в LTS $\boldsymbol{L_\gamma 2 L_{\phi\gamma}}(\mathbf{S})$ переход по ϕ-символу – это петля.
3) Если ϕ-трасса $\boldsymbol{\mu}$ завершается ϕ-символом o некоторого стабильного состояния s, а символ $z\notin o_r$, то в состоянии s есть трасса $\langle z\rangle$, следовательно, в LTS есть ϕ-трасса $\boldsymbol{\mu}\cdot\langle z\rangle$.
4) Если ϕ-трасса $\boldsymbol{\mu}$ завершается ϕ-символом o некоторого стабильного состояния s, а символ $z\in o_g$, то в состоянии s есть трасса $\langle z,\gamma\rangle$, следовательно, в LTS есть ϕ-трасса $\boldsymbol{\mu}\cdot\langle z,\gamma\rangle$.

56. Доказательство Утверждение 56:

Поскольку для каждого стабильного состояния s его ***ref***-множество $\phi(s)_r = C\backslash \boldsymbol{init}(s)$, βδ-отказы, порождаемые этим состоянием, и βδ-отказы, вложенные в ***ref***-множество этого ϕ-символа, совпадают:
$\beta\delta(\phi(s)_r) = \beta\delta(C\backslash \boldsymbol{init}(s)) = \beta\delta(s)$.

Поскольку βδ-отказы и ϕ-символы состояния моделируются петлями в этом состоянии и βγδ-трасса LTS является βγδ-трассой некоторого маршрута этой LTS, то всегда найдётся такая ϕ-трасса этого маршрута, которая порождает эту βγδ-трассу. Наоборот, βγδ-трасса, порождаемая ϕ-трассой любого маршрута LTS, является одной из βγδ-трасс этого маршрута.

57. Доказательство Утверждение 57:

Обозначим: $a=\phi(k)$, $b=\phi(l)$.
Заметим, что $\boldsymbol{init}(k)=A\backslash a_r$ и $\boldsymbol{init}(l)=B\backslash b_r$.

a) Композиция kl стабильных состояний k и l стабильна тогда и только тогда, когда невозможен синхронный переход, то есть в этих состояниях нет переходов по противоположным символам: $\boldsymbol{init}(k)\cap\underline{\boldsymbol{init}(l)}=\varnothing$, что эквивалентно $(A\backslash a_r)\cap(\underline{B\backslash b_r})=\varnothing$.

b) Пусть состояние kl стабильно. Обозначим: $c=\phi(kl)$. Нам нужно доказать, что $c=a\|b$.

В состоянии kl не определён переход по символу $z\in A\backslash\underline{B}$ тогда и только тогда, когда по нему нет перехода в состоянии k, то есть $z\in a_r$. Соответственно, в состоянии kl не определён переход по символу $z\in B\backslash\underline{A}$ тогда и только тогда, когда по нему нет перехода в состоянии l, то есть $z\in b_r$. Тем самым, множество символов, по которым в состоянии kl не определены переходы, равно $c_r=(A\backslash\underline{B})\cap a_r\cup(B\backslash\underline{A})\cap b_r$.

В состоянии kl определён переход по непосредственно разрушающему символу $z\in A\backslash\underline{B}$ тогда и только тогда, когда этот символ непосредственно разрушающий в состоянии k, то есть $z\in a_g$. Соответственно, в состоянии kl определён переход по непосредственно разрушающему символу $z\in B\backslash\underline{A}$ тогда и только тогда, когда этот символ непосредственно разрушающий в состоянии l, то есть $z\in b_g$. Тем самым, множество символов, непосредственно разрушающих в состоянии kl, равно $c_g=(A\backslash\underline{B})\cap a_g\cup(B\backslash\underline{A})\cap b_g$.

Итак, мы имеем:
$c=(c_r,c_g)=((A\backslash\underline{B})\cap a_r\cup(B\backslash\underline{A})\cap b_r,\ (A\backslash\underline{B})\cap a_g\cup(B\backslash\underline{A})\cap b_g)=a\|b$.

58. **Доказательство Утверждение 58:**

Непосредственно следует из очевидной коммутативности композиции ϕ-символов и определения композиции ϕ-трасс: симметричность операндов в правилах $(\phi 0,3\div 6)$ и взаимная симметричность правил $(\phi 1)$ и $(\phi 2)$.

59. **Доказательство Утверждение 59:**

Пусть $A,B\subseteq Z$, $\mathbf{K}\in\boldsymbol{LTS}_{\beta\gamma\delta}(A)$, $\mathbf{L}\in\boldsymbol{LTS}_{\beta\gamma\delta}(B)$.

Поскольку ϕ-трасса LTS не продолжается после разрушения, нам достаточно рассматривать только такие маршруты, которые не продолжаются после перехода по разрушению.

1. $\forall K \in \boldsymbol{runs}^{\omega}(\mathbf{K})\ \ \forall L \in \boldsymbol{runs}^{\omega}(\mathbf{L})\ \ \forall M \in K \between L\ \ \forall \boldsymbol{\mu} \in \boldsymbol{traces}_{\phi}^{\omega}(M)$
$\exists \boldsymbol{\kappa} \in \boldsymbol{traces}_{\phi}^{\omega}(K)\ \ \exists \boldsymbol{\lambda} \in \boldsymbol{traces}_{\phi}^{\omega}(L)\ \ \boldsymbol{\mu} \in \boldsymbol{\kappa} \between \boldsymbol{\lambda}$.

Очевидно, нам достаточно рассмотреть три случая в зависимости от того, конечны или бесконечны ϕ-трасса $\boldsymbol{\mu}$ и маршрут M.

1.1. Пусть ϕ-трасса $\boldsymbol{\mu}$ и маршрут M конечны.

Каждый базовый символ ϕ-трассы $\boldsymbol{\mu}$ является символом некоторого перехода маршрута M, а ϕ-символ – ϕ-символом стабильного состояния маршрута. Рассмотрим последовательность $k_0 l_0, k_1 l_1, \ldots$ тех стабильных композиционных состояний маршрута M, ϕ-символы которых попали в $\boldsymbol{\mu}$. Этой последовательности соответствуют две последовательности, очевидно, также стабильных состояний маршрутов K и L: $k_0, k_1, \ldots$ и $l_0, l_1, \ldots$. Заметим, что любые ϕ-трассы $\boldsymbol{\kappa}$ $(\boldsymbol{\lambda})$ маршрута K (L) имеют одну и ту же подпоследовательность базовых символов $\boldsymbol{\kappa}{\downarrow}A_{\gamma}$ $(\boldsymbol{\lambda}{\downarrow}B_{\gamma})$, поэтому они различаются только ϕ-символами. Выберем ϕ-трассы $\boldsymbol{\kappa}$ и $\boldsymbol{\lambda}$ маршрутов K и L так, чтобы для каждого i в ϕ-трассы $\boldsymbol{\kappa}$ и $\boldsymbol{\lambda}$ помещалось столько же ϕ-символов состояний k_i и l_i, соответственно, сколько ϕ-символов состояния $k_i l_i$ помещается в ϕ-трассу $\boldsymbol{\mu}$.

Выбор начального состояния $k_0 l_0$ соответствует композиции пустых ϕ-трасс по правилу вывода $(\phi 0)$. Применение каждого из правил вывода асинхронного перехода $(\between 1 \div 2)$ к маршрутам-операндам для символа $z \neq \tau$ добавляет в ϕ-трассу $\boldsymbol{\mu}$ символ z, что соответствует применению соответствующего правила вывода $(\phi 1 \div 2)$ к ϕ-трассам-операндам. Для символа $z=\tau$ применение правил вывода $(\between 1 \div 2)$ не меняет ϕ-трассу $\boldsymbol{\mu}$ и ϕ-трассы-операнды. Применение правила вывода синхронного перехода $(\between 3)$ к маршрутам также не меняет ϕ-трассу $\boldsymbol{\mu}$, что соответствует применению соответствующего правила вывода $(\phi 3)$ к ϕ-трассам-операндам. Учитывая описанные выше правила выбора ϕ-символов в ϕ-трассах $\boldsymbol{\kappa}$ и $\boldsymbol{\lambda}$, размещение в ϕ-трассе $\boldsymbol{\mu}$ ϕ-символа состояния $k_i l_i$ соответствует применению правила вывода $(\phi 4)$ к ϕ-трассам-операндам, при условии, что $\phi(k_i l_i) = \phi(k_i) \between \phi(l_i)$. Выполнение последнего условия доказано в Утверждение 57:.

Утверждение 1.1 доказано.

1.2. Теперь рассмотрим случай, когда ϕ-трасса $\boldsymbol{\mu}$ конечна, а маршрут M бесконечен.

Это может быть только в том случае, когда маршрут M может быть представлен в виде $M=M_1 \cdot M_2$, где M_2 – бесконечный τ-постфикс. Очевидно, что маршрут M_1 является результатом композиции $M_1 \in K_1 \wr\!\downarrow L_1$ конечных префиксов маршрутов-операндов $K=K_1 \cdot K_2$ и $L=L_1 \cdot L_2$. А тогда $M_2 \in K_2 \wr\!\downarrow L_2$.

Далее возможны два случая.

1.2.1. ϕ-трасса $\boldsymbol{\mu}$ не завершается разрушением.

Тогда $\boldsymbol{\mu}$ является также одной из ϕ-трасс конечного префикса M_1. Поэтому, по 1.1, $\exists \boldsymbol{\kappa}_1 \in \boldsymbol{traces}_\phi(K_1)\ \exists \boldsymbol{\lambda}_1 \in \boldsymbol{traces}_\phi(L_1)\ \boldsymbol{\mu} \in \boldsymbol{\kappa}_1 \wr\!\downarrow \boldsymbol{\lambda}_1$.

Далее возможны два случая.

1.2.1.1. Один из постфиксов маршрутов-операндов K_2 или L_2 является бесконечным τ-маршрутом.

Пусть это будет постфикс K_2. Поскольку K_2 и L_2 компонуются, L_2 также является τ-маршрутом (конечным или бесконечным). Тогда $\boldsymbol{\kappa}_1 \in \boldsymbol{traces}_\phi(K)$ и $\boldsymbol{\lambda}_1 \in \boldsymbol{traces}_\phi(L)$. Аналогично доказывается случай бесконечного постфикса L_2.

1.2.1.2. Ни один из постфиксов маршрутов-операндов K_2 или L_2 не является бесконечным τ-маршрутом.

Тогда трассы этих постфиксов противоположны друг другу $\boldsymbol{trace}(K_2)=\underline{\boldsymbol{trace}(L_2)} \in (A \cap \underline{B})^\infty$, что даёт бесконечную цепочку синхронных переходов (быть может, перемежающуюся τ-переходами в маршрутах-операндах). Тогда $\boldsymbol{\kappa}=\boldsymbol{\kappa}_1 \cdot \boldsymbol{trace}(K_2) \in \boldsymbol{traces}_\phi(K)$ и $\boldsymbol{\lambda}=\boldsymbol{\lambda}_1 \cdot \boldsymbol{trace}(L_2) \in \boldsymbol{traces}_\phi(L)$.

По правилам вывода (ϕ3) и (ϕ6), $\boldsymbol{trace}(K_2) \wr\!\downarrow \boldsymbol{trace}(L_2)=\{\epsilon\}$.

А тогда, по правилу индуктивного предела (ϕ6), $\boldsymbol{\mu} \in \boldsymbol{\kappa} \wr\!\downarrow \boldsymbol{\lambda}$.

1.2.2. ϕ-трасса $\boldsymbol{\mu}$ завершается разрушением.

Тогда ϕ-трассу можно представить в виде $\boldsymbol{\mu}=\boldsymbol{\mu}_1\cdot\langle\gamma\rangle$, где $\boldsymbol{\mu}_1$ является одной из ϕ-трасс маршрута M_1. Поэтому, по 1.1, $\exists\boldsymbol{\kappa}_1\in\boldsymbol{traces}_\phi(K_1)$ $\exists\boldsymbol{\lambda}_1\in\boldsymbol{traces}_\phi(L_1)$ $\boldsymbol{\mu}_1\in\boldsymbol{\kappa}_1\rceil\lfloor\boldsymbol{\lambda}_1$.

Далее возможны два случая.

1.2.2.1. Один из постфиксов маршрутов-операндов K_2 или L_2 является бесконечным τ-маршрутом.

Пусть это будет постфикс K_2. Поскольку K_2 и L_2 компонуются, L_2 также является τ-маршрутом (конечным или бесконечным). Тогда $\boldsymbol{\kappa}_1\cdot\langle\gamma\rangle\in\boldsymbol{traces}_\phi(K)$ и $\boldsymbol{\lambda}_1\in\boldsymbol{traces}_\phi(L)$. По правилу вывода (ϕ1), $\boldsymbol{\mu}=\boldsymbol{\mu}_1\cdot\langle\gamma\rangle\in\boldsymbol{\kappa}_1\cdot\langle\gamma\rangle\rceil\lfloor\boldsymbol{\lambda}_1$. Аналогично доказывается случай бесконечного постфикса L_2: по правилу (ϕ2), $\boldsymbol{\mu}=\boldsymbol{\mu}_1\cdot\langle\gamma\rangle\in\boldsymbol{\kappa}_1\rceil\lfloor\boldsymbol{\lambda}_1\cdot\langle\gamma\rangle$.

1.2.2.2. Ни один из постфиксов маршрутов-операндов K_2 или L_2 не является бесконечным τ-маршрутом.

Тогда трассы этих постфиксов противоположны друг другу $\boldsymbol{trace}(K_2)=\underline{\boldsymbol{trace}(L_2)}\in(A\cap\underline{B})^\infty$, что даёт бесконечную цепочку синхронных переходов (быть может, перемежающуюся τ-переходами в маршрутах-операндах). Тогда $\boldsymbol{\kappa}=\boldsymbol{\kappa}_1\cdot\boldsymbol{trace}(K_2)\in\boldsymbol{traces}_\phi(K)$ и $\boldsymbol{\lambda}=\boldsymbol{\lambda}_1\cdot\boldsymbol{trace}(L_2)\in\boldsymbol{traces}_\phi(L)$. По правилу моделирования дивергенции разрушением (ϕ5), $\boldsymbol{\mu}=\boldsymbol{\mu}_1\cdot\langle\gamma\rangle\in\boldsymbol{\kappa}\rceil\lfloor\boldsymbol{\lambda}$.

Утверждение 1.2 доказано.

1.3. Пусть ϕ-трасса $\boldsymbol{\mu}$ бесконечна.

Этот случай аналогичен случаю 1.1, за исключением того, что правила вывода применяются бесконечное число раз и хотя бы одна из ϕ-трасс $\boldsymbol{\kappa}$ или $\boldsymbol{\lambda}$ получается бесконечной. По правилу индуктивного предела (ϕ6), имеем $\boldsymbol{\mu}\in\boldsymbol{\kappa}\rceil\lfloor\boldsymbol{\lambda}$.

Утверждение 1.3 доказано.

Утверждение 1 доказано.

2. $\forall K\in \boldsymbol{runs}^{\omega}(\mathbf{K})\ \forall L\in \boldsymbol{runs}^{\omega}(\mathbf{L})\ \forall \boldsymbol{\kappa}\in \boldsymbol{traces}_{\phi}^{\omega}(K)\ \forall \boldsymbol{\lambda}\in \boldsymbol{traces}_{\phi}^{\omega}(L)$
$\forall \boldsymbol{\mu}\in \boldsymbol{\kappa}\between\boldsymbol{\lambda}\ \exists M\in K\between L\ \boldsymbol{\mu}\in \boldsymbol{traces}_{\phi}^{\omega}(M)$.

Нам достаточно рассмотреть три случая в зависимости от того, применяются или нет для построения ϕ-трассы $\boldsymbol{\mu}$ правило моделирования дивергенции разрушением $(\phi 5)$ и правило индуктивного предела $(\phi 6)$. Заметим, что эти два правила не могут применяться вместе.

2.1. Пусть правила вывода $(\phi 5)$ и $(\phi 6)$ не применяются.

ϕ-трасса $\boldsymbol{\mu}$ однозначно определяется правилом вывода $(\phi 0)$ и конечной последовательностью применения правил вывода $(\phi 1\div 4)$ к заданным ϕ-трассам-операндам $\boldsymbol{\kappa}$ и $\boldsymbol{\lambda}$.

Каждому применению правила вывода $(\phi 1)$ для символа z к ϕ-трассам-операндам соответствует применение правила вывода $(\between 1)$ для z к маршрутам-операндам. Это может не получиться, если левый маршрут продолжается не переходом по символу z, а τ-переходом. Тогда мы сначала применяем правило вывода $(\between 1)$ для τ и левого маршрута. Аналогично для правил вывода $(\phi 2)$ и $(\between 2)$.

Каждому применению правила вывода $(\phi 3)$ к ϕ-трассам-операндам соответствует применение правила вывода $(\between 3)$ к маршрутам-операндам. Это может не получиться, если один из маршрутов-операндов продолжается τ-переходом. Тогда мы сначала применяем правила вывода $(\between 1\div 2)$ для τ и того маршрута, который продолжается τ-переходом. Для однозначности, если оба маршрута продолжаются τ-переходом, сначала выбираем τ-переход левого маршрута.

Таким образом мы однозначно получим маршрут $M\in K\between L\subseteq \boldsymbol{runs}(\mathbf{K}\between\mathbf{L})$. Далее рассмотрим применение правила вывода $(\phi 4)$. Оно компонует два ϕ-символа a и b в один ϕ-символ $a\between b$, если $a\between b\neq\tau$, или не меняет компонуемую ϕ-трассу, если $a\between b=\tau$. Этим ϕ-символам a и b соответствуют два состояния k и l LTS-операндов $a=\phi(k)$ и $b=\phi(l)$. Нам достаточно выполнения условия: $a\between b=\tau$ влечёт

нестабильность состояния kl, а, если $a\between b\neq\tau$, то состояние kl стабильно, и $\phi(kl)=a\between b$. Выполнение этого условия доказано в Утверждение 57:.

Утверждение 2.1 доказано.

2.2. Пусть применяется моделирование дивергенции разрушением $(\phi 5)$.

Это правило формирует конечную ϕ-трассу, заканчивающуюся разрушением. Для такой ϕ-трассы никакое правило вывода далее не может применяться.

Итак, ϕ-трассы можно представить в виде $\boldsymbol{\mu}=\boldsymbol{\mu}_1\cdot\langle\gamma\rangle$, $\boldsymbol{\kappa}=\boldsymbol{\kappa}_1\cdot\boldsymbol{\kappa}_2$, $\boldsymbol{\lambda}=\boldsymbol{\lambda}_1\cdot\boldsymbol{\lambda}_2$, где $\boldsymbol{\kappa}_2=\underline{\boldsymbol{\lambda}_2}\in(A\cap\underline{B})^{\infty}$ и $\boldsymbol{\mu}_1\in\boldsymbol{\kappa}_1\between\boldsymbol{\lambda}_1$. Очевидно, ϕ-трасса $\boldsymbol{\mu}_1$ получена с помощью правил вывода $(\phi 1\div 4)$. Тогда маршруты можно представить в виде $K=K_1\cdot K_2$ и $L=L_1\cdot L_2$, причём $\boldsymbol{\kappa}_1\in\boldsymbol{traces}_{\phi}(K_1)$ и $\boldsymbol{\lambda}_1\in\boldsymbol{traces}_{\phi}(L_1)$. Построим маршруты $M\in K\between L$ и $M_1\in K_1\between L_1$ по описанным выше однозначным правилам, определяемым последовательностью применения правил вывода $(\phi 1\div 3)$ и описанной дисциплиной выбора τ-переходов. Очевидно, $M=M_1\cdot M_2$, где M_2 – бесконечный τ-маршрут. По доказанному случаю, $\boldsymbol{\mu}_1\in\boldsymbol{traces}_{\phi}(M_1)$. Тогда $\boldsymbol{\mu}=\boldsymbol{\mu}_1\cdot\langle\gamma\rangle\in\boldsymbol{traces}_{\phi}^{\omega}(M)$.

Утверждение 2.2 доказано.

2.3. Пусть не применяется правило вывода $(\phi 5)$, но применяется правило индуктивного предела $(\phi 6)$.

Этот случай отличается от случая 2.1 только тем, что, по правилу индуктивного предела, ϕ-трасса $\boldsymbol{\mu}$ формируется бесконечной последовательностью применения правил вывода $(\phi 0\div 4)$. Одна из ϕ-трасс-операндов бесконечна.

Если оба маршрута K и L конечные, то каждая из ϕ-трасс $\boldsymbol{\kappa}$ и $\boldsymbol{\lambda}$ заканчивается бесконечным повторением ϕ-символа конечного состояния маршрута K и L, соответственно. Тогда маршрут M конечен, $M\in K\between L$ и $\boldsymbol{\mu}\in\boldsymbol{traces}_{\phi}^{\omega}(M)$.

Если хотя бы один из маршрутов K или L бесконечный, то маршрут M получается с использование правила индуктивного предела. Также $M \in K \between L$ и $\mu \in \boldsymbol{traces}_{\phi}^{\omega}(M)$.

Утверждение 2.3 доказано.

Утверждение 2 доказано.

60. Доказательство Утверждение 60:

ϕ-трасса LTS является ϕ-трассой некоторого маршрута LTS, а маршрут композиции LTS принадлежит композиции некоторых маршрутов LTS-операндов. Поэтому требуемое утверждение непосредственно следует из Утверждение 59:.

61. Доказательство Утверждение 61:

Рефлексивность и транзитивность мажорирования ϕ-трасс следует из рефлексивности и транзитивности мажорирования базовых и ϕ-символов.

Мажорирование базовых символов рефлексивно и транзитивно, поскольку совпадает с равенством: $a \preccurlyeq b = a=b$.

Мажорирование ϕ-символов рефлексивно, поскольку

$a \preccurlyeq a = a_r \subseteq a_r \cup a_g \ \& \ a_g \subseteq a_g$,

и транзитивно, поскольку

$$a \preccurlyeq b \preccurlyeq c = a_r \subseteq b_r \cup b_g \ \& \ a_g \subseteq b_g \ \& \ b_r \subseteq c_r \cup c_g \ \& \ b_g \subseteq c_g$$
$$\Rightarrow \ a_r \subseteq b_r \cup b_g \subseteq c_r \cup c_g \cup c_g \ \& \ a_g \subseteq b_g \subseteq c_g$$
$$\Rightarrow \ a_r \subseteq c_r \cup c_g \ \& \ a_g \subseteq c_g.$$

62. Доказательство Утверждение 62:

Непосредственно следует из Утверждение 61:.

63. Доказательство Утверждение 63:

Конечная ϕ-трасса мажорируется только конечной ϕ-трассой, а бесконечная ϕ-трасса порождает пустое множество βγδ-трасс, которое мажорируется любым множеством βγδ-трасс. Поэтому нам достаточно рассмотреть случай конечных ϕ-трасс.

В ϕ-трассе два идущих подряд ϕ-символа совпадают. Для каждого ϕ-символа a, очевидно, $\xi(\langle a,a\rangle)=\xi(\langle a\rangle)\cdot\xi(\langle a\rangle)=\beta\delta(a_r)^*\cdot\beta\delta(a_r)^*=\beta\delta(a_r)^*=\xi(\langle a\rangle)$. Таким образом, повторные ϕ-символы ничего не добавляют к множеству порождаемых βγδ-трасс. Тем самым, нам достаточно ограничиться нормальными ϕ-трассами, получаемыми удалением повторных ϕ-символов.

Пусть $C\subseteq Z$, $\mathbf{I},\mathbf{S}\in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$, $\boldsymbol{\mu}\in \boldsymbol{traces}_{\phi}(\mathbf{I})$ нормальная ϕ-трасса, $\boldsymbol{\sigma}\in \boldsymbol{traces}_{\phi}(\mathbf{S})$, $\boldsymbol{\mu}\preccurlyeq\boldsymbol{\sigma}$. Нам надо показать, что $\xi(\boldsymbol{\mu})\preccurlyeq\cup\circ\xi\circ\boldsymbol{traces}_{\phi}(\mathbf{S})$.

Мы покажем, что βγδ-трасса $\mu\in\xi(\boldsymbol{\mu})$ либо γ-мажорируется βγδ-трассой, порождаемой некоторой ϕ-трассой LTS $\mathbf{S}$, либо порождается ϕ-трассой $\boldsymbol{\sigma}$.

Будем вести доказательство индукцией по длине n ϕ-трассы $\boldsymbol{\mu}$.

<u>База индукции</u>.

Пустая ($n=0$) ϕ-трасса $\boldsymbol{\mu}=\epsilon$ порождает только пустую βγδ-трассу $\mu=\epsilon$.

Пустая ϕ-трасса $\boldsymbol{\mu}=\epsilon$ либо 1) γ-мажорируется ϕ-трассой $\boldsymbol{\sigma}=\langle\gamma\rangle$, либо 2) α-мажорируется пустой ϕ-трассой $\boldsymbol{\sigma}=\epsilon$.

В случае 1) $\mu=\epsilon$ γ-мажорируется βγδ-трассой $\langle\gamma\rangle$, порождаемой ϕ-трассой $\langle\gamma\rangle\in\boldsymbol{traces}_{\phi}(\mathbf{S})$.

В случае 2) $\mu=\epsilon$ порождается ϕ-трассой $\boldsymbol{\sigma}=\epsilon$.

<u>Шаг индукции</u>.

Допустим утверждение верно для всех ϕ-трасс длины меньше $n>0$.
Рассмотрим ϕ-трассу $\boldsymbol{\mu}$ длины n.
Имеем $\boldsymbol{\mu}\preccurlyeq\boldsymbol{\sigma}$.

Возможны два варианта.

1. $\boldsymbol{\mu}\preccurlyeq^{\alpha}\boldsymbol{\sigma}$.

Поскольку $n>0$, ϕ-трассу $\boldsymbol{\mu}$ можно представить в виде $\boldsymbol{\mu}=\boldsymbol{\mu}`\cdot\langle a\rangle$, где $a\in C_{\gamma}$ или $a\in\Phi(C)$.
Тогда βγδ-трассу $\mu\in\xi(\boldsymbol{\mu})$ можно представить в виде $\mu=\mu`\cdot\mathbf{o}$, где $\mu`\in\xi(\boldsymbol{\mu}`)$, $\mathbf{o}=\langle a\rangle$, если $a\in C_{\gamma}$, или $\mathbf{o}\in\xi(a)=\beta\delta(a_r)^*$, если $a\in\Phi(C)$.

Поскольку ϕ-трассы α-мажорируются, $\boldsymbol{\mu}$ и $\boldsymbol{\sigma}$ отличаются только ϕ-символами, стоящими в соответствующих позициях, причём ϕ-символ мажорируемой ϕ-трассы мажорируется ϕ-символом мажорирующей ϕ-трассы. Следовательно, ϕ-трассу $\boldsymbol{\sigma}$ можно представить в виде $\boldsymbol{\sigma}=\boldsymbol{\sigma}`\cdot\langle b\rangle$, где $b=a\in C_\gamma$ или $a,b\in\Phi(C)$ и $a\preccurlyeq b$, и, кроме того, $\boldsymbol{\mu}`\preccurlyeq\boldsymbol{\sigma}`$ ($\boldsymbol{\mu}`\preccurlyeq^{\alpha}\boldsymbol{\sigma}`$).

Итак, $\boldsymbol{\mu}`\preccurlyeq\boldsymbol{\sigma}`$ и $\mu`\in\xi(\boldsymbol{\mu}`)$.

Поскольку ϕ-трасса $\boldsymbol{\mu}`$ имеет длину меньше n, для неё, по предположению шага индукции, утверждение верно.
Возможны три варианта.

a) $\mu`\preccurlyeq^{\gamma}\sigma`$, где $\sigma`$ порождается некоторой ϕ-трассой LTS $\mathbf{S}$.

По определению мажорирования βγδ-трасс, $\mu=\mu`\cdot o$ и $\mu`\preccurlyeq^{\gamma}\sigma`$ влечёт $\mu\preccurlyeq^{\gamma}\sigma`$, что и требовалось доказать.

b) $\mu`\in\xi(\boldsymbol{\sigma}`)$ и $b=a\in C_\gamma$.

Тогда $\mu`\in\xi(\boldsymbol{\sigma}`)$ влечёт $\mu=\mu`\cdot\langle a\rangle\in\xi(\boldsymbol{\sigma}`\cdot\langle a\rangle)=\xi(\boldsymbol{\sigma})$, что и требовалось доказать.

c) $\mu`\in\xi(\boldsymbol{\sigma}`)$ и $a,b\in\Phi(C)$, $a\preccurlyeq b$ и $o\in\beta\delta(a_r)^*$.

Возможны два случая.

i. $o\in\beta\delta(b_r)^*$.

Тогда $\mu`\in\xi(\boldsymbol{\sigma}`)$ влечёт $\mu=\mu`\cdot o\in\xi(\boldsymbol{\sigma}`\cdot\langle b\rangle)=\xi(\boldsymbol{\sigma}`)$, что и требовалось доказать.

ii. $o\notin\beta\delta(b_r)^*$.

Пусть u первый βδ-отказ в o, который $u\notin\beta\delta(b_r)$.

По условию мажорируемости ϕ-символов $a\preccurlyeq b$, имеем $u\subseteq b_r\cup b_g$.

Значит есть такой символ $z\in u$, что $z\in b_g$ и в LTS $\mathbf{S}$ есть ϕ-трасса $\boldsymbol{\sigma}`\cdot\langle z,\gamma\rangle$.

А тогда $\mu`\in\xi(\boldsymbol{\sigma}`)$ влечёт $\mu`\cdot\langle z,\gamma\rangle\in\xi(\boldsymbol{\sigma}`\cdot\langle z,\gamma\rangle)$.

Кроме того, $\mu=\mu`\cdot o\preccurlyeq^{\gamma}\mu`\cdot\langle z,\gamma\rangle$, что и требовалось доказать.

2. $\boldsymbol{\mu} \preccurlyeq^{\gamma} \boldsymbol{\sigma}$.

В этом случае $\exists \boldsymbol{\mu}` \leq \boldsymbol{\mu}\ \exists \boldsymbol{\sigma}`\ \ \boldsymbol{\mu}` \preccurlyeq^{\alpha} \boldsymbol{\sigma}`\ \ \&\ \ \boldsymbol{\sigma} = \boldsymbol{\sigma}` \cdot \langle\gamma\rangle$.

Если $\boldsymbol{\mu}` = \boldsymbol{\mu}$, то, по доказанному варианту 1, либо $\mu$ $\gamma$-мажорируется $\beta\gamma\delta$-трассой, порождаемой некоторой $\phi$-трассой LTS $\mathbf{S}$, либо $\mu \in \xi(\boldsymbol{\sigma}`)$. В последнем случае $\mu \cdot \langle\gamma\rangle \in \xi(\boldsymbol{\sigma}` \cdot \langle\gamma\rangle) = \xi(\boldsymbol{\sigma})$ и $\mu \preccurlyeq^{\gamma} \mu \cdot \langle\gamma\rangle$, что и требовалось доказать.

Если $\boldsymbol{\mu}` \neq \boldsymbol{\mu}$, то $\phi$-трасса $\boldsymbol{\mu}`$ имеет длину меньше n и для неё, по предположению шага индукции, утверждение верно.
Префикс порождающей ϕ-трассы порождает префикс порождаемой $\beta\gamma\delta$-трассы: $\exists \mu` \in \xi(\boldsymbol{\mu}`)\ \ \mu` \leq \mu$.

Возможны два варианта.

a) $\mu` \preccurlyeq^{\gamma} \sigma`$, где $\beta\gamma\delta$-трасса $\sigma`$ порождается некоторой $\phi$-трассой LTS $\mathbf{S}$. Тогда, по определению мажорирования $\beta\gamma\delta$-трасс, $\mu` \leq \mu$ и $\mu` \preccurlyeq^{\gamma} \sigma`$ влечёт $\mu \preccurlyeq^{\gamma} \sigma`$, что и требовалось доказать.

b) $\mu` \in \xi(\boldsymbol{\sigma}`)$. Тогда $\mu` \cdot \langle\gamma\rangle \in \xi(\boldsymbol{\sigma}` \cdot \langle\gamma\rangle) = \xi(\boldsymbol{\sigma})$ и $\mu` \preccurlyeq^{\gamma} \mu` \cdot \langle\gamma\rangle$. Поскольку $\mu` \leq \mu$, имеем $\mu \preccurlyeq^{\gamma} \mu` \cdot \langle\gamma\rangle$, что и требовалось доказать.

Утверждение доказано.

64. Доказательство Утверждение 64:

Пусть $A, B \subseteq Z$, $\mathbf{I}_1, \mathbf{S}_1 \in \boldsymbol{LTS}_{\beta\gamma\delta}(A)$, $\mathbf{I}_2 \in \boldsymbol{LTS}_{\beta\gamma\delta}(B)$.

Обозначим $\mathbf{l_1} = \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{I}_1)$, $\mathbf{l_2} = \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{I}_2)$, $\boldsymbol{\Sigma_1} = \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S}_1)$.

Пусть $\boldsymbol{\kappa} \in \mathbf{l_1}$, $\boldsymbol{\lambda} \in \mathbf{l_2}$, $\boldsymbol{\kappa}` \in \boldsymbol{\Sigma_1}$ и $\boldsymbol{\kappa} \preccurlyeq \boldsymbol{\kappa}`$.

Возьмём произвольную ϕ-трассу $\boldsymbol{\mu} \in \boldsymbol{\kappa} \between \boldsymbol{\lambda}$.

Нам надо показать, что $\{\boldsymbol{\mu}\} \preccurlyeq \cup(\boldsymbol{\Sigma_1} \between \mathbf{l_2})$.

Рассмотрим два возможных случая.

1. $\boldsymbol{\kappa} \preccurlyeq^{\alpha} \boldsymbol{\kappa}`$.

Для данных ϕ-трасс **κ** и **λ** ϕ-трасса **μ** однозначно определяется последовательностью применения правил композиции (ϕ1÷4) и, быть может, дополнительным правилом (ϕ5) для построения ϕ-трасс, завершающихся разрушением, или правилом (ϕ6) для построения бесконечных ϕ-трасс.

Из определения α-мажорирования следует, что ϕ-трассы **κ** и **κ`** отличаются только i-ыми ϕ-символами $a=\kappa_{\downarrow}\Phi(A)(i)$ и $a`=\kappa`_{\downarrow}\Phi(A)(i)$. Отсюда следует, что для любой последовательности применения правил композиции ϕ-трасс **κ** и **λ**, определяющей ϕ-трассу **μ**, эту последовательность можно применить к ϕ-трассам **κ`** и **λ**, и получить некоторую ϕ-трассу **μ`**. При этом ϕ-трассы **μ** и **μ`** могут отличаться только результатами композиций ϕ-символов: для $b=\lambda_{\downarrow}\Phi(B)(i)$ могут различаться $a\wr\!\downarrow b$ и $a`\wr\!\downarrow b$.

Рассмотрим два возможных варианта.

а. Для каждого i-ого ϕ-символа верно: $\forall z\in A\cap\underline{B}\cap a`_{g}\ \underline{z}\in b_{r}$.
Мы покажем, что существуют такие ϕ-трассы $\kappa`_0\in\Sigma_1$ и $\lambda_0\in I_2$, которые получаются из ϕ-трасс **κ`** и **λ**, соответственно, удалением некоторых ϕ-символов, и такие, что некоторая ϕ-трасса $\mu`_0\in\kappa`_0\wr\!\downarrow\lambda_0$ мажорирует $\mu\preccurlyeq^{\alpha}\mu`_0$.

Отсюда будет следовать нужное утверждение. Действительно, по Утверждение 55:, $\kappa`_0\in\Sigma_1$ и $\lambda_0\in I_2$, что влечёт $\mu`_0\in\kappa`_0\wr\!\downarrow\lambda_0\subseteq\cup(\Sigma_1\wr\!\downarrow I_2)$. Тем самым, $\mu\preccurlyeq\cup(\Sigma_1\wr\!\downarrow I_2)$.

Нам достаточно показать, что $a\wr\!\downarrow b\neq\tau$ влечёт $a`\wr\!\downarrow b\neq\tau$ и $a\wr\!\downarrow b\preccurlyeq a`\wr\!\downarrow b$.

Действительно, если это условие выполнено, то мажорирование $\mu\preccurlyeq^{\alpha}\mu`$ может быть нарушено только из-за того, что при построении ϕ-трассы **μ** композиция ϕ-символов $a\wr\!\downarrow b=\tau$, а при построении ϕ-трассы **μ`** композиция ϕ-символов $a`\wr\!\downarrow b\neq\tau$.
Удаление всех таких ϕ-символов a` из ϕ-трассы **κ`** и соответствующих им ϕ-символов b из ϕ-трассы **λ** даёт ϕ-трассы

$\boldsymbol{\kappa}`_0$ и $\boldsymbol{\lambda}_0$, композиция которых даёт ϕ-трассу $\boldsymbol{\mu}`_0\in\boldsymbol{\kappa}`_0\between\boldsymbol{\lambda}_0$, которая и будет искомой мажорирующей ϕ-трассой $\boldsymbol{\mu}\preccurlyeq^{\alpha}\boldsymbol{\mu}`_0$.

По Утверждение 55:, $\boldsymbol{\kappa}`_0\in\boldsymbol{\Sigma}_1$ и $\boldsymbol{\lambda}_0\in\boldsymbol{I}_2$, что влечёт $\boldsymbol{\mu}`_0\in\boldsymbol{\kappa}`_0\between\boldsymbol{\lambda}_0\subseteq\cup(\boldsymbol{\Sigma}_1\between\boldsymbol{I}_2)$.

Тем самым, $\{\boldsymbol{\mu}\}\preccurlyeq\cup(\boldsymbol{\Sigma}_1\between\boldsymbol{I}_2)$, что и требовалось доказать.

Покажем, что $a\between b\neq\tau$ влечёт $a`\between b\neq\tau$ и $a\between b\preccurlyeq a`\between b$.

Сначала покажем, что $a`\between b\neq\tau$.

Если $a\between b\neq\tau$, то, по определению композиции ϕ-символов, $(A\backslash a_r)\cap(\underline{B}\backslash b_r)=\varnothing$.

По определению мажорирования ϕ-трасс, $a_r\subseteq a`_r\cup a`_g$ и $a_g\subseteq a`_g$.

Покажем, что $(A\backslash a`_r)\cap(\underline{B}\backslash b_r)=\varnothing$.

Для произвольного $z\in A$ рассмотрим все возможные варианты (напомним, что $a`_r$ и $a`_g$ не пересекаются):

i. $z\notin A\cap\underline{B}$.
Тогда $z\notin(A\backslash a`_r)\cap(\underline{B}\backslash b_r)$.

ii. $z\in a`_r$.
Тогда $z\notin A\backslash a`_r$ и, следовательно, $z\notin(A\backslash a`_r)\cap(\underline{B}\backslash b_r)$.

iii. $z\in a`_g$.
Тогда, по условию рассматриваемого варианта a, $\underline{z}\in b_r$, что влечёт $z\notin\underline{B}\backslash b_r$ и, следовательно, $z\notin(A\backslash a`_r)\cap(\underline{B}\backslash b_r)$.

iv. $z\notin a`_r$ и $z\notin a`_g$.
Тогда $z\notin a`_r\cup a`_g$. Отсюда, поскольку $a_r\subseteq a`_r\cup a`_g$, имеем $z\notin a_r$. Следовательно, $z\in A\backslash a_r$. Отсюда, поскольку $(A\backslash a_r)\cap(\underline{B}\backslash b_r)=\varnothing$, имеем $z\notin\underline{B}\backslash b_r$, что влечёт $z\notin(A\backslash a`_r)\cap(\underline{B}\backslash b_r)$.

Мы доказали, что $(A\backslash a`_r)\cap(\underline{B}\backslash b_r)=\varnothing$.

Отсюда, по определению композиции ϕ-символов, $a`\between b\neq\tau$.

Теперь покажем, что $a\between b\preccurlyeq a`\between b$.

По определению композиции ϕ-символов,
во-первых, $(a⫛b)_r = (A\underline{B})\cap a_r \cup (B\underline{A})\cap b_r$
и $(a`⫛b)_r = (A\underline{B})\cap a`_r \cup (B\underline{A})\cap b_r$,
во-вторых, $(a⫛b)_g = (A\underline{B})\cap a_g \cup (B\underline{A})\cap b_g$
и $(a`⫛b)_g = (A\underline{B})\cap a`_g \cup (B\underline{A})\cap b_g$.
Поскольку $a_r \subseteq a`_r \cup a`_g$, имеем $(a⫛b)_r \subseteq (a`⫛b)_r \cup (a`⫛b)_g$.
Поскольку, $a_g \subseteq a`_g$, имеем $(a⫛b)_g \subseteq (a`⫛b)_g$.
Отсюда, по определению мажорирования ϕ-символов, $a⫛b \preccurlyeq a`⫛b$.

Утверждение варианта a доказано.

b. Для некоторого i-ого ϕ-символа имеет место: $\exists z \in A\cap\underline{B}\cap a`_g \ \underline{z} \notin b_r$.
Мы покажем, что $\boldsymbol{\mu} \preccurlyeq^{\gamma} \cup(\boldsymbol{\Sigma_1}⫛\boldsymbol{I_2})$.

Выберем самый первый такой индекс i.
Тогда ϕ-трассы можно представить в таком виде $\boldsymbol{\kappa}=\boldsymbol{\kappa}_1\cdot\langle a\rangle\cdot\boldsymbol{\kappa}_2$, $\boldsymbol{\kappa}`=\boldsymbol{\kappa}`_1\cdot\langle a`\rangle\cdot\boldsymbol{\kappa}`_2$, $\boldsymbol{\lambda}=\boldsymbol{\lambda}_1\cdot\langle b\rangle\cdot\boldsymbol{\lambda}_2$, $\boldsymbol{\mu}=\boldsymbol{\mu}_1\cdot(\langle a⫛b\rangle\uparrow\{\tau\})\cdot\boldsymbol{\mu}_2$, $\boldsymbol{\mu}`=\boldsymbol{\mu}`_1\cdot(\langle a`⫛b\rangle\uparrow\{\tau\})\cdot\boldsymbol{\mu}`_2$, что $\boldsymbol{\mu}_1 \in \boldsymbol{\kappa}_1⫛\boldsymbol{\lambda}_1$, $\boldsymbol{\mu}`_1 \in \boldsymbol{\kappa}`_1⫛\boldsymbol{\lambda}_1$.

Здесь последовательность правил $(\phi 1\div 4)$, порождающих $\boldsymbol{\mu}_1$ и $\boldsymbol{\mu}`_1$, является префиксом последовательности правил $(\phi 1\div 4)$, порождающих $\boldsymbol{\mu}$ и $\boldsymbol{\mu}`$.

По Утверждение 55:, $\boldsymbol{\kappa}_1\cdot\langle a\rangle \in \boldsymbol{I_1}$, $\boldsymbol{\kappa}`_1\cdot\langle a`\rangle \in \boldsymbol{\Sigma_1}$ и $\boldsymbol{\lambda}_1\cdot\langle b\rangle \in \boldsymbol{I_2}$.
Тогда, по доказанному утверждению варианта a, существуют ϕ-трассы $\boldsymbol{\kappa}`_{10} \in \boldsymbol{\Sigma_1}$ и $\boldsymbol{\lambda}_{10} \in \boldsymbol{I_2}$, которые получаются из ϕ-трасс $\boldsymbol{\kappa}`_1$ и $\boldsymbol{\lambda}_1$, соответственно, удалением некоторых ϕ-символов, и такие, что некоторая ϕ-трасса $\boldsymbol{\mu}`_{10} \in \boldsymbol{\kappa}`_{10}⫛\boldsymbol{\lambda}_{10}$ мажорирует $\boldsymbol{\mu}_1 \preccurlyeq^{\alpha} \boldsymbol{\mu}`_{10}$.

По Утверждение 55:, $\boldsymbol{\kappa}`_{10}\cdot\langle a`\rangle \in \boldsymbol{\Sigma_1}$ и $\boldsymbol{\lambda}_{10}\cdot\langle b\rangle \in \boldsymbol{I_2}$.

По правилам вывода $(\phi 3)$ и $(\phi 1)$, $\boldsymbol{\mu}`_{10} \in \boldsymbol{\kappa}`_{10}⫛\boldsymbol{\lambda}_{10}$ влечёт $\boldsymbol{\mu}`_{10}\cdot\langle\gamma\rangle \in (\boldsymbol{\kappa}`_{10}\cdot\langle z,\gamma\rangle)⫛(\boldsymbol{\lambda}_{10}\cdot\langle\underline{z}\rangle)$.

Поскольку $z\in a`_g$ и $\boldsymbol{\kappa}`_{10}\cdot\langle a`\rangle\in\boldsymbol{\Sigma_1}$, по Утверждение 55:, $\boldsymbol{\kappa}`_{10}\cdot\langle z,\gamma\rangle\in\boldsymbol{\Sigma_1}$.

Поскольку, по условию рассматриваемого варианта b, $\underline{z}\in A\cap\underline{B}\setminus b_r$ и $\boldsymbol{\lambda}_{10}\cdot\langle b\rangle\in\boldsymbol{I_2}$, то по Утверждение 55:, $\boldsymbol{\lambda}_{10}\cdot\langle\underline{z}\rangle\in\boldsymbol{I_2}$.

Следовательно, $\boldsymbol{\mu}`_{10}\cdot\langle\gamma\rangle\in\cup(\boldsymbol{\Sigma_1}\upharpoonright\downarrow\boldsymbol{I_2})$.

По определению мажорирования ϕ-трасс, $\boldsymbol{\mu}_1\leq\boldsymbol{\mu}$ и $\boldsymbol{\mu}_1\preccurlyeq^{\alpha}\boldsymbol{\mu}`_{10}$ влечёт $\boldsymbol{\mu}\preccurlyeq\boldsymbol{\mu}`_{10}\cdot\langle\gamma\rangle$.

Итак, $\boldsymbol{\mu}\preccurlyeq\boldsymbol{\mu}`_{10}\cdot\langle\gamma\rangle$, $\boldsymbol{\mu}`_{10}\cdot\langle\gamma\rangle\in\cup(\boldsymbol{\Sigma_1}\upharpoonright\downarrow\boldsymbol{I_2})$, что и требовалось доказать.

Для варианта b утверждение доказано.

Для случая 1 утверждение доказано.

2. $\boldsymbol{\kappa}\preccurlyeq^{\gamma}\boldsymbol{\kappa}`$.

В этом случае ϕ-трассы можно представить в таком виде $\boldsymbol{\kappa}=\boldsymbol{\kappa}_1\cdot\boldsymbol{\kappa}_2$, $\boldsymbol{\kappa}`=\boldsymbol{\kappa}_1\cdot\langle\gamma\rangle$, $\boldsymbol{\lambda}=\boldsymbol{\lambda}_1\cdot\boldsymbol{\lambda}_2$, $\boldsymbol{\mu}=\boldsymbol{\mu}_1\cdot\boldsymbol{\mu}_2$, что $\boldsymbol{\mu}_1\in\boldsymbol{\kappa}_1\upharpoonright\downarrow\boldsymbol{\lambda}_1$.

Здесь последовательность правил $(\phi 1\div 4)$, порождающих $\boldsymbol{\mu}_1$, является префиксом последовательности правил $(\phi 1\div 4)$, порождающих $\boldsymbol{\mu}$.

По доказанному случаю α-мажорирования, возможна два варианта.

a. Существуют такие ϕ-трассы $\boldsymbol{\kappa}`_0\in\boldsymbol{\Sigma_1}$ и $\boldsymbol{\lambda}_0\in\boldsymbol{I_2}$, которые получаются из ϕ-трасс $\boldsymbol{\kappa}`$ и $\boldsymbol{\lambda}$, соответственно, удалением некоторых ϕ-символов, и такие, что некоторая ϕ-трасса $\boldsymbol{\mu}`_0\in\boldsymbol{\kappa}`_0\upharpoonright\downarrow\boldsymbol{\lambda}_0$ мажорирует $\boldsymbol{\mu}\preccurlyeq^{\alpha}\boldsymbol{\mu}`_0$.

По правилу вывода $(\phi 1)$, $\boldsymbol{\mu}`_{10}\in\boldsymbol{\kappa}`_{10}\upharpoonright\downarrow\boldsymbol{\lambda}_{10}$ влечёт $\boldsymbol{\mu}`_{10}\cdot\langle\gamma\rangle\in(\boldsymbol{\kappa}`_{10}\cdot\langle\gamma\rangle)\upharpoonright\downarrow\boldsymbol{\lambda}_{10}$.

По Утверждение 55:, $\boldsymbol{\kappa}`_{10}\cdot\langle\gamma\rangle\in\boldsymbol{\Sigma_1}$.

Тем самым, $\boldsymbol{\mu}\preccurlyeq^{\gamma}\boldsymbol{\mu}`_{10}\cdot\langle\gamma\rangle\in(\boldsymbol{\kappa}`_{10}\cdot\langle\gamma\rangle)\upharpoonright\downarrow\boldsymbol{\lambda}_{10}\subseteq\cup(\boldsymbol{\Sigma_1}\upharpoonright\downarrow\boldsymbol{I_2})$.

Следовательно, $\boldsymbol{\mu}\preccurlyeq^{\gamma}\cup(\boldsymbol{\Sigma_1}\upharpoonright\downarrow\boldsymbol{I_2})$.

b. $\boldsymbol{\mu}_1\preccurlyeq^{\gamma}\cup(\boldsymbol{\Sigma_1}\upharpoonright\downarrow\boldsymbol{I_2})$.

Тогда $\boldsymbol{\mu}_1\leq\boldsymbol{\mu}$ влечёт $\boldsymbol{\mu}\preccurlyeq^{\gamma}\cup(\boldsymbol{\Sigma_1}\upharpoonright\downarrow\boldsymbol{I_2})$.

Для случая 2 утверждение доказано.

Утверждение доказано.

65. **Доказательство Утверждение 65:**

Пусть для $\mathbf{I_1, I_2, S_1, S_2} \in \boldsymbol{LTS}_{\beta\gamma\delta}$ имеет место мажорирование:
$\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{I_1}) \preccurlyeq \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S_1})$ & $\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{I_2}) \preccurlyeq \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S_2})$.

По композиционности мажорирования ϕ-трасс слева (Утверждение 64:),
$\cup(\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{I_1}) \upharpoonright\!\downarrow \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{I_2})) \preccurlyeq \cup(\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S_1}) \upharpoonright\!\downarrow \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{I_2}))$.

По коммутативности композиции ϕ-трасс (Утверждение 58:),
$\cup(\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S_1}) \upharpoonright\!\downarrow \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{I_2})) = \cup(\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{I_2}) \upharpoonright\!\downarrow \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S_1}))$.

По композиционности мажорирования ϕ-трасс слева (Утверждение 64:),
$\cup(\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{I_2}) \upharpoonright\!\downarrow \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S_1})) \preccurlyeq \cup(\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S_2}) \upharpoonright\!\downarrow \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S_1}))$.

По коммутативности композиции ϕ-трасс (Утверждение 58:),
$\cup(\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S_2}) \upharpoonright\!\downarrow \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S_1})) = \cup(\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S_1}) \upharpoonright\!\downarrow \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S_2}))$.

По транзитивности мажорирования ϕ-трасс (Утверждение 61:),
$\cup(\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{I_1}) \upharpoonright\!\downarrow \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{I_2})) \preccurlyeq \cup(\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S_1}) \upharpoonright\!\downarrow \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S_2}))$.

66. **Доказательство Утверждение 66:**

1) Power-состояние $[\mu]=(\mathbf{S}\ \boldsymbol{after}\ \mu)$ τ-замкнуто по определению ***after***.

2) В силу конвергентности βγδ-модели тау-трасса $\mu \in \boldsymbol{T}(\Sigma)$ полна или продолжается некоторым βδ-отказом. Тем самым для некоторой (быть может, пустой) трассы βδ-отказов $\mathbf{o}$ βγδ-трасса $\mu\cdot\mathbf{o}$ стабильна. А тогда, по правилам вывода, в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ существует τ-переход $[\mu] \text{—}\tau\rightarrow [\mu\cdot\mathbf{o}]\mathtt{t}$, где $\mathtt{t} \in !\mathtt{C}_{\gamma\delta}$.

3) По правилам вывода, τ-переход начинается в power-состоянии вида $[\mu]$, а γ-переход определяется только в γ-состоянии. Тем самым, из power-состояния вида $[\mu]\mathtt{t}$ не выходят τ- и γ-переходы, то есть оно стабильно.

67. **Доказательство Утверждение 67:**

Будем вести доказательство индукцией по βγδ-трассе μ.
Обозначим $\Sigma = \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$.

База индукции. Поскольку в γ-состоянии определён только один переход – γ-петля, то из существования power-состояния $[\epsilon]$ следует, что это начальное power-состояние и $\mathbf{S}$ ***safe***, следовательно, $\epsilon \in \boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})$. В этом случае также $\epsilon \in \boldsymbol{T}(\Sigma)$.

Шаг индукции. Допустим утверждение верно для βγδ-трассы $\mu \in \boldsymbol{T}(\Sigma)$ и докажем его для βγδ-трасс 1) $\mu\cdot\langle z\rangle \in \boldsymbol{T}(\Sigma)$, 2) $\mu\cdot o\cdot\langle z\rangle \in \boldsymbol{T}(\Sigma)$ и 3) $\mu\cdot o\cdot\langle z\rangle \in \boldsymbol{S}(\Sigma)$, где $z \in C$ и $o \in \beta\delta(C)^*$.

Для каждого возможного здесь случая есть соответствующее правило вывода, которое определяет переход для $t \in !C_{\gamma\delta}$:

1) $[\mu] \text{—} z \rightarrow [\mu\cdot\langle z\rangle]$,

2) $[\mu] \text{—} \tau \rightarrow [\mu\cdot o]t \text{—} z \rightarrow [\mu\cdot o\cdot\langle z\rangle]$,

3) $[\mu] \text{—} \tau \rightarrow [\mu\cdot o]t$.

Нам надо показать, что $[\mu] \in (\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ ***after*** $\mu)$ влечёт:

1) $[\mu\cdot\langle z\rangle] \in (\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ ***after*** $\mu\cdot\langle z\rangle)$,

2) $[\mu\cdot o\cdot\langle z\rangle] \in (\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ ***after*** $\mu\cdot o\cdot\langle z\rangle)$,

3) $[\mu\cdot o]t \in (\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ ***after*** $\mu\cdot o)$.

Случай 1) очевиден. Для остальных случаев достаточно показать, что в power-состоянии $[\mu\cdot o]t$ есть трасса отказов o, то есть в этом power-состоянии реализуется каждый отказ $r \in \boldsymbol{Im}(o)$. А это непосредственно следует из правил вывода, определяющих в таком power-состоянии переход по стимулу или реакции u только в том случае, когда $\mu\cdot o\cdot\langle u\rangle \in \boldsymbol{T}(\Sigma)$, то есть когда $\beta\delta(u) \notin \boldsymbol{Im}(o)$.

68. **Доказательство Утверждение 68:**

Обозначим: $\mathbf{T}=\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ и $\Sigma=\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$.

Сначала рассмотрим случай $t_0=\gamma$.

В LTS $\mathbf{T}$ есть только пустая βγδ-трасса и βγδ-трасса $\langle\gamma\rangle$. Этот случай возможен только тогда, когда LTS $\mathbf{S}$ опасна, то есть в ней тоже есть пустая

βγδ-трасса и βγδ-трасса $\langle\gamma\rangle$. Имеем: $\epsilon \preccurlyeq \epsilon \in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$ и $\langle\gamma\rangle \preccurlyeq \langle\gamma\rangle \in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$.

Теперь рассмотрим случай $t_0=[\epsilon]$.
Доказательство будем вести индукцией по маршруту P.

База индукции. Пустой маршрут $P=\epsilon$ заканчивается в начальном power-состоянии $t_0=[\epsilon]$, которое, по Утверждение 66: (2), нестабильно. Поэтому $\mathit{traces}_{\beta\gamma\delta}(\epsilon)=\{\epsilon\}$. Таким образом, $\sigma=\epsilon$ и для $\mu=\epsilon$ имеем $\sigma \in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$, $\omega(P)=[\mu]$ и $[\mu]\subseteq[\sigma]$.

Шаг индукции. Предположим, что утверждение доказано для маршрута P, заканчивающегося в power-состоянии $[\mu]$ или $[\mu]t$, где $t \in !C_{\gamma\delta}$, и докажем его для маршрута $P`$, получающегося из P с помощью ещё одного перехода. Мы рассмотрим четыре случая.

A. Переход по безопасному стимулу или реакции z.

По правилам вывода, такой переход ведёт в power-состояние $[\mu\cdot\langle z\rangle] \in \boldsymbol{T}(\Sigma)$.

По Утверждение 66: (2), power-состояние $[\mu\cdot\langle z\rangle]$ нестабильно.

Поэтому любая βγδ-трасса из $\mathit{traces}_{\beta\gamma\delta}(P`)$ имеет вид $\sigma\cdot\langle z\rangle$, где $\sigma \in \mathit{traces}_{\beta\gamma\delta}(P)$.
По предположению индукционного шага, $[\mu]\subseteq[\sigma]$ и $\sigma \in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$. Тогда
$[\mu\cdot\langle z\rangle] = \cup([\mu]\ \mathit{after}\ \langle z\rangle) \subseteq \cup([\sigma]\ \mathit{after}\ \langle z\rangle) = [\sigma\cdot\langle z\rangle]$.
Поскольку $[\mu\cdot\langle z\rangle]\neq\varnothing$, имеем $[\sigma\cdot\langle z\rangle]\neq\varnothing$, то есть $\sigma\cdot\langle z\rangle \in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$.
Итак, $\sigma\cdot\langle z\rangle \in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$, $\omega(P`)=[\mu\cdot\langle z\rangle]$ и $[\mu\cdot\langle z\rangle]\subseteq[\sigma\cdot\langle z\rangle]$.

B. Переход по опасному стимулу или реакции z.

По правилам вывода, такой переход ведёт в γ-состояние, которое, очевидно, нестабильно.

Поэтому любая βγδ-трасса из $\mathit{traces}_{\beta\gamma\delta}(P`)$ имеет вид $\sigma\cdot\langle z\rangle$, где $\sigma \in \mathit{traces}_{\beta\gamma\delta}(P)$.
По предположению индукционного шага, $\sigma \in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$.

Если $z=?x$ стимул, то он разрушающий.

Поэтому $\sigma\cdot\langle ?x\rangle\in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$ и $\sigma\cdot\langle ?x\rangle\preccurlyeq\sigma\cdot\langle ?x\rangle$.

Если $z=!y$ реакция, то существует разрушающая реакция $!t$, то есть $\sigma\cdot\langle !t,\gamma\rangle\in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$.

Поэтому $\sigma\cdot\langle !t,\gamma\rangle\in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$ и $\sigma\cdot\langle !y\rangle\preccurlyeq\sigma\cdot\langle !t,\gamma\rangle$.

C. τ-переход вида $[\mu]\text{—}\tau\rightarrow[\mu]t$, где $t\in !C_{\gamma\delta}$.

Такой переход существует, если μ стабильная тау-трасса, то есть полная трасса. Тогда, по правилам вывода, множество состояний $[\mu]$ полное. Тем самым, power-состояние $[\mu]t$ не порождает $\beta\delta$-отказов.
Поэтому любая $\beta\gamma\delta$-трасса $\sigma\in \mathit{traces}_{\beta\gamma\delta}(P`)$ также $\sigma\in \mathit{traces}_{\beta\gamma\delta}(P)$.
По предположению индукционного шага, $[\mu]\subseteq[\sigma]$ и $\sigma\in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$.
Итак, $\sigma\in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$, $\omega(P`)=[\mu]$ и $[\mu]\subseteq[\sigma]$.

D. τ-переход вида $[\mu]\text{—}\tau\rightarrow[\mu\cdot o]t$, где $t\in !C_{\gamma\delta}$ и o непустая трасса $\beta\delta$-отказов.

По Утверждение 66: (3), power-состояние $[\mu\cdot o]t$ стабильно.
Любая $\beta\gamma\delta$-трасса из $\mathit{traces}_{\beta\gamma\delta}(P`)$ имеет вид $\sigma\cdot o`$, где $\sigma\in \mathit{traces}_{\beta\gamma\delta}(P)$ и $o`\in\beta\delta([\mu\cdot o]t)^*$.
Очевидно, что каждое состояние $s\in[\mu\cdot o]$ стабильно.
По правилам вывода, $\{?x\}\in\beta\delta([\mu\cdot o]t)$, если в каждом состоянии $s\in[\mu\cdot o]$ нет перехода по $?x$. Поскольку состояние стабильно, это означает, что $\{?x\}\in\beta\delta(s)$.
Также, по правилам вывода, $\delta\in\beta\delta([\mu\cdot o]t)$, если в каждом состоянии $s\in[\mu\cdot o]$ нет переходов по реакциям. Поскольку состояние стабильно, это означает, что $\delta\in\beta\delta(s)$.
Таким образом, $o`\in\beta\delta(s)^*$ для каждого состояния $s\in[\mu\cdot o]$.
Следовательно, поскольку, по предположению индукционного шага, $[\mu]\subseteq[\sigma]$, $[\mu\cdot o]\subseteq[\mu]\subseteq[\sigma]$.
Отсюда, поскольку $\sigma\in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$, $\sigma\cdot o`\in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$.
Итак, $\sigma\cdot o`\in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$, $\omega(P`)=[\mu\cdot o]$ и $[\mu\cdot o]\subseteq[\sigma\cdot o`]$.

69. **Доказательство Утверждение 69:**

По Утверждение 68:, $\mathit{traces}_{\beta\gamma\delta}\circ\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})\preccurlyeq\mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$.

А тогда, по Утверждение 54:, $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ ***ioco***$_{\beta\gamma\delta}$ $\mathbf{S}$.

70. Доказательство Утверждение 70:

Если состояние s достижимо по безопасной βγδ-трассе, то в *S*-ветвящейся LTS дерево τ-маршрутов, начинающихся в s, конечно-ветвящееся. Поскольку состояние s и все состояния, достижимые из s по τ-переходам, конвергентны, в этом дереве нет бесконечного τ-маршрута (бесконечно-возрастающей цепочки τ-маршрутов). Следовательно, по теореме Кёнига [KÖNIG], это дерево конечно, то есть число τ-маршрутов, начинающихся в состоянии s, конечно.

Тем самым, для пустой безопасной βγδ-трассы $\sigma=\epsilon$ утверждение доказано.

Далее по индукции: пусть для некоторой безопасной βγδ-трассы σ множество $\mathbf{S}$ ***after*** σ конечно. Рассмотрим её безопасные продолжения.

Сначала рассмотрим продолжение стимулом или реакцией z, то есть $\sigma\cdot\langle z\rangle\in$ ***safe***$_{\beta\gamma\delta}(\mathbf{S})$.

Имеем $\mathbf{S}$ ***after*** $\sigma\cdot\langle z\rangle = \{r\in V_s \mid \exists s\in(\mathbf{S}$ ***after*** $\sigma)\ \exists s`\ s\text{—}z\rightarrow s`\Rightarrow r\}$. Для *S*-ветвящейся LTS и символа (стимула или реакции) z, безопасного после σ, в каждом из конечного числа состояний $s\in(\mathbf{S}$ ***after*** $\sigma)$ определено конечное число переходов $s\text{—}z\rightarrow s`$, то есть число таких состояний $s`$ конечно. Каждое состояние $s`$ достижимо по безопасной βγδ-трассе $\sigma\cdot\langle z\rangle$. Следовательно, дерево τ-маршрутов, начинающихся в $s`$, конечно. Тем самым, конечно множество конечных состояний этих маршрутов. В результате конечно множество $\mathbf{S}$ ***after*** $\sigma\cdot\langle z\rangle$.

Теперь рассмотрим продолжение βδ-отказом r. Поскольку отказы – это виртуальные петли в состояниях, имеет место вложение $\mathbf{S}$ ***after*** $\sigma\cdot\langle r\rangle \subseteq \mathbf{S}$ ***after*** σ. Поскольку множество $\mathbf{S}$ ***after*** σ конечно, его подмножество тоже конечно.

71. Доказательство Утверждение 71:

Обозначим $U = \mathbf{S}$ ***after*** σ и $U` = \cap(\mathbf{S}$ ***after*** $\{\sigma\}\cdot\mathbf{O})$.

Поскольку $\sigma \in \mathit{safe}_{\beta\gamma\delta}(\mathbf{S})$, $U \neq \varnothing$.

Допустим утверждение не верно, то есть $\neg\exists o \in \mathbf{O}\ \mathbf{S}\ \mathit{after}\ \sigma\cdot o = U` \neq \varnothing$.

Поскольку βδ-отказы – это виртуальные петли в стабильных состояниях, для каждой трассы βδ-отказов $o_i \in \mathbf{O}$ имеет место $\mathbf{S}\ \mathit{after}\ \sigma\cdot o_i \subseteq U$. Тем самым, $U` \subseteq U$.

Если $U`=U$, то для $o=\epsilon$ имеем $\mathbf{S}\ \mathit{after}\ \sigma\cdot o = U = U` \neq \varnothing$, что противоречит допущению.

Значит, $U` \subset U$.

По Утверждение 70:, множество состояний U конечно.
Поэтому множество $U\backslash U`$ конечно и, поскольку $U` \subset U$, не пусто.
Перенумеруем его состояния $s_1, \ldots, s_n$.

Поскольку множество $\mathbf{O}$ не пусто, для каждого состояния $s_i \in U\backslash U`$, найдётся такая трасса βδ-отказов $o_i \in \mathbf{O}$, что $s_i \notin \mathbf{S}\ \mathit{after}\ \sigma\cdot o_i$.
Рассмотрим конкатенацию $o=o_1\cdot\ldots\cdot o_n$.
По замкнутости множества $\mathbf{O}$ по конкатенации, $o \in \mathbf{O}$. Следовательно, $\sigma\cdot o \in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$ и $U` \subseteq \mathbf{S}\ \mathit{after}\ \sigma\cdot o \neq \varnothing$.
С другой стороны, трасса $\sigma\cdot o$ не может заканчиваться ни в одном из состояний s_i, так как тогда в этом состоянии заканчивалась бы трасса $\sigma\cdot o_i$, получаемая из $\sigma\cdot o$ удалением βδ-отказов.
Следовательно, $\mathbf{S}\ \mathit{after}\ \sigma\cdot o \subseteq U`$.
Таким образом, $\mathbf{S}\ \mathit{after}\ \sigma\cdot o = U` \neq \varnothing$.

Мы пришли к противоречию, и утверждение можно считать доказанным.

72. **Доказательство Утверждение 72:**

Действительно, если это не так, то βγδ-трассу $\mu\cdot o_1\cdot o_2$ можно представить в виде $\mu\cdot o_1\cdot o_2=\mu\cdot o_1\cdot o_{21}\cdot\langle r\rangle\cdot o_{22}$, где $\mu\cdot o_1\cdot o_{21} \in \mathit{safe}_{\beta\gamma\delta}(\mathbf{S})$, а $r \in \beta\delta(C)$ опасен после $\mu\cdot o_1\cdot o_{21}$.
Тогда существует $z \in r$ такой, что $\mu\cdot o_1\cdot o_{21}\cdot\langle z,\gamma\rangle \in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$.

Но тогда также $\mu\cdot o_{21}\cdot\langle z,\gamma\rangle \in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$, то есть $\mu\cdot o_2 \notin \mathit{safe}_{\beta\gamma\delta}(\mathbf{S})$, что не верно.

73. Доказательство Утверждение 73:

1. По определению преобразования $\mathcal{T}_{\beta\gamma\delta}$, если в power-состоянии LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ определён переход по разрушающей реакции, то в нём определены переходы по всем реакциям, и все такие переходы ведут в одно состояние γ. Тем самым, если $\beta\gamma\delta$-трасса продолжается некоторой реакцией и далее разрушением, то она это делает через некоторое power-состояние, через которое она также продолжается каждой реакцией и далее разрушением. Поэтому $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ γ-однородна.

2. Для power-состояния вида $[\mu]$ и $[\mu]t$, где $t \in !C_{\gamma\delta}$, $\beta\gamma\delta$-трасса μ безопасна. Поэтому, по Утверждение 70:, она заканчивается в конечном множестве состояний.

3. Пусть исходная LTS $\mathbf{S}$ *S*-ветвящаяся. Нам достаточно показать выполнение в преобразованной LTS соответствующего свойства для power-состояний вида $[\mu]$ и $[\mu]t$, где $t \in !C_{\gamma\delta}$. Согласно правилам вывода, в каждом таком power-состоянии определено не более одного перехода по каждому стимулу или реакции z. В нестабильном power-состоянии вида $[\mu]$ определено не большее число τ-переходов, чем число непустых подмножеств множества $[\mu]$, умноженное на число реакций. Поэтому достаточно конечности множества $[\mu]$, что доказано в п.2.

74. Доказательство Утверждение 74:

Пусть $s \in \mathit{der} \circ \mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$. Нам надо показать конечность числа переходов из состояния s по каждому стимулу или реакции z и по τ.

По построению, в $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ есть не более одного перехода $s \text{—} z \rightarrow s`$.

Если $s=\gamma$, то в γ-состоянии нет τ-переходов.

Если $s \neq \gamma$, то, по Утверждение 67:, состояние s достижимо в $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ по $\beta\gamma\delta$-трассе μ, безопасной в $\mathbf{S}$.
По конформности преобразования (Утверждение 69:), $\beta\gamma\delta$-трасса μ, имеющаяся в $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ и безопасная в $\mathbf{S}$, безопасна в $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$.
Следовательно, состояние s *S*-достижимо в $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$.
По Утверждение 73: (3), LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ *S*-ветвящаяся.
Следовательно, состояние s *S*-ветвящееся, то есть в нём определено конечное число τ-переходов.

В LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ все τ-переходы ведут в стабильные состояния.
Следовательно, каждое состояние τ-ограниченное.

75. **Доказательство Утверждение 75:**

Доказательство будем вести индукцией по ϕ-трассе $\boldsymbol{\mu}$.

База индукции. Пустая ϕ-трасса, очевидно, является нормальной и принадлежит любой LTS. Для $\boldsymbol{\mu}=\epsilon$ положим $P=\epsilon$, $\boldsymbol{\sigma}=\epsilon$ и $\mu=\epsilon$.

Имеем: $1{:}\,\epsilon\in \boldsymbol{runs}_{\circ}\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$; $2{:}\,\epsilon\in \boldsymbol{traces}_{\phi}(\epsilon)$; $3{:}\,\epsilon\preccurlyeq\epsilon$.

Если LTS $\mathbf{S}$ опасна, её начальное состояние $t_0=\gamma$ и левая часть импликации $4{:}$ ложна, следовательно, импликация истинна.

Если $\mathbf{S}$ ***safe***, то $t_0=[\epsilon]$ и левая часть импликации $4{:}$ истинна.

Тогда $5{:}\,\epsilon\preccurlyeq^{\alpha}\epsilon$; $6{:}\,\epsilon\in\xi(\epsilon)\cap\xi(\epsilon)$; $7{:}\,\epsilon\in \boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})$; поскольку $\boldsymbol{\mu}=\epsilon$, левая часть импликации $8{:}$ ложна и импликация истинна; далее $\omega(\epsilon)=t_0=[\epsilon]$, то есть правая часть импликации $9{:}$ истинна и, значит, импликация истинна.

Шаг индукции. Пусть утверждение верно для ϕ-трассы $\boldsymbol{\mu}$, и маршрут P заканчивается в power-состоянии $\omega(P)=\gamma$ или в power-состоянии $\omega(P)=[\mu]$ или $\omega(P)=[\mu]t$, где $t\in !C_{\gamma\delta}$. Будем доказывать утверждение для ϕ-трассы $\boldsymbol{\mu}`{>}\boldsymbol{\mu}$.

Сначала рассмотрим случай, когда маршрут P заканчивается в γ-состоянии: $\omega(P)=\gamma$.

Если последний переход маршрута $\gamma{\rightarrow}\gamma{\rightarrow}\gamma$, то имеется маршрут $P`=P$ с ϕ-трассой $\boldsymbol{\sigma}`=\boldsymbol{\sigma}$, заканчивающейся γ.
В противном случае, по правилу вывода (γ), в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ имеется переход $\gamma{\rightarrow}\gamma{\rightarrow}\gamma$. Следовательно, имеется маршрут $P`=P{\cdot}\langle\gamma{\rightarrow}\gamma{\rightarrow}\gamma\rangle$ с ϕ-трассой $\boldsymbol{\sigma}`=\boldsymbol{\sigma}{\cdot}\langle\gamma\rangle$.

Если $\boldsymbol{\mu}\preccurlyeq\boldsymbol{\sigma}$, то $\boldsymbol{\mu}`{>}\boldsymbol{\mu}$ влечёт $\boldsymbol{\mu}`\preccurlyeq^{\gamma}\boldsymbol{\sigma}`$. В этом случае условия $1{:},2{:},3{:}$ выполнены, а маршрут $P`$ заканчивается в power-состоянии

$\omega(P)=\gamma$, то есть левая часть импликации 4: ложна, следовательно, импликация истинна.

Теперь рассмотрим случай, когда маршрут P не заканчивается в γ-состоянии: $\omega(P)\neq\gamma$. Тогда дано:
$\mathbf{I}\in\mathfrak{I}(\mathbf{S})$, $\boldsymbol{\mu}\in\boldsymbol{traces}_{\phi}(\mathbf{I})$ нормальна,

1: $P\in\boldsymbol{runs}\circ\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$,
2: $\boldsymbol{\sigma}\in\boldsymbol{traces}_{\phi}(P)$,
3: $\boldsymbol{\mu}\preccurlyeq\boldsymbol{\sigma}$,
4: $\omega(P)=[\mu]$ или $\omega(P)=[\mu]t$, где $t\in!C_{\gamma\delta}$,
5: $\boldsymbol{\mu}\preccurlyeq^{\alpha}\boldsymbol{\sigma}$,
6: $\mu\in\xi(\boldsymbol{\mu})\cap\xi(\boldsymbol{\sigma})$,
7: $\mu\in\boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})$,
8: если $\boldsymbol{\mu}\neq\epsilon$ & $\boldsymbol{\mu}(|\boldsymbol{\mu}|)=o$ & $!y\notin o_r$ & $\mu\cdot\langle!y\rangle\in\boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})$,
где $o\in\Phi(C)$, то $\omega(P)=[\mu]!y$,
9: если $\boldsymbol{\mu}=\epsilon\ \vee\ \boldsymbol{\mu}(|\boldsymbol{\mu}|)\in C$, то $\omega(P)=[\mu]$.

Мы будем доказывать утверждение для всех возможных минимальных, но не пустых, продолжений ϕ-трассы $\boldsymbol{\mu}`>\boldsymbol{\mu}$, которые оставляют ϕ-трассу принадлежащей конформной реализации и нормальной: 1) продолжение базовым символом $z\in C_{\gamma}$, опасным в $\mathbf{S}$ после μ, 2) продолжение базовым символом $z\in C_{\gamma}$, безопасным в $\mathbf{S}$ после μ, 3) продолжение ϕ-символом $o\in\Phi(C)$.

Прежде всего, заметим, что $z\neq\gamma$. Действительно, если бы это было так, то в конформной реализации была бы ϕ-трасса $\boldsymbol{\mu}`=\boldsymbol{\mu}\cdot\langle\gamma\rangle$. Поскольку $\mu\in\xi(\boldsymbol{\mu})$ влечёт $\mu\cdot\langle\gamma\rangle\in\xi(\boldsymbol{\mu}\cdot\langle\gamma\rangle)$, была бы $\beta\gamma\delta$-трасса $\mu\cdot\langle\gamma\rangle$. А это противоречит гипотезе о безопасности, поскольку $\mu\in\boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})$.

Следовательно, в случаях 1) и 2) $z\in C$.

1. Продолжение опасным базовым символом
$z\in C$ и $z\notin\cap\circ\boldsymbol{safesymbols}_{\beta\gamma\delta}([\mu])$.

Обозначим $\boldsymbol{\mu}`=\boldsymbol{\mu}\cdot\langle z\rangle$, $\boldsymbol{\sigma}`=\boldsymbol{\sigma}\cdot\langle z\rangle$ и $P`=P\cdot\langle\omega(P)\text{—}z\rightarrow\gamma\rangle$.

Рассмотрим все возможные варианты:

a. $\omega(P)=[\mu]$ и $z=!y$. Тогда μ гамма-трасса, применимо правило вывода $(\mathbf{T}!\gamma)$, и в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ $\omega(P)\text{—}z\rightarrow\gamma$.

b. $\omega(P)=[\mu]$ и $z=?x$. Тогда применимо правило вывода $(\mathbf{T}?\gamma)$, и в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ $\omega(P)\text{—}z\rightarrow\gamma$.

c. $\omega(P)=[\mu]!y$. В этом случае реакции безопасны после μ, поэтому $z=?x$. Тогда применимо правило вывода $(\mathbf{S}?\gamma)$, и в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ $\omega(P)\text{—}z\rightarrow\gamma$.

d. $\omega(P)=[\mu]\gamma$ и $z=!y$. Тогда применимо правило вывода $(\mathbf{G}!\gamma)$, и в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ $\omega(P)\text{—}z\rightarrow\gamma$.

e. $\omega(P)=[\mu]\gamma$ и $z=?x$. Тогда применимо правило вывода $(\mathbf{G}?\gamma)$, и в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ $\omega(P)\text{—}z\rightarrow\gamma$.

f. $\omega(P)=[\mu]\delta$. В этом случае реакции не продолжают μ и, тем самым, безопасны после μ, поэтому $z=?x$. Тогда применимо правило вывода $(\mathbf{D}?\gamma)$, и в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ $\omega(P)\text{—}z\rightarrow\gamma$.

Итак, во всех вариантах в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ $\omega(P)\text{—}z\rightarrow\gamma$. Поскольку $P`=P\cdot\langle\omega(P)\text{—}z\rightarrow\gamma\rangle$, имеем:

$1: P`\in \boldsymbol{runs}\circ\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$.

Далее $\boldsymbol{\sigma}\in \boldsymbol{traces}_{\phi}(P)$ влечёт $\boldsymbol{\sigma}\cdot\langle z\rangle\in \boldsymbol{traces}_{\phi}(P\cdot\langle\omega(P)\text{—}z\rightarrow\gamma\rangle)$, то есть

$2: \boldsymbol{\sigma}`\in \boldsymbol{traces}_{\phi}(P`)$.

Далее $\boldsymbol{\mu}\preccurlyeq^{\alpha}\boldsymbol{\sigma}$ влечёт $\boldsymbol{\mu}\cdot\langle z\rangle\preccurlyeq^{\alpha}\boldsymbol{\sigma}\cdot\langle z\rangle$, то есть

$3: \boldsymbol{\mu}`\preccurlyeq\boldsymbol{\sigma}`$.

Поскольку $\omega(P`)=\gamma$, больше нам проверять ничего не надо: левая часть импликации $4:$ ложна и, следовательно, импликация истинна.

Для случая 1) утверждение доказано.

2. Продолжение безопасным базовым символом
$z\in C$ и $z\in\cap\circ\boldsymbol{safesymbols}_{\beta\gamma\delta}([\mu])$.

Рассмотрим два случая, в каждом из которых выберем маршрут P:

1. Символ $z=!y$ реакция, ϕ-трасса $\boldsymbol{\mu}$ заканчивается на ϕ-символ o. Тогда реакция $!y$ не принадлежит ***ref***-множеству ϕ-символа o.
$\boldsymbol{\mu}\neq\epsilon$ & $\boldsymbol{\mu}(|\boldsymbol{\mu}|)=o$ & $!y\notin o_r$.

Поскольку z безопасен после μ, по условию 8:, мы можем выбрать такой маршрут P, который заканчивается в power-состоянии $\omega(P)=[\mu]!y$.

2. Если условия п.1 не выполнены, мы выбираем любой маршрут P, удовлетворяющий условиям для ϕ-трассы $\boldsymbol{\mu}$.

Теперь обозначим $\boldsymbol{\mu}`=\boldsymbol{\mu}\cdot\langle z\rangle$, $\boldsymbol{\sigma}`=\boldsymbol{\sigma}\cdot\langle z\rangle$, $\mu`=\mu\cdot\langle z\rangle$ и $P`=P\cdot\langle\omega(P)\text{—}z\rightarrow[\mu\cdot\langle z\rangle]\rangle$.

По предположению индукционного шага, $\mu\in\xi(\boldsymbol{\mu})$, что влечёт $\mu\cdot\langle z\rangle\in\xi(\boldsymbol{\mu}\cdot\langle z\rangle)$.

Поскольку $\boldsymbol{\mu}\cdot\langle z\rangle\in\boldsymbol{traces}_{\phi}(\mathbf{I})$, имеем $\mu\cdot\langle z\rangle\in\cup\circ\xi\circ\boldsymbol{traces}_{\phi}(\mathbf{I})$.

По генеративности ϕ-трасс, $\cup\circ\xi\circ\boldsymbol{traces}_{\phi}(\mathbf{I})=\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{I})$.

Следовательно, $\mu\cdot\langle z\rangle\in\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{I})$.

По конформности, $\mu\cdot\langle z\rangle\in\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{I})$, $\mu\in\boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})$ и безопасность z в $\mathbf{S}$ после μ влекут $\mu\cdot\langle z\rangle\in\boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})$.

Рассмотрим все возможные варианты:

a. $\omega(P)=[\mu]$ и $z=!y$. Тогда применимо правило вывода $(\mathbf{T}!\mathbf{T})$, и в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ $\omega(P)\text{—}z\rightarrow[\mu\cdot\langle z\rangle]$.

b. $\omega(P)=[\mu]$ и $z=?x$. Тогда применимо правило вывода $(\mathbf{T}?\mathbf{T})$, и в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ $\omega(P)\text{—}z\rightarrow[\mu\cdot\langle z\rangle]$.

c. $\omega(P)=[\mu]!y$ и $z=!y$. Тогда применимо правило вывода $(\mathbf{S}!\mathbf{T})$, и в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ $\omega(P)\text{—}z\rightarrow[\mu\cdot\langle z\rangle]$.

d. $\omega(P)=[\mu]!y$ и $z=?x$. Тогда применимо правило вывода $(\mathbf{S}?\mathbf{T})$, и в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ $\omega(P)\text{—}z\rightarrow[\mu\cdot\langle z\rangle]$.

e. $\omega(P)=[\mu]\gamma$. В этом случае реакции опасны после μ, поэтому $z=?x$. Тогда применимо правило вывода $(\mathbf{G}?\mathbf{T})$, и в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ $\omega(P)\text{—}z\rightarrow[\mu\cdot\langle z\rangle]$.

f. $\omega(P)=[\mu]\delta$. В этом случае μ не продолжается реакциями и ϕ-трасса $\boldsymbol{\mu}$, порождающая μ, не должна продолжаться реакциями. Поэтому $z=?x$. Тогда применимо правило вывода $(\mathbf{D}?\mathbf{T})$, и в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ $\omega(P)\text{—}z\rightarrow[\mu\cdot\langle z\rangle]$.

Итак, во всех вариантах в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ $\omega(P)$—z→$[\mu\cdot\langle z\rangle]$. Поскольку $P`=P\cdot\langle\omega(P)$—$z$→$[\mu\cdot\langle z\rangle]\rangle$, имеем:

1: $P`\in \boldsymbol{runs}_{\circ}\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$.

Далее $\boldsymbol{\sigma}\in \boldsymbol{traces}_{\phi}(P)$ влечёт $\boldsymbol{\sigma}\cdot\langle z\rangle\in \boldsymbol{traces}_{\phi}(P`\cdot\langle\omega(P)$—$z$→$[\mu\cdot\langle z\rangle]\rangle)$, то есть

2: $\boldsymbol{\sigma}`\in \boldsymbol{traces}_{\phi}(P`)$.

Далее $\boldsymbol{\mu}\preccurlyeq^{\alpha}\boldsymbol{\sigma}$ влечёт $\boldsymbol{\mu}\cdot\langle z\rangle\preccurlyeq^{\alpha}\boldsymbol{\sigma}\cdot\langle z\rangle$, то есть

3: $\boldsymbol{\mu}`\preccurlyeq\boldsymbol{\sigma}`$ и 5: $\boldsymbol{\mu}`\preccurlyeq^{\alpha}\boldsymbol{\sigma}`$.

Поскольку $\omega(P`)=[\mu\cdot\langle z\rangle]$,

4: $\omega(P`)=[\mu`]$.

Далее $\mu\in\xi(\boldsymbol{\mu})\cap\xi(\boldsymbol{\sigma})$ влечёт $\mu\cdot\langle z\rangle\in\xi(\boldsymbol{\mu}\cdot\langle z\rangle)\cap\xi(\boldsymbol{\sigma}\cdot\langle z\rangle)$, то есть

6: $\mu`\in\xi(\boldsymbol{\mu}`)\cap\xi(\boldsymbol{\sigma}`)$.

Далее $\mu\cdot\langle z\rangle\in \boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})$, то есть

7: $\mu`\in \boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})$.

8: Поскольку $\boldsymbol{\mu}`=\boldsymbol{\mu}\cdot\langle z\rangle$ не заканчивается на ϕ-символ, левая часть импликации ложна и импликация истинна.

Поскольку $\omega(P`)=[\mu\cdot\langle z\rangle]$,

9: $\omega(P`)=[\mu`]$, то есть правая часть импликации истинна и, значит, импликация истинна

Для случая 2) утверждение доказано.

3. Продолжение ϕ-символом $o\in\Phi(C)$.

Поскольку мы рассматриваем только нормальные ϕ-трассы, ϕ-символом может продолжаться только такая ϕ-трасса, которая не заканчивается ϕ-символом. Из определения ϕ-трассы также следует, что разрушение может быть только последним символом ϕ-трассы. Следовательно, ϕ-трасса $\boldsymbol{\mu}$ пуста или заканчивается стимулом или реакцией, а тогда по условию 9:, $\omega(P)=[\mu]$.

Обозначим $\boldsymbol{\mu}`=\boldsymbol{\mu}\cdot\langle o\rangle$, $\boldsymbol{\sigma}`=\boldsymbol{\sigma}\cdot\langle o`\rangle$, $\mu`=\mu\cdot\mathbf{o}$, $P`=P\cdot\langle\omega(P)$—$\tau$→$([\mu\cdot o],t)\rangle$ и $o`=\phi([\mu\cdot o]t))$, а трассу βδ-отказов $\mathbf{o}$ и $t\in !C_{\gamma\delta}$ мы определим ниже.

A. Рассмотрим множество $\mathbf{O}$ трасс βδ-отказов, порождаемых ***ref***-множеством o_r, и таких, которые оставляют βγδ-трассу μ безопасной в $\mathbf{S}$: $\mathbf{O}=\{\mathbf{o}_i\in\beta\delta(o_r)^* \mid \mu\cdot\mathbf{o}_i\in \boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})\}$.
Покажем, что $\exists\mathbf{o}\in\mathbf{O}\ [\mu\cdot\mathbf{o}] = \cap(\{[\mu\cdot\mathbf{o}_i] \mid \mathbf{o}_i\in\mathbf{O}\} \neq \varnothing$.

a) Поскольку $\mu\in \boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})$, а пустая трасса $\epsilon\in\beta\delta(o_r)^*$, имеем $\mu\cdot\epsilon=\mu\in \boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})$, то есть множество $\mathbf{O}$ не пусто.
b) Поскольку $\mu\in\xi(\boldsymbol{\mu})$ для каждого $\mathbf{o}_i\in\beta\delta(o_r)^*$ имеет место $\mu\cdot\mathbf{o}_i\in\xi(\boldsymbol{\mu}\cdot\langle o\rangle)$.
   Поскольку $\boldsymbol{\mu}\cdot\langle o\rangle\in \boldsymbol{traces}_{\phi}(\mathbf{I})$, $\mu\cdot\mathbf{o}_i\in\cup_\circ\xi_\circ \boldsymbol{traces}_{\phi}(\mathbf{I})$.
   По генеративности ϕ-трасс, $\cup_\circ\xi_\circ \boldsymbol{traces}_{\phi}(\mathbf{I})=\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{I})$.
   Значит, $\mu\cdot\mathbf{o}_i\in \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{I})$.
   Отсюда, поскольку множество $\beta\delta(o_r)^*$ замкнуто по конкатенации, а реализация конформна $\mathbf{I}\in\mathfrak{I}(\mathbf{S})$, по Утверждение 72:, множество $\mathbf{O}$ замкнуто по конкатенации.
c) $\mu\in \boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})$.
d) $\mathbf{S}$ *S*-ветвящаяся.

Тем самым, выполнены условия a÷d Утверждение 71:, и, значит,
$\exists\mathbf{o}\in\mathbf{O}\ [\mu\cdot\mathbf{o}] = \cap(\{[\mu\cdot\mathbf{o}_i] \mid \mathbf{o}_i\in\mathbf{O}\} \neq \varnothing$.

Поскольку $\mathbf{o}\in\beta\delta(o_r)^*$, $\mu\cdot\mathbf{o}\in\xi(\boldsymbol{\mu}\cdot\langle o\rangle)$.

B. Покажем, что βγδ-трасса $\mu\cdot\mathbf{o}$ стабильна и не продолжается стимулом $?x$ тогда и только тогда, когда его блокировка $\{?x\}\in\beta\delta(o_r)$ и безопасна после $\mu\cdot\mathbf{o}$, а также не продолжается реакциями тогда и только тогда, когда стационарность $\delta\in\beta\delta(o_r)$ и безопасна после $\mu\cdot\mathbf{o}$.

a. Сначала рассмотрим случай, когда $[\mu\cdot\mathbf{o}]$ содержит только стабильные состояния.

Обозначим $B\Delta([\mu\cdot\mathbf{o}]) = \cap\{\beta\delta(s) \mid s\in[\mu\cdot\mathbf{o}]\}$.
Нам нужно показать, что
$B\Delta([\mu\cdot\mathbf{o}]) = (\cap_\circ \boldsymbol{safesymbols}_{\beta\gamma\delta}([\mu\cdot\mathbf{o}]))\cap\beta\delta(o_r)$.

Сначала докажем, что
$B\Delta([\mu\cdot\mathbf{o}]) \supseteq (\cap_\circ \boldsymbol{safesymbols}_{\beta\gamma\delta}([\mu\cdot\mathbf{o}]))\cap\beta\delta(o_r)$.

Поскольку $[\mu\cdot\mathbf{o}] = \cap(\{[\mu\cdot\mathbf{o}_i] | \mathbf{o}_i \in \mathbf{O}\}$, каждое состояние $s \in [\mu\cdot\mathbf{o}]$ порождает все трассы βδ-отказов из $\mathbf{O}$: $\forall \mathbf{o}_i \in \mathbf{O}$ $\mathbf{o}_i \in \beta\delta(s)^*$.

Рассмотрим βδ-отказ $r \in (\cap\circ \boldsymbol{safesymbols}_{\beta\gamma\delta}([\mu\cdot\mathbf{o}])) \cap \beta\delta(o_r)$.

Поскольку $\mathbf{o} \in \beta\delta(o_r)^*$ и $r \in \beta\delta(o_r)$, $\mathbf{o}\cdot\langle r\rangle \in \beta\delta(o_r)^*$.

Поскольку $\mathbf{o}\cdot\langle r\rangle \in \beta\delta(o_r)^*$ и $\mu \in \xi(\boldsymbol{\mu})$, имеем $\mu\cdot\mathbf{o}\cdot\langle r\rangle \in \xi(\boldsymbol{\mu}\cdot\langle o\rangle)$.

Поскольку $\boldsymbol{\mu}\cdot\langle o\rangle \in \boldsymbol{traces}_{\phi}(\mathbf{I})$, имеем $\mu\cdot\mathbf{o}\cdot\langle r\rangle \in \cup\circ\xi\circ\boldsymbol{traces}_{\phi}(\mathbf{I})$.

По генеративности ϕ-трасс, $\cup\circ\xi\circ\boldsymbol{traces}_{\phi}(\mathbf{I}) = \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{I})$.

Значит, $\mu\cdot\mathbf{o}\cdot\langle r\rangle \in \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{I})$.

Поскольку $\mu\cdot\mathbf{o} \in \boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})$, из конформности следует

либо 1) $\mu\cdot\mathbf{o}\cdot\langle r\rangle \in \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$, если r безопасен в $\mathbf{S}$ после $\mu\cdot\mathbf{o}$,

либо 2) r опасен в $\mathbf{S}$ после $\mu\cdot\mathbf{o}$.

Поскольку $r \in \cap\circ\boldsymbol{safesymbols}_{\beta\gamma\delta}([\mu\cdot\mathbf{o}])$, случай 2) невозможен.

Тогда $\mu\cdot\mathbf{o}\cdot\langle r\rangle \in \boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})$, что влечёт $\mathbf{o}\cdot\langle r\rangle \in \mathbf{O}$.

Далее $[\mu\cdot\mathbf{o}] = \cap(\{[\mu\cdot\mathbf{o}_i] | \mathbf{o}_i \in \mathbf{O}\}$ влечёт $[\mu\cdot\mathbf{o}] \subseteq [\mu\cdot\mathbf{o}\cdot\langle r\rangle]$.

Поэтому $\forall s \in [\mu\cdot\mathbf{o}]$ $r \in \beta\delta(s)$, то есть $r \in B\Delta([\mu\cdot\mathbf{o}])$.

Теперь докажем, что
$B\Delta([\mu\cdot\mathbf{o}]) \subseteq (\cap\circ\boldsymbol{safesymbols}_{\beta\gamma\delta}([\mu\cdot\mathbf{o}])) \cap \beta\delta(o_r)$.

Рассмотрим отказ $r \in B\Delta([\mu\cdot\mathbf{o}])$.

Поскольку $B\Delta([\mu\cdot\mathbf{o}]) = \cap\{\beta\delta(s) | s \in [\mu\cdot\mathbf{o}]\}$, отказ r реализуется в каждом состоянии после $\mu\cdot\mathbf{o}$, то есть в каждом состоянии после $\mu\cdot\mathbf{o}$ нет переходов по символам из r, в том числе и разрушающих.

Поэтому $r \in \cap\circ\boldsymbol{safesymbols}_{\beta\gamma\delta}([\mu\cdot\mathbf{o}])$.

Допустим $r \notin \beta\delta(o_r)$.

Тогда, поскольку $\boldsymbol{\mu}\cdot\langle o\rangle \in \boldsymbol{traces}_{\phi}(\mathbf{I})$, существует такой символ $z \in r$, что $\boldsymbol{\mu}\cdot\langle o, z\rangle \in \boldsymbol{traces}_{\phi}(\mathbf{I})$.

Поскольку $\mathbf{o} \in \beta\delta(o_r)^*$ и $\mu \in \xi(\boldsymbol{\mu})$, имеем $\mu\cdot\mathbf{o}\cdot\langle z\rangle \in \xi(\boldsymbol{\mu}\cdot\langle o, z\rangle)$.

Отсюда $\mu\cdot\mathbf{o}\cdot\langle z\rangle \in \cup\circ\xi\circ\boldsymbol{traces}_{\phi}(\mathbf{I})$.

По генеративности ϕ-трасс, $\cup\circ\xi\circ\boldsymbol{traces}_{\phi}(\mathbf{I}) = \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{I})$.

Значит, $\mu\cdot\mathbf{o}\cdot\langle z\rangle \in \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{I})$.

Поскольку $\mu\cdot\mathbf{o} \in \boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})$, из конформности следует

либо 1) $\mu\cdot o\cdot\langle z\rangle\in \mathit{traces}_{\beta\gamma\delta}(S)$, если z безопасен в S после $\mu\cdot o$,

либо 2) z опасен в S после $\mu\cdot o$, следовательно $r=\beta\delta(z)$ опасен после $\mu\cdot o$.

Случай 1) невозможен, так как в каждом состоянии после $\mu\cdot o$ нет переходов по символам из r.

Случай 2) невозможен, так как $r\in\cap_\circ \mathit{safesymbols}_{\beta\gamma\delta}([\mu\cdot o])$.

Мы пришли к противоречию и, значит, наше допущение было неверно и $r\in\beta\delta(o_r)$.

Тем самым, $r\in(\cap_\circ \mathit{safesymbols}_{\beta\gamma\delta}([\mu\cdot o]))\cap\beta\delta(o_r)$.

Для случая a, когда $[\mu\cdot o]$ содержит только стабильные состояния, утверждение B доказано.

b. Теперь рассмотрим случай, когда $[\mu\cdot o]$ содержит нестабильное состояние.

Поскольку $[\mu\cdot o]=\cap(\{[\mu\cdot o_i]\mid o_i\in O\}$, в каждом состоянии $s\in[\mu\cdot o]$ реализуется каждая трасса отказов $o_i\in O$.

Для $[\mu\cdot o]$, содержащего нестабильное состояние, это возможно только в том случае, когда все эти трассы отказов пустые, то есть $O=\{\epsilon\}$, $o=\epsilon$ и $[\mu\cdot o]=[\mu]$.

Если $r\in\beta\delta(o_r)$, то, поскольку $O=\{\epsilon\}$, $\mu\cdot\langle r\rangle\notin \mathit{safe}_{\beta\gamma\delta}(S)$.

Тем самым, каждый отказ, реализуемые ϕ-символом o, опасен после μ.

Нам осталось показать, что μ стабильная тау-трасса, то есть полная: продолжается каждым стимулом и хотя бы одной реакцией.

Рассмотрим произвольный стимул $?x\in C$.

Если $\{?x\}\in\beta\delta(o_r)$, то $\mu\cdot\langle\{?x\}\rangle\notin \mathit{safe}_{\beta\gamma\delta}(S)$.

Поскольку $\mu\in \mathit{safe}_{\beta\gamma\delta}(S)$, то, значит, $\mu\cdot\langle ?x,\gamma\rangle\in \mathit{traces}_{\beta\gamma\delta}(S)$.

Если $\{?x\}\notin\beta\delta(o_r)$, то, очевидно, $\mu\cdot\langle ?x\rangle\in \mathit{traces}_{\beta\gamma\delta}(I)$.

Поскольку $\mu\in \mathit{safe}_{\beta\gamma\delta}(S)$, из конформности следует

либо 1) $\mu\cdot\langle ?x\rangle\in \mathit{traces}_{\beta\gamma\delta}(S)$, если $?x$ безопасен в S после μ,

либо 2) $\mu\cdot\langle ?x,\gamma\rangle\in \mathit{traces}_{\beta\gamma\delta}(S)$, в противном случае.

Таким образом, βγδ-трасса μ продолжается каждым стимулом.

Если $\delta \in \beta\delta(o_r)$, то $\mu\cdot\langle\delta\rangle \notin \mathit{safe}_{\beta\gamma\delta}(\mathbf{S})$.

Так как $\mu \in \mathit{safe}_{\beta\gamma\delta}(\mathbf{S})$, то $\exists !y \in C\ \mu\cdot\langle !y,\gamma\rangle \in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$.

Если $\delta \notin \beta\delta(o_r)$, то, очевидно, $\exists !y \in C\ \mu\cdot\langle !y\rangle \in \mathit{traces}_{\beta\gamma\delta}(\mathbf{I})$.

Поскольку $\mu \in \mathit{safe}_{\beta\gamma\delta}(\mathbf{S})$, из конформности следует один из двух вариантов:

либо 1) $\mu\cdot\langle !y\rangle \in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$, если реакции безопасны в $\mathbf{S}$ после μ,

либо 2) $\exists !y` \in C\ \mu\cdot\langle !y`,\gamma\rangle \in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$, в противном случае.

Таким образом, $\beta\gamma\delta$-трасса μ продолжается хотя бы одной реакцией.

Для случая b, когда $[\mu\cdot o]$ содержит нестабильное состояние, утверждение B доказано.

Тем самым, утверждение B доказано.

C. Покажем, что, если стимул $?x$ разрушающий в реализации $\mathbf{I}$ после ϕ-трассы $\boldsymbol{\mu}\cdot\langle o\rangle$, то есть в $\mathbf{I}$ есть ϕ-трасса $\boldsymbol{\mu}\cdot\langle o,?x,\gamma\rangle$, он разрушающий в спецификации после $\beta\gamma\delta$-трассы $\mu\cdot o$, то есть в $\mathbf{S}$ есть $\beta\gamma\delta$-трасса $\mu\cdot\langle o,?x,\gamma\rangle$. Также, если реакция $!y$ разрушающая в реализации $\mathbf{I}$ после ϕ-трассы $\boldsymbol{\mu}\cdot\langle o\rangle$, то есть в $\mathbf{I}$ есть ϕ-трасса $\boldsymbol{\mu}\cdot\langle o,!y,\gamma\rangle$, то некоторая (быть может, другая) реакция $!y`$ разрушающая в спецификации после $\beta\gamma\delta$-трассы $\mu\cdot o$, то есть в $\mathbf{S}$ есть $\beta\gamma\delta$-трасса $\mu\cdot\langle o,!y`,\gamma\rangle$.

Если стимул разрушающий в реализации $\mathbf{I}$ после ϕ-трассы $\boldsymbol{\mu}\cdot\langle o\rangle$, то, по генеративности ϕ-трасс, он является разрушающим в $\mathbf{I}$ после порождаемой ϕ-трассой $\boldsymbol{\mu}\cdot\langle o\rangle$ $\beta\gamma\delta$-трассы $\mu\cdot o$. Поскольку $\mu\cdot o \in \mathit{safe}_{\beta\gamma\delta}(\mathbf{S})$, из конформности (гипотезы о безопасности) следует, что стимул разрушающий в спецификации после $\beta\gamma\delta$-трассы $\mu\cdot o$.

Если реакция разрушающая в реализации $\mathbf{I}$ после ϕ-трассы $\boldsymbol{\mu}\cdot\langle o\rangle$, то, по генеративности ϕ-трасс, она является разрушающей в $\mathbf{I}$ после

порождаемой ϕ-трассой $\mu\cdot\langle o\rangle$ βγδ-трассы $\mu\cdot o$. Поскольку $\mu\cdot o \in \mathit{safe}_{\beta\gamma\delta}(S)$, из конформности (гипотезы о безопасности) следует, что найдётся (быть может, другая) реакция, разрушающая в спецификации после βγδ-трассы $\mu\cdot o$.

D. Покажем, что в LTS $\mathcal{T}_{\beta\gamma\delta}(S)$ $[\mu] \text{—}\tau\rightarrow [\mu\cdot o]t$ и $o \preccurlyeq o` = \phi([\mu\cdot o]t)$, где $t \in !C_{\gamma\delta}$.

Поскольку βγδ-трасса $\mu\cdot o$ стабильна, существует power-состояние вида $[\mu\cdot o]t$ и $[\mu]\text{—}\tau\rightarrow[\mu\cdot o]t$. Значение t определяется в зависимости от типа βγδ-трассы $\mu\cdot o$. Нам достаточно доказать следующие три утверждения:

a. если $\mu\cdot o$ гамма-трасса, то
   в LTS $\mathcal{T}_{\beta\gamma\delta}(S)$ $[\mu]\text{—}\tau\rightarrow[\mu\cdot o]\gamma$ и $o \preccurlyeq o` = \phi([\mu\cdot o]\gamma)$;

b. если $\mu\cdot o$ дельта-трасса, то
   в LTS $\mathcal{T}_{\beta\gamma\delta}(S)$ $[\mu]\text{—}\tau\rightarrow[\mu\cdot o]\delta$ и $o \preccurlyeq o` = \phi([\mu\cdot o]\delta)$;

c. иначе, для каждой реакции $!y \notin o_r$
   в LTS $\mathcal{T}_{\beta\gamma\delta}(S)$ $[\mu]\text{—}\tau\rightarrow[\mu\cdot o]!y$ и $o \preccurlyeq o` = \phi([\mu\cdot o]!y)$.

Сначала рассмотрим пп.a÷c порознь и покажем существование требуемых переходов и выполнение условия $!o_g \subseteq !o`_g$ & $!o_r \subseteq !o`_r \cup !o`_g$.

a. $\mu\cdot o$ гамма-трасса.

   По правилу вывода $(G!\gamma)$, $[\mu]\text{—}\tau\rightarrow[\mu\cdot o]\gamma$ и для каждой реакции $[\mu\cdot o]\gamma\text{—}!y\rightarrow\gamma$.
   Следовательно, $!o`_g = !C$,
   что влечёт $!o_g \subseteq !o`_g$ и $!o_r \subseteq !o`_r \cup !o`_g$.

b. $\mu\cdot o$ дельта-трасса.

   По правилу вывода (D), $[\mu]\text{—}\tau\rightarrow[\mu\cdot o]\delta$.

   Допустим существует разрушающая реакция $!y \in !o_g$.
   Тогда, по утверждению C, найдётся реакция $!y`$ разрушающая в спецификации после βγδ-трассы $\mu\cdot o$.

А это противоречит тому, что $\mu\cdot\mathbf{o}$ не гамма-трасса.
Следовательно, $!o_g=\varnothing$, что влечёт $!o_g\subseteq!o`_g$.

Поскольку из $[\mu\cdot\mathbf{o}]\delta$ нет переходов по реакциям, $!o`_r=!C$.
Следовательно, $!o_r\subseteq!o`_r\cup!o`_g$.

c. Если $!y\notin o_r$, то в реализации $\mathbf{I}$ есть ϕ-трасса $\boldsymbol{\mu}\cdot\langle o,!y\rangle$, порождающая $\beta\gamma\delta$-трассу $\mu\cdot\mathbf{o}\cdot\langle!y\rangle$, которая, по генеративности ϕ-трасс, должна быть в $\mathbf{I}$.
Поскольку $\mu\in\boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})$, из конформности следует
либо 1) $\mu\cdot\mathbf{o}\cdot\langle!y\rangle\in\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$, если реакция $!y$ безопасна в $\mathbf{S}$ после μ,
либо 2) $\exists!y`\in r\ \mu\cdot\mathbf{o}\cdot\langle!y`,\gamma\rangle\in\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$, в противном случае.
Случай 2) невозможен, поскольку тогда все реакции были бы опасны в $\mathbf{S}$ после $\mu\cdot\mathbf{o}$, то есть это была бы гамма-трасса.
В случае 1) имеем $\mu\cdot\mathbf{o}\cdot\langle!y\rangle\in\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$.

Поскольку реакция $!y$ безопасна в $\mathbf{S}$ после μ, то $\mu\cdot\mathbf{o}\cdot\langle!y\rangle$ безопасна в $\mathbf{S}$.

Тем самым, по правилу вывода $(\mathbf{S}!\mathbf{T})$, $[\mu]\text{—}\tau\text{→}[\mu\cdot\mathbf{o}]!y$.

Итак, мы показали, что для каждой реакции $!y\notin o_r$ $[\mu]\text{—}\tau\text{→}[\mu\cdot\mathbf{o}]!y$.

Допустим существует разрушающая реакция $!y\in!o_g$.
Тогда, по утверждению C, найдётся реакция $!y`$ разрушающая в спецификации после $\beta\gamma\delta$-трассы $\mu\cdot\mathbf{o}$.
А это противоречит тому, что $\mu\cdot\mathbf{o}$ не гамма-трасса.
Следовательно, $!o_g=\varnothing$, что влечёт $!o_g\subseteq!o`_g$.

По правилу вывода $(\mathbf{S}!\mathbf{T})$, в power-состоянии $[\mu\cdot\mathbf{o}]!y$ определяется переход по реакции $!y$ и не определяются переходы по другим реакциям.
Следовательно, $!o`_r=!C\backslash\{!y\}$.
Поскольку $!y\notin o_r$, имеем $!o_r\subseteq!o`_r\cup!o`_g$.

Теперь рассмотрим общую часть пп.a÷c, мы покажем выполнение условия $?o_g\subseteq?o`_g$ & $?o_r\subseteq?o`_r\cup?o`_g$.

Пусть разрушающий стимул $?x \in ?o_g$.
Тогда, по утверждению C, стимул ?x разрушающий в спецификации после βγδ-трассы $\mu\cdot\mathbf{o}$.
Поэтому, по правилам вывода $(\mathbf{T}?\gamma)$, $(\mathbf{G}?\gamma)$ или $(\mathbf{D}?\gamma)$, $[\mu\cdot o]t \text{—}?x\rightarrow\gamma$.
Отсюда, $?x \in o`_g$.
Следовательно, $?o_g \subseteq ?o`_g$.

Если стимул $?x \in o_r$, то $\{?x\} \in \beta\delta(o_r)$.
По утверждению B,
либо 1) блокировка {?x} опасна после $\mu\cdot\mathbf{o}$,
либо 2) $\mu\cdot\mathbf{o}$ не продолжается стимулом ?x.

1) В LTS **S** есть βγδ-трасса $\mu\cdot\mathbf{o}\cdot\langle ?x,\gamma\rangle$. Поэтому, по правилам вывода $(\mathbf{T}?\gamma)$, $(\mathbf{G}?\gamma)$ или $(\mathbf{D}?\gamma)$, $[\mu\cdot o]t \text{—}?x\rightarrow\gamma$. Следовательно, $?x \in ?o`_g$.
2) В LTS **S** нет βγδ-трассы $\mu\cdot\mathbf{o}\cdot\langle ?x\rangle$. Поэтому, по правилам вывода, $[\mu\cdot o]t \text{—}?x\not\rightarrow$. Следовательно, $?x \in ?o`_r$.

В любом случае $?x \in ?o`_r \cup ?o`_g$.
Следовательно, $?o_r \subseteq ?o`_r \cup ?o`_g$.

Итак, доказано:
$!o_g \subseteq !o`_g$ & $!o_r \subseteq !o`_r \cup !o`_g$ & $?o_g \subseteq ?o`_g$ & $?o_r \subseteq ?o`_r \cup ?o`_g$.
Отсюда, $o_g \subseteq o`_g$ & $o_r \subseteq o`_r \cup o`_g$, что влечёт $o \preccurlyeq o`$.

Из существования перехода $\omega(P) \text{—}\tau\rightarrow [\mu\cdot o]t$ следует:
$1: P` \in \boldsymbol{runs}\circ\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$.
Поскольку $\boldsymbol{\sigma}` = \boldsymbol{\sigma}\cdot\langle o`\rangle$, $o` = \phi([\mu\cdot o]t)$ и $\omega(P`) = [\mu\cdot o]t$, имеем:
$2: \boldsymbol{\sigma}` \in \boldsymbol{traces}_{\phi}(P`)$.
Далее $\boldsymbol{\mu} \preccurlyeq^{\alpha} \boldsymbol{\sigma}$ и $o \preccurlyeq o`$ влечёт $\boldsymbol{\mu}\cdot\langle o\rangle \preccurlyeq^{\alpha} \boldsymbol{\sigma}\cdot\langle o`\rangle$, то есть
$3: \boldsymbol{\mu}` \preccurlyeq \boldsymbol{\sigma}`$ и $5: \boldsymbol{\mu}` \preccurlyeq^{\alpha} \boldsymbol{\sigma}`$.
Далее $\omega(P`) = [\mu\cdot o]t$, следовательно
$4: \omega(P`) = [\mu`]t)$.

Далее $\mu\cdot o\in\xi(\boldsymbol{\mu}\cdot\langle o\rangle)$. По утверждениям B и C, трасса отказов o реализуется в power-состоянии $[\mu\cdot o]t$. Поскольку $\mu\in\xi(\boldsymbol{\sigma})$, $\mu\cdot o\in\xi(\boldsymbol{\sigma}\cdot\langle o`\rangle)$.

6: $\mu`\in\xi(\boldsymbol{\mu`})\cap\xi(\boldsymbol{\sigma`})$.

По A, $\mu\cdot o\in\boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})$, то есть

7: $\mu`\in\boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})$.

По D пп.c, если реакции безопасны, то для каждой реакции, которой может продолжаться ϕ-трасса $\boldsymbol{\mu}\cdot\langle o\rangle$, найдётся маршрут $P`$ с $\omega(P`)=[\mu\cdot o]!y$.

8: $!y\notin o_r$ & $\mu`\cdot\langle!y\rangle\in\boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})\ \Rightarrow\ \omega(P)=[\mu`]!y$.

9: Поскольку ϕ-трасса $\boldsymbol{\mu}\cdot\langle o\rangle$ заканчивается на ϕ-символ, левая часть импликации ложна, и импликация истинна.

Для случая 3) утверждение доказано.

Утверждение доказано.

76. Доказательство Утверждение 76:

Нам нужно доказать: $\forall\mathbf{S}\in\boldsymbol{SBLTS}_{\beta\gamma\delta}\ \cup_\circ\boldsymbol{traces}_{\phi^\circ}\mathfrak{I}(\mathbf{S})\ \preccurlyeq\ \boldsymbol{traces}_{\phi^\circ}\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$.

Пусть заданы алфавит $C\subseteq Z$, *S*-ветвящаяся LTS-спецификация $\mathbf{S}$ и $\mathbf{T}=\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$.

Сначала рассмотрим случай $t_0=\gamma$. В этом случае в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ есть ϕ-трасса $\langle\gamma\rangle$, которая, очевидно, γ-мажорирует любую ϕ-трассу.

Теперь рассмотрим случай $t_0=[\epsilon]$.

Из определения мажорируемости следует, что $\boldsymbol{\mu}\cdot\langle o\rangle\cdot\boldsymbol{\mu`}\preccurlyeq\boldsymbol{\sigma}\cdot\langle o\rangle\cdot\boldsymbol{\sigma`}$, где $o\in\Phi(C)$, влечёт $\boldsymbol{\mu}\cdot\langle o,o\rangle\cdot\boldsymbol{\mu`}\preccurlyeq\boldsymbol{\sigma}\cdot\langle o,o\rangle\cdot\boldsymbol{\sigma`}$. Поскольку ϕ-символ стабильного состояния может повторяться в ϕ-трассе LTS любое число раз, из наличия в LTS ϕ-трассы $\boldsymbol{\sigma}\cdot\langle o\rangle$ следует наличие в ней ϕ-трассы $\boldsymbol{\sigma}\cdot\langle o,o\rangle$. Поэтому нам достаточно доказать конечную мажорантность пребразования только для нормальных ϕ-трасс. А это непосредственно следует из Утверждение 75:.

77. **Доказательство Утверждение 77:**

Пусть $C \subseteq Z$, $S \in \mathit{LTS}_{\beta\gamma\delta}(C)$ локально-конечно-ветвящаяся LTS. Нам нужно доказать, что каждая бесконечная ϕ-трасса $\mu \in \mathit{traces}_{\phi}^{\infty} \circ \mathfrak{I}(S)$ мажорируется некоторой ϕ-трассой $\sigma \in \mathit{traces}_{\phi}^{\omega}(S)$. Обозначим $\Sigma = \mathit{traces}_{\phi}^{\omega}(S)$.

В рамках данного доказательства нам будет удобно ϕ-трассу $\sigma \in \Sigma$, мажорирующую какой-нибудь префикс ϕ-трассы μ, называть *удобной ϕ-трассой*. Поскольку конечные ϕ-трассы конформных реализаций мажорируются конечными ϕ-трассами LTS S, все конечные префиксы бесконечной ϕ-трассы μ мажорируются удобными ϕ-трассами.

Рассмотрим два случая.

1. Одна из удобных ϕ-трасс σ продолжается разрушением.

Тогда $\sigma \cdot \langle \gamma \rangle \in \mathit{traces}_{\phi} \circ \mathcal{T}_{\beta\gamma\delta}(S)$ и $\mu \preccurlyeq^{\gamma} \sigma \cdot \langle \gamma \rangle$.

2. Ни одна из удобных ϕ-трасс не продолжается разрушением.

Поскольку префикс мажорирующей ϕ-трассы мажорирует некоторый префикс мажорируемой ϕ-трассы, префикс удобной ϕ-трассы удобен.
А тогда удобная ϕ-трасса не заканчивается разрушением (иначе был бы её префикс, продолжающийся разрушением).
В этом случае каждая удобная ϕ-трасса α-мажорирует соответствующий префикс ϕ-трассы.

По определению мажорирования ϕ-трасс, бесконечная ϕ-трасса не может мажорировать конечную ϕ-трассу.
Поэтому нам достаточно показать, что существует бесконечная удобная ϕ-трасса. Она и будет α-мажорировать ϕ-трассу μ.

Рассмотрим два подслучая.

2.1. ϕ-трасса μ имеет конечную подтрассу базовых символов.

Такая ϕ-трасса заканчивается бесконечным повторением одного и того же ϕ-символа o и представима в виде $\mu = \mu` \cdot \{o\}^{\infty}$.
Для конечного префикса $\mu` \cdot \langle o \rangle$ должна существовать α-мажоранта $\mu` \cdot \langle o \rangle \preccurlyeq^{\alpha} \sigma \cdot \langle o` \rangle \in \Sigma$.

А в этом случае существует бесконечная ϕ-трасса $\boldsymbol{\sigma}\cdot\{o`\}^{\infty}\in\boldsymbol{\Sigma}$, которая является искомой α-мажорантой: $\boldsymbol{\mu}=\boldsymbol{\mu}`\cdot\{o\}^{\infty}\preccurlyeq^{\alpha}\boldsymbol{\sigma}\cdot\{o`\}^{\infty}$.

2.2. ϕ-трасса $\boldsymbol{\mu}$ имеет бесконечную подтрассу базовых символов.

Маршрут преобразованной спецификации, имеющий (в числе прочих) удобную ϕ-трассу, будем называть удобным маршрутом. Множество удобных маршрутов обозначим:

$P=\{p\in \boldsymbol{runs}^{\omega}_{\circ}\mathcal{T}_{\beta\gamma\delta}(S)\mid\exists\boldsymbol{\sigma}\in\boldsymbol{traces}_{\phi}^{\omega}(p)\ \ \exists\boldsymbol{\mu}_1\le\boldsymbol{\mu}\ \ \boldsymbol{\mu}_1\preccurlyeq^{\alpha}\boldsymbol{\sigma}\}$.

Покажем, что множество P дерево.

Префикс удобного маршрута, очевидно, имеет (в числе прочих) префикс его удобной ϕ-трассы, который, как показано выше, удобен.
Следовательно, удобен префикс удобного маршрута.

Покажем, что дерево P конечно-ветвящееся.

Удобная ϕ-трасса $\boldsymbol{\sigma}$ не содержит разрушения и её подтрасса базовых символов является префиксом подтрассы базовых символов ϕ-трассы $\boldsymbol{\mu}$: $\boldsymbol{\sigma}_{\downarrow}C\le\boldsymbol{\mu}_{\downarrow}C$.
Поэтому удобная ϕ-трасса $\boldsymbol{\sigma}$ может удобно продолжаться только одним стимулом или реакцией $\boldsymbol{\mu}_{\downarrow}C(|\boldsymbol{\sigma}_{\downarrow}C|+1)$.
Отсюда следует, что удобный маршрут может удобно продолжаться либо τ-переходом (не меняющим его ϕ-трассы), либо переходами по одному стимулу или реакции z.
В локально-конечно-ветвящееся LTS число таких переходов конечно.
Следовательно, удобный маршрут удобно продолжается конечным числом переходов.
А это и означает, что дерево удобных маршрутов конечно-ветвящееся.

Покажем, что дерево P бесконечно.

Допустим противное: P конечно.
Поскольку бесконечный удобный маршрут имеет бесконечное число префиксов и они удобны, P не может содержать бесконечных маршрутов.
Следовательно, P состоит из конечного числа конечных маршрутов.
Тогда длина подтрассы базовых символов удобной ϕ-трассы ограничена сверху некоторым числом n.

Поскольку ϕ-трасса μ имеет бесконечную подтрассу базовых символов, существуют префиксы μ с подтрассой базовых символов длины больше n.
Поскольку при α-мажорировании подтрассы базовых символов совпадают, эти префиксы не имеют α-мажорант.
Мы пришли к противоречию и, следовательно, дерево P бесконечно.

Итак, дерево P конечно-ветвящееся и бесконечное.
Тогда, по теореме Кёнига [KÖNIG], существует бесконечный удобный маршрут p.[75]

Нам достаточно показать, что p имеет бесконечную удобную ϕ-трассу.

Покажем, что маршрут p имеет бесконечную трассу.

Если p имеет конечную трассу, он заканчивается бесконечным постфиксом τ-маршрутов.
Любая ϕ-трасса маршрута p заканчивается или продолжается разрушением (моделирующим дивергенцию).
Следовательно, удобная ϕ-трасса этого маршрута также продолжается или заканчивается разрушением, что неверно.

Итак, маршрут p имеет бесконечную трассу и, следовательно, все его ϕ-трассы бесконечны.
Следовательно, удобная ϕ-трасса маршрута p бесконечна.

78. **Доказательство Утверждение 78:**

По Утверждение 76:, преобразование $\tau_{\beta\gamma\delta}$ конечно-мажорантно.
Докажем его мажорантность.
По Утверждение 74:, LTS $\tau_{\beta\gamma\delta}(s)$ локально-конечно-ветвящаяся.
Тогда, по Утверждение 77:, LTS $\tau_{\beta\gamma\delta}(s)$ мажорантна.

79. **Доказательство Утверждение 79:**

У нас выполнены все восемь условий монотонности для соответствия $ioco_{\beta\gamma\delta}$ и преобразования $\tau_{\beta\gamma\delta}$ на классе $SBLTS_{\beta\gamma\delta}$:

---

[75] В отличие от трасс, βγδ-трасс и ϕ-трасс существование такого маршрута в дереве означает его существование в LTS.

1. Эквивалентность соответствия мажорированию βγδ-трасс (Утверждение 54:).
2. Генеративность ϕ-трасс (Утверждение 56:).
3. Аддитивность ϕ-трасс относительно композиции (Утверждение 60:).
4. Генеративность мажорирования ϕ-трасс (Утверждение 63:).
5. Композиционность мажорирования ϕ-трасс на классе всех LTS $\boldsymbol{LTS}_{\beta\gamma\delta}$ (Утверждение 65:).
6. Конформность преобразования $\mathcal{T}_{\beta\gamma\delta}$ на классе всех LTS $\boldsymbol{LTS}_{\beta\gamma\delta}$ (Утверждение 69:).
7. Мажорантность преобразования $\mathcal{T}_{\beta\gamma\delta}$ на классе *S*-ветвящихся LTS $\boldsymbol{SBLTS}_{\beta\gamma\delta}$ (Утверждение 78:).
8. Рефлексивность мажорирования ϕ-трасс на классе всех LTS $\boldsymbol{LTS}_{\beta\gamma\delta}$ (Утверждение 62:).

Поэтому, по Утверждение 52:, соответствие $\boldsymbol{ioco}_{\beta\gamma\delta}$ $\mathcal{T}_{\beta\gamma\delta}$-левомонотонно и $\mathcal{T}_{\beta\gamma\delta}$-монотонно на классе $\boldsymbol{SBLTS}_{\beta\gamma\delta}$.

80. Доказательство Утверждение 80:

Поскольку, по Утверждение 79:, преобразование $\mathcal{T}_{\beta\gamma\delta}$ инвариантно, $\mathcal{T}_{\gamma\delta}(\mathbf{S}) \sim_{\boldsymbol{ioco}_{\beta\gamma\delta}} \mathbf{S}$.
Следовательно, $\boldsymbol{safe}_{\beta\gamma\delta}(\mathbf{S})=\boldsymbol{safe}_{\beta\gamma\delta}\circ\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$.
Поэтому доказываемое утверждение следует из аналогичного Утверждение 67:, в котором показывается, что βγδ-трасса μ безопасна в **S**.

81. Доказательство Утверждение 81:

По Утверждение 74:, LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ локально-конечно-ветвящаяся и τ-ограниченная. Для конечного алфавита локально-конечно-ветвимость совпадает с конечно-ветвимостью.

Поскольку конечная LTS в конечном алфавите, очевидно, τ-ограничена, вторая часть утверждения также верна.

82. Доказательство Утверждение 82:

Поскольку итераторы стимулов и реакций не меняются при преобразовании, нам нужно построить оракулы 1) $\mathbf{Safe}_{\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})}()$ и 2) $\mathbf{Gamma1}_{\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})}(\mathbf{v},\mathbf{z})$ и итераторы 3) $\mathbf{Extend1}_{\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})}(\mathbf{v},\mathbf{z})$ и 4) $\mathbf{Tau1}_{\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})}(\mathbf{v})$, где power-состояние $\mathbf{v}\in\boldsymbol{der}(\mathcal{T}_{\beta\gamma\delta}(\mathbf{S}))$ и $\mathbf{v}=[\mu]$ или $\mathbf{v}=[\mu]\mathbf{t}$, где $\mathbf{t}\in !C_{\gamma\delta}$, βγδ-трасса μ безопасна и множество состояний $[\mu]$ конечно.

1) Очевидно, $\mathbf{Safe}_{\tau_{\beta\gamma\delta}(s)}() = \mathbf{Safe}_s()$.

2) Оракул $\mathbf{Gamma1}_{\tau_{\beta\gamma\delta}(s)}(v,z)$. Этот оракул мы построим для любого конечного множества состояний $U \subseteq \mathit{der}(S)$, а не только для $U=[\mu]$, где βγδ-трасса μ безопасна.

   Стимул $z=?x$ разрушающий в v, если он разрушающий хотя бы в одном состоянии $s \in U$. Это проверяется за конечное время с помощью оракула $\mathbf{Gamma1}_s(s,z)$, где s пробегает конечное множество состояний U. Если z реакция, то она разрушающая в v, если хотя бы одна реакция разрушающая хотя бы в одном состоянии $s \in U$. Это проверяется за конечное время с помощью оракула $\mathbf{Gamma1}_s(s,!y)$, где $!y$ пробегает конечное множество реакций $!C$, а s пробегает конечное множество состояний U.

3) Итератор $\mathbf{Extend1}_{\tau_{\beta\gamma\delta}(s)}(v,z)$. Этот итератор мы построим для любого конечного множества состояний $U \subseteq \mathit{der}(S)$, а не только для $U=[\mu]$, где βγδ-трасса μ безопасна.

   Этот итератор перечисляет переходы из v по безопасным стимулам и реакциям. Стимул z безопасен в v, если он неразрушающий в v, что проверяется за конечное время с помощью построенного оракула $\mathbf{Gamma1}_{\tau_{\beta\gamma\delta}(s)}(v,z)$. Реакция z безопасна в v, если каждая реакция неразрушающая в v, что проверяется с помощью построенного оракула $\mathbf{Gamma1}_{\tau_{\beta\gamma\delta}(s)}(v,!y)$, где $!y$ пробегает конечное множество реакций $!C$.

   Для безопасного стимула или реакции z с помощью итератора $\mathbf{Extend1}_s(s,z)$, где s пробегает конечное множество состояний U, за конечное время перечисляем конечное множество $U`$ постсостояний переходов по $z$ из всех состояний $s \in U$. Если оно пусто, итератор $\mathbf{Extend1}_{\tau_{\beta\gamma\delta}(s)}(v,z)$ сообщает, что переходов нет. В противном случае с помощью итератора $\mathbf{Tau1}_s$ выполняем τ-замыкание множества $U`$. Результатом будет конечное множество состояний, которое является единственным значением, возвращаемым итератором $\mathbf{Extend1}_{\tau_{\beta\gamma\delta}(s)}(v,z)$.

4) Итератор **Tau1**$_{\tau_{\beta\gamma\delta(s)}}$(v).

Если v=[μ]t, то τ-переходов нет.
Пусть v=[μ].

Тогда итератор перечисляет power-постсостояния τ-переходов из power-состояния [μ]. Заметим, что все эти power-постсостояния будут стабильны: они имеют вид [μ·o]t.

Нам нужно проверить все трассы βδ-отказов o, при которых βγδ-трасса μ·o стабильна. Очевидно, достаточно ограничиться трассами без повторных βδ-отказов (каждый βδ-отказ встречается в трассе не более одного раза). При конечном алфавите каждый βδ-отказ (как множество) конечен, число βδ-отказов также конечно и, следовательно, число трасс без повторных βδ-отказов конечно, и мы можем их все перечислять, используя итераторы стимулов и реакций, и отбирая стабильные трассы. Одновременно будем строить множество состояний [μ·o].

Проверка стабильности βγδ-трассы μ·o для βδ-отказа o вместе с построением множества состояний [μ·o] выполняется следующим образом.

Пустая βγδ-трасса ϵ безопасна (так как безопасна βγδ-трасса μ), и [μ·ϵ]=[μ]. Пустая βγδ-трасса стабильна, если она полна, то есть продолжается каждым стимулом и хотя бы одной реакцией. Для проверки этого мы можем использовать построенный итератор **Extend1**$_{\tau_{\beta\gamma\delta(s)}}$([μ],z), где z пробегает конечный алфавит стимулов и реакций.

Пусть для некоторой трассы βδ-отказов o βγδ-трасса μ·o безопасна. Тогда её продолжение βδ-отказом r, то есть βγδ-трасса μ·o·⟨r⟩ безопасна и, следовательно, стабильна, если a) в каждом состоянии s∈[μ·o] каждый символ z∈r неразрушающий, и b) хотя бы в одном таком состоянии нет перехода ни по одному такому z.

Для проверки a) мы можем использовать построенный оракул **Gamma1**$_{\tau_{\beta\gamma\delta(s)}}$([μ·o],z), где z пробегает конечное множество

r.[76] Далее для проверки b) мы можем использовать построенный итератор $\mathbf{Extend1}_{\tau_{\beta\gamma\delta}(s)}([\mu\cdot o],z)$, где z пробегает конечное множество r. Множество $[\mu\cdot o\cdot\langle r\rangle]$ строится как множество тех состояний $s\in[\mu\cdot o]$, для которых итератор при любом $z\in r$ сообщает, что переходов нет.

Если $\beta\gamma\delta$-трасса $\mu\cdot o$ стабильна и множество $[\mu\cdot o]$ не пусто, итератор $\mathbf{Tau1}_{\tau_{\beta\gamma\delta}(s)}(v)$ перечисляет соответствующие постсостояния τ-переходов из $v=[\mu]$:

- перечисляется $[\mu\cdot o]\delta$, если в каждом состоянии $s\in[\mu\cdot o]$ нет переходов по реакциям, что проверяется итератором $\mathbf{Extend1}_{\tau_{\beta\gamma\delta}(s)}([\mu\cdot o],z)$, где z пробегает конечное множество реакций;
- перечисляется $[\mu\cdot o]\gamma$, если в некотором состоянии $s\in[\mu\cdot o]$ некоторая реакция разрушающая, что проверяется итератором $\mathbf{Gamma1}_{\tau_{\beta\gamma\delta}(s)}([\mu\cdot o],z)$, где z пробегает конечное множество реакций;
- перечисляется $[\mu\cdot o]!y$ для каждой реакции $!y$ такой, что, во-первых, в каждом состоянии $s\in[\mu\cdot o]$ каждая реакция неразрушающая, что проверяется итератором $\mathbf{Gamma1}_{\tau_{\beta\gamma\delta}(s)}([\mu\cdot o],z)$, где z пробегает конечное множество реакций, и, во-вторых, в некотором состоянии $s\in[\mu\cdot o]$ есть переход $s\text{—}!y\rightarrow$, что проверяется итератором $\mathbf{Extend1}_{\tau_{\beta\gamma\delta}(s)}([\mu\cdot o],!y)$.

83. Доказательство Утверждение 83:

Итераторы стимулов и реакций одинаковы для **L1**- и **L2**-LTS.

Если $\mathbf{Safe}_s()$ возвращает ***false***, значит LTS опасна. Тогда считаем, что начальное состояние $s_0=\gamma$.

В γ-состоянии определим $\mathbf{Gamma2}_s(\gamma)=\mathit{true}$, $\mathbf{Extend2}_s(\gamma,z)=\epsilon$ и $\mathbf{Tau2}_s(\gamma)=\epsilon$.

---

[76] Если r блокировка, это одноэлементное множество, а если r стационарность, то это конечное множество всех реакций.

По Утверждение 80:, в преобразованной LTS любое состояние $s\neq\gamma$ *S*-достижимо в этой LTS. Поэтому в нём определены оракул $\mathbf{Gamma1_s}(s,z)$ и итераторы $\mathbf{Extend1_s}(s,z)$ и $\mathbf{Tau1_s}(s)$.

Проверка того, является ли символ (стимул или реакция) z разрушающим, выполняется с помощью оракула $\mathbf{Gamma1_s}(s,z)$.

Для LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ все переходы по стимулам и реакциям, опасным в состоянии, разрушающие в этом состоянии и ведут в стандартное состояние γ. Поэтому итератор $\mathbf{Extend2_s}(s,z)$ для неразрушающего символа z совпадает с итератором $\mathbf{Extend1_s}(s,z)$, а для разрушающего символа z перечисляется единственное состояние γ.

Оракул $\mathbf{Gamma2_s}(s)$=***false***.

Итератор $\mathbf{Tau2_s}(s)=\mathbf{Tau1_s}(s)$.

84. Доказательство Утверждение 84:

Нам надо показать, что композиция локально-конечно-ветвящихся LTS с конечным алфавитом реакций локально-конечно-ветвящаяся: в каждом достижимом композиционном состоянии ab определено конечное число переходов по каждому символу.

Очевидно, что состояния-операнды a и b достижимы в соответствующих LTS-операндах. Поскольку LTS-операнды локально-конечно-ветвящиеся, в каждом из состояний a и b определено конечное число переходов по каждому символу.

Каждый переход по стимулу или реакции из состояния ab является асинхронным переходом, то есть соответствует одному переходу в одном из операндов. Следовательно, число переходов по каждому стимулу или реакции из состояния ab конечно.

Каждый τ-переход из состояния ab либо асинхронный τ-переход в одном из операндов, либо синхронный переход по стимулу $?z$ в одном из операндов и по противоположной реакции $!z$ в другом операнде. Поскольку число асинхронных τ-переходов в каждом операнде конечно, суммарное число асинхронных τ-переходов также конечно. Поскольку конечны число реакций и число переходов по каждой реакции $!z$ и

каждому противоположному стимулу $?z$, число синхронных τ-переходов из состояния ab также конечно.

85. **Доказательство Утверждение 85:**

Композиция локально-конечно-ветвящихся LTS с конечными алфавитами реакций локально-конечно-ветвящаяся по Утверждение 84:.

Сначала покажем, что композиция локально-конечно-ветвящихся, τ-ограниченных и слабо-регулярных LTS с конечными алфавитами реакций τ-ограниченная.

Нам надо показать, что любое достижимое композиционное состояние ab τ-ограничено, то есть дерево τ-маршрутов, начинающихся в состоянии ab, либо a) конечно, либо b) перечислимо-ветвящееся и содержит маршрут с самопересечением или заканчивающийся в состоянии с γ-переходом. Перечислимо-ветвимость дерева τ-маршрутов следует из локально-конечно-ветвимости композиции (конечно число переходов по каждому символу, в том числе, по τ). Пусть это дерево содержит бесконечный τ-маршрут R. Нам надо показать, что в дереве найдётся маршрут с самопересечением или заканчивающийся в состоянии с γ-переходом.

Маршрут R является композицией двух маршрутов R^a и R^b в LTS-операндах, начинающихся в состояниях a и b, соответственно. Возможны два варианта: 1) один из маршрутов R^a или R^b имеет разрушающую трассу, 2) оба маршрута R^a и R^b имеют бесконечные неразрушающие трассы.

1) Пусть, для определённости, маршрут R^a имеет разрушающую трассу. Тогда, поскольку LTS-операнды τ-ограниченные, существует конечный маршрут R^a_1, начинающийся в состоянии a, имеющий в качестве трассы префикс трассы маршрута R^a, и заканчивающийся в состоянии a_1, в котором начинается τ-маршрут R^a_2, который либо a) заканчивается в состоянии a_2 с γ-переходом, либо b) имеет самопересечение. Очевидно, маршрут R^b имеет конечный префикс R^b_1, заканчивающийся в состоянии b_1, компонуемый с маршрутом R^a_1 и дающий композиционный τ-маршрут R_1. Поскольку τ-переходы LTS-операндов выполняются асинхронно, τ-маршруту R^a_2 соответствует композиционный τ-маршрут R_2, начинающийся в состоянии a_1b_1 и заканчивающийся в состоянии a_2b_1 (второй компонент

композиционного состояния не меняется на этом композиционном маршруте).

a) В композиции LTS имеем τ-маршрут $R_1 \cdot R_2$, начинающийся в состоянии ab и заканчивающийся в состоянии $a_2 b_1$, в котором определён асинхронный γ-переход.
b) В композиции LTS имеем τ-маршрут с самопересечением $R_1 \cdot R_2$.

2) Поскольку LTS-операнды слабо-регулярные, найдутся регулярные маршруты $R^a{}_1$ и $R^b{}_1$, начинающиеся также в состояниях a и b, имеющие те же трассы, что маршруты R^a и R^b, соответственно. Очевидно, эти маршруты $R^a{}_1$ и $R^b{}_1$ компонуются и в композиции дают регулярный τ-маршрут R_1 (проходящий через конечное множество композиционных состояний как подмножество декартового произведения двух конечных множеств). Тем самым, τ-маршрут R_1, начинающийся в состоянии ab, имеет самопересечение.

Теперь покажем, что в композиции слабо-регулярных LTS каждое достижимое композиционное состояние слабо-регулярно.

Пусть в композиционном состоянии ab начинается бесконечная неразрушающая трасса σ. Выберем любой маршрут R, начинающийся в состоянии ab и имеющий трассу σ. Маршрут R является композицией двух маршрутов R^a и R^b в LTS-операндах, начинающихся в состояниях a и b, соответственно, и имеющих трассы σ^a и σ^b, соответственно.

Если бы одна из этих трасс, например, σ^a была разрушающей, то у неё был бы конечный префикс $\sigma^a{}_1 \le \sigma^a$, который имел бы продолжение разрушением. Тогда существовал бы маршрут $R^a{}_1$, начинающийся в состоянии a и имеющий трассу $\sigma^a{}_1 \cdot \langle\gamma\rangle$. Соответственно, у маршрута R^b был бы префикс, компонуемый с маршрутом $R^a{}_1$, и дающий в композиции маршрут R_1, начинающийся в состоянии ab и имеющий трассу $\sigma_1 \cdot \langle\gamma\rangle$, где $\sigma_1 \le \sigma$. Но тогда трасса σ была бы разрушающей, что неверно. Следовательно, обе трассы σ^a и σ^b неразрушающие.

Если трасса σ^a или σ^b конечна, то имеется конечный маршрут с этой трассой и регулярный, проскольку регулярны все конечные маршруты. Если же трасса σ^a или σ^b бесконечна, то, поскольку LTS-операнды слабо-регулярные, также имеется регулярный маршрут с этой трассой. Такие маршруты компонуются и дают композиционный маршрут, начинающийся в состоянии ab, имеющий трассу σ и регулярный

(проходящий через конечное число композиционных состояний как подмножество декартового произведения двух конечных множеств). Тем самым, состояние ab слабо-регулярно.

86. Доказательство Утверждение 86:

1. **Необходимость**. Пусть LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ слабо-регулярна. Рассмотрим в $\mathbf{S}$ произвольную бесконечную безопасную βγδ-трассу $\mu\cdot\lambda$, где λ трасса. Нам надо показать, что в $\mathbf{S}$ существует регулярная безопасная βγδ-трасса $\mu\cdot\sigma$, где $\sigma{\downarrow}C=\lambda$.

   Обозначим через P_μ power-маршрут, заканчивающийся в power-состоянии $[\mu]$ и имеющий βγδ-трассу μ. По Утверждение 67:, такой power-маршрут должен существовать.

   Очевидно, в LTS $\mathbf{S}$ в некоторых состояниях из $[\mu]$ начинается трасса λ. Поскольку $\mu\cdot\lambda$ безопасна, а λ не содержит βδ-отказов, у λ нет ответвлений по разрушению. Для любого префикса $\lambda`\cdot\langle z\rangle<\lambda$ в каждом состоянии из $\mu\cdot\lambda`$ символ z неразрушающий. Следовательно, в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ в power-состоянии $v=[\mu]$ или $v=[\mu]t$, где $t\in !C_{\gamma\delta}$ также начинается трасса λ.

   Поскольку LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ слабо-регулярна, существует регулярный power-маршрут P_λ, начинающийся в power-состоянии v и имеющий трассу λ. Power-маршрут P_λ проходит последовательность power-состояний $v=v_1, v_2, v_3, \ldots$, где $v_i=[\kappa_i]$ или $v_i=[\kappa_i]t_i$, где $t_i\in !C_{\gamma\delta}$ ($\kappa_1=\mu$).

   Построим для power-маршрута P_λ βγδ-трассу σ следующим образом.

   Сначала имеем пустой префикс βγδ-трассы σ.
   Каждый раз, когда префикс маршрута P_λ продолжается в P_λ power-переходом по стимулу или реакции $v_i \text{—} z_{i+1} \rightarrow v_{i+1}$ имеет место $[\kappa_{i+1}]=[\kappa_i\cdot\langle z_{i+1}\rangle]$; в этом случае уже построенный префикс βγδ-трассы σ продолжаем символом z_{i+1}. Каждый раз, когда префикс P_λ продолжается в P_λ τ-power-переходом $v_i \text{—}\tau\rightarrow v_{i+1}$ имеет место $[\kappa_{i+1}]=[\kappa_i\cdot\mathbf{o}_{i+1}]$, где $\mathbf{o}_{i+1}$ трасса βδ-отказов; в этом случае уже построенный префикс βγδ-трассы σ продолжаем трассой βδ-отказов $\mathbf{o}_{i+1}$.

Очевидно, что $\sigma_{\downarrow C}=\lambda$.

Теперь рассмотрим все маршруты LTS **S**, которые начинаются в состояниях $s\in[\mu]$ и имеют $\beta\gamma\delta$-трассу σ.

Power-переходу $v_i\text{—}z_{i+1}\rightarrow v_{i+1}$ соответствуют все маршруты в LTS **S** вида $s_i=z_{i+1}\Rightarrow s_{i+1}$, где $s_i\in[\kappa_i]$, и множество концов s_{i+1} всех таких маршрутов равно $\{s_{i+1}\,|\,s_i\in[\kappa_i]\ \&\ s_i=z_{i+1}\Rightarrow s_{i+1}\} = [\kappa_i\cdot\langle z_{i+1}\rangle] = [\kappa_{i+1}]$. Тем самым, power-переходу $v_i\text{—}z_{i+1}\rightarrow v_{i+1}$ соответствуют в LTS **S** все маршруты, начинающиеся в состояниях $s_i\in[\kappa_i]$, которые имеют $\beta\gamma\delta$-трассу $\langle z_{i+1}\rangle$.

Также τ-power-переходу $v_i\text{—}\tau\rightarrow v_{i+1}$ соответствуют в LTS **S** все состояния $s_i\in[\kappa_i]$, в которых есть трасса отказов o_{i+1}, где $\kappa_{i+1}=\kappa_i\cdot o_{i+1}$, и множество таких состояний равно $\{s_i\,|\,s_i\in[\kappa_i]\ \&\ o_{i+1}\in \mathit{traces}_{\beta\gamma\delta}(s_i)\} = [\kappa_i\cdot o_{i+1}] = [\kappa_{i+1}]$. Тем самым, τ-power-переходу $v_i\text{—}\tau\rightarrow v_{i+1}$ соответствуют в LTS **S** все (пустые) маршруты, начинающиеся (и заканчивающиеся) в состояниях $s_i\in[\kappa_i]$, которые имеют $\beta\gamma\delta$-трассу o_{i+1}.

Отсюда следует, что любой маршрут в LTS **S**, начинающийся в состоянии $s\in[\mu]$ и имеющий $\beta\gamma\delta$-трассу σ, проходит только через те состояния, которые принадлежат множествам состояний $[\kappa_i]$.

Поскольку power-маршрут P_λ регулярен, число множеств состояний $[\kappa_i]$ конечно. Для *S*-ветвящейся LTS **S** каждое из множеств состояний $[\kappa_i]$ также конечно. Следовательно, конечно их объединение. Тем самым, каждый маршрут в LTS **S**, начинающийся в состоянии $s\in[\mu]$ и имеющий $\beta\gamma\delta$-трассу σ, регулярен.

Очевидно, каждый маршрут R в LTS **S**, начинающийся в начальном состоянии и имеющий $\beta\gamma\delta$-трассу $\mu\cdot\sigma$, является конкатенацией $R=R_\mu\cdot R_\lambda$ маршрута R_μ, начинающегося в начальном состоянии, имеющего $\beta\gamma\delta$-трассу μ и заканчивающегося в состоянии из $[\mu]$, и маршрута R_λ, начинающегося в конце маршрута R_μ и имеющего $\beta\gamma\delta$-трассу σ. Тем самым, каждый такой маршрут R регулярен, то есть $\beta\gamma\delta$-трасса $\mu\cdot\sigma$ регулярна.

Необходимость доказана.

2. **Достаточность**. Пусть LTS $\mathbf{S}$ сильно-регулярна. Поскольку в γ-состоянии определён единственный переход γ-петля, это состояние регулярно. Рассмотрим произвольное power-состояние $v=[\mu]$ или $v=[\mu]t$, где $t\in !C_{\gamma\delta}$, и произвольную бесконечную безопасную трассу λ, начинающуюся в power-состоянии v. Нам надо показать, что в $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ существует регулярный маршрут, начинающийся в v и имеющий трассу λ.

   Без ограничения общности мы можем считать, что $v=[\mu]$.

   Действительно, если $v=[\mu]t$, где $t\in !C_{\gamma\delta}$, то power-состояние v стабильно и в нём должен быть переход по символу $\lambda(1)$, ведущий в power-состояние $v`=[\mu\cdot\langle\lambda(1)\rangle]$, а в этом power-состоянии должна начинаться трасса $\lambda`=\lambda[2..\infty]$. Из существования регулярного маршрута $P`$, начинающегося в $v`$ и имеющего трассу $\lambda`$, очевидно, следует существование регулярного маршрута $P=\langle(v,\lambda(1),v`)\rangle\cdot P`$, начинающегося в v и имеющего трассу λ.

   Покажем, что $\beta\gamma\delta$-трасса $\mu\cdot\lambda$ безопасна в LTS $\mathbf{S}$.

   Действительно, $\beta\gamma\delta$-трасса μ безопасна в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ и, в силу $\boldsymbol{ioco}_{\beta\gamma\delta}$-эквивалентности LTS $\mathbf{S}$ и $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$, $\beta\gamma\delta$-трасса μ безопасна в LTS $\mathbf{S}$. Допустим, $\beta\gamma\delta$-трасса $\mu\cdot\lambda$ опасна в LTS $\mathbf{S}$. Тогда, поскольку λ не содержит $\beta\delta$-отказов, либо a) существует такой префикс $\lambda`<\lambda$, что в $\mathbf{S}$ есть $\beta\gamma\delta$-трасса $\mu\cdot\lambda`\cdot\langle\gamma\rangle$, либо b) для некоторой реакции $!y$ существует такой префикс $\lambda`\cdot\langle !y\rangle<\lambda$, что в $\mathbf{S}$ есть $\beta\gamma\delta$-трасса $\mu\cdot\lambda`\cdot\langle !y`,\gamma\rangle$. В случае a) в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ есть трасса $\lambda`\cdot\langle\gamma\rangle$, начинающаяся в power-состоянии $[\mu]$. По Утверждение 73: (1), LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ γ-однородна. Поэтому в случае b) в ней есть трасса $\lambda`\cdot\langle !y,\gamma\rangle$, начинающаяся в power-состоянии $[\mu]$. В обоих случаях трасса λ не была бы безопасной в LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$, что неверно. Мы пришли к противоречию и, следовательно, $\beta\gamma\delta$-трасса $\mu\cdot\lambda$ безопасна в LTS $\mathbf{S}$.

Тогда, в силу сильно-регулярности LTS $\mathbf{S}$, существует регулярная безопасная βγδ-трасса $\mu\cdot\sigma$, где $\sigma_{\downarrow}C=\lambda$. Построим power-маршрут P_λ, начинающийся в power-состоянии $[\mu]$ и имеющий βγδ-трассу σ. Он будет проходить через состояния $v=v_1,v_2,v_3,\ldots$, где $v_i=[\kappa_i]$ или $v_i=[\kappa_i]t_i$, где $t_i\in !C_{\gamma\delta}$ ($\kappa_1=\mu$).

Каждый раз, когда префикс (начиная с пустого префикса) βγδ-трассы σ продолжается стимулом или реакцией z_{i+1}, мы продолжаем power-маршрут переходом $v_i\text{—}z_{i+1}\rightarrow v_{i+1}$, где $v_{i+1} = [\kappa_{i+1}] = [\kappa_i\cdot\langle z_{i+1}\rangle] = \{s_{i+1}\,|\,s_i\in[\kappa_i]\ \&\ s_i=z_{i+1}\Rightarrow s_{i+1}\}$. Каждый раз, когда префикс βγδ-трассы σ продолжается трассой отказов o_{i+1} до следующего стимула или реакции z_{i+2}, мы продолжаем power-маршрут τ-переходом $v_i\text{—}\tau\rightarrow v_{i+1}$, где $v_{i+1} = [\kappa_{i+1}]t_{i+1}$, $[\kappa_{i+1}] = [\kappa_i\cdot o_{i+1}] = \{s_i\,|\,s_i\in[\kappa_i]\ \&\ o_{i+1}\in \mathit{traces}_{\beta\gamma\delta}(s_i)\}$. У нас может быть несколько power-состояний с одним и тем же множеством состояний $[\kappa_{i+1}]$, различающихся $t_{i+1}\in !C_{\gamma\delta}$. Если $z_{i+2}=!y$ реакция, выбираем $t_{i+1}=!y$, в противном случае нам годится любое power-состояние вида $[\kappa_{i+1}]t_{i+1}$.

Легко видеть, что маршрут P_λ будет иметь βγδ-трассу σ.

Теперь рассмотрим маршруты LTS S, соответствующие power-маршруту P_λ:

Каждому power-переходу $v_i\text{—}z_{i+1}\rightarrow v_{i+1}$ соответствуют все маршруты, начинающиеся в состояниях из $[\kappa_i]$ и имеющие трассу $\langle z_{i+1}\rangle$, причём для power-состояния v_{i+1} множество состояний $[\kappa_{i+1}]$ – это в точности концы таких маршрутов. Также каждому power-τ-переходу $v_i\text{—}\tau\rightarrow v_{i+1}$ соответствуют все (пустые) маршруты, начинающиеся в состояниях из $[\kappa_i]$ и имеющие βγδ-трассу o_{i+1}, причём для power-состояния v_{i+1} множество состояний $[\kappa_{i+1}]$ – это в точности концы таких маршрутов.

Тем самым, каждое множество состояний $[\kappa_i]$ состоит только из тех состояний, через которые проходят маршруты $R_1,R_2,R_3,\ldots$ LTS $\mathbf{S}$, начинающиеся в состояниях из $[\mu]$ и имеющие βγδ-трассу σ. Каждый такой маршрут R_i является постфиксом маршрута, начинающегося в начальном состоянии и имеющего βγδ-трассу $\mu\cdot\sigma$. В силу сильно-

регулярности LTS $\mathbf{S}$, каждый такой маршрут R_i проходит через конечное множество состояний $\boldsymbol{Im}_{\circ}\boldsymbol{pre}(R_i)$.

Нам осталось показать, что объединение множеств $\boldsymbol{Im}_{\circ}\boldsymbol{pre}(R_i)$ также конечно.

Поскольку для *S*-ветвящейся LTS $\mathbf{S}$ множество $[\mu]$ конечно, нам достаточно показать, что конечно объединение множеств $\boldsymbol{Im}_{\circ}\boldsymbol{pre}(R_i)$ для всех маршрутов R_i, начинающихся в одном состоянии $s\in[\mu]$.

Рассмотрим дерево R таких маршрутов R_i и их префиксов. Поскольку LTS $\mathbf{S}$ *S*-ветвящаяся, состояние s *S*-достижимо и все маршруты имеют одну и ту же безопасную трассу, это дерево конечно-ветвящееся. Так как каждый маршрут R_i регулярен, у него есть конечный префикс $R`_i<R_i$, проходящий через все состояний маршрута $R_i$: $\boldsymbol{Im}_{\circ}\boldsymbol{pre}(R`_i)\cup\{\omega(R`_i)\}=\boldsymbol{Im}_{\circ}\boldsymbol{pre}(R_i)$. Поддерево $R`\subset R$ таких префиксов (и их префиксов), очевидно, также конечно-ветвящееся, но содержит только конечные маршруты. Следовательно, по теореме Кёнига [KÖNIG], это поддерево конечно. Множество состояний, через которые проходят конечные маршруты из конечного множества, конечно. Тем самым конечно объединение множеств $\boldsymbol{Im}_{\circ}\boldsymbol{pre}(R_i)$ для всех маршрутов R_i, начинающихся в одном состоянии $s\in[\mu]$, конечно, что и требовалось доказать.

Достаточность доказана.

87. **Доказательство Утверждение 87:**

Эквивалентность по мажорированию βγδ-трасс.

По Утверждение 79:, $\mathcal{T}_{\beta\gamma\delta}$-преобразование инвариантно.
Следовательно, $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S}) \sim_{\boldsymbol{ioco}_{\beta\gamma\delta}} \mathbf{S}$.
По Утверждение 54:, эквивалентность по мажорированию βγδ-трасс совпадает с $\boldsymbol{ioco}_{\beta\gamma\delta}$-эквивалентностью.
В силу транзитивности и симметричности $\boldsymbol{ioco}_{\beta\gamma\delta}$-эквивалентности, нам достаточно показать $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S}) \sim_{\boldsymbol{ioco}_{\beta\gamma\delta}} \mathbf{S}$.

По Утверждение 54:, $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S}) \sim_{\boldsymbol{ioco}_{\beta\gamma\delta}} \mathbf{S}$ влечёт взаимное мажорирование βγδ-трасс этих LTS:

$$\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})\preccurlyeq\boldsymbol{traces}_{\beta\gamma\delta}\circ\mathcal{T}_{\beta\gamma\delta}(\mathbf{S}) \;\&\; \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})\succcurlyeq\boldsymbol{traces}_{\beta\gamma\delta}\circ\mathcal{T}_{\beta\gamma\delta}(\mathbf{S}).$$

Очевидно, что в этом случае также верно:

$$\mathit{traces}_{\beta\gamma\delta}(\mathbf{S}) \preccurlyeq \mathit{traces}_{\beta\gamma\delta}(\mathbf{S}) \cup \mathit{traces}_{\beta\gamma\delta} \circ \mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$$

$$\& \ \mathit{traces}_{\beta\gamma\delta}(\mathbf{S}) \succcurlyeq \mathit{traces}_{\beta\gamma\delta}(\mathbf{S}) \cup \mathit{traces}_{\beta\gamma\delta} \circ \mathcal{T}_{\beta\gamma\delta}(\mathbf{S}).$$

Поэтому нам достаточно показать, что множество βγδ-трасс двойной LTS равно объединению множеств βγδ-трассы исходной и $\mathcal{T}_{\beta\gamma\delta}$-преобразованной LTS:

$$\mathit{traces}_{\beta\gamma\delta} \circ \mathcal{W}_{\beta\gamma\delta}(\mathbf{S}) = \mathit{traces}_{\beta\gamma\delta}(\mathbf{S}) \cup \mathit{traces}_{\beta\gamma\delta} \circ \mathcal{T}_{\beta\gamma\delta}(\mathbf{S}).$$

В двойной LTS маршруты делятся на три категории:

a) маршрут $\mathcal{T}_{\beta\gamma\delta}$-преобразованной LTS, то есть не содержащий τ-переходов, ведущих из power-состояний $\mathcal{T}_{\beta\gamma\delta}$-преобразованной LTS в состояния исходной LTS;
b) маршрут исходной LTS, точнее маршрут, начинающийся с τ-перехода $[\epsilon] \text{—}\tau\rightarrow s_0$ из начального (нестабильного) power-состояния $[\epsilon]$ двойной LTS в начальное состояние исходной LTS;
c) смешанный маршрут P, образованный конечным маршрутом Q LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$, заканчивающимся в (нестабильном) power-состоянии $[\mu]$, далее τ-переходом $[\mu] \text{—}\tau\rightarrow s$ в состояние исходной LTS $s \in [\mu]$, и, наконец, маршрутом R исходной LTS, начинающимся в состоянии s.

Очевидно, маршруты первых двух категорий вносят только такие βγδ-трассы, которые уже есть либо в исходной, либо в $\mathcal{T}_{\beta\gamma\delta}$-преобразованной LTS.
Смешанный маршрут имеет вид $P=Q\cdot\langle([\mu],\tau,s)\rangle\cdot R$, где $Q \in \mathit{runs} \circ \mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$, $[\mu]=\omega(Q)$, $\mu \in \mathit{safe}_{\beta\gamma\delta}(\mathbf{S})$, $s \in [\mu]$, $R \in \mathit{runs}(s)$.
Любая βγδ-трасса $\sigma \in \mathit{traces}_{\beta\gamma\delta}(P)$ может быть представлена в виде: $\sigma=\mu\cdot\lambda$, где $\mu \in \mathit{traces}_{\beta\gamma\delta}(Q)$ и $\lambda \in \mathit{traces}_{\beta\gamma\delta}(R)$.
Очевидно, $\mu \in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$ и $\lambda \in \mathit{traces}_{\beta\gamma\delta}(s)$.
Поскольку, $s \in (\mathbf{S}\ \mathit{after}\ \mu)$, $\lambda \in \cup\circ \mathit{traces}_{\beta\gamma\delta}(\mathbf{S}\ \mathit{after}\ \mu)$.
Но тогда $\sigma=\mu\cdot\lambda \in \mathit{traces}_{\beta\gamma\delta}(\mathbf{S})$.

Таким образом, множество βγδ-трасс двойной LTS равно объединению множеств βγδ-трассы исходной и $\mathcal{T}_{\beta\gamma\delta}$-преобразованной LTS, что и требовалось доказать.

Эквивалентность по мажорированию ϕ-трасс.

По построению, $\mathit{traces}_{\phi}{}^{\omega} \circ \mathcal{W}_{\beta\gamma\delta}(\mathbf{S}) \supseteq \mathit{traces}_{\phi}{}^{\omega} \circ \mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$, что влечёт $\mathit{traces}_{\phi}{}^{\omega} \circ \mathcal{W}_{\beta\gamma\delta}(\mathbf{S}) \succcurlyeq \mathit{traces}_{\phi}{}^{\omega} \circ \mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$.

По доказанному, $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S}) \sim_{\boldsymbol{ioco}_{\beta\gamma\delta}} \mathbf{S}$, следовательно, $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S}) \in \mathfrak{I}(\mathbf{S})$.

По мажорантности преобразования $\mathcal{T}_{\beta\gamma\delta}$ (Утверждение 78:), $\boldsymbol{traces}_{\phi}^{\omega} \circ \mathcal{W}_{\beta\gamma\delta}(\mathbf{S}) \preccurlyeq \boldsymbol{traces}_{\phi}^{\omega} \circ \mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$.

Тем самым, $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ и $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ эквивалентны по мажорированию ϕ-трасс.

88. Доказательство Утверждение 88:

По Утверждение 52:, нам достаточно показать конформность и мажорантность преобразования $\mathcal{W}_{\beta\gamma\delta}$.

Конформность.

По Утверждение 87:, $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ и $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ эквивалентны по мажорированию βγδ-трасс.
Отсюда, по Утверждение 54:, $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S}) \; \boldsymbol{ioco}_{\beta\gamma\delta} \; \mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$.
По конформности преобразования $\mathcal{T}_{\beta\gamma\delta}$ (Утверждение 69:), $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S}) \; \boldsymbol{ioco}_{\beta\gamma\delta} \; \mathbf{S}$.
По транзитивности предпорядка $\boldsymbol{ioco}_{\beta\gamma\delta}$, имеем $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S}) \; \boldsymbol{ioco}_{\beta\gamma\delta} \; \mathbf{S}$, что и требовалось доказать.

Мажорантность.

По Утверждение 87:, $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ и $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ эквивалентны по мажорированию ϕ-трасс.
По Утверждение 78:, преобразование $\mathcal{T}_{\beta\gamma\delta}$ мажорантно.
Следовательно, по транзитивности мажорирования ϕ-трасс (Утверждение 61:), преобразование $\mathcal{W}_{\beta\gamma\delta}$ также мажорантно.

89. Доказательство Утверждение 89:

Пусть LTS $\mathbf{S}$ *S*-τ-ограниченная и *S*-регулярная. Нам надо показать, что этими свойствами обладает также LTS $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$.

S-τ-ограниченность.

Поскольку для каждого нестабильного power-состояния $[\mu]$ множество $[\mu]$ состояний LTS $\mathbf{S}$ конечно, в каждом power-состоянии $[\mu]$ мы добавляем только конечное число τ-переходов $[\mu] \text{—}\tau\text{→} s$ в состояния исходной LTS $s \in [\mu]$.

Отсюда, а также из *S*-τ-ограниченности LTS $\mathbf{S}$ и $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ непосредственно следует *S*-τ-ограниченность LTS $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$.

<u>*S*-регулярность</u>.

По определению *S*-регулярности и двойной LTS, нам достаточно доказать следующие утверждения:

1. Состояние LTS $\mathbf{S}$, *S*-достижимое в $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$, *S*-достижимо в $\mathbf{S}$.
2. Состояние LTS $\mathbf{S}$, *S*-регулярное в $\mathbf{S}$, *S*-регулярно в $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$.
3. Power-состояние LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$, отличное от γ-состояния, *S*-достижимо и *S*-регулярно в $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$.

1. Пусть состояние $s \in V_{\mathbf{S}}$ *S*-достижимо в $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$.

   Тогда в $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ имеется маршрут P, заканчивающийся в состоянии s и имеющий безопасную в $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ βγδ-трассу μ.

   Такой маршрут можно представить в виде $P=P_1\cdot\langle[\mu_1]\text{—}\tau\rightarrow s_1\rangle\cdot P_2$, где $\mu_1 \in \textit{safe}_{\beta\gamma\delta}(\mathbf{S})$, $s_1 \in [\mu_1]$ и $\omega(P_2)=s$.

   Тогда $\mu=\mu`_1\cdot\mu_2$, где $\mu`_1 \in \textit{traces}_{\beta\gamma\delta}(P_1)$ и $\mu_2 \in \textit{traces}_{\beta\gamma\delta}(P_2)$.

   Очевидно, $s_1 \in (\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})\ \textit{after}\ \mu`_1)$ и $\mu_2 \in \textit{traces}_{\beta\gamma\delta}(s_1)$.

   Поскольку также $\mu \in \textit{safe}_{\beta\gamma\delta^\circ}\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ и $\mu=\mu`_1\cdot\mu_2$, имеем $\mu_2 \in \textit{safe}_{\beta\gamma\delta}(s_1)$ в $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$.

   Поскольку из состояний $\mathbf{S}$ нет переходов в power-состояния $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$, $\mu_2 \in \textit{safe}_{\beta\gamma\delta}(s_1)$ в $\mathbf{S}$.

   Поскольку $\mu_1 \in \textit{safe}_{\beta\gamma\delta}(\mathbf{S})$, $s_1 \in [\mu_1]$ и $\mu_2 \in \textit{safe}_{\beta\gamma\delta}(s_1)$ в $\mathbf{S}$, имеем $\mu_1\cdot\mu_2 \in \textit{safe}_{\beta\gamma\delta}(\mathbf{S})$, то есть состояние s достижимо в $\mathbf{S}$ по безопасной βγδ-трассе $\mu_1\cdot\mu_2$.

2. Пусть состояние $s \in V_{\mathbf{S}}$ *S*-достижимо в $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$.

   Рассмотрим в $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ произвольную бесконечную трассу λ, начинающуюся в s и проходящую только через *S*-достижимые состояния $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$.

   Поскольку в $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ нет переходов из состояний LTS $\mathbf{S}$ в power-состояния LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$, λ проходит только через состояния LTS $\mathbf{S}$.

   Тогда, по доказанному утверждению 1, все эти состояния, включая состояние s, *S*-достижимы в $\mathbf{S}$.

Поскольку LTS $\mathbf{S}$ *S*-регулярна, в ней и, следовательно, в $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$, существует регулярный маршрут, начинающийся в s и имеющий трассу λ.
А это означает, что состояние s *S*-регулярно в $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$, что и требовалось доказать.

3. Пусть $v \neq \gamma$ power-состояние LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$.

По Утверждение 80:, power-состояние v *S*-достижимо в $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$.
По Утверждение 87:, LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ и $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ эквивалентны по мажорированию βγδ-трасс.
По Утверждение 54:, $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S}) \sim_{\mathit{ioco}_{\beta\gamma\delta}} \mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$.
По Утверждение 21:, LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$ и $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ имеют одно и тоже множество безопасных βγδ-трасс.
Поэтому power-состояние v *S*-достижимо в $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$.

Рассмотрим в LTS $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ бесконечную трассу λ, начинающуюся в power-состоянии v и проходящую только через *S*-достижимые состояния.

Возможны два случая.

а) Power-состояние v нестабильно.
Тогда оно имеет вид $v=[\mu]$, где $\mu \in \mathit{safe}_{\beta\gamma\delta}(\mathbf{S})$.
Выберем произвольный маршрут W, начинающийся в power-состоянии $[\mu]$ и имеющий трассу λ.
Поскольку λ проходит только через *S*-достижимые состояния, она не проходит через состояние γ.
Следовательно, маршруту W в LTS $\mathbf{S}$ соответствует хотя бы один маршрут с той же трассой λ, начинающийся в некотором состоянии $s \in [\mu]$.
Мы показали, что существует состояние $s \in [\mu]$ и в LTS $\mathbf{S}$ существует маршрут с трассой λ, начинающийся в состоянии s.
Теперь рассмотрим произвольный такой маршрут S.
Поскольку $\mu \in \mathit{safe}_{\beta\gamma\delta}(\mathbf{S})$, состояние $s \in [\mu]$ достижимо в LTS $\mathbf{S}$ по безопасной βγδ-трассе μ, следовательно, оно *S*-достижимо в $\mathbf{S}$.
Маршруту S соответствует в $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ маршрут $S` = \langle [\mu] \text{—} \tau \rightarrow s \rangle \cdot S$, очевидно, имеющий ту же трассу λ.

Поскольку трасса λ проходит в $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ только через *S*-достижимые состояния, маршрут $S`$ проходит в $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ только через *S*-достижимые состояния.

Поскольку в $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ нет переходов из состояний LTS $\mathbf{S}$ в power-состояния LTS $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})$, маршрут S проходит в $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ только через состояния, принадлежащие $\mathbf{S}$, и *S*-достижимые в $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$.

По доказанному утверждению 2, все такие состояния *S*-достижимы в $\mathbf{S}$.

Поскольку мы выбирали в LTS $\mathbf{S}$ произвольный маршрут S с трассой λ, начинающийся в s, трасса λ проходит только через *S*-достижимые состояния LTS $\mathbf{S}$.

А тогда, поскольку LTS $\mathbf{S}$ *S*-регулярна, в ней среди всех её маршрутов S, начинающих в s и имеющих трассу λ, найдётся регулярный маршрут S_1.

Но тогда и в LTS $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ маршрут $S_1`=\langle[\mu]—\tau\rightarrow s\rangle\cdot S_1$ начинается в power-состоянии $[\mu]$, имеет ту же трассу λ и также регулярный.

Что и требовалось доказать.

b) Power-состояние v стабильно.

Тогда оно имеет вид $v=[\mu]t$, где $\mu\in \mathit{safe}_{\beta\gamma\delta}(\mathbf{S})$.

Трассу λ можно представить в виде $\lambda=\langle z\rangle\cdot\lambda_z$, где z стимул или реакция.

Такая трасса λ_z в LTS $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ также проходит только через *S*-достижимые состояния и начинается в нестабильном power-состоянии $[\mu\cdot\langle z\rangle]$.

По доказанному утверждению a) для нестабильного power-состояния, в LTS $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ найдётся маршрут S_z, начинающийся в power-состоянии $[\mu\cdot\langle z\rangle]$, имеющий трассу λ_z и регулярный.

Но тогда в LTS $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$ найдётся маршрут $S=\langle[\mu]—z\rightarrow[\mu\cdot\langle z\rangle]\rangle\cdot S_z$, начинающийся в power-состоянии $[\mu]$, имеющий трассу λ и регулярный.

Что и требовалось доказать.

90. Доказательство Утверждение 90:

Итераторы стимулов и реакций одинаковы для **L2**-LTS $\mathbf{S}$ и $\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$.

В состоянии s исходной LTS $\mathbf{S}$ сохраним те алгоритмы, которые были в нём определены:

$\mathbf{Extend2}_{\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})}(s,z) = \mathbf{Extend2}_{\mathbf{S}}(s,z)$,

$\mathbf{Tau2}_{w_{\beta\gamma\delta(s)}(s)} = \mathbf{Tau2}_s(s)$,
$\mathbf{Gamma2}_{w_{\beta\gamma\delta(s)}(s)} = \mathbf{Gamma2}_s(s)$.

В состоянии v исходной LTS $\tau_{\beta\gamma\delta}(\mathbf{S})$ сохраним алгоритмы:
$\mathbf{Extend2}_{w_{\beta\gamma\delta(s)}(v,z)} = \mathbf{Extend2}_{\tau_{\beta\gamma\delta(s)}(v,z)}$,
$\mathbf{Gamma2}_{w_{\beta\gamma\delta(s)}(s)} = \mathbf{Gamma2}_{\tau_{\beta\gamma\delta(s)}(s)}$.

В состоянии v исходной LTS $\tau_{\beta\gamma\delta}(\mathbf{S})$ переопределим итератор **Tau2**:

Если состояние v стабильно или $v=\gamma$, определим
$\mathbf{Tau2}_{w_{\beta\gamma\delta(s)}(s)} = \mathbf{Tau2}_{\tau_{\beta\gamma\delta(s)}(s)}$.

Если состояние v нестабильно и $v\neq\gamma$, то $v=[\mu]$ и мы определим итератор $\mathbf{Tau2}_{w_{\beta\gamma\delta(s)}(s)}$ как перечисляющий все τ-переходы, перечисляемые итератором $\mathbf{Tau2}_{\tau_{\beta\gamma\delta(s)}(s)}$, и дополнительно τ-переходы вида $[\mu]\text{—}\tau\rightarrow s$ для каждого состояния $s\in\mu$. Поскольку итератор $\mathbf{Tau2}_{\tau_{\beta\gamma\delta(s)}(s)}$ перечисляет конечное число τ-переходов и множество $[\mu]$ также конечно, итератор $\mathbf{Tau2}_{w_{\beta\gamma\delta(s)}(s)}$ перечисляет конечное множество τ-переходов.

91. **Доказательство Утверждение 91:**

Пусть **A** и **B** две всюду-заданные **L2**-LTS с оракулами стимулов в алфавитах A и B, соответственно.

Сначала построим итераторы стимулов и реакций, а также оракул стимулов для LTS $\mathbf{A}\!\upharpoonright\!\downharpoonright\!\mathbf{B}$.

Итератор стимулов $\mathbf{X}_{A\upharpoonright\!\downharpoonright B}$. Итерируем поочерёдно стимулы обеих LTS с помощью итераторов $\mathbf{X}_A$ и $\mathbf{X}_B$, проверяя каждый стимул на синхронность. Стимул $?x\in?A$ синхронный, если $!x\in!B$, что за конечное время проверяется с помощью итератора конечного множества реакций $\mathbf{Y}_B$. Аналогично, для $?x\in?B$.

Итератор реакций $\mathbf{Y}_{A\upharpoonright\!\downharpoonright B}$. Итерируем поочерёдно реакции обеих LTS с помощью итераторов $\mathbf{Y}_A$ и $\mathbf{Y}_B$, проверяя каждую реакцию на синхронность. Реакция $!y\in!A$ синхронная, если $?y\in?B$, что за конечное

время проверяется с помощью оракула стимулов $\mathbf{isX}_B$. Аналогично, для $!y \in !B$.

Оракул стимулов $\mathbf{isX}_{A||B}$. Имеем $?(A||B)=?A\setminus !B \cup ?B\setminus !A$. Для проверки $?x \in ?A$ используем оракул $\mathbf{isX}_A$, для проверки $?x \in ?B$ используем оракул $\mathbf{isX}_B$, далее $?x \in !B$ эквивалентно $!x \in !B$, что проверяется с помощью итератора конечного множества реакций $\mathbf{Y}_B$, соответственно $?x \in !A$ эквивалентно $!x \in !A$, что проверяется с помощью итератора конечного множества реакций $\mathbf{Y}_A$.

Поскольку LTS-операнды всюду-заданы, для композиционного состояния ab в каждом из состояний-операндов a и b, определены оракул **Gamma2** и итераторы **Extend2** и **Tau2**. Определим эти алгоритмы в композиционном состоянии ab.

Композиционный γ-переход всегда является асинхронным переходом в одном из LTS-операндов.
Поэтому $\mathbf{Gamma2}_{A||B}(ab) = \mathbf{Gamma2}_A(a) \vee \mathbf{Gamma2}_B(b)$.

Построим итератор $\mathbf{Extend2}_{A||B}(ab,z)$ для $z \in A||B$. Композиционный переход по стимулу или реакции z всегда является асинхронным переходом в одном из LTS-операндов.

Проверка $z \in A$ выполняется для стимула $z=?x$ с помощью оракула стимулов $\mathbf{isX}_A$, а для реакции $z=!y$ с помощью итератора конечного множества реакций реакций $\mathbf{Y}_A$.

Если $z \in A$, то для каждого постсостояния $a`$, возвращаемого итератором $\mathbf{Extend2}_A(a,z)$, нужно вернуть композиционное постсостояние $a`b$. В противном случае $z \in B$ и для каждого постсостояния $b`$, возвращаемого итератором $\mathbf{Extend2}_B(b,z)$, нужно вернуть композиционное постсостояние $ab`$.

Построим итератор $\mathbf{Tau2}_{A||B}(ab)$. Композиционный τ-переход является либо асинхронным τ-переходом в одном из LTS-операнде, либо синхронным переходом. Для каждого постсостояния $a`$, возвращаемого итератором $\mathbf{Tau2}_A(a)$, нужно вернуть композиционное постсостояние $a`b$.

Аналогично, для каждого постсостояния $b`$, возвращаемого итератором $\mathbf{Tau2}_B(b,z)$, нужно вернуть композиционное постсостояние $ab`$.

Кроме этого, для каждого синхронного стимула или реакции $z \in A \cap \underline{B}$ возвращаем композиционное постсостояние $a`b`$.

Итератор $\mathbf{Y}_A$ даёт нам конечное множество реакций $!y \in A$. Для каждой такой реакции проверяем её синхронность, то есть $?y \in ?B$, с помощью оракула стимулов $\mathbf{isX}_B$. Для каждой синхронной реакции $!y \in A \cap \underline{B}$ возвращаем композиционное постсостояние $a`b`$, если итератор $\mathbf{Extend2}_A(a,!y)$ перечисляет $a`$, а итератор $\mathbf{Extend2}_B(b,?y)$ перечисляет $b`$.

Аналогично, для конечного множества реакций $!y \in B$.

92. **Доказательство Утверждение 92:**

Если LTS $\mathbf{S}$ опасна, то в ней есть трасса $\langle\gamma\rangle$. Тогда $\mathcal{C}(\mathbf{S})$ состоит только из начального состояния s_0, состояния $\gamma`$ и двух переходов $s_0 \text{—}\tau\rightarrow \gamma`$ и $\gamma` \text{—}\gamma\rightarrow \gamma`$. Такая LTS, очевидно, обладает всеми требуемыми свойствами.

Далее будем считать, что LTS $\mathbf{S}$ безопасна.

Конечная ветвимость.

По построению, в LTS $\mathcal{C}(\mathbf{S})$ в состоянии $\gamma`$ определён только $\gamma$-переход-петля. Любое достижимое в $\mathcal{C}(\mathbf{S})$ состояние $s \neq \gamma`$ *S*-достижимо в безопасной LTS $\mathbf{S}$. Поскольку LTS $\mathbf{S}$ *S*-ветвимая и правильно раскрашенная, в таком состоянии s имеется только конечное число τ-переходов. Если символ z безопасен после некоторой безопасной $\beta\gamma\delta$-трассы, заканчивающейся в s, то число переходов $s \text{—}z\rightarrow$ также конечно. В противном случае, по *S*-τ-ограниченности LTS $\mathbf{S}$, множество U постсостояний переходов $s \text{—}z\rightarrow$ τ-ограничено. Это означает, что оно перечислимо и дерево τ-маршрутов, начинающихся в состояниях из U и не проходящих через состояния с γ-переходом (кроме, быть может, конечного состояния), либо a) конечно, либо b) перечислимо-ветвящееся и содержит маршрут с самопересечением или заканчивающийся в состоянии с γ-

переходом. В случае a) число переходов $s \text{—} z \rightarrow$ конечно, а в случае b) в LTS $\mathcal{C}(\mathbf{S})$ имеется только один переход $s \text{—} z \rightarrow \gamma`$.

τ-ограниченность.

По построению, в LTS $\mathcal{C}(\mathbf{S})$ в состоянии $\gamma`$ определён только γ-переход-петля. Все остальные состояния LTS $\mathcal{C}(\mathbf{S})$ *S*-достижимы в безопасной LTS **S** и, следовательно, в каждом из них определено конечное число τ-переходов. Бесконечного τ-маршрута быть не может, так как это противоречило бы *S*-достижимости состояний в **S**. Следовательно, по теореме Кёнига [KÖNIG], дерево τ-маршрутов, начинающихся в каждом таком состоянии, конечно.

Слабо-регулярность.

В LTS $\mathcal{C}(\mathbf{S})$ все достижимые состояния, кроме состояния $\gamma`$, *S*-достижимы в безопасной LTS **S**. В состоянии $\gamma`$ нет неразрушающих трасс. Пусть в LTS $\mathcal{C}(\mathbf{S})$ в некотором состоянии s начинается неразрушающая трасса λ. Тогда s *S*-достижимо в **S**, а каждый конечный префикс $\sigma<\lambda$ в $\mathcal{C}(\mathbf{S})$ заканчивается не в $\gamma`$, то есть в *S*-достижимом состоянии **S**. Поэтому, по *S*-регулярности LTS **S**, в ней есть регулярный маршрут P, начинающийся в s и имеющий трассу λ.

Покажем, что маршрут P остаётся в $\mathcal{C}(\mathbf{S})$. Действительно, в противном случае у маршрута P есть префикс $P`$, который есть в $\mathcal{C}(\mathbf{S})$, а следующий в $P$ переход $\omega(P`) \text{—} z \rightarrow$ отсутствует в $\mathcal{C}(\mathbf{S})$. По построению, это означает, что в $\mathcal{C}(\mathbf{S})$ есть переход $\omega(P`) \text{—} z \rightarrow \gamma`$. Но тогда трасса λ разрушающая в $\mathcal{C}(\mathbf{S})$, что неверно.

93. **Доказательство Утверждение 93:**

По определению преобразования $\mathcal{C}$, такое состояние s *S*-достижимо в безопасной LTS **S**. Каждый переход $s \text{—} z \rightarrow s`$ в **S** либо сохраняется в $\mathcal{C}(\mathbf{S})$, либо заменяется на переход $s \text{—} z \rightarrow \gamma`$. Других переходов из s не добавляется. Отсюда непосредственно следует доказываемое утверждение.

94. **Доказательство Утверждение 94:**

Если LTS $\mathbf{S}$ опасна, то в ней есть трасса $\langle\gamma\rangle$. Тогда $\mathcal{C}(\mathbf{S})$ состоит только из начального состояния s_0, состояния $\gamma`$ и двух переходов $s_0 \text{—}\tau\rightarrow \gamma`$ и $\gamma` \text{—}\gamma\rightarrow \gamma`$. Для таких LTS $\mathbf{S}$ и $\mathcal{C}(\mathbf{S})$, очевидно, выполняются все требуемые отношения.

Далее будем считать, что LTS $\mathbf{S}$ безопасна.

Конформность: $\mathcal{C}(\mathbf{S})$ $\boldsymbol{ioco}_{\beta\gamma\delta}$ $\mathbf{S}$.

По Утверждение 54:, конформность эквивалентна мажорированию βγδ-трасс. Пусть βγδ-трасса $\sigma\in \boldsymbol{traces}_{\beta\gamma\delta^\circ}\mathcal{C}(\mathbf{S})$. Нам надо показать, что существует мажоранта $\sigma\preccurlyeq\lambda\in \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$. Рассмотрим два случая.

1. $\sigma\in \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$. Тогда $\sigma\preccurlyeq\lambda=\sigma\in \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$.
2. $\sigma\notin \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$.
   Выберем максимальный префикс $\mu<\sigma$, принадлежащий $\mathbf{S}$.
   Если μ продолжается в σ стимулом или реакцией z, то, поскольку, по Утверждение 93:, $\boldsymbol{init}_{\mathcal{C}(\mathbf{S})}(s)=\boldsymbol{init}_{\mathbf{S}}(s)$, имеет место $\mu\cdot\langle z\rangle\in \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$, что противоречит максимальности префикса μ.
   Если μ продолжается в σ βδ-отказом $r\in\beta\delta(\phi_{\mathcal{C}(\mathbf{S})}(s)_r)$, то, поскольку, по Утверждение 93:, $\phi_{\mathcal{C}(\mathbf{S})}(s)_r=\phi_{\mathbf{S}}(s)_r$, имеет место $\mu\cdot\langle r\rangle\in \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$, что также противоречит максимальности префикса μ.
   Следовательно, μ продолжается в σ разрушением: $\mu\cdot\langle\gamma\rangle\leq\sigma$, следовательно, $\mu\cdot\langle\gamma\rangle=\sigma$.
   Поскольку $\mathbf{S}$ безопасна, μ можно представить в виде $\mu=\mu`\cdot\langle z\rangle$, где z стимул или реакция.
   По построению, наличие в $\mathcal{C}(\mathbf{S})$ βγδ-трассы $\sigma=\mu`\cdot\langle z\rangle\cdot\langle\gamma\rangle$ может быть только в том случае, когда в $\mathbf{S}$ βγδ-трасса $\mu`\cdot\langle z\rangle$ заканчивается в жёлтом или красном состоянии.
   Следовательно, $\mu`\cdot\langle z\rangle$ опасна в $\mathbf{S}$.
   Тогда, по Утверждение 53: и Утверждение 54:, в $\mathbf{S}$ есть γ-мажоранта $\mu`\cdot\langle z\rangle\preccurlyeq^{\gamma}\lambda\in \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$.

Поскольку $\mu`\cdot\langle z\rangle<\sigma$, имеем $\sigma\preccurlyeq^{\gamma}\lambda\in \mathit{traces}_{\beta\gamma\delta}(S)$.

Мажорирование: $\mathit{traces}_{\phi}^{\omega}\circ C(S)\succcurlyeq \mathit{traces}_{\phi}^{\omega}(S)$.

Пусть ϕ-трасса $\sigma\in \mathit{traces}_{\phi}^{\omega}(S)$. Нам надо показать, что существует мажоранта $\sigma\preccurlyeq\sigma`\in \mathit{traces}_{\phi}^{\omega}\circ C(S)$. Рассмотрим два случая.

1. $\sigma\in \mathit{traces}_{\phi}^{\omega}\circ C(S)$. Тогда $\sigma\preccurlyeq\sigma`=\sigma\in \mathit{traces}_{\phi}^{\omega}\circ C(S)$.
2. $\sigma\notin \mathit{traces}_{\phi}^{\omega}\circ C(S)$.
   Рассмотрим в S маршрут S с ϕ-трассой σ и максимальный его префикс $M<S$, имеющийся в $C(S)$ и имеющий в S ϕ-трассу $\mu<\sigma$.
   Очевидно, все состояния маршрута M отличны от $\gamma`$.
   Тогда, по Утверждение 93:, маршрут M имеет в $C(S)$ ϕ-трассу $\mu`$, являющуюся α-мажорантой $\mu\preccurlyeq^{\alpha}\mu`$.

   Следующий в маршруте S после маршрута M переход $\omega(M)\text{—}z\rightarrow$, по построению, заменяется в $C(S)$ на переход $\omega(M)\text{—}z\rightarrow\gamma`$.

   Тогда маршрут $M`=M\cdot\langle\omega(M)\text{—}z\rightarrow\gamma`,\gamma`\text{—}\gamma\rightarrow\gamma`\rangle$ имеет в $C(S)$ ϕ-трассу $\mu`\cdot\langle z,\gamma\rangle$.

   Тогда, очевидно, $\mu\cdot\langle z\rangle<\sigma$ и, поскольку $\mu\preccurlyeq\mu`$ влечёт $\mu\cdot\langle z\rangle\preccurlyeq^{\gamma}\mu`\cdot\langle z,\gamma\rangle$, имеем $\sigma\preccurlyeq\sigma`=\mu`\cdot\langle z,\gamma\rangle\in \mathit{traces}_{\phi}^{\omega}\circ C(S)$.

95. Доказательство Утверждение 95:

По Утверждение 52:, нам достаточно показать конформность и мажорантность преобразования $C\circ W_{\beta\gamma\delta}$.

Конформность.
По Утверждение 88:, $W_{\beta\gamma\delta}(S)$ $\mathit{ioco}_{\beta\gamma\delta}$ S.
По Утверждение 94: (1), $C\circ W_{\beta\gamma\delta}(S)$ $\mathit{ioco}_{\beta\gamma\delta}$ $W_{\beta\gamma\delta}(S)$.
По Утверждение 20:, отношение $\mathit{ioco}_{\beta\gamma\delta}$ транзитивно.
Следовательно, $C\circ W_{\beta\gamma\delta}(S)$ $\mathit{ioco}_{\beta\gamma\delta}$ S.

Мажорантность.
По Утверждение 61:, мажорирование множеств ϕ-трасс LTS транзитивно.

Поэтому, по Утверждение 87: и Утверждение 78:, преобразование $\mathcal{W}_{\beta\gamma\delta}$ мажорантно.

По Утверждение 94: (2), $\mathit{traces}_{\phi}^{\omega}\circ\mathcal{C}\circ\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})\succcurlyeq\mathit{traces}_{\phi}^{\omega}\circ\mathcal{W}_{\beta\gamma\delta}(\mathbf{S})$.

Следовательно, преобразование $\mathcal{C}\circ\mathcal{W}_{\beta\gamma\delta}$ мажорантно.

96. Доказательство Утверждение 96:

Итераторы стимулов и реакций одинаковы для **L2**-LTS $\mathbf{S}$ и $\mathcal{C}(\mathbf{S})$.

В состоянии $\gamma`$ определим:
$\mathbf{Extend2}_{\mathcal{C}(\mathbf{S})}(\gamma`,z)=\epsilon$, $\mathbf{Tau2}_{\mathcal{C}(\mathbf{S})}(\gamma`)=\epsilon$, $\mathbf{Gamma2}_{\mathcal{C}(\mathbf{S})}(\gamma`)=\mathit{true}$.

В состоянии s_0 сначала строим τ-замыкание с контролем разрушения с помощью итератора $\mathbf{Tau2}_{\mathbf{S}}$ и оракула $\mathbf{Gamma2}_{\mathbf{S}}$. Если разрушение или дивергенция обнаружены, определяем $\mathbf{Extend2}_{\mathcal{C}(\mathbf{S})}(s_0,z)=\epsilon$, $\mathbf{Tau2}_{\mathcal{C}(\mathbf{S})}(s_0)=\langle\gamma`\rangle$ (перечисляется единственное постсостояние τ-перехода $\gamma`$), $\mathbf{Gamma2}_{\mathcal{C}(\mathbf{S})}(s_0)=\mathit{false}$. Если разрушение не обнаружено, для состояния s_0 строим алгоритмы так же, как для любого другого зелёного состояния.

В зелёном состоянии s исходной LTS $\mathbf{S}$ сохраним итератор $\mathbf{Tau2}_{\mathcal{C}(\mathbf{S})}(s)=\mathbf{Tau2}_{\mathbf{S}}(s)$ и оракул $\mathbf{Gamma2}_{\mathcal{C}(\mathbf{S})}(s)=\mathbf{Gamma2}_{\mathbf{S}}(s)$.

Построим итератор: $\mathbf{Extend2}_{\mathcal{C}(\mathbf{S})}(s,z)$.
Для этого с помощью алгоритмов исходной LTS $\mathbf{S}$ перечисляем постсостояния переходов $s\text{—}z\rightarrow$, проверяя цвет каждого постсостояния.
По S-τ-ограниченности, если все постсостояния зелёные, то их число конечно. Тогда определим $\mathbf{Extend2}_{\mathcal{C}(\mathbf{S})}(s,z)=\mathbf{Extend2}_{\mathbf{S}}(s,z)$.
В противном случае определим $\mathbf{Extend2}_{\mathcal{C}(\mathbf{S})}(s,z)=\langle\gamma`\rangle$ (перечисляется единственное постсостояние τ-перехода $\gamma`$).

97. Доказательство Утверждение 97:

Непосредственно следует из Утверждение 79: и равенства $\mathcal{T}_{\beta\gamma\delta}(\mathbf{S})=\mathcal{T}_{\beta\delta}(\mathbf{S})$ для LTS $\mathbf{S}$ без разрушения.

98. **Доказательство Утверждение 98:**

Сохранение свойств конечно-ветвимости и τ-ограниченности непосредственно следует из Утверждение 81: и определения преобразования $\mathcal{T}_{\beta\delta}$ как совпадающего с $\mathcal{T}_{\beta\gamma\delta}$ на классе LTS без разрушения. В результате преобразования $\mathcal{T}_{\beta\delta}$, очевидно, разрушение не появляется.

99. **Доказательство Утверждение 99:**

По определению сильного мажорирования, $\mu \preccurlyeq\preccurlyeq \sigma \;=\; \mu=\sigma \;\vee\; \mu \preccurlyeq\preccurlyeq^{\gamma} \sigma$.

Если мажорируемая βγδ-трасса μ конечна, то для $\mu=\sigma$ βγδ-трасса σ также конечна. А если $\mu \preccurlyeq\preccurlyeq^{\gamma} \sigma$, то σ конечна по определению (заканчивается разрушением).

100. **Доказательство Утверждение 100:**

Непосредственно следует из определения мажорирования и сильного мажорирования βγδ-трасс.

101. **Доказательство Утверждение 101:**

Рефлексивность и транзитивность сильного мажорирования множеств βγδ-трасс непосредственно следует из рефлексивности и транзитивности сильного мажорирования отдельных βγδ-трасс, которое мы сейчас и докажем.

Рефлексивность.

По определению сильного мажорирования βγδ-трасс, $\mu=\lambda \;\Rightarrow\; \mu \preccurlyeq\preccurlyeq \lambda$.

Транзитивность.

Пусть $\mu \preccurlyeq\preccurlyeq \lambda \preccurlyeq\preccurlyeq \sigma$. Возможны два варианта.

1. $\mu=\lambda \;\vee\; \lambda=\sigma$. Тогда также $\mu \preccurlyeq\preccurlyeq \sigma$.

2. $\mu \preccurlyeq\preccurlyeq^{\gamma} \lambda \;\&\; \lambda \preccurlyeq\preccurlyeq^{\gamma} \sigma$.

$\mu \preccurlyeq\preccurlyeq^{\gamma} \lambda$ влечёт $\exists \pi_1 \le \mu \;\; \lambda=\pi_1\cdot\langle\gamma\rangle \;\vee\; \exists a_1 \in C \;\; \lambda=\pi_1\cdot\langle a_1,\gamma\rangle \;\&\; \pi_1\cdot\langle\beta\delta(a_1)\rangle \le \mu$.

$\lambda \preccurlyeq\preccurlyeq^{\gamma} \sigma$ влечёт $\exists \pi_2 \le \lambda \;\; \sigma=\pi_2\cdot\langle\gamma\rangle \;\vee\; \exists a_2 \in C \;\; \sigma=\pi_2\cdot\langle a_2,\gamma\rangle \;\&\; \pi_2\cdot\langle\beta\delta(a_2)\rangle \le \lambda$.

Рассмотрим подварианты.

2.1. $\lambda=\pi_1\cdot\langle\gamma\rangle \;\&\; \sigma=\pi_2\cdot\langle\gamma\rangle$.

Так как $\sigma=\pi_2\cdot\langle\gamma\rangle$, π_2 не заканчивается на γ.

Поэтому $\pi_2\le\lambda=\pi_1\cdot\langle\gamma\rangle$ влечёт $\pi_2\le\pi_1$.

Поскольку также $\pi_1\le\mu$, имеем $\pi_2\le\mu$.

Итак: $\pi_2\le\mu$ & $\sigma=\pi_2\cdot\langle\gamma\rangle$, что влечёт $\mu\preccurlyeq^{\gamma}\sigma$.

2.2. $\lambda=\pi_1\cdot\langle\gamma\rangle$ & $\sigma=\pi_2\cdot\langle a_2,\gamma\rangle$ & $\pi_2\cdot\langle\beta\delta(a_2)\rangle\le\lambda$.

Так как $\pi_2\cdot\langle\beta\delta(a_2)\rangle\le\lambda=\pi_1\cdot\langle\gamma\rangle$ и $\beta\delta(a_2)\ne\gamma$, то $\pi_2\cdot\langle\beta\delta(a_2)\rangle\le\pi_1$.

Поскольку также $\pi_1\le\mu$, имеем $\pi_2\cdot\langle\beta\delta(a_2)\rangle\le\mu$.

Итак: $\sigma=\pi_2\cdot\langle a_2,\gamma\rangle$ & $\pi_2\cdot\langle\beta\delta(a_2)\rangle\le\mu$, что влечёт $\mu\preccurlyeq^{\gamma}\sigma$.

2.3. $\lambda=\pi_1\cdot\langle a_1,\gamma\rangle$ & $\pi_1\cdot\langle\beta\delta(a_1)\rangle\le\mu$ & $\sigma=\pi_2\cdot\langle\gamma\rangle$.

Так как $\sigma=\pi_2\cdot\langle\gamma\rangle$, π_2 не заканчивается на γ.

Поэтому $\pi_2\le\lambda=\pi_1\cdot\langle a_1,\gamma\rangle$ влечёт $\pi_2\le\pi_1\cdot\langle a_1\rangle$.

Если $\pi_2=\pi_1\cdot\langle a_1\rangle$, то имеем $\sigma=\pi_1\cdot\langle a_1,\gamma\rangle$ & $\pi_1\le\mu$, что влечёт $\mu\preccurlyeq^{\gamma}\sigma$.

Если $\pi_2<\pi_1\cdot\langle a_1\rangle$, то $\pi_2\le\pi_1$.

Поскольку $\pi_2\le\pi_1\le\mu$, то имеем $\sigma=\pi_2\cdot\langle\gamma\rangle$ & $\pi_2\le\mu$, что также влечёт $\mu\preccurlyeq^{\gamma}\sigma$.

2.4. $\lambda=\pi_1\cdot\langle a_1,\gamma\rangle$ & $\pi_1\cdot\langle\beta\delta(a_1)\rangle\le\mu$ & $\sigma=\pi_2\cdot\langle a_2,\gamma\rangle$ & $\pi_2\cdot\langle\beta\delta(a_2)\rangle\le\lambda$.

Так как $\pi_2\cdot\langle\beta\delta(a_2)\rangle\le\lambda=\pi_1\cdot\langle a_1,\gamma\rangle$ и $\gamma\ne\beta\delta(a_2)\ne a_1$, то $\pi_2\cdot\langle\beta\delta(a_2)\rangle\le\pi_1$.

Поскольку также $\pi_1\le\mu$, имеем $\pi_2\cdot\langle\beta\delta(a_2)\rangle\le\mu$.

Итак: $\sigma=\pi_2\cdot\langle a_2,\gamma\rangle$ & $\pi_2\cdot\langle\beta\delta(a_2)\rangle\le\mu$, что влечёт $\mu\preccurlyeq^{\gamma}\sigma$.

102. **Доказательство Утверждение 102:**

Мажорирование множеств ϕ-трасс **A** и **B** определяется как сильное мажорирование множеств βγδ-трасс, порождаемых ϕ-трассами этих множеств **A** и **B**: $\mathbf{A}\preccurlyeq\mathbf{B} = \cup_{\circ}\xi^{\omega}(\mathbf{A}) \preccurlyeq\preccurlyeq \cup_{\circ}\xi^{\omega}(\mathbf{B})$. Поэтому рефлексивность и транзитивность мажорирования множеств ϕ-трасс непосредственно следует из рефлексивности и транзитивности сильного мажорирования βγδ-трасс, которое доказано в Утверждение 101:.

103. **Доказательство Утверждение 103:**

Мажорирование множеств ϕ-трасс **A** и **B** определяется как сильное мажорирование множеств βγδ-трасс, порождаемых ϕ-трассами этих множеств **A** и **B**:

$\mathbf{A}\preccurlyeq\mathbf{B} = \cup_\circ\xi^\omega(\mathbf{A}) \preccurlyeq\preccurlyeq \cup_\circ\xi^\omega(\mathbf{B})$.

По Утверждение 99:, сильная мажоранта конечной βγδ-трассы конечна:

$\{\mu\in\cup_\circ\xi^\omega(\mathbf{A})\,|\,|\mu|\neq\infty\} \preccurlyeq\preccurlyeq \{\sigma\in\cup_\circ\xi^\omega(\mathbf{B})\,|\,|\sigma|\neq\infty\}$.

По Утверждение 100:, сильное мажорирование конечных βγδ-трасс влечёт обычное мажорирование этих βγδ-трасс:

$\{\mu\in\cup_\circ\xi^\omega(\mathbf{A})\,|\,|\mu|\neq\infty\} \preccurlyeq \{\sigma\in\cup_\circ\xi^\omega(\mathbf{B})\,|\,|\sigma|\neq\infty\}$, что эквивалентно

$\cup_\circ\xi(\mathbf{A}) \preccurlyeq \cup_\circ\xi(\mathbf{B})$.

104. **Доказательство Утверждение 104:**

Пусть $A_1, A_2\subseteq Z$, $\mathbf{I_1}\in\boldsymbol{LTS}_{\beta\gamma\delta}(A_1)$, $\mathbf{I_2}\in\boldsymbol{LTS}_{\beta\gamma\delta}(A_2)$, $\mathbf{S_1}\in\boldsymbol{LTS}_{\gamma\delta}(A_1)$, $\mathbf{S_2}\in\boldsymbol{LTS}_{\gamma\delta}(A_2)$, $\mathbf{I_1}=\boldsymbol{traces}_\phi^\omega(\mathbf{I_1})$, $\mathbf{I_2}=\boldsymbol{traces}_\phi^\omega(\mathbf{I_2})$, $\mathbf{\Sigma_1}=\boldsymbol{traces}_\phi^\omega(\mathbf{S_1})$, $\mathbf{\Sigma_2}=\boldsymbol{traces}_\phi^\omega(\mathbf{S_2})$, $\mathbf{I_1}\preccurlyeq\mathbf{\Sigma_1}$, $\mathbf{I_2}\preccurlyeq\mathbf{\Sigma_2}$, $A=A_1\between A_2$.

Пусть также $\boldsymbol{\kappa_1}\in\mathbf{I_1}$, $\boldsymbol{\kappa_2}\in\mathbf{I_2}$, $\boldsymbol{\kappa}\in\boldsymbol{\kappa_1}\between\boldsymbol{\kappa_2}$ и $\kappa\in\xi^\omega(\boldsymbol{\kappa})$.

Нам надо доказать, что существуют $\boldsymbol{\lambda_1}\in\mathbf{\Sigma_1}$, $\boldsymbol{\lambda_2}\in\mathbf{\Sigma_2}$, $\boldsymbol{\lambda}\in\boldsymbol{\lambda_1}\between\boldsymbol{\lambda_2}$ и $\lambda\in\xi^\omega(\boldsymbol{\lambda})$ такие, что $\kappa\preccurlyeq\preccurlyeq\lambda$.

Каждый βδ-отказ r βγδ-трассы κ порождается некоторым ϕ-символом o ϕ-трассы $\boldsymbol{\kappa}$, который в свою очередь, является результатом композиции двух ϕ-символов o_1 и o_2 ϕ-трасс $\boldsymbol{\kappa_1}$ и $\boldsymbol{\kappa_2}$, соответственно: $o=o_1\between o_2$.

Если βδ-отказ r это блокировка стимула $\{?x\}$, то он наследуется от одного из операндов.

Если βδ-отказ r это стационарность δ, то каждому ответвлению по синхронной реакции в одном операнде соответствует блокировка противоположного стимула в другом операнде:
$A_1\cap\underline{A_2}\setminus o_{1r}\subseteq\underline{o_{2r}}$ & $A_2\cap\underline{A_1}\setminus o_{2r}\subseteq\underline{o_{1r}}$.

Нас будут интересовать два случая:

- *Удобная стационарность*: нет ответвлений по синхронным реакциям, то есть ϕ-символ каждого операнда порождает стационарность: $!A_1 \subseteq o_{1r}$ и $!A_2 \subseteq o_{2r}$.
- *Неудобная стационарность*: есть ответвление по синхронной реакции в одном операнде, причём в каждом таком случае, например, $!x \notin o_{2r}$, противоположный стимул блокируется в другом операнде $?x \in o_{1r}$.

1. Сначала рассмотрим случай, когда в βγδ-трассе κ нет блокировок и неудобных стационарностей.

Построим βγδ-трассы операндов $\kappa_1 \in \xi^{\omega}(\boldsymbol{\kappa_1})$ и $\kappa_2 \in \xi^{\omega}(\boldsymbol{\kappa_2})$.
Подтрассы базовых символов однозначно определяются порождающими ϕ-трассами: $\kappa_1 \downarrow A_{1\gamma} = \boldsymbol{\kappa_1} \downarrow A_{1\gamma}$ и $\kappa_2 \downarrow A_{2\gamma} = \boldsymbol{\kappa_2} \downarrow A_{2\gamma}$.
Каждой стационарности βγδ-трассы κ будет соответствовать стационарность в κ_1 и стационарность в κ_2.

Так как $\boldsymbol{l_1} \preccurlyeq \boldsymbol{\Sigma_1}$ и $\boldsymbol{l_2} \preccurlyeq \boldsymbol{\Sigma_2}$, эти βγδ-трассы должны иметь сильные мажоранты:

$\exists \boldsymbol{\lambda_1} \in \boldsymbol{\Sigma_1}\ \exists \boldsymbol{\lambda_2} \in \boldsymbol{\Sigma_2},\ \exists \lambda_1 \in \xi^{\omega}(\boldsymbol{\lambda_1})\ \exists \lambda_2 \in \xi^{\omega}(\boldsymbol{\lambda_2})\ \kappa_1 \preccurlyeq\preccurlyeq \lambda_1\ \&\ \kappa_2 \preccurlyeq\preccurlyeq \lambda_2$.

Рассмотрим два подслучая.

1.1. βγδ-трассы κ_1 и κ_2 мажорируются по равенству: $\kappa_1 = \lambda_1$, $\kappa_2 = \lambda_2$ и не имеют γ-мажорирования, за исключением, быть может, случая продолжения разрушением: $\forall \mu_i \in \cup_{\circ} \xi^{\omega}(\boldsymbol{\Sigma_i})\ \kappa_i \preccurlyeq\preccurlyeq^{\gamma} \mu_i \Rightarrow \kappa_i \cdot \langle\gamma\rangle = \mu_i$, где $i=1,2$.

Удалим из ϕ-трасс $\boldsymbol{\lambda_1}$ и $\boldsymbol{\lambda_2}$ те ϕ-символы, которые не участвуют в порождении βγδ-трасс λ_1 и λ_2.
По построению, подтрассы базовых символов ϕ-трасс $\boldsymbol{\lambda_1}$ и $\boldsymbol{\kappa_1}$ совпадают: $\boldsymbol{\lambda_1} \downarrow A_{1\gamma} = \boldsymbol{\kappa_1} \downarrow A_{1\gamma}$. То же самое верно для ϕ-трасс $\boldsymbol{\lambda_2}$ и $\boldsymbol{\kappa_2}$.
Каждому ϕ-символу ϕ-трассы $\boldsymbol{\lambda_1}$ соответствует ϕ-символ ϕ-трассы $\boldsymbol{\kappa_1}$, и оба они порождают стационарность. То же самое верно для ϕ-трасс $\boldsymbol{\lambda_2}$ и $\boldsymbol{\kappa_2}$.
Поэтому из компонуемости ϕ-трасс $\boldsymbol{\kappa_1}$ и $\boldsymbol{\kappa_2}$ следует компонуемость ϕ-трасс $\boldsymbol{\lambda_1}$ и $\boldsymbol{\lambda_2}$.

Более того, поскольку $\boldsymbol{\kappa\in\kappa_1\wr\!\mid\kappa_2}$ и $\kappa\in\xi^{\omega}(\boldsymbol{\kappa})$, по построению, найдётся ϕ-трасса $\boldsymbol{\lambda\in\lambda_1\wr\!\mid\lambda_2}$, которая порождает ту же самую βγδ-трассу $\kappa\in\xi^{\omega}(\boldsymbol{\lambda})$.[77]

Поскольку $\kappa\preccurlyeq\preccurlyeq\kappa$, утверждение для этого подслучая доказано.

1.2. Хотя бы одна из βγδ-трасс κ_1 и κ_2 имеет γ-мажоранту, отличную от продолжения этой βγδ-трассы разрушением: $\kappa_1\preccurlyeq\preccurlyeq^{\gamma}\lambda_1\neq\kappa_1\cdot\langle\gamma\rangle \vee \kappa_2\preccurlyeq\preccurlyeq^{\gamma}\lambda_2\neq\kappa_2\cdot\langle\gamma\rangle$.

У βγδ-трассы κ всегда существует такой префикс, для которого имеет место подслучай 1.1. В частности, таким префиксом является пустой префикс, поскольку в каждой LTS имеются пустая βγδ-трасса и пустая ϕ-трасса, композиция пустых ϕ-трасс даёт пустую ϕ-трассу, пустая ϕ-трасса порождает пустую βγδ-трассу и пустая βγδ-трасса γ-мажорируется только трассой $\langle\gamma\rangle$, которая является продолжением пустой βγδ-трассы разрушением.

Выберем максимальный такой префикс βγδ-трассы $\kappa`<\kappa$ и соответствующие ему (по 1.1) префиксы ϕ- и βγδ-трасс, которые мы будем снабжать штрихом "`":

$\boldsymbol{\kappa`\leq\kappa}$, $\boldsymbol{\kappa`_1\leq\kappa_1}$, $\boldsymbol{\kappa`_2\leq\kappa_2}$, $\boldsymbol{\lambda`\leq\lambda}$, $\boldsymbol{\lambda`_1\leq\lambda_1}$, $\boldsymbol{\lambda`_2\leq\lambda_2}$, $\kappa`_1\leq\kappa_1$, $\kappa`_2\leq\kappa_2$

такие, что

$\boldsymbol{\kappa`\in\kappa`_1\wr\!\mid\kappa`_2}$, $\boldsymbol{\lambda`\in\lambda`_1\wr\!\mid\lambda`_2}$, $\kappa`\in\xi^{\omega}(\boldsymbol{\kappa`})$, $\kappa`\in\xi^{\omega}(\boldsymbol{\lambda`})$,

$\kappa`_1\in\xi^{\omega}(\boldsymbol{\kappa`_1})$, $\kappa`_1\in\xi^{\omega}(\boldsymbol{\lambda`_1})$, $\kappa`_2\in\xi^{\omega}(\boldsymbol{\kappa`_2})$, $\kappa`_2\in\xi^{\omega}(\boldsymbol{\lambda`_2})$.

Пусть, для определённости, $\kappa`_1$ продолжается в $\kappa_1$ символом $u$ и $\kappa`_1\cdot\langle u\rangle$ имеет γ-мажоранту, отличную от продолжения разрушением $\kappa`_1\cdot\langle u,\gamma\rangle$.

Поскольку $\kappa`_1$ не имеет γ-мажоранты, отличной от $\kappa`_1\cdot\langle\gamma\rangle$, возможны два варианта:

$\exists\boldsymbol{\lambda``_1}\cdot\langle\gamma\rangle\in\boldsymbol{\Sigma_1}\ \kappa`_1\cdot\langle\gamma\rangle\in\xi^{\omega}(\boldsymbol{\lambda``_1}\cdot\langle\gamma\rangle)$

$\vee\ u=\delta\ \ \&\ \exists!y\in A_1\ \exists\boldsymbol{\lambda``_1}\cdot\langle!y,\gamma\rangle\in\boldsymbol{\Sigma_1}\ \kappa`_1\in\xi^{\omega}(\boldsymbol{\lambda``_1})$.

---

[77] Этот подслучай покрывает композицию не только конечных ϕ-трасс, но и бесконечных ϕ-трасс как по правилу вывода $(\phi 6)$, когда получается бесконечная композиционная ϕ-трасса, так и по правилу вывода $(\phi 5)$, когда получается конечная композиционная ϕ-трасса, заканчивающаяся разрушением, моделирующим дивергентность.

1.2.1. $\exists \boldsymbol{\lambda}``_1 \cdot \langle \gamma \rangle \in \boldsymbol{\Sigma}_1 \;\; \kappa`_1 \cdot \langle \gamma \rangle \in \xi^{\omega}(\boldsymbol{\lambda}``_1 \cdot \langle \gamma \rangle)$.

Аналогично 1.1, можно считать, что в ϕ-трассе $\boldsymbol{\lambda}``_1$ «лишние» ϕ-символы удалены и существует $\boldsymbol{\lambda}`' \in \boldsymbol{\lambda}``_1 \upharpoonright \boldsymbol{\lambda}`_2$ такая, что $\kappa` \in \xi^{\omega}(\boldsymbol{\lambda}`')$.

Тогда $\boldsymbol{\lambda}`' \cdot \langle \gamma \rangle \in \boldsymbol{\lambda}``_1 \cdot \langle \gamma \rangle \upharpoonright \boldsymbol{\lambda}`_2$ и $\kappa` \cdot \langle \gamma \rangle \in \xi^{\omega}(\boldsymbol{\lambda}`' \cdot \langle \gamma \rangle)$.

Следовательно, $\kappa` \preccurlyeq \preccurlyeq^{\gamma} \kappa` \cdot \langle \gamma \rangle$, что для $\kappa` < \kappa$ влечёт $\kappa \preccurlyeq \preccurlyeq^{\gamma} \kappa` \cdot \langle \gamma \rangle$.

1.2.2. $u = \delta \;\; \& \;\; \exists !y \in A_1 \;\; \exists \boldsymbol{\lambda}``_1 \cdot \langle !y, \gamma \rangle \in \boldsymbol{\Sigma}_1 \;\; \kappa`_1 \in \xi^{\omega}(\boldsymbol{\lambda}``_1)$.

Аналогично 1.1, можно считать, что в ϕ-трассе $\boldsymbol{\lambda}``_1$ «лишние» ϕ-символы удалены и существует $\boldsymbol{\lambda}`' \in \boldsymbol{\lambda}``_1 \upharpoonright \boldsymbol{\lambda}`_2$ такая, что $\kappa` \in \xi^{\omega}(\boldsymbol{\lambda}`')$.

Здесь возможны два варианта в зависимости от того, реакция $!y$ асинхронная или синхронная.

1.2.2.1. Асинхронная реакция $!y$.

Тогда $\boldsymbol{\lambda}`' \cdot \langle !y, \gamma \rangle \in \boldsymbol{\lambda}``_1 \cdot \langle !y, \gamma \rangle \upharpoonright \boldsymbol{\lambda}`_2$ и $\kappa` \cdot \langle !y, \gamma \rangle \in \xi^{\omega}(\boldsymbol{\lambda}`' \cdot \langle !y, \gamma \rangle)$.

Следовательно, $\kappa` \cdot \langle \delta \rangle \preccurlyeq \preccurlyeq^{\gamma} \kappa` \cdot \langle !y, \gamma \rangle$, что для $\kappa` \cdot \langle \delta \rangle \le \kappa$ влечёт $\kappa \preccurlyeq \preccurlyeq^{\gamma} \kappa` \cdot \langle !y, \gamma \rangle$.

1.2.2.2. Синхронная реакция $!y$.

Поскольку в LTS $\boldsymbol{S}_2$ нет блокировок, каждая её ϕ-трасса продолжается каждым стимулом из A_2, в частности, $\boldsymbol{\lambda}`_2 \cdot \langle ?y \rangle \in \boldsymbol{\Sigma}_2$.

Тогда $\boldsymbol{\lambda}`' \cdot \langle \gamma \rangle \in \boldsymbol{\lambda}``_1 \cdot \langle !y, \gamma \rangle \upharpoonright \boldsymbol{\lambda}`_2 \cdot \langle ?y \rangle$ и $\kappa` \cdot \langle \gamma \rangle \in \xi^{\omega}(\boldsymbol{\lambda}`' \cdot \langle \gamma \rangle)$.

Следовательно, $\kappa` \preccurlyeq \preccurlyeq^{\gamma} \kappa` \cdot \langle \gamma \rangle$, что для $\kappa` < \kappa$ влечёт $\kappa \preccurlyeq \preccurlyeq^{\gamma} \kappa` \cdot \langle \gamma \rangle$.

2. Теперь рассмотрим случай, когда в βγδ-трассе κ есть блокировка или неудобная стационарность.

Рассмотрим первое вхождение блокировки или неудобной стационарности u в βγδ-трассу κ: $\kappa \ge \kappa` \cdot \langle u \rangle$ и $\kappa`$ не содержит блокировок и неудобных стационарностей.

βγδ-трасса $\kappa`$ удовлетворяет условиям случая 1.

Если βγδ-трасса $\kappa`$ удовлетворяет условиям подслучая 1.2, она имеет γ-мажоранту $\kappa` \preccurlyeq \preccurlyeq^{\gamma} \kappa`` \cdot \langle\gamma\rangle$, что для $\kappa` < \kappa$ влечёт $\kappa \preccurlyeq \preccurlyeq^{\gamma} \kappa`` \cdot \langle\gamma\rangle$.

Далее будем считать, что βγδ-трасса $\kappa`$ удовлетворяет условиям подслучая 1.1.
Тогда для βγδ-префикса $\kappa` < \kappa$ можно выбрать соответствующие ему (по 1.1) ϕ- и βγδ-трассы, которые мы будем снабжать штрихом "`":
$\boldsymbol{\kappa`} \le \boldsymbol{\kappa}$, $\boldsymbol{\kappa`_1} \le \boldsymbol{\kappa_1}$, $\boldsymbol{\kappa`_2} \le \boldsymbol{\kappa_2}$, $\boldsymbol{\lambda`}$, $\boldsymbol{\lambda`_1} \in \boldsymbol{\Sigma_1}$, $\boldsymbol{\lambda`_2} \in \boldsymbol{\Sigma_2}$,
такие, что
$\boldsymbol{\kappa`} \in \boldsymbol{\kappa`_1} \wr\lfloor \boldsymbol{\kappa`_2}$, $\boldsymbol{\lambda`} \in \boldsymbol{\lambda`_1} \wr\lfloor \boldsymbol{\lambda`_2}$, $\kappa` \in \xi^{\omega}(\boldsymbol{\kappa`})$, $\kappa` \in \xi^{\omega}(\boldsymbol{\lambda`})$,
$\kappa`_1 \in \xi^{\omega}(\boldsymbol{\kappa`_1})$, $\kappa`_1 \in \xi^{\omega}(\boldsymbol{\lambda`_1})$, $\kappa`_2 \in \xi^{\omega}(\boldsymbol{\kappa`_2})$, $\kappa`_2 \in \xi^{\omega}(\boldsymbol{\lambda`_2})$.

Поскольку u βδ-отказ, он порождается некоторым ϕ-символом $o \in \Phi(A)$, которым ϕ-префикс $\boldsymbol{\kappa`}$ продолжается в ϕ-трассе $\boldsymbol{\kappa}$: $\boldsymbol{\kappa`} \cdot \langle o \rangle \le \boldsymbol{\kappa}$.
Этот ϕ-символ получается в результате композиции двух ϕ-символов $o_1 \in \Phi(A_1)$ и $o_2 \in \Phi(A_2)$, которыми ϕ-префиксы $\boldsymbol{\kappa`_1}$ и $\boldsymbol{\kappa`_2}$ продолжается в ϕ-трассах $\boldsymbol{\kappa_1}$ и $\boldsymbol{\kappa_2}$, соответственно: $\boldsymbol{\kappa`_1} \cdot \langle o_1 \rangle \le \boldsymbol{\kappa_1}$ и $\boldsymbol{\kappa`_2} \cdot \langle o_2 \rangle \le \boldsymbol{\kappa_2}$.

Если в $\cup_{\circ} \xi^{\omega}(\boldsymbol{\Sigma_1})$ есть βγδ-трасса $\kappa`_1 \cdot \langle\gamma\rangle$, то она порождается некоторой ϕ-трассой $\boldsymbol{\lambda``_1} \cdot \langle\gamma\rangle \in \boldsymbol{\Sigma_1}$.
Аналогично 1.1, можно считать, что в ϕ-трассе $\boldsymbol{\lambda``_1}$ «лишние» ϕ-символы удалены и существует $\boldsymbol{\lambda``} \in \boldsymbol{\lambda``_1} \wr\lfloor \boldsymbol{\lambda`_2}$ такая, что $\kappa` \in \xi^{\omega}(\boldsymbol{\lambda``})$.
А тогда $\boldsymbol{\lambda``} \cdot \langle\gamma\rangle \in \boldsymbol{\lambda``_1} \cdot \langle\gamma\rangle \wr\lfloor \boldsymbol{\lambda`_2}$, $\kappa` \cdot \langle\gamma\rangle \in \xi^{\omega}(\boldsymbol{\lambda``} \cdot \langle\gamma\rangle)$, что для $\kappa` < \kappa$ влечёт $\kappa \preccurlyeq \preccurlyeq^{\gamma} \kappa` \cdot \langle\gamma\rangle$.

Аналогично для случая, когда в $\cup_{\circ} \xi^{\omega}(\boldsymbol{\Sigma_2})$ есть βγδ-трасса $\kappa`_2 \cdot \langle\gamma\rangle$.

Далее будем считать, что $\kappa`_1 \cdot \langle\gamma\rangle \notin \cup_{\circ} \xi^{\omega}(\boldsymbol{\Sigma_1})$ и $\kappa`_2 \cdot \langle\gamma\rangle \notin \cup_{\circ} \xi^{\omega}(\boldsymbol{\Sigma_2})$.

Мы рассмотрим два подслучая в зависимости от того, является символ u блокировкой или неудобной стационарностью.

2.1. Блокировка $u = \{?x\}$.

Стимул $?x$ асинхронный.
Пусть, для определённости, $?x \in A_1 \setminus A_2$.

Тогда $\kappa`_1\cdot\langle\{?x\}\rangle\in\xi^{\omega}(\boldsymbol{\kappa`_1}\cdot\langle o_1\rangle)$.

Поскольку $\kappa`_1\cdot\langle\gamma\rangle\notin\cup_{\circ}\xi^{\omega}(\boldsymbol{\Sigma_1})$ и в LTS $\mathbf{S_2}$ нет блокировок, $\kappa`_1\cdot\langle\{?x\}\rangle$ имеет мажоранту $\kappa`_1\cdot\langle?x,\gamma\rangle$.

Эта мажоранта порождается некоторой ϕ-трассой $\boldsymbol{\lambda``_1}\cdot\langle?x,\gamma\rangle\in\boldsymbol{\Sigma_1}$.

Аналогично 1.1, можно считать, что в ϕ-трассе $\boldsymbol{\lambda``_1}$ «лишние» ϕ-символы удалены и существует $\boldsymbol{\lambda``}\in\boldsymbol{\lambda``_1}\downharpoonright\!\upharpoonright\boldsymbol{\lambda`_2}$ такая, что $\kappa`\in\xi^{\omega}(\boldsymbol{\lambda``})$.

А тогда $\boldsymbol{\lambda``}\cdot\langle?x,\gamma\rangle\in\boldsymbol{\lambda``_1}\cdot\langle?x,\gamma\rangle\downharpoonright\!\upharpoonright\boldsymbol{\lambda`_2}$, $\kappa`\cdot\langle?x,\gamma\rangle\in\xi^{\omega}(\boldsymbol{\lambda``}\cdot\langle?x,\gamma\rangle)$, что для $\kappa`\cdot\langle\{?x\}\rangle<\kappa$ влечёт $\kappa\preccurlyeq\preccurlyeq^{\gamma}\kappa`\cdot\langle?x,\gamma\rangle$.

2.2. Неудобная стационарность $u=\delta$.

В этом случае есть ответвление по синхронной реакции хотя бы в одном операнде. Для определённости, пусть такое ответвление есть во втором операнде, то есть существует такая синхронная реакция $!x\notin o_{2r}$, что противоположный стимул блокируется в первом операнде $?x\in o_{1r}$.

Также как в 2.1, есть βγδ-трасса $\kappa`_1\cdot\langle\{?x\}\rangle\in\xi^{\omega}(\boldsymbol{\kappa`_1}\cdot\langle o_1\rangle)$, которая имеет мажоранту $\kappa`_1\cdot\langle?x,\gamma\rangle$.

Эта мажоранта порождается некоторой ϕ-трассой $\boldsymbol{\lambda``_1}\cdot\langle?x,\gamma\rangle\in\boldsymbol{\Sigma_1}$ такой, что существует $\boldsymbol{\lambda``}\in\boldsymbol{\lambda``_1}\downharpoonright\!\upharpoonright\boldsymbol{\lambda`_2}$ и $\kappa`\in\xi^{\omega}(\boldsymbol{\lambda``})$.

Также должно быть $\boldsymbol{\kappa`_2}\cdot\langle!x\rangle\in\mathbf{I_2}$ и $\kappa`_2\cdot\langle!x\rangle\in\xi^{\omega}(\boldsymbol{\kappa`_2}\cdot\langle!x\rangle)$.

Поскольку $\kappa`_2\cdot\langle\gamma\rangle\notin\cup_{\circ}\xi^{\omega}(\boldsymbol{\Sigma_2})$, $\kappa`_2\cdot\langle!x\rangle$ имеет мажоранту $\kappa`_2\cdot\langle!x\rangle$.

Эта мажоранта порождается некоторой ϕ-трассой $\boldsymbol{\lambda``_2}\cdot\langle!x\rangle\in\boldsymbol{\Sigma_2}$.

Аналогично 1.1, можно считать, что в ϕ-трассе $\boldsymbol{\lambda``_2}$ «лишние» ϕ-символы удалены и существует $\boldsymbol{\lambda```}\in\boldsymbol{\lambda``_1}\downharpoonright\!\upharpoonright\boldsymbol{\lambda``_2}$ такая, что $\kappa`\in\xi^{\omega}(\boldsymbol{\lambda```})$.

А тогда $\boldsymbol{\lambda```}\cdot\langle\gamma\rangle\in\boldsymbol{\lambda``_1}\cdot\langle?x,\gamma\rangle\downharpoonright\!\upharpoonright\boldsymbol{\lambda``_2}\cdot\langle!x\rangle$, $\kappa`\cdot\langle\gamma\rangle\in\xi^{\omega}(\boldsymbol{\lambda```}\cdot\langle\gamma\rangle)$, что для $\kappa`<\kappa$ влечёт $\kappa\preccurlyeq\preccurlyeq^{\gamma}\kappa`\cdot\langle\gamma\rangle$.

105. **Доказательство Утверждение 105:**

γ-однородность преобразованной LTS непосредственно следует из определения преобразования.

106. Доказательство Утверждение 106:

Преобразование $\mathcal{T}_{\gamma\delta}$ не удаляет ни одного перехода и добавляет переходы по реакциям только в таких состояниях, в которых определены переходы по разрушающим реакциям. Поэтому преобразование $\mathcal{T}_{\gamma\delta}$ не изменяет множество безопасных βγδ-трасс. Тогда, по Утверждение 21:, $\mathcal{T}_{\gamma\delta}(\mathbf{S}) \sim_{\boldsymbol{ioco}_{\beta\gamma\delta}} \mathbf{S}$, что и требовалось доказать.

107. Доказательство Утверждение 107:

Обычное мажорирование βγδ-трасс отличается от сильного мажорирования тем, что 1) определено только для конечных βγδ-трасс, и 2) разрешено продолжение мажоранты реакцией и разрушением, когда мажорируемая βγδ-трасса продолжается *другой* реакцией.
В γ-однородных спецификациях, если после βγδ-трассы одна реакция разрушающая, то и другая реакция разрушающая.
Поэтому, если LTS $\mathbf{S}$ γ-однородна, то сильное и обычное мажорирование конечных βγδ-трасс совпадают.

108. Доказательство Утверждение 108:

Пусть задана γ-однородная LTS-спецификация $\mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}$ и конформная LTS-реализация $\mathbf{I}$ $\boldsymbol{ioco}_{\beta\gamma\delta}$ $\mathbf{S}$.

Нам нужно доказать, что $\boldsymbol{traces}_{\phi}(\mathbf{I}) \preccurlyeq \boldsymbol{traces}_{\phi}(\mathbf{S})$.

По Утверждение 54:, конформность эквивалентна обычному мажорированию βγδ-трасс, следовательно:

$\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{I}) \preccurlyeq \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$.

По Утверждение 107:, для γ-однородных спецификаций (правые операнды мажорирования) сильное и обычное мажорирование конечных βγδ-трасс совпадают, следовательно:

$\boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{I}) \preccurlyeq\preccurlyeq \boldsymbol{traces}_{\beta\gamma\delta}(\mathbf{S})$.

По Утверждение 56:, ϕ-трассы генеративны, следовательно:

$\cup_{\circ}\xi_{\circ}\boldsymbol{traces}_{\phi}(\mathbf{I}) \preccurlyeq\preccurlyeq \cup_{\circ}\xi_{\circ}\boldsymbol{traces}_{\phi}(\mathbf{S})$.

Мажорирование множеств ϕ-трасс сводится к сильному мажорированию множеств βγδ-трасс, порождаемых этими ϕ-трассами, следовательно:

$\boldsymbol{traces}_{\phi}(\mathbf{I}) \preccurlyeq \boldsymbol{traces}_{\phi}(\mathbf{S})$, что и требовалось доказать.

109. **Доказательство Утверждение 109:**

Пусть заданы алфавит $C \subseteq Z$, $\mathbf{S} \in \boldsymbol{LTS}_{\beta\gamma\delta}(C)$ γ-однородная S-ветвящаяся конечно-мажорантная LTS и конформная реализация $\mathbf{I}$ $\boldsymbol{ioco}_{\beta\gamma\delta}$ $\mathbf{S}$.

Нам нужно доказать: $\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{I}) \preccurlyeq \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S})$.

Мажорирование множеств ϕ-трасс сводится к сильному мажорированию множеств $\beta\gamma\delta$-трасс, порождаемых этими ϕ-трассами.

Следовательно, нам нужно доказать: $\cup \circ \xi^{\omega} \circ \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{I}) \preccurlyeq\preccurlyeq \cup \circ \xi^{\omega} \circ \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S})$.

Конечно-мажорантность спецификации означает $\boldsymbol{traces}_{\phi}(\mathbf{I}) \preccurlyeq \boldsymbol{traces}_{\phi}(\mathbf{S})$.

По Утверждение 56:, ϕ-трассы генеративны, следовательно:

$\cup \circ \xi \circ \boldsymbol{traces}_{\phi}(\mathbf{I}) \preccurlyeq\preccurlyeq \cup \circ \xi \circ \boldsymbol{traces}_{\phi}(\mathbf{S})$.

Для конечных ϕ-трасс ξ-оператор и ξ^{ω}-оператор совпадают, следовательно:

$\cup \circ \xi^{\omega} \circ \boldsymbol{traces}_{\phi}(\mathbf{I}) \preccurlyeq\preccurlyeq \cup \circ \xi^{\omega} \circ \boldsymbol{traces}_{\phi}(\mathbf{S})$.

Конечная ϕ-трасса порождает только конечные $\beta\gamma\delta$-трассы.
Бесконечная ϕ-трасса также может порождать конечные $\beta\gamma\delta$-трассы, если она имеет конечную подтрассу базовых символов.
Такая ϕ-трасса заканчивается бесконечным повторением одного и того же ϕ-символа.
Однако в этом случае все порождаемые ею конечные $\beta\gamma\delta$-трассы порождаются её конечным префиксом, заканчивающимся этим ϕ-символом.

Поэтому нам достаточно доказать, что каждая бесконечная $\beta\gamma\delta$-трасса реализации сильно мажорируется $\beta\gamma\delta$-трассами спецификации.

Обозначим: $\mathbf{I} = \subset_{\beta\gamma\delta}^{\infty} \cap (\cup \circ \xi^{\omega} \circ \boldsymbol{traces}_{\phi}^{\infty}(\mathbf{I}))$, $\mathbf{\Sigma} = \cup \circ \xi^{\omega} \circ \boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S})$.

Нам надо доказать: $\forall \mu \in \mathbf{I}\ \ \mu \preccurlyeq\preccurlyeq \mathbf{\Sigma}$.

Рассмотрим два случая.

1. Некоторый конечный префикс $\beta\gamma\delta$-трассы $\mu_1 < \mu$ имеет γ-мажоранту: $\exists \sigma \in \mathbf{\Sigma}\ \ \mu_1 \preccurlyeq\preccurlyeq^{\gamma} \sigma$.

Тогда, по определению сильного мажорирования $\beta\gamma\delta$-трасс, $\mu \preccurlyeq\preccurlyeq^{\gamma} \sigma$ и $\mu \preccurlyeq\preccurlyeq^{\gamma} \mathbf{\Sigma}$.

2. Каждый конечный префикс βγδ-трассы $\mu_1<\mu$ не имеет в $\mathbf{\Sigma}$ γ-мажоранты.

Поскольку конечные βγδ-трассы реализации сильно-мажорируются конечными βγδ-трассами спецификации, должно быть мажорирование по равенству, то есть каждый конечный префикс βγδ-трассы $\mu_1<\mu$ принадлежит спецификации $\mu_1\in\mathbf{\Sigma}$.

По Утверждение 53:, опасная βγδ-трасса имеет γ-мажоранту.
Следовательно, каждый конечный префикс $\mu_1<\mu$ безопасен в $\mathbf{S}$.

Докажем, что в этом случае $\mu\in\mathbf{\Sigma}$ и, тем самым, $\mu\preccurlyeq\preccurlyeq\mathbf{\Sigma}$.

Рассмотрим два подслучая.

2.1. βγδ-трасса μ имеет конечную подтрассу базовых символов.

Такая βγδ-трасса заканчивается бесконечным постфиксом βδ-отказов $\mathbf{o}$ и представима в виде $\mu=\mu`\cdot\mathbf{o}$.
Далее βγδ-трасса μ порождается в реализации некоторой ϕ-трассой $\boldsymbol{\mu}$, которая имеет ту же самую конечную подтрассу базовых символов и, следовательно, заканчивается бесконечным повторением одного и того же ϕ-символа o, то есть представима в виде $\boldsymbol{\mu}=\boldsymbol{\mu}`\cdot\{o\}^{\infty}$, причём $\boldsymbol{Im}(o)\subseteq o_r$.
Такая ϕ-трасса $\boldsymbol{\mu}$ в реализации заканчивается в таком состоянии s, что $\beta\delta(s)=o_r$.
Следовательно, βγδ-трасса $\mu`$ в реализации также заканчивается в состоянии s.
Поскольку βγδ-трасса $\mu`$ как конечный префикс $\mu$ безопасна в $\mathbf{S}$, по Утверждение 70:, $\mathbf{S}$ ***after*** $\mu`$ конечно.
Перенумеруем его состояния $s_1,\ldots,s_n$.
Покажем, что для некоторого i имеет место: $\boldsymbol{Im}(o)\subseteq\beta\delta(s_i)$.

Допустим противное.
Тогда для каждого i можно выбрать βδ-отказ $r_i\in\boldsymbol{Im}(o)\setminus\beta\delta(s_i)$.
Поскольку число состояний конечно, мы имеем конечное множество таких βδ-отказов $r=\{r_1,\ldots,r_n\}$.
Тогда в βδ-трассе $\mathbf{o}$ найдётся конечный префикс, содержащий все эти βδ-отказы: $o`<o \;\&\; r\subseteq\boldsymbol{Im}(o`)$.
Поскольку βγδ-трасса $\mu`\cdot o`<\mu$, она есть в спецификации $\mu`\cdot o`\in\mathbf{\Sigma}$.

Такая βγδ-трасса $\mu`\cdot o`$ должна заканчиваться в некотором состоянии $s_i \in (\mathbf{S}\ \textit{after}\ \mu`)$, что противоречит $r_i \in \textit{Im}(o) \setminus \beta\delta(s_i)$.

Тогда в спецификации βγδ-трасса $\mu`$ заканчивается в таком состоянии s_i, что $\textit{Im}(o) \subseteq \beta\delta(s_i)$.
Тем самым в спецификации имеется βγδ-трасса $\mu = \mu`\cdot o$.

2.2. βγδ-трасса μ имеет бесконечную подтрассу базовых символов.

В рамках данного доказательства нам будет удобно префикс βγδ-трассы μ называть *удобной βγδ-трассой*. Маршрут спецификации, имеющий (в числе прочих) удобную βγδ-трассу, будем называть удобным маршрутом. Множество удобных маршрутов обозначим $P = \{p \in \textit{runs}^{\omega}(\mathbf{S}) \mid \exists \mu` \in \textit{traces}_{\beta\gamma\delta}^{\omega}(p)\ \ \mu` \le \mu\}$.

Покажем, что множество P дерево.

Префикс удобного маршрута, очевидно, имеет (в числе прочих) префикс его удобной βγδ-трассы, который также удобен.
Следовательно, удобен префикс удобного маршрута.

Покажем, что дерево P конечно-ветвящееся.

Удобная βγδ-трасса $\mu`$ безопасна в спецификации и её подтрасса базовых символов является префиксом подтрассы базовых символов βγδ-трассы $\mu$: $\mu`{\downarrow}C \le \mu{\downarrow}C$.
Поэтому удобная βγδ-трасса $\mu`$ может удобно продолжаться только одним стимулом или реакцией $\mu{\downarrow}C(|\mu`{\downarrow}C|+1)$.
Отсюда следует, что удобный маршрут с βγδ-трасса $\mu`$ может удобно продолжаться либо τ-переходом (не меняющим его βγδ-трассы), либо переходами по одному стимулу или реакции z.
Поскольку $\mu`$ и $\mu`\cdot\langle z\rangle$ безопасны в спецификации, удобный маршрут заканчивается только в *S*-достижимых состояниях и z безопасен в этих состояниях.
Поэтому в *S*-ветвящейся LTS число таких переходов (по τ и по z) конечно.
Следовательно, удобный маршрут удобно продолжается конечным числом переходов.
А это и означает, что дерево удобных маршрутов конечно-ветвящееся.

Покажем, что дерево P бесконечно.

Допустим противное: P конечно.
Поскольку бесконечный удобный маршрут имеет бесконечное число префиксов и они удобны, P не может содержать бесконечных маршрутов.
Следовательно, P состоит из конечного числа конечных маршрутов.
Тогда длина подтрассы базовых символов удобной βγδ-трассы ограничена сверху некоторым числом n, что противоречит тому, что βγδ-трасса μ имеет бесконечную подтрассу базовых символов.
Мы пришли к противоречию и, следовательно, дерево P бесконечно.

Итак, дерево P конечно-ветвящееся и бесконечное.
Тогда, по теореме Кёнига [KÖNIG], существует бесконечный удобный маршрут p.[78]

Нам достаточно показать, что p имеет бесконечную удобную βγδ-трассу, поскольку такой βγδ-трассой может быть только βγδ-трасса μ.

Покажем, что маршрут p имеет бесконечную трассу.

Если p имеет конечную трассу, он заканчивается бесконечным постфиксом τ-маршрутов.
Любая βγδ-трасса маршрута p заканчивается или продолжается разрушением (моделирующим дивергенцию).
Следовательно, удобная βγδ-трасса этого маршрута также продолжается или заканчивается разрушением, что противоречит её безопасности в спецификации.

Итак, маршрут p имеет бесконечную трассу и, следовательно, все его βγδ-трассы бесконечны.
Следовательно, удобная βγδ-трасса маршрута p бесконечна.

110. **Доказательство Утверждение 110:**

По Утверждение 105:, $\tau_{\gamma\delta}(\mathbf{s})$ γ-однородная LTS.

---

[78] В отличие от трасс, βγδ-трасс и ϕ-трасс существование такого маршрута в дереве означает его существование в LTS.

Тогда, по Утверждение 108:, преобразование $\mathcal{T}_{\gamma\delta}$ конечно-мажорантно на классе $\boldsymbol{LTS}_{\beta\gamma\delta}$ и, следовательно, на его подклассе $\boldsymbol{SBLTS}_{\beta\gamma\delta}$.
По построению, преобразование $\mathcal{T}_{\gamma\delta}$ сохраняет *S*-ветвимость LTS.
Следовательно, по Утверждение 109:, преобразование $\mathcal{T}_{\gamma\delta}$ мажорантно на классе *S*-ветвящихся LTS.

111. Доказательство Утверждение 111:

У нас выполнены все восемь условий монотонности для соответствия $\boldsymbol{ioco}_{\beta\gamma\delta}$ и преобразования $\mathcal{T}_{\gamma\delta}$ на классе $\boldsymbol{SBLTS}_{\gamma\delta}$:

1. Эквивалентность соответствия мажорированию βγδ-трасс (Утверждение 54:).
2. Генеративность ϕ-трасс (Утверждение 56:).
3. Аддитивность ϕ-трасс относительно композиции (Утверждение 60:).
4. Генеративность мажорирования ϕ-трасс (Утверждение 103:).
5. Композиционность мажорирования ϕ-трасс на классе всех LTS $\boldsymbol{LTS}_{\beta\gamma\delta}$ (Утверждение 104:).
6. Конформность преобразования $\mathcal{T}_{\gamma\delta}$ на классе всех LTS $\boldsymbol{LTS}_{\beta\gamma\delta}$ (Утверждение 106:).
7. Мажорантность преобразования $\mathcal{T}_{\gamma\delta}$ на классе *S*-ветвящихся LTS без блокировок $\boldsymbol{SBLTS}_{\gamma\delta}$ (Утверждение 110:).
8. Рефлексивность мажорирования ϕ-трасс на классе LTS без блокировок $\boldsymbol{LTS}_{\gamma\delta}$ (Утверждение 102:).

Поэтому, по Утверждение 52:, соответствие $\boldsymbol{ioco}_{\beta\gamma\delta}$ $\mathcal{T}_{\gamma\delta}$-левомонотонно и $\mathcal{T}_{\gamma\delta}$-монотонно на классе $\boldsymbol{SBLTS}_{\gamma\delta}$.

112. Доказательство Утверждение 112:

Сохранение свойств *S*-τ-ограниченности и *S*-регулярности непосредственно следует из того, что преобразование $\mathcal{T}_{\gamma\delta}$ не удаляет переходы и для каждой реакции $!y$, опасной в состоянии s, добавляет ровно один переход $s—!y→\gamma$.

113. Доказательство Утверждение 113:

Итераторы стимулов и реакций не меняются при преобразовании.
Нам нужно построить итераторы $\mathbf{Extend2}_{\mathcal{T}_{\gamma\delta}(s)}(s,z)$, $\mathbf{Tau2}_{\mathcal{T}_{\gamma\delta}(s)}(s)$ и оракул $\mathbf{Gamma2}_{\mathcal{T}_{\gamma\delta}(s)}(s)$, где s *S*-достижимое состояние $\mathcal{T}_{\gamma\delta}(\mathbf{S})$, а также оракул $\mathbf{Gamma2}_{\mathcal{T}_{\gamma\delta}(s)}(\gamma)=\boldsymbol{true}$.

Очевидно, s S-достижимое состояние $\mathbf{S}$.
Поэтому $\mathbf{Tau2}_{\tau_{\gamma\delta(\mathbf{S})}}(s)=\mathbf{Tau2}_{\mathbf{S}}(s)$ и
$\mathbf{Gamma2}_{\tau_{\gamma\delta(\mathbf{S})}}(s)=\mathbf{Gamma2}_{\mathbf{S}}(s)$.

Построим итератор $\mathbf{Extend2}_{\tau_{\gamma\delta(\mathbf{S})}}(s,z)$.

Если $z=?x$ стимул, то $\mathbf{Extend2}_{\tau_{\gamma\delta(\mathbf{S})}}(s,?x)=\mathbf{Extend2}_{\mathbf{S}}(s,?x)$.

Если $z=!y$ реакция, то нам нужно проверить, безопасны ли реакции в состоянии s. В доказательстве Утверждение 48: показано, что это можно сделать за конечное время.

Если реакции безопасны, то $\mathbf{Extend2}_{\tau_{\gamma\delta(\mathbf{S})}}(s,!y)=\mathbf{Extend2}_{\mathbf{S}}(s,!y)$.
В противном случае $\mathbf{Extend2}_{\tau_{\gamma\delta(\mathbf{S})}}(s,!y)$ начинает перечисление с состояния γ, а далее перечисляет те же постсостояния, что итератор $\mathbf{Extend2}_{\mathbf{S}}(s,!t)$.

114. Доказательство Утверждение 114:

Мажорирование множеств ϕ-трасс сводится к сильному мажорированию множеств порождаемых этими ϕ-трассами βγδ-трасс. Поэтому нам достаточно доказать, что каждая βγδ-трасса $\sigma\in\cup_{\circ}\xi^{\omega}_{\circ}\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S})$ имеет сильную мажоранту $\sigma\preccurlyeq\preccurlyeq\sigma`\in\cup_{\circ}\xi^{\omega}_{\circ}\boldsymbol{traces}_{\phi}^{\omega}{}_{\circ}\mathcal{C}(\mathbf{S})$.

Рассмотрим два случая.

1. $\sigma\in\cup_{\circ}\xi^{\omega}_{\circ}\boldsymbol{traces}_{\phi}^{\omega}{}_{\circ}\mathcal{C}(\mathbf{S})$. Тогда $\sigma\preccurlyeq\preccurlyeq\sigma$.
2. $\sigma\notin\cup_{\circ}\xi^{\omega}_{\circ}\boldsymbol{traces}_{\phi}^{\omega}{}_{\circ}\mathcal{C}(\mathbf{S})$.
   Рассмотрим в $\mathbf{S}$ маршрут S с βγδ-трассой σ и максимальный его конечный префикс $M<S$, имеющийся в $\mathcal{C}(\mathbf{S})$ и имеющий в $\mathbf{S}$ βγδ-трассу $\mu<\sigma$.
   Очевидно, все состояния маршрута M отличны от $\gamma`$.
   Тогда, по Утверждение 93:, маршрут M имеет в $\mathcal{C}(\mathbf{S})$ βγδ-трассу μ.
   Следующий в маршруте S после маршрута M переход $\omega(M)$—z→, по построению, заменяется в $\mathcal{C}(\mathbf{S})$ на переход $\omega(M)$—z→$\gamma`$.
   Тогда маршрут $M`=M\cdot\langle\omega(M)$—$z$→$\gamma`,\gamma`$—$\gamma$→$\gamma`\rangle$ имеет в $\mathcal{C}(\mathbf{S})$ βγδ-трассу $\mu\cdot\langle z,\gamma\rangle$.

Тогда, очевидно, $\mu\cdot\langle z\rangle<\sigma$ и, поскольку $\mu\cdot\langle z\rangle\preccurlyeq^{\gamma}\mu\cdot\langle z,\gamma\rangle$, имеем $\sigma\preccurlyeq\preccurlyeq\mu\cdot\langle z,\gamma\rangle\in\cup\circ\xi^{\omega}\circ\boldsymbol{traces}_{\phi}^{\omega}\circ\mathcal{C}(\mathbf{S})$.

115. Доказательство Утверждение 115:

Доказательство полностью аналогично доказательству Утверждение 95:. Отличие только в ссылках на используемые утверждения и преобразованиях: $\mathcal{W}_{\beta\gamma\delta}$ или $\mathcal{T}_{\gamma\delta}$.

По Утверждение 52:, нам достаточно показать конформность и мажорантность преобразования $\mathcal{C}\circ\mathcal{T}_{\gamma\delta}$.

Конформность.
По Утверждение 106:, $\mathcal{T}_{\gamma\delta}(\mathbf{S})\ \boldsymbol{ioco}_{\beta\gamma\delta}\ \mathbf{S}$.
По Утверждение 94: (1), $\mathcal{C}\circ\mathcal{T}_{\gamma\delta}(\mathbf{S})\ \boldsymbol{ioco}_{\beta\gamma\delta}\ \mathcal{T}_{\gamma\delta}(\mathbf{S})$.
По Утверждение 20:, отношение $\boldsymbol{ioco}_{\beta\gamma\delta}$ транзитивно.
Следовательно, $\mathcal{C}\circ\mathcal{T}_{\gamma\delta}(\mathbf{S})\ \boldsymbol{ioco}_{\beta\gamma\delta}\ \mathbf{S}$.

Мажорантность.
По Утверждение 110:, преобразование $\mathcal{T}_{\gamma\delta}$ мажорантно.
По Утверждение 114:, $\boldsymbol{traces}_{\phi}^{\omega}\circ\mathcal{C}(\mathbf{S})\succcurlyeq\boldsymbol{traces}_{\phi}^{\omega}(\mathbf{S})$.
По Утверждение 102:, мажорирование множеств ϕ-трасс LTS транзитивно.
Следовательно, преобразование $\mathcal{C}\circ\mathcal{T}_{\gamma\delta}$ мажорантно.

116. Доказательство Утверждение 116:

Непосредственно следует из Утверждение 111:, поскольку преобразование $\mathcal{T}_{\gamma\delta}$ для LTS без блокировок и разрушения совпадает с тождественным преобразованием.